QOS PREDICTION AND EVALUATION FOR NETWORKED TELELEARNING APPLICATIONS

by

Yu Chen

B.Sc., Beijing University of Posts & Telecoms, 1992

THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

In The School Of

Engineering Science

Yu Chen 1999

SIMON FRASER UNIVERSITY

March 1999

Approval

Name: Yu Chen

Degree: Master of Applied Science

Title of thesis: QoS Prediction and Evaluation for Networked Telelearning Applications

Examining Committee:

Chair: Dr. M. Saif, P.Eng.

_____
Dr. Jacques Vaisey, P.Eng.
Associate Professor
School of Engineering Science

_____
Dr. Steve Hardy
Professor
School of Engineering Science

_____
Dr. Paul Ho
Professor
School of Engineering Science

Date Approved: _____

Abstract

Telelearning is a collection of strategies and techniques for instruction at a distance. With the relatively recent success of the Internet and its almost universal accessibility, the web has become a very attractive vehicle for the delivery of courses. Depending on the course content and the amount of synchronous interaction between students, the amount of network traffic can vary tremendously – and this can have a huge impact on the quality of service (QoS) experienced by the system users. Thus, in planning to deliver courses, network administrators and course designers need answers to the following questions:

- What is the "capacity" for a given system configuration and QoS criteria?
- What is the effect of changing the course content on the QoS?
- How do the critical resources in the system affect the overall QoS?
- What is the most effective systems architecture?
- What system resources need to be present to offer a course.

The answer to each of these questions can be quite complicated, since systems have many parameters and each user will interact with the environment in unique and often complicated ways .

This thesis describes our efforts to answer these questions with the OPNET computer simulation package. We have studied the configurations and usage of the experimental Virtual-U, and have built computer models of the following main system components: the network, the server system, and the user model for interacting with the system in the specific courses. In simulations, parameters such as the server

processing rate, the intensity of background traffic and the course content are varied to see the effect of these changes and predict the system capacity.

From the simulation results, we discovered that the system bottleneck in the current Virtual-U system is in the server. From the network bandwidth perspective, the server's subnet is most likely to suffer bandwidth starvation. The user's activity and the course material content all have effects on the system performance. Adding more multimedia content in a course will increase the load to the system, but if we control it at a reasonable level under the system capacity, the degradation of QoS is not dramatic. Although some results need additional work for improved accuracy, these results show that our method of carrying out QoS research through computer simulation is valid and that the simulation tool we have built is able to aid the development and application of telelearning systems such as Virtual-U.

Dedication

To my wife Jun Yan.

Acknowledgments

Thanks to Cindy Xin of the Telelearning Research Lab for providing VU server log data.

Table of Contents

List of Tables

List of Figures

Abbreviations

CD	Collision Detect
CGI	Common gateway Interface
CRC	Cyclic Redundancy Check
CSMA	Carrier Sense Medium Access
FTP	File Transfer Protocol
GIF	Graphics Interchange Format
HTML	HyperText Markup Language
HTTP	Hypter Text Transfre Protocol
IEEE	Institution of Electrical and Electronic Engineers
IP	Internet Protocol
ISO	International Standards Organization
JPEG	Joint Photographic Experts Group
LAN	Local Area Network
MAC	Medium Access Control
MPEG	Moving Picture Experts Group
NNTP	Network News Transport Protocol
OSI	Open Systems Interconnection
QoS	Quality of Service
SMTP	Simple Mail Transfer Protocol
TCP	Transmission Control Protocol
TL-NCE	Telelearning Network of Centers of Excellence
UDP	User Datagram Protocol
VBR	Variable Bit Rate
VOD	Video On Demand
VU	Virtual-U

WWW Word Wide Web

Chapter I
Introduction

Telelearning is a collection of methods and technologies of using networked computer environment and tools for education and training. It will provide “virtual universities” on the network and enable students to access learning resources beyond the “border” of conventional classrooms. The Telelearning Network of Centers of Excellence (TL-NCE) is conducting the research and development of telelearning systems to make them able to "support the development of a knowledge economy and learning society in Canada" [Telelearning Web]. In order to make a better design of telelearning systems, and also a better usage of telelearning systems to deliver courses, system designers and educators need answers to the following questions:

- What is the capacity of a system for a given system configuration and QoS criteria?
- What is the effect of changing the course content on the QoS?
- How do the critical resources in the system affect overall QoS?
- What is the most effective system architecture?
- What system resources are required to offer a course?

The answer to each of these questions can be quite complicated, since systems have many parameters and each user will interact with the environment in unique and often complicated ways (which may interact with metrics such as the QoS). Nevertheless, it is possible to have a better understanding of the systems and their key factors by studying the behaviors of an experimental telelearning system in some “typical” conditions. Our research started by studying Virtual-U (VU), an

experimental Internet-based course delivery environment developed by researchers in Simon Fraser University.

Virtual-U [VU Web] is a web-based software system which allows universities or organizations to offer post-secondary level courses online. It includes tools for course design and facilitation, class discussion and presentation, course resource managing, class management and evaluation, and system administration. The basic Virtual-U components are:

1) VGroup Conferencing System (VG): "VGroup" supports group communication and collaboration in a secure newsgroup-style. Instructors can set up collaborative groups and define structures, tasks and objectives. Any user can learn to moderate conferences and to create sub-conferences. Users can easily sort messages in different ways to follow conversational threads, and view as a list of message titles or view the whole message.
2) Course Structuring Tool: This tool enables instructors to organize the resources of an on-line course.
3) Assignment Submission Tool: This tool enables instructors to request, receive and comment on course activity and assignment files which are submitted by students (to the course server) in a Virtual-U course.
4) Grade-book tool: This tool manages the database of students grades for each course delivered with Virtual-U.
5) System Administration Tool: This tool assists the system administrators in installing and maintaining Virtual-U.

Virtual-U is a learning environment based on the World-Wide-Web (WWW). This design makes Virtual-U easy to access and ready to integrate multimedia content. The "Web" [Tanenbaum, 1996], is an

architectural frame-work for accessing linked documents stored on the Internet. The richness of HTML, the language of the web, makes it relatively easy to add multimedia content to a web page. In addition, commonly available web browsers support multimedia content such as images, rich-text, and audio/video through third-party applications. By being centered in the Web, Virtual-U has inherited all of these strong points and is using them to build a "virtual university".

Virtual-U's 1997 design uses Common Gateway Interface (CGI) programs [Schwartz, 1993] because CGI is widely used in the development of Web servers and has a large amount of libraries available. The pages viewed by the user are not pre-made static Web pages, but generated dynamically by the responding CGI program according to the user's request. Inside each page there are some icons representing the choices available for the user, any choice made will invoke another CGI program to generate a new page. The communication protocols that Virtual-U based on are the TCP/IP [Miller, 1992] communication protocols. TCP/IP is very popular in today's network environments and widely supported by vendors and products. TCP/IP's popularity provides high accessibility to the applications based on them (such as VU), but it does not distinguish traffic according to its QoS requirements and do not provide QoS guarantees, thus it is difficult to support high quality multimedia services.

As of 1997, the version our work is based on, Virtual-U courses were based on an ascii-text conference model (using the VGroup conferencing system), where each course contains numerous conference threads in which students can post contributions or read what others have written. When taking a course offered by VU, a student mostly

interacts with the “VGroup” system. As the first step, a student logins into the system to view the “VGroup welcome page”. He or she then has several choices, such as reading course introduction material and listing the available conferences. If the student wants to see the conference list they simply click on the appropriate icon and list will be displayed. After the student finds the conference he or she is most interested in, he or she can then list all the unread messages. After that, what the student can do is similar to that in a regular Email system; i.e. the student can read a message or put his or her comments on a new message and post it . In each of the steps above, the student is able to move backwards and make other choices.

Virtual-U is evolving and improving. In order to provide a better learning environment, future versions of the system will include options for multimedia technologies such as “Video on Demand” (VOD) streaming applications, where the students can view a lecture stored on the server by playing with a “virtual VCR”; and video conferencing where a student can interact with the teacher and other students through their computers. These technologies are already available in the market, and there may be more compelling possibilities in the near future. In our research, we studied two types of courses: VU conference based courses (referred as VU course) and “VOD” courses that allow streaming video. More possibilities will be added to the simulation system in the future.

As we mentioned earlier, Virtual-U utilizes the popular TCP/IP based network to achieve high accessibility. However, the current (as opposed to next generation) TCP/IP protocols were designed for data-oriented services that do not have QoS constraints other than reliable delivery. These are “best effort” protocols in the sense that they do not

distinguish traffic according to their QoS requirements and do not provide QoS guarantees. For the a telelearning system that may have a lot of multimedia content, this kind of protocol is problematic, since multimedia services have strict QoS requirements, such as delay and delay jitter, that really need to be guaranteed. For multimedia services, "best effort" networks should be used with care. Good estimates of the expected traffic and available bandwidth are required so that the whole system can be run at a "safe" load and so that QoS degradation due to resource starvation will rarely happen. In the future, this problem will likely be solved through the usage of network protocols that support multiple QoS classes and provide QoS guarantees. Many protocols and proposals of this type are underway , such as IEEE 802.12 [Watson, 1995], Ethernet++ [Edwarts, 1995], IP version 6 [Huitema, 1998] et cetera. Some of them will be discussed in this thesis.

Quality of service (QoS) is one of the main interests of our research and in our context, it refers to performance measures such as delay, delay jitter, packet-loss rate, and distortion, as seen by the users and applications. Distortion is the reduction in the quality of the information perceived by the user because of quantization, compression, and loss et cetera. that may be necessary because of the limited bandwidth available on the channel. In telelearning systems such as Virtual-U, different types of applications and courses are run and they may have different QoS requirements. For example, delay and/or delay jitter must be upper-bounded to ensure in-time delivery. Loss and distortion must also be bounded to ensure a reasonable subjective quality. For interactive applications such as video conferencing, a long time delay will make the interaction difficult. On the other hand, for the VOD courses using

streaming video, delay jitter is even more harmful then the delay itself. In order to smooth the delay jitter caused by the network's traffic fluctuation, there is a play buffer in the viewer's play station. When the incoming (from the source) traffic rate is higher than the play station's displaying rate, the amount of data in the buffer increases; if the incoming traffic rate is lower than the displaying rate, the amount of data in the buffer decreases. Since the buffer has a limited volume, it can only handle fluctuations in the arrival rate that are within a certain level. Above this level, play buffer overflow or underflow will happen, and the perceived quality will degrade due to the loss of a video frame or the repetition of an old one. Compared courses rich in multimedia content, text based courses can tolerate much more delay; i.e., students will generally not complain about waiting for 3~5 seconds while retrieving a message from the server (but 30 seconds is definitely a problem). The wide range of requirements, from those with relaxed to stringent QoS parameters, suggests that it would be highly advantageous for applications to have control over the QoS provided to them.

Before any mechanisms are implemented to control the QoS of telelearning applications, it is very important to define appropriate QoS measures (or benchmarks) that can capture the essence of the impairments. At this stage, there are two considerations in selecting the QoS measures. Firstly, as we are mostly interested in the overall system capacity and QoS, the measures should be able to take into account the (QoS) contributions from all the sub-systems and reflect the "end to end" performance. Secondly, we should use both statistical and subjective measures. The statistical characteristics such as "maximum value", "minimum value", "mean" and "standard deviation" are widely used in

QoS analysis, but they do not always reflect the subjective quality of service. We should find ways to directly reflect the user's subjective QoS. As we know, QoS requirements are different for different types of services. Even for the same type of service, different users may have different QoS requirements. Therefore, we should define several QoS levels based on the "satisfactory levels" and study the percentage of services that fit into each of these QoS levels. As a reasonable starting point, three QoS levels are used in this thesis:

"Good":	Very good quality.
"Not good":	The quality is not good, but will be acceptable to most users.
"Bad":	Bad and not acceptable.

The QoS measures and parameters are chosen based on the above considerations. For example, for the Virtual-U conference based courses, we will measure the request response time (T), which is defined as the time from when a user sends out the request until the requested information is received. The parameters for the three QoS levels are set as: T<5s, 5s<=T<20s, T>20s respectively. These QoS measures will be used for system performance evaluation.

Our approach in carrying out research on the QoS of telelearning systems and to answer the above questions consists of computer simulations using a commercial package called OPNET[1]. We have

[1] For information about OPNET, please refer to Appendix A.

studied the configurations and usage of the experimental Virtual-U system, and have built computer models of the following main system components: the network, the server system, and the user's that interact with the system in the specific courses. In the simulations, parameters such as the server processing rate, the intensity of background traffic and the course content are varied to see the effect of these changes and to predict the system capacity.

From the simulation results, we have learned quite a lot about the system's capacity and were able to generate quantitative information on the effect of varying critical factors. For the system configuration of 1997, and with text-based courses, we have confirmed that the system bottleneck is not the network, but rather the server. The capacity of the system depends on the server's processing ability, which is determined by both hardware power and software design. Also, from the network bandwidth perspective, the server's subnet is most likely to suffer bandwidth starvation. The user's activity and the course content also effect the system performance. Adding more multimedia content to a course will increase the load on both the server and the network, but if we control the amount of traffic generated to be a reasonable amount below the network bandwidth limit, the degradation in the QoS is not dramatic. Although some results need additional work for improved accuracy, these results show that our method of carrying out QoS research through computer simulation is valid and the simulation tool we have built is able to provide useful information about the expected performance of telelearning systems.

In conclusion, the contributions we made in this research are: 1) We analyzed the system structure of Virtual-U (1997) and the usage

pattern of test courses, and constructed simulation models. 2) We implemented the simulation models into computer code and built a simulation tool in OPNET. 3) We executed experiments with the simulation system and obtained results about telelearning QoS issues.

The remaining sections of this thesis are organized as follows. In Chapter 2, we provide background information about computer networks and computer simulation. Chapter 3 describes our method of building a simulation system and presents a simulation system of Virtual-U as an example. Some experiment results generated by our simulation system are then discussed in Chapter 4. Finally, we evaluate our simulation system and point out further improvements in Chapter 5.

Chapter II
Computer Network and Computer Simulation

2.1 Computer Network and Protocols

A telelearning system is to provide a networked learning environment and its success relies on the supporting network. Modern computer networks are designed in a highly structured way to reduce the design complexity. That is, the networks are organized as a series of layers, each layer is responsible for a certain group of functions. The layers are independent and communicate through interfaces. A popular reference model for the structure of computer networks is the ISO OSI (Open Systems Interconnection) Reference Model [Tanenbaum, 1996], which is developed by the International Standards Organization (ISO) as a step toward international standardization of various protocols. Standardized protocols have enabled many communications systems to respond to the demand for network interoperability. This is very important for telelearning systems that ideally need a network with no border.

In the OSI model (refer to Figure 2.1) there are seven functional layers. The application layer defines the tasks performed from the user's perspective. Examples include Email, file transfer and Web browsing. The Virtual-U course environment belongs to this layer. The presentation layer handles any conversions required to prepare files for presentation to the user. The session layer establishes sessions for users on different machines. One example of the session layer services is the "token

management" which makes sure that the two communication sides will not attempt the conflicting operations at the same time.

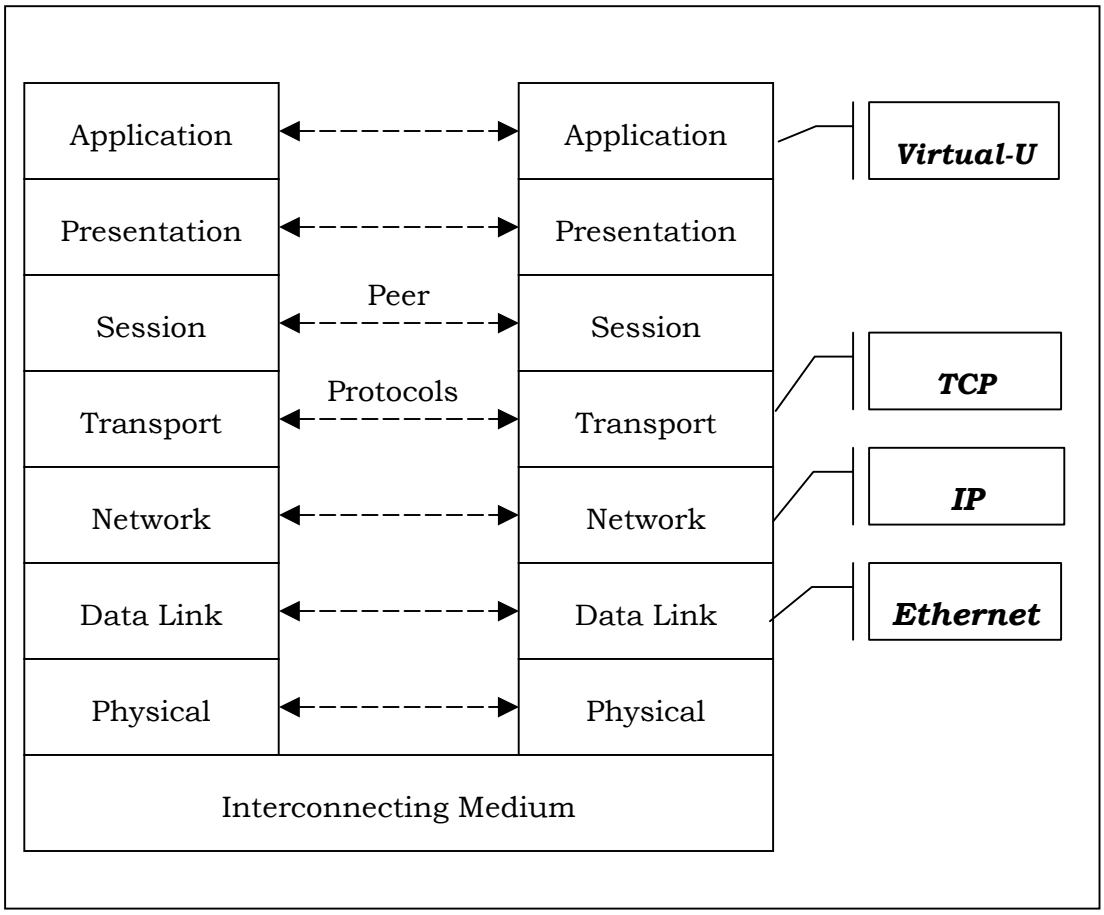

Figure 2.1 The OSI Model[2] [Freeman, 1995]

[2] This figure is drawn based on the Figure 5.20 "The OSI Model" of [Freeman, 1995].

The transport layer is responsible for end-to-end file delivery. This layer includes protocols for detecting and correcting errors that occur during file transfer. The popular Transmission Control Protocol (TCP) is an example of this layer. The network layer is responsible for delivering packets of information that will be assembled into files in the transport layer. Routers, the equipment broadly used in today's network to move traffic from node to node, operate in the network layer to determine which path can be used to access the destination computer. Routers can also implement congestion control algorithms, which can improve overall system performance. The Internet Protocol (IP) in common use is a protocol of the network layer. The data-link layer defines the frames of data traveling through a network, as well as error-correction and retransmission schemes. The frame format may include CRC checks and error-correction codes, which allow the link to appear virtually error-free to the user. LAN protocols, such as Ethernet and Token Ring, belong to this layer. Bridges, which are used for connecting multiple LANs together, implement some of the data-link functions. The physical layer defines how unstructured binary digits (bits) travel over the physical media between machines. Physical-layer protocols define pin configurations for cables and voltage levels.

The OSI model is a conceptual model. In real networks, the functions are not always implemented according to the seven layer definition. For example, the web systems in the Internet do not have explicitly defined "presentation " and "session" layers. The necessary functions are instead implemented in the "Application layer" for system simplicity and efficiency. The stack model of Virtual–U and its supporting

network environment include Ethernet, the TCP/IP protocol suite and the Virtual-U applications.

The Ethernet is a bus-based local area network (LAN) technology whose operation is managed by a medium access control (MAC)[3] protocol, which has been standardized by the Institution of Electrical and Electronic Engineers (IEEE) under the name 802.3 [Tanenbaum, 1996]. The role of this MAC protocol is to provide efficient and fair sharing of the communication channel, which in our case is the 10Mbps bus connecting the stations of the LAN. The Ethernet MAC accepts data packets from a higher layer protocol (such as IP) and transmits them at appropriate times to other stations on the bus. Because the higher layer protocols can forward data at any time and the bus is a broadcast medium, collisions are unavoidable in the Ethernet protocol. Ethernet therefore attempts to provide efficient mechanisms for handling collisions, i.e. carrier sensing and collision detecting (CSMA/CD). When a station wants to transmit, it listens to the bus. If the bus is busy, the station waits until it goes idle, otherwise it transmits immediately. If two or more stations simultaneously begin transmitting on the "idle" bus, they will collide. All colliding stations then terminate their transmission, wait a random time, and repeat the whole process all over again. In June 1995, IEEE approved 802.3u (which is commonly called "Fast Ethernet")[Johnson, 1996]. This protocol keeps the frame format of 802.3 and the CSMA/CD, but makes it go faster-100Mbps. Technically, 802.3u is an addendum to the 802.3 rather than a new protocol.

---

[3] MAC is a sub-layer of the data link layer in the OSI model.

The Ethernet is widely used in today's network because its algorithm is simple and stations can be installed without taking the network down. Furthermore, the delay at low load is practically zero. Stations can transmit data as soon as the bus is free and do not have to wait for tokens as in other systems such as Token Ring [Tanenbaum, 1996]. However, "802.3" is non-deterministic, which is often inappropriate for real time work because it does not distinguish traffic with different priorities. At heavy load, collisions become frequent, which has a serious impact on the throughput. An overloaded Ethernet will collapse totally and the data throughput will go to zero. The utilization of the Ethernet depends on many factors, such as the number of stations, the frame size, the traffic load of each station et cetera. As a general rule, the actual capacity of an Ethernet system is approximately 80% of its stated maximum; i.e. on our subnets, the maximum throughput is about 8Mbps.

The Internet Protocol [Freeman, 1995] is a connectionless network layer protocol whose task is to interconnect multiple networks. It provides services to transport layer protocols (e.g. TCP and UDP) and relies on the services of data link layer protocols (e.g. Ethernet and Token Ring) to relay packets to other IP modules. Within a single host or router in the network, IP may have several interfaces to different types of data link layer protocols. This property provides the ability to route packets between different types of networks. IP essentially works like a "glue" that binds the different networks together into one network. Packets that are created and forwarded by IP modules are called datagrams. IP datagrams carry headers that hold control information such as the source and destination addresses, type of service, identifier and so on.

Because IP connects different types of networks that may support different maximum packet lengths, datagrams may have to be broken into fragments, which are also considered datagrams. The various resulting fragments may travel independently through the network, following completely different routes and possibly arriving out of order at the destination. IP therefore has the task of reassembling the fragments before it can deliver data to the higher level protocols. This process is transparent to the upper layer for data services such as file transfer. But for multimedia services such as streaming video that require sequential delivery, this fragmentation is problematic. Packets' "arriving out of order" may cause delay and delay jitter which will seriously affect the perceived quality of service.

The Transmission Control Protocol (TCP) [Freeman, 1995], typically used with IP, is a widely used connection-oriented transport layer protocol that provides reliable, ordered packet delivery over an unreliable network. TCP forms a communication channel between two higher layer entities, referred to as applications that operate across a network. Another popular transport layer protocol is the User Datagram Protocol (UDP) [Tanenbaum, 1996]. Unlike TCP, UDP is a connectionless protocol; i.e. it allows users to send messages without first establishing a connection. However, UDP does not provide a guarantee of delivery and sequencing and can be viewed as simply a user interface to IP.

In conclusion, today's computer networks are organized in a structured manner. Different types of networks and systems communicate each other through standard protocols. Applications are provided with vehicles and routes from the network to reach destinations far away. Telelearning systems like Virtual-U use popular network

technologies (e.g. TCP/IP, Ethernet) to achieve their high accessibility and take advantages of the considerable operational experience. However, these mature technologies do not distinguish traffic according to their QoS requirements and transmit them with "best efforts". They should be used with care when carrying multimedia services.

2.2 Computer Simulation

While making applications like Virtual-U more and more far reaching, today's communication networks and computer systems are getting more and more complex. It is too difficult to analyze the system performance with only paper and a pencil, because the systems are mathematically intractable and simplistic models are not representative of the true complexity. Queuing theory based analytical models [Schwartz, 1987] are very useful for the performance analysis of telephone networks, where only the number and the duration of calls are interesting from the traffic perspective. On the contrary, telelearning systems contain users, servers, and network components each of which is very complex by itself. It is impossible to use a single analytic model to represent all of them. Meanwhile, the behavior of the components is not independent. For example, too many user requests to the server will cause long response times , since requests will get piled up in server's waiting queue. The requests will be re-sent by the users or the application software if their waiting timers expire. This re-sending will continue until either the requested data from the server arrives or a certain number of retries fail. This re-sending mechanism can cause congestion of the network connecting the user and the server, which will make it harder for the server to finish its current work. This example

tells us that separately considering system components such as the network and the server will not lead to a correct estimation of the performance. Thus, the approach of decomposing the whole system into separate analytic models will not allow us to produce satisfactory results.

Fortunately, computer simulation is at the stage where it can play becoming an important role in performance analysis. The type of simulation that we are interested in is called discrete-event system-level simulation [MacDougall, 1987]. Discrete-event systems change state at discrete points in time, as opposed to continuous systems which change state over time. Real systems can be modeled at several levels of detail. Since we are interested in the overall system from a performance standpoint, we should represent only those elements of the system pertinent to the performance issues.

The three basic components of the simulation are the input, the simulation system and the output. Simulation input data may either be generated probabilistically within the simulation program according to models obtained from the real system, or it may be generated externally. For example, in a trace-driven simulation, the input is obtained from a trace of real system execution.

In modeling a system, we need to describe both its structure and the way in which it accomplishes work. Developing a model to represent the real system has two tasks: developing a representation of the system, and developing a representation of the work to be done by the system (this is also called workload characterization). The simulation outputs are the functions of the input to the simulation system. The analysis of

these functions leads to system understanding and performance evaluation.

The modeling and analysis process is outlined in Figure 2.2. The first step is to describe the system operation from a performance viewpoint and then to abstract the description into models which include both the represented facilities and their attributes according to the analysis objectives. In the next step, the appropriate analysis method is chosen, which includes the definition of a set of performance measures. Thirdly, a model implementation is developed. This also includes the program debugging and verification. Verification insures that the simulation program is indeed an implementation of the model. The following step is to validate that the model (and of course the simulation system implemented in computer programs) is a reasonable representation of the real system. This can be done by executing some experiments and comparing the results with real systems. Finally, simulations are executed and results are analyzed.

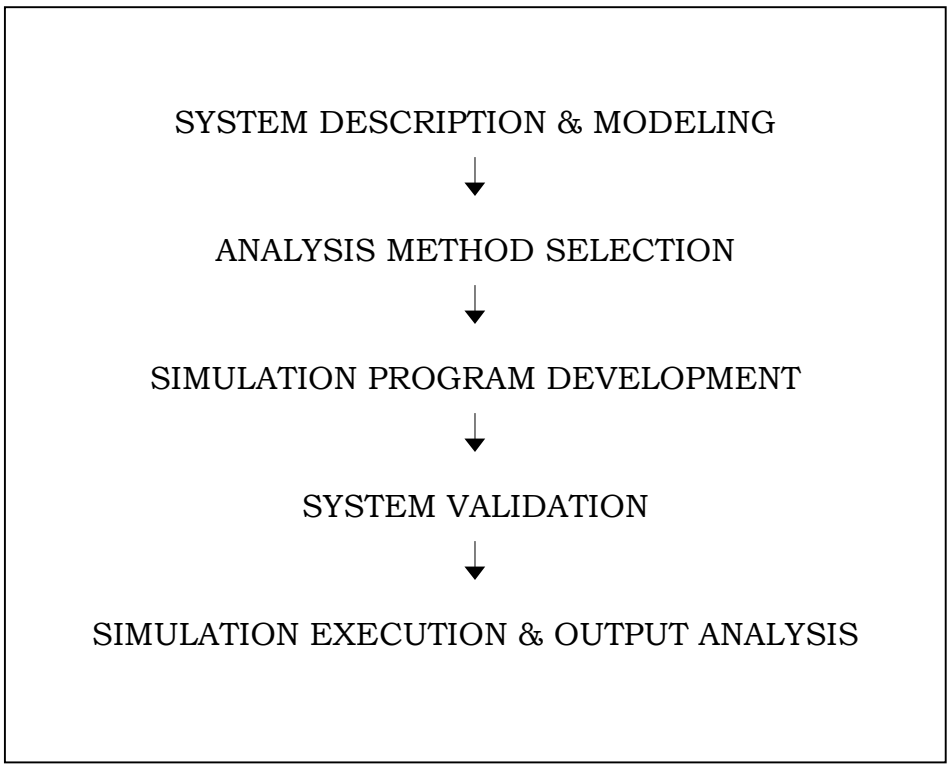

Figure 2.2 The Modeling and Analysis Process

In the implementation of a complicated simulation system, it is possible to use existing software packages such as OPNET [OPNET Web] to speed up the development and to focus on the issues at hand. OPNET provides a "comprehensive software environment for modeling, simulating, and analyzing the performance of communications networks, computer systems and applications, and distributed systems"[OPNET Web] that organizes models in a hierarchical way (i.e. network model , node model and process model). The network models define the interconnections between the communicating entities (nodes). Each node is described by a block structured data flow diagram, which defines the interrelation of processes in a sub-system. Every programmable

block in a node model has its functionality defined by a process model which "combines the graphical power of a state-transition diagram with the flexibility of a standard programming language and a broad library of pre-defined modeling functions" [OPNET Web]. Our simulation of the Virtual-U is implemented and executed in OPNET.

2.3 Traffic Modeling

A communications network is used to carry traffic. In telelearning systems and all other networked systems, the applications interact with the network and other components in the network (e.g. the server) through traffic. Users consume system resources through their generated traffic. Traffic is thus a key component throughout all network related issues and modeling is crucial in network performance research. In our simulation, two types of traffic models are constructed; i.e. ,the model describing the traffic generated by the telelearning users model for background traffic, which is defined as the traffic on the system corresponding to non-telelearning usage. Telelearning systems may provide a lot of multimedia services. Therefore, during the traffic modeling, not only the number of requests but also the statistical properties of the information transmitted for the requests is important. Based on statistical empirical data on multimedia traffic, a number of models have been advanced to capture the statistical nature of multimedia traffic. Stochastic source models simulate the behavior of the traffic generated by a terminal (e.g. a computer terminal, a video-on-demand server et cetera.), while more general models represent the multiplexed traffic of many sources. The set of traffic models examined in the literature is rather large, but we can nevertheless make a broad

distinction between two categories, according to the purpose for which they are usually used. On the one hand, there is a search for models that capture the relevant statistical properties of a specific kind of traffic from a source as accurately as possible. A good traffic model will not only capture the first few moments of the statistics, but will render higher order statistics as well. For example, there have been many proposals for modeling variable-bit-rate (VBR) video traffic, using first order auto-regressive models [Maglaris, 1988] and Markov Modulated Poisson processes [Schwartz, 1996]. On the other hand, we also have models that are based solely on a few parameters extracted from the traffic characteristics. These parameters (peak rate, burstiness, average rate, and some tolerances) are insufficient to fully describe a traffic stream; however, they can be used to generate upper bounds on loss and delay of traffic. These models are called "bounded traffic models." [Michiel, 1997]. Both stochastic models and bounded models are used in developing our simulation models.

One popular traffic model is the Poisson model [Schwartz, 1987] [Frost, 1994]. This model has been used since the telephone era, where it was effective at modeling the times at which telephone calls arrived at a switch. A Poisson process is a memoryless, independent and identical (i.i.d) process. The interarrival times are exponentially distributed and the number of arrivals in disjoint intervals are statistically independent. The exponential distribution is:

$$f(x) = \frac{1}{\lambda} e^{(-x\lambda)}; \quad \mu = \sigma = \lambda,$$

while the Poisson distribution is:

$$f(x) = \frac{\lambda^x}{x!} e^{-\lambda}; \quad \mu = \sigma^2 = \lambda ,$$

Where λ is the rate parameter; μ and □ represent the expectation and standard deviation respectively.

One important property of Poisson process is that the superposition of independent Poisson processes results in a new Poisson process whose rate is the sum of the component rates. Also, the memoryless property of Poisson processes can greatly simplify queuing problems involving Poisson arrivals. Poisson processes are fairly common in models for traffic that are made up of a large number of independent traffic sources. "The theoretical basis for this phenomenon is known as Palm's Theorem [Larson, 1979]. It roughly states that under suitable but mild conditions, such multiplexed streams, the statistics of the sum approaches a Poisson process as the number of streams grows." [Frost, 1994] Thus, traffic streams on main communications trunks are commonly believed to follow Poisson arrival statistics, as opposed to traffic on upstream links, which is less likely to be Poisson. The Poisson model is simple and can be used with care for simulating the background traffic of a network, although there are problems with certain kinds of traffic.

As opposed to Poisson, recent studies have discovered that packet traffic appears to be statistically self-similar [Beranetal, 1992]. A self-similar phenomenon exhibits structural similarities across all (or at least a wide range) time scales. In the case of packet traffic, self-similarity is manifested by the absence of a natural length of a burst: at every time scale ranging form a few milliseconds to minutes and hours, similar-

looking traffic bursts are evident. In this case, self-similar models are better at simulating the traffic generated by multimedia sources that Poisson models because they capture the traffic burstiness, which directly affects the QoS of the network. Poisson models assume that the traffic arrivals are independent on time, which is not the case for some types of traffic such as the VBR video. If traffic follows a Poisson arrival process, it would have a characteristic burst length which would tend to be smoothed by averaging over a long enough time scale. However, measurements of real VBR traffic indicate that significant traffic burstiness is present on a wide range of time scale [Beran, 1995]. An example of the self-similar stochastic model is the fractional Gaussian noise [Mandelbrot,1968].

In the next chapter, we will present a simulation system for Virtual-U, a networked learning system.

Chapter III
The Simulation System

3.1 Overview

The goals of our work are to build a simulation system that can represent a real telelearning environment and to carry out research on network-related performance issues. In order to achieve these goals, we followed a four-step procedure:

Step 1: We studied the architecture, operation and functionality of the experimental Virtual-U system to determine the appropriate building blocks for our simulation. At this point, we also identified the most important system parameters.

Step 2: We studied the individual components of the test system and built a simulation model for each of them. We also obtained values for each of the simulation parameters in this step. This process needs to be done for each distinct type of course that will be offered. For example, the models for conference based courses are based on the statistical data in the web-server log files from a "text-based" VU course (i.e. BUS362), while the models for Video on Demand (VOD) courses are based on the network traffic traces obtained when a student attended a streaming video lecture from a Stanford University on-line course.

Step 3: We implemented the simulation models on the computer simulation using the "industrial strength" OPNET package environment.

Step 4: We then studied the capacity and performance issues using our simulation system. Specifically, we predicted the system capacity (in terms of the number of users the system will support with a reasonable QoS) and determined how varying various resource parameters affected this capacity.

After analysis of the 1997 Virtual-U system design and the network environment (SFU CSSnet) it is implemented on, we built a simulation system that includes a network model, a background traffic model, a server model, and a user model; this system is shown in Figure 3.1.

While modeling all these components, we kept in mind three following principles:

1) The models are derived from the current Virtual-U design, however, it must be possible to extend them in order to predict the capabilities and characteristics of future versions of Virtual-U – especially when the use of rich-text and multimedia becomes prevalent.
2) The models should represent the generic and universal characteristics of networked multimedia systems, which will make our simulation system robust enough to be valid for telelearning systems other than Virtual-U.
3) All the models should be simple and easy to implement in computer simulation.

3.2 Simulation System Construction

The simulation system includes a network model, a background traffic model, a server model, and a user model. These will now be discussed individually.

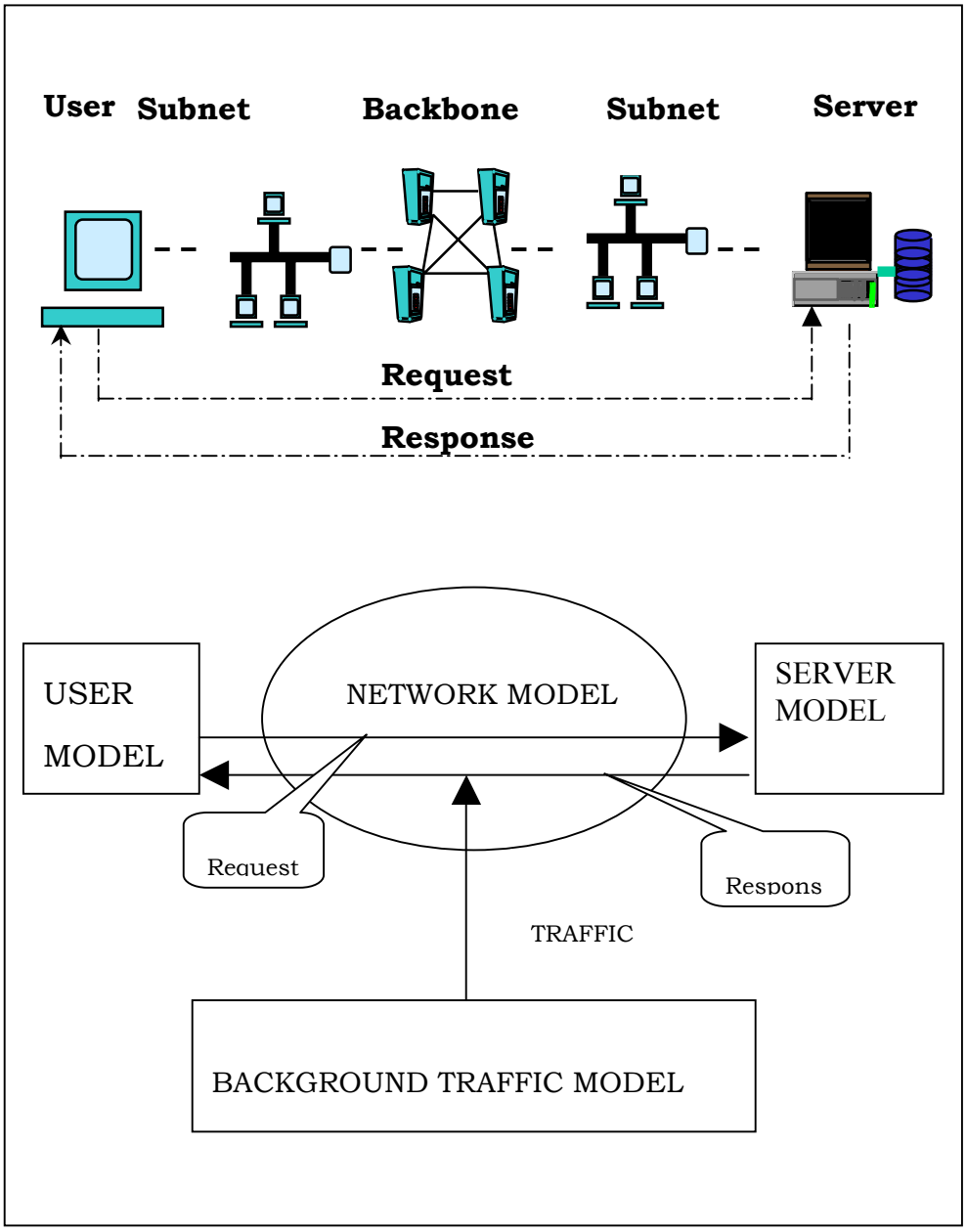

Figure 3.1 Simulation Model

3.2.1 The Network Model

In order to build a scaleable network model that can be straightforwardly adapted to different network environments, we used hierarchical building blocks (which can be easily implemented in OPNET), i.e. the network model consists of four levels: Network → subnet → station → process. The current network model is based on a simple 2-layer network architecture that consists of a high-speed backbone (100Mbps) and 10base_T subnets (10Mbps). The backbone network is "organized" by an eight-port Multi-ports Intelligent Bridge, as is used in the SFU CSSnet. Eight-port bridges are also common in most of today's LAN environments and are thus appropriate for "general" telelearning environments. In our simulation model, all of the eight ports are used, seven ports are connected to the subnets and one port is connected to what we call the "internet-backbone". Each subnet is a 10Base_T Ethernet subnet that can be populated with users (we mean user's computers), servers, bridges, hubs, routers et cetera, all the network subscribers (i.e. users, servers) are called stations. There is currently a limit of 32 stations per subnet in the existing setup, so we chose the 32-ports hub model (for the same reason as the bridge stated above). In this thesis, we have "user subnets" that only contain client workstations and "server subnets", which contain servers and clients in both types of subnet, we also provide choice to include workstations to generate background traffic, as required by the simulation.

As mentioned earlier, the backbone network is also connected by a router to the "Internet backbone network", which is in turn connected to another Ethernet subnet, which could contain other servers. This

architecture can potentially simulate a student logging onto a remote server if this is a common activity in a course.

3.2.2 The Server Model

The Virtual-U server provides the web based course material to the client workstations when this material is requested. In order to facilitate the course offering, it maintains a course database that has all the related course material and administrative information (e.g. the students' grade books); a user interface that receives user requests and returns required information; and of course a whole package of controlling programs that are responsible for processing the user requests and controlling the system resources. In the 1997 version of the system, the interface between user requests and the course database was implemented by Common Gateway Interface(CGI) programs. When a user request is received, the corresponding CGI program (or set of programs) is executed to process the request and gather the information required. The processing of the request may require database queries or updates, information comparisons or calculations. After all the information is ready , a web-page is generated and sent back to the user.

In the real server, the crucial resources include the CPU, memory, I/O bus, disks, et cetera. As our goal is to simulate the system delays of the user requests contributed by the server, we do not have to repeat all the components in building the server model, instead we need to build a model which can represent the server's structure and functionality from the point of view of processing the user request. Therefore, we modeled the server as an interface to the users, a queue, a processor and a database. (See Figure 3.2)

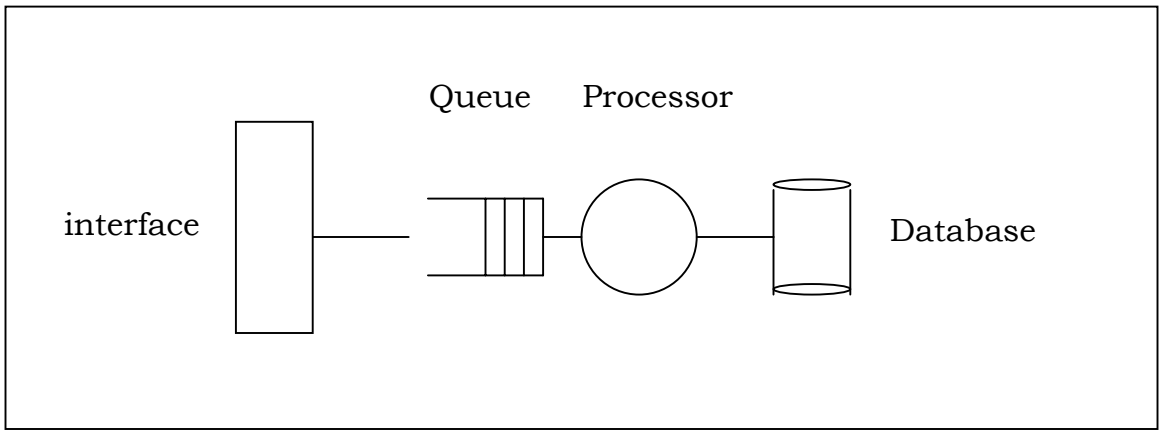

Figure 3.2 Server Model

In our server model, the “interface” is responsible for the communication with the user , i.e. receiving a request and sending backed required data. It also determines what type of processing will be needed in the server when receiving a request. The “queue” and the “processor” are modeled as a single processor with a single First Come First Serve (FCFS) queue. The processor‘s processing ability is represented by a single “processing power” parameter having units of “jobs/second”. We define a “ job” as the basic request from a user, such as “get a message” or “put a message” and the resulting communication with the course. Ideally, this processing parameter should be a function of the server load, the CPU power, the cache size, the I/O speed et cetera. Of course, we should also take into account other factors such as the type of file system and the code efficiency. However, for simplicity, we have assumed that the parameter can be effectively represented as a constant and that it is obtainable through experiments. The processing delay in the server is calculated the server’s processing power and the

amount of data required by the user request. Future work should address the improvements possible through more accurate server models.

Another important factor is that the user requests are not always identical and thus different types of requests will consume different amounts of the server's processing power. In the Virtual-U system, each user request is processed by the corresponding CGI program. When a CGI program is being executed, it may invoke other programs and spawn several sub-tasks, all of which consume the server's processing power and thus add to the total processing delay in the server. For example, a HTML request will spawn many processes to form the components of the page, all of which require processing. Another example is a request that requires the comparison of several pieces of database information. During the processing of this kind of request, the server will have to search the database to obtain the required information and then make the comparison. Although the requested result itself may be very concise, the resources allocated to such a task may be large. Therefore the number of bytes in a users request does not always directly represent the real processing load in a server. To account for the factor of processing power consumption for different type of user request, we used a parameter called "Number of processing power units consumed", which is denoted by "N". In our simulations, this "property" is implemented as one parameter in the user's request. In truth, this parameter depends both on the type of the request (what information and operation will be required in the sever) and on the current state of the system; however, for simplicity, we currently assume that "N" can be modeled as a constant in each main application category; i.e., streaming

video, or web page downloads (which may contain images). Future work should attempt to improve the model of the server and the user to make a more accurate representation of this factor.

In conclusion, the processing delay associated with a server request is a function of the server's processing power(P), the amount of data required by the request(L) and the type of request(N). Until a more complex and accurate model is available, the processing delay (T) in the server is calculated with the following formula, based on the assumptions mentioned earlier, where the parameter "A" is chosen to fit the real system according to the experiments of the real system :

$$T = A \times \frac{N \times L}{P}$$

Databases also play a very important role in the Virtual-U system (although the 1997 version used a simple "flat" file structure); however, the goal of our simulation at this stage can be achieved without a model of the internal database mechanisms. Our main interest is in the traffic rather than how the information is organized. Nonetheless, the simulation system has the capability of modeling the database in detail should it be required in the future.

In estimating the server's processing power, we based our analysis on the hardware specification. Since our applications are I/O intensive tasks, we assumed the bottleneck in the sever is in the I/O bus. It is 10Mbps for a Sun Ultra1 work station which is the hardware of the VU sever in 1997. If we use the average message length (800 Bytes) as the length of a standard job, we get the number of 1500 jobs/sec. This

estimate does not take into account the fact that software factors which may also play important roles in determining this parameter. The overall "effective" processing power also depends on software factors which we use "N" to represent. Although our most interests are in the overall "effective" processing power, in reality the improvement of software and hardware are usually though separate processes. We will experiment with the various numbers for "N" and use the "combination" to represent the "effective" processing power.

3.2.3 The Background Traffic

The background traffic is the basic traffic (as opposed to the telelearning traffic under study) on the local subnets and the backbones. The amount of background traffic will significantly affect the amount of the congestion and will thus contribute to the delay experienced by telelearning users. Background traffic will typically be generated by applications such as Telnet, FTP, HTTP, NNTP, and SMTP, which have historically accounted for a high percentage of the traffic on real network, however, newer multimedia applications such as video and audio are expected to take up a quickly growing share of the network bandwidth [Crovella, 1996], [Paxon, 1994]. As discussed in Chapter 2, the Poisson model is a simple and popular model in modeling network traffic. Recent studies have shown that some traffic, such as user initiated TCP session arrivals, can be well modeled as Poisson process but that other types of traffic deviate considerably from Poisson statistics [Paxon, 1995]; what model is best depends on what type of traffic is running on the network. Initially, we develop a model that is simple and "reasonable" accurate. The model chosen in the first step is the Poisson model. Although it is

not perfect, it does to some extent reflect the characteristics of the traffic running on the CSSNet[4].

3.2.4 The User Model

Our simulation uses a computer model to represent telelearning users generating requests to the server. This model must be simple enough to be implemented in our simulation, but it must also be flexible enough to capture the essence of the "typical" user's behavior, which may depend on such variables as the course content, the user's knowledge about the course material, the user's thinking habit and the design of the system. As many of these factors and interactions are not well understood, it is very hard to build a deterministic model , or to hope to be able capture the users behavior exactly. However, it is possible to build a user model with stochastic process and try to reflect the "large scale" interaction in such a way as to have predictive power for estimating things like the quality of service. To model the network activity when users attend specific courses, we only need to know the possible actions and the likelihood of them being undertaken. Because these actions are usually limited by the technological reasons as well as the system design (e.g. in a web based system the choices at any given time are limited by the icons currently available) and/or human thinking patterns (e.g. people usually read posted messages before posting replies), we are also able to obtain the probabilities for each choice using

---

[4] We are presently lacking the data for the traffic running on the network. Until February 1998, there is no formal report on the CSSNet traffic analysis. The understanding of the traffic on CSSnet is based on the discussion with the network administrators.

statistical analysis of the server log-files for different users. In this way, we treat the user “activities” as states and obtain a Markov chain model (as shown in the Figure 3.3) that can stochastically represent the average behavior of the users.

Thus, to achieve our goal of building a user model that represents the real telelearning users generating requests and the resulting traffic produced by the server, our strategy is to first quantify the typical user's behavior for a specific kind of course content by examining the server log files for specific courses and then to build up a Markov chain model that consists of several "states". Each model state represents the user’s “main activity” when taking a certain course and is associated with probability density functions for the file sizes (traffic) requested and the time between requests. We are not modeling the dependencies of the intrastate file sizes in the models being built to reflect the “large scale” interaction (as we mentioned earlier). If the user’s activities are too complex , we may also decrease the grain by analyzing the log files to identify patterns of "meta" activities. The model cycles between these states according to transition probabilities in the way that users “typically” work with the real system . Also, users in the various states will wait different amounts of time between requests, which we term the states “request inter-arrival time”.

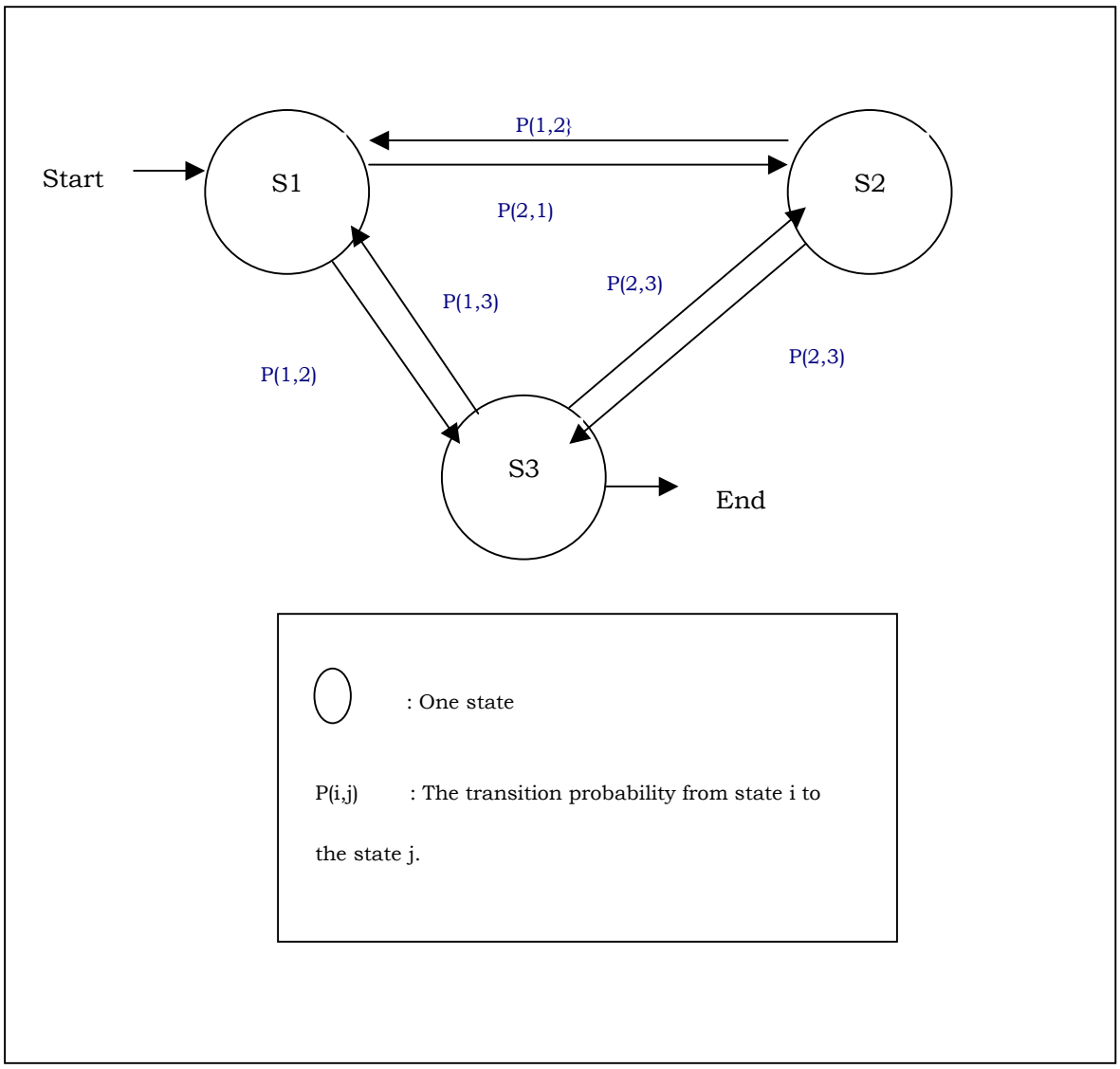

Figure 3.3 : An Example of Markov Chain Model

The above model will help us to study the effect of changing the course content on the system performance. If the change is not to the design of the course itself, the task may be as simple as changing the probability density functions associated with each state; however, if the design of the course is substantially changed, then the whole model must be reworked in a reasonable way.

In order to increase the system flexibility, we propose classifying courses into several different categories, which can then be used as models for new courses. For example, one category may be the Web based conference courses offered by Virtual-U (VU course), another category may have lectures based on streaming video content (Video On Demand course). These possibilities will now be discussed in more detail.

VU Conference-Based Courses

As we mentioned earlier, the VU conference-based course is organized like a news group. Students read and post messages according to specific topics raised in the course. Each topic leads to a "conference", a student can join any "conference" they have permission to. When students take such a course, they first login to the system and choose the course/conferences that they are interested in; they then read newly posted messages and post their own. Handouts and assignments are also available though the messaging system.

To get a user model for the students taking VU courses through Virtual-U, we need to analyze and obtain the usage pattern of students who are taking the test courses. Fortunately, the usage of the VU system are (at least partially) recorded in the server's log files. The log files of the

VU server contain the information about user request to the server. Each user request will generate a record. In such a record, there is information as the user's IP address, the user's login ID, the date and time of the request, the CGI program called (and the arguments), the size of data transferred and the result of the request (i.e. success or failure). Analyzing these log files and building a user model was done in collaboration with the researchers in Virtual-U Research Laboratory. Some of our work is based on their research results on the typical usage patterns, and our traffic modeling was achieved using the log files they provided. According to the analysis of log files for the test courses, several typical user's usage pattern were discovered [Zaiane , 1998]. For example, new users like to try various options, while experienced users are more focused. There is a strong usage pattern of "start V-Group", "list conferences", "list unread messages", "display a message" and " preview/add a message" for all users. This pattern is understandable because it is the result of both "natural" human behavior as well as the system design. For our study, it makes more sense to use the pattern of users already experienced with the VU system to build the simulation model. Based on this discovery from analyzing log data of the testing course BUS362, , we have built a five-state model for the VU conference based courses.

State 1: "Start V_Group" represents the state in which a user logs into to Virtual-U and views the VU welcome page.

State 2: "List Conferences" represents the situation where users list the conferences that they have joined.

State 3: "List Unread Messages" represents the situation where users list all the unread messages.

State 4: "Display a message" is the state for displaying and reading messages.

State 5: "Preview/Add a message" is the state in which a user previews and posts composed messages.

In order to get models for the traffic of each of the states and to form the state transition probability matrix, the raw server log files were then processed to remove erroneous records and unrelated domains and the records sorted by course. We then used the following analysis procedure on the data corresponding to our chosen date-range:

1. Sort the records by user. In the log files, the original records are ordered by time, here we sort them by user-ID.
2. Parse the records into sessions. A session here refers to the period of time during which a student is actively using the VU system. It starts from the time when a user logins into VU and ends when they log out. There is no "logout" record in the log file; however, if a user has no communication with the VU server for more than 30 minutes, then they are treated as "logged out". The 30 minute parameter comes from observation (VU Research Lab) that very few sessions last more than 30. The records are parsed according to the "user_ID" and time stamps.
3. Label the records according to the states. The state number (as we defined] is labeled for each record. The state in the model represents a student's usage of a specific function provided by the VU system. A state can be "recognized" by the CGI program being called by the user request, which is available in one of the domains in the records.

4. Calculate the transition probability matrix. The transition probability is calculated state-by-state using the state label of each record. For each state, the calculations are done for each student separately and then the results are averaged for all students. Please note that the transition between states across a session (defined in step 2) is regarded as an invalid transition and is not counted.
5. Calculate and obtain the histogram of the traffic attached to the request in each state. The quantity of "data transferred" in the log file is used as the traffic size of the request.
6. Calculate and obtain the histogram of the "request inter-arrival time" for each state. The "request inter-arrival time" is not included in the log file as a domain, but it can be obtained by determining the time between two consecutive (within one session) requests.
7. Find the statistical models for the traffic size and the request inter-arrival time of each state. With the completion of this step, we have a model that can be used to represent the user's request and the traffic incurred; this model is then used in the OPNET simulations.

Using the procedure described above, we analyzed a log file containing 24 hours of data for the 48-student course BUS 362. Since the log file was obtained in the second half of the semester, most of the students will be experienced with the VU System.

The parameters of the user model such as the "request inter-arrival time" should only reflect the user's behavior, but the information obtained from the log file also contains "noise" which depends on the conditions of the testing system. The system dependency should be

filtered out from the user model. If the system is not congested, the "noise" is only in the forms of system "processing delay" or "transmission delay" which are relatively small compared to users' thinking and reading. However, if the system is congested, this effect will not be small any more. Unfortunately, the congestion status of the system is not recorded by the log file, it is difficult to filter the system dependency with current log data. At this stage, we assume the error caused by the system dependency is not severe and the user model obtained from the log data can reasonably approximate the user's behavior. Efforts on building the model with no system dependency should be made in the future.

The accuracy of the statistical model depends on the number of samples being processed and the complexity of the model to be implemented into the simulation. However, at this stage of the project, our goal is to obtain "reasonable" approximation of the behaviors of a "typical" user. More accurate models that better catch the interaction between the system and the users need to be developed, but this is a matter for future work. With simplicity in mind, we used two approaches to obtain the "first cut" approximation. The first approach was to fit the histogram of real data with a well-known distribution by "eyeballing", while the second approach, used when a simple statistical distribution could not be found, was to represent the data using a "pessimistic" constant that was worse than 90% of the samples. Since we are doing research about QoS/capacity issues, under-estimation of the traffic is more harmful than over-estimation. This approximation leads us to a conservative prediction of the system capacity and QoS. The "pessimistic level" (e.g. the 90% above) will determine the traffic of the model. The

more “pessimistic” of the model, the more conservative an capacity estimation will be make. It is always a trade off between the number of users in the system and the QoS can be guaranteed all the users. This judgement is beyond the scope of our study. We will do some experiments to see how changing the “pessimistic level” affects the results.

When the work in a state is done, a state transition will happen according to the transition probability matrix. The transition probability matrix that we obtained from the BUS 362 course is given in the following table.

Table 3.1:Transition Probability For the VU Conference Based Course

	state 1	*state 2*	*state 3*	*state 4*	*state 5*
state 1	2.5%	96.5%	1%	0	0
state 2	21.5%	5.6%	63.3%	1.7%	7.9%
state 3	15.3%	0.4%	24.5%	54.2%	5.6%
state 4	17.3%	0.4%	24.2%	54.4%	5.6%
state 5	15.3%	0	27.1%	18.6%	39.0%

Details of traffic observed in each of the 5 states are as follows:

State 1: "Start V_Group" The traffic in this state is modeled as a constant 6094 bytes, because the login process and the information to the user (i.e. the information in the welcome page) are determined by the design of system interface and will not vary from user to user.

State 2: "List Conferences" The traffic in this state shows some humps in the histogram curve (Figure 3.4), since users may join different numbers of conferences. We could not fit this histogram it with any well-known simple distribution. Until a more accurate model can be found, we used the “pessimistic” constant approach as mentioned earlier and simply model this traffic by a constant 10kB (90% of the traffic in this state is less than 10kB according to the analysis of the log file).

State 3: "List Unread Messages" It is very difficult to model how many unread messages are there since the user last login. Like State 2, the histogram of traffic in this state could not be fit with any simple distribution. So until a more accurate model can be found, the traffic

associated with this state is modeled as a constant 35kB (90% of the traffic in this state is less than 35kB according to the analysis of the log file).

State 4,5 : "Display a message" & "Preview/Add a message" These states represent a user reading or writing a message. If we look at a course for the whole time it is running, the messages a user can read are the messages other users have written and posted. The distribution of message size in these two states should be approximately the same (we assume the error caused by the fact that "a message can only be viewed after it is posted" is minimum), although most messages will be read many times. Figure 3.5 shows the histogram of the message sizes[5]. The curve's bell-shape makes us think about making an approximation with a truncated Normal distribution. By eyeballing, we found that a Normal distribution with μ=300Bytes and σ=450Bytes fits this histogram fairly well (see Figure 3.6). However, this model under-estimates the right tail of the histogram. As we stated earlier, at this stage, we would like to choose a simple model with a conservative estimate. So until a more accurate model can be found, the message size is modeled as a constant 2.5kB (90% of the message size is less than 2.5kB according to the analysis of the log file). Please note that it does not mean we can not have a more accurate model in the future based on the Normal distribution. On the contrary, we think that it

[5] The data for this figured is not the one-day log file used for other states, we were able to use the file from the VU research Laboratory which contains the information of the messages in this course (BUS362) during a period of 3 months.

may be possible to model it by an Normal distribution with a large variance if more data in the future can prove that the "tail problem" is not severe. Otherwise, some "adjusted" Normal distribution (e.g. a combination of a Normal distribution and another type of distribution to handle the tail) will be the approaches for an accurate model. The interesting thing is that when we looked at another VU course, the histogram of message size also showed a bell shape. We suspect that the law of large numbers [Hogg, 1997] is at work here.

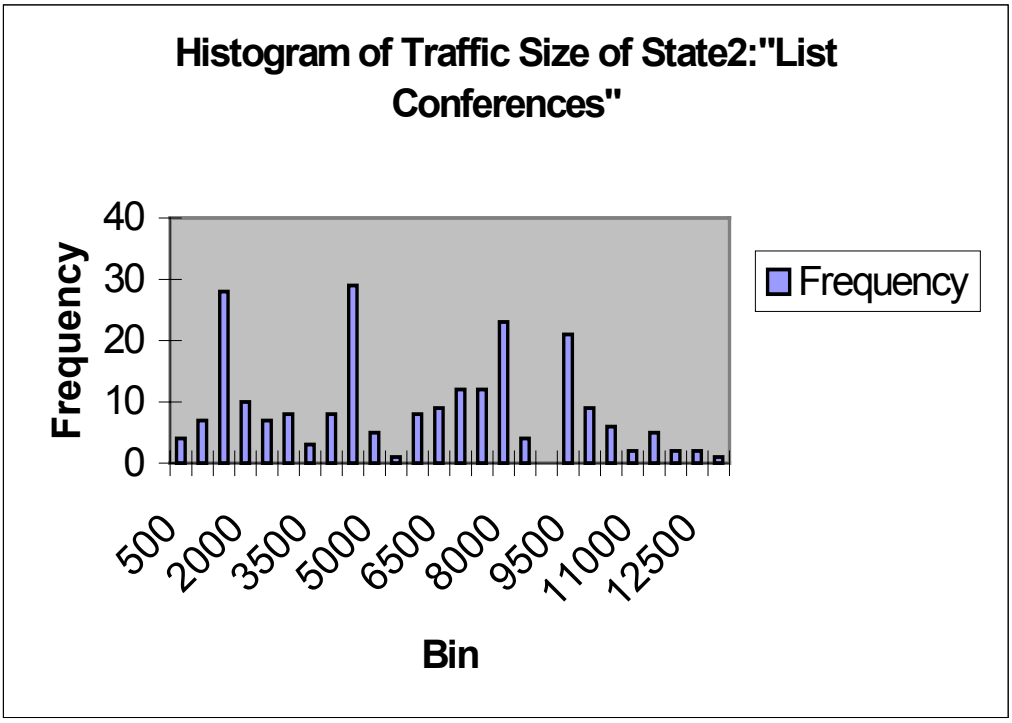

Figure 3.4 Histogram of the Traffic in State 2

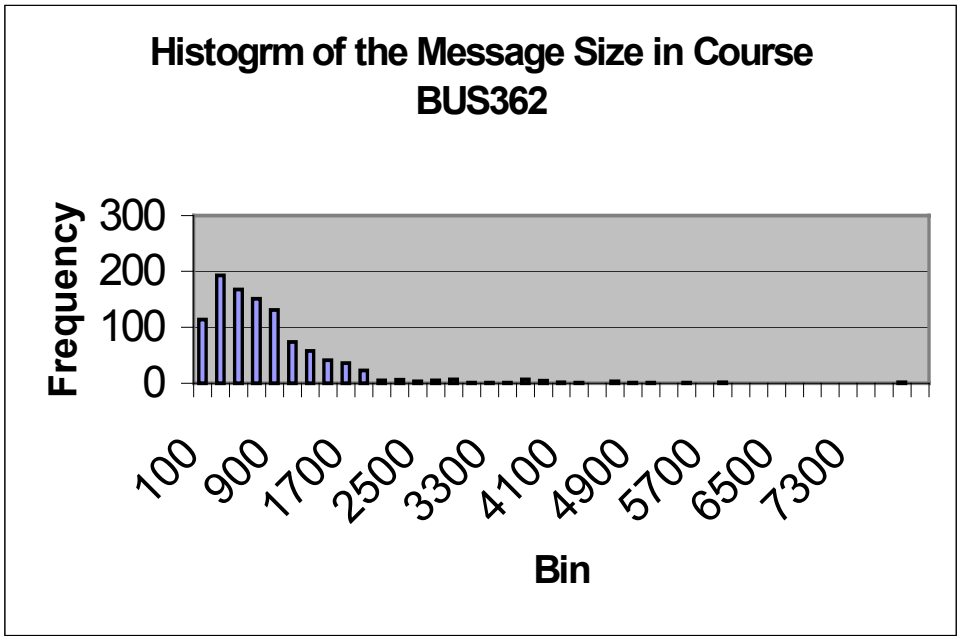

Figure 3.5 Histogram of the Traffic in State 4 & 5.

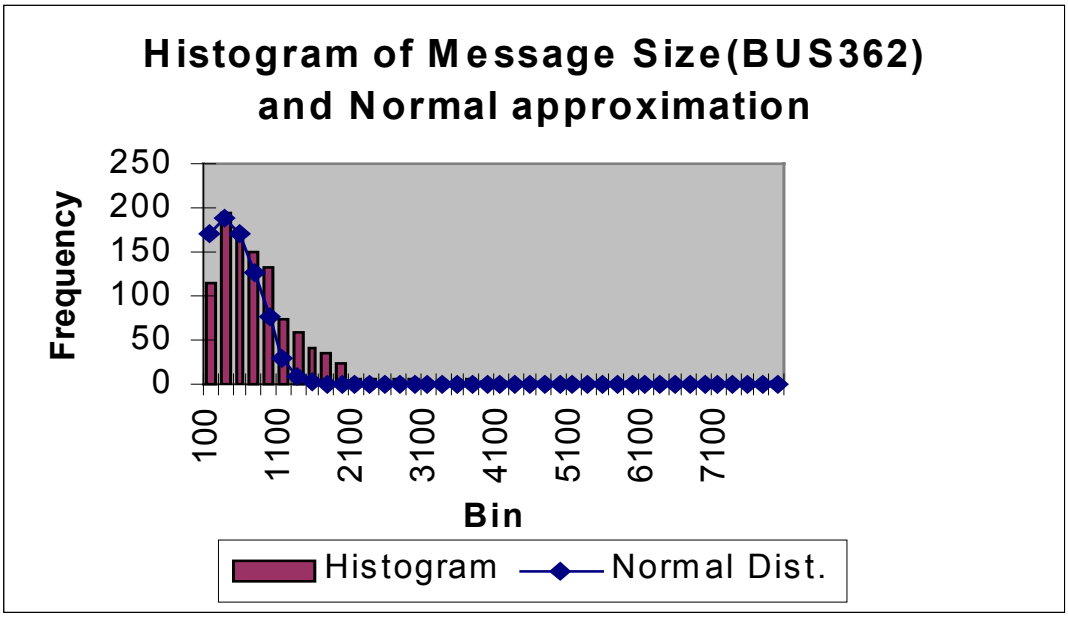

Figure 3.6 Normal Approximation of the Traffic in State 4 & 5.

Regarding the request inter-arrival times in states 1-3, the users always take a quick look and go to another state. So, if we use a "pessimistic" single number, which is higher than 90% of the data samples, to model the request inter-arrival time in these states, the numbers that we get are 5 seconds, 5 seconds, and 10 seconds respectively for the above 3 states.

The request inter-arrival time for a user reading the messages of a course, state 4, is mainly determined by the course content and the student's reading/thinking habits. A long message by itself will take a long time to read; a message full of hard questions will also take a long time to read. All these types of messages will make for a long request interarrival time. A student who wish to read and think through the messages one by one will have a different interarrival time model than the student who likes to read many related messages first and then make a thorough thinking about them. For these reasons, an accurate and representative model for the interarrival times is not easy to obtain and we make the simplifying assumption of defining this behavior for a "typical" user. In the future, improvements may be possible by dividing the users into several groups according to the expected distribution of "thinking habits".

The histogram of the interarrival time for course BUS 362 is shown in Figure 3.7. Through "eyeballing", we approximated the histogram by an exponential distribution with $\mu=\sigma=168s$.

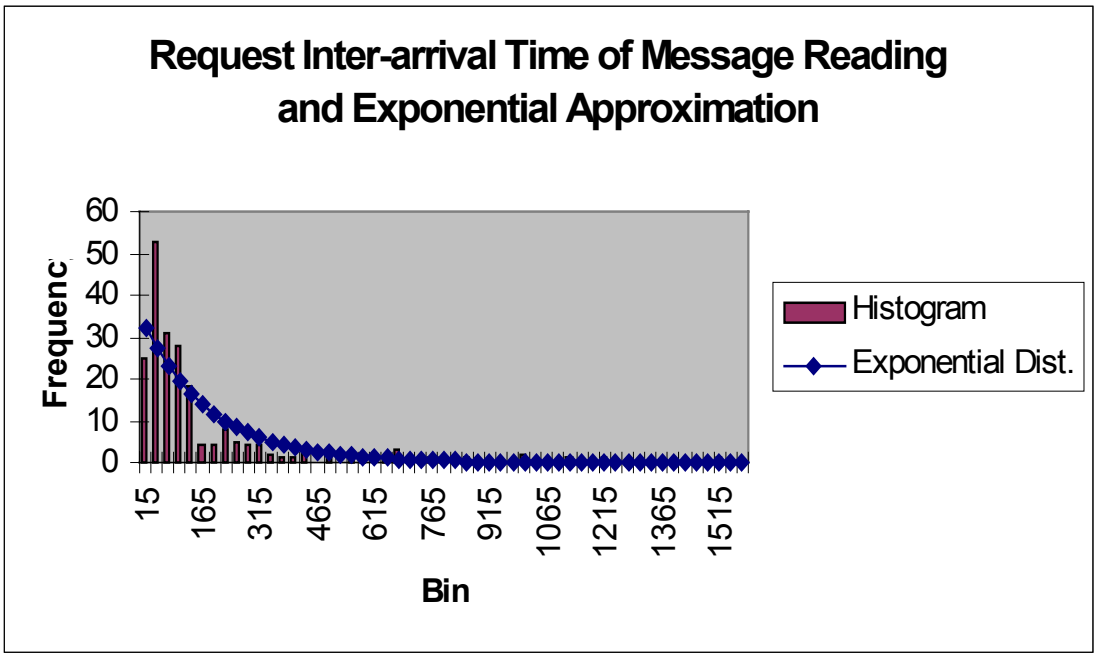

Figure 3.7 Request Interarrival Time of State 4.

Conference based courses with image loading

Today's technologies are able to provide ever increasing amounts of multimedia content, such as images, rich-text and audio/video over the Internet. We believe that multimedia adds educational value in many situations and that future telelearning systems must support these types of applications. In order to examine the effect of having additional multimedia content in a course, we built a new traffic model, which is based on the basic model above, and new "invented" state, State 6: "Loading an image", to represent what happens when users request multimedia information such as images or large rich-text files, such as WORD documents; other types of information such as video will be considered later in this section. For simplicity, everything downloaded in this state is referred to as an "image".

The key attributes of the State 6 are the frequency of requests for images size of the image being loaded. The projected usage manner of loading the image is to download an image referred by the course material, so we will model the loading frequency by the percentage of messages that refers to an image, i.e. the transition probability from state 5 to state 6, P(5,6). The size of the image (i.e. image file) depends on many technical factors such as the data format (e.g. JPEG, MPEG, GIF et cetera.), image displaying size, resolution and the content. An accurate model of the image file size will thus be quite course dependent and a thus course data is needed to produce a truly accurate model. Unfortunately, the current version of Virtual-U (for which test data is available) runs essentially text-based courses and there is no data available for multimedia rich cases. However, in order to come up with a

generic situation, we decided that it would be reasonable to assume that the file sizes random with a truncated Normal distribution. Our projected behavior pattern is as follows: after a student reads a message, he or she loads an image referred by that message, then after loading and viewing the image, the student may read another message or load another image if there is a series of them. If we assume there is a 25% chance a user will load an image after reading a message, and after viewing the image, a 60% chance the student will view another message while a 40% chance the student will load another image. After analyzing 14 "typical" images [GIF and JPEG) on a computer (we picked the scanned pictures rather than the computer icons), we developed a few different test models. For a course that demands many high-quality images as a core component of its content, such as an art history course, we set μ=128KB and σ=40KB; for the next rung of courses with moderate image demands, such as geography and biology ,we set , μ=64KB and σ=20KB. For both of these types of courses, we set the nominal image loading frequency to be 25% of all requests; i.e., P(5,6) = 25%. For different categories of courses the frequency of loading images and the mean size of loaded image may vary. We vary these parameters in the simulations and examine how adding this type of multimedia content affects the system QoS.

The new set of transition probabilities after inserting this state is shown in Table 3.2.

TABLE 3.2: Transition Probabilities for VU Course[6]

	state 1	*state 2*	*state 3*	*state 4*	*state 5*	*state 6*
state 1	2.5%	96.5%	0.9%	0	0	0
state 2	21.5%	5.6%	63.3%	1.7%	7.9%	0
state 3	15.3%	0.4%	24.5%	54.2%	5.6%	0
state 4	17.3%	0.4%	24.2%	39.4%	5.6%	25%
state 5	15.3%	0	27.1%	18.6%	39%	0
state 6	0	0	0	60%	0	40%

Video On Demand Course

In the future, it is expected that most telelearning courses will include increasing amounts of multimedia content. In addition to the downloading of images and rich-text files, another possibility is the inclusion of streaming-video (Video On Demand, or VOD) lectures on the server that can be viewed over the Internet using applications such as Microsoft's NetShow [NetShow Web]. A VOD session can be a type of course by itself, or it can be integrated into a VU conference course as part of the course material (as an additional state). In our simulation, we make it separate from the VU courses. However, from the perspective of analyzing the system QoS, the two usage modes of VOD sessions do no have significant difference, because the system performance is affected

[6] As we stated above, we would vary the transition probability for state 6 in the simulation, the values in this table about state 6 are just an example.

by the traffic and, as far as a VOD session goes it will impose the same amount of traffic on the system no matter what the usage mode.

The NetShow product was formerly called “Vxtreme” (Microsoft acquired Vxteme in 1997). We used NetShow for VOD course analysis because NetShow has already had examples of usage for education purposes and provided sample VOD courses for evaluation. In the NetShow application, students typically log into the server and select specific lectures from a list; viewing begins by clicking on the “play” button on the screen. Students can pause the lecture when desired, or rewind/fast forward to quickly move around in the lecture. Using streaming video lectures, students can “attend” a lecture while also having the freedom to appreciate the lecture content at their own pace. We believe that this kind of application provides a very good learning environment and that is very attractive for many telelearning courses. For this reason, we built a streaming video “VOD” course model into our simulation. Alternative video delivery mechanisms such as “multicast” are also possible; however, they don’t fit well into the “any time, any place” paradigm of telelearning.

It is possible to use TCP or UTP to setup VOD sessions (NetShow provides both options). As we discussed in Section 2.1, TCP guarantees delivery while UTP does not. For VOD sessions, there is not really real tine interaction, TCP can make sure the viewer misses nothing due to temporary (as opposed to a severe problem cause the network goes down for a log time) network congestion.

Based on the interface of NetShow and our knowledge of the way a VOD course is supposed to be, we created a three state Markov chain model.

State 1: " Play " In this state, a student views the lecture at the normal speed. The VOD server delivers data to the viewer's computer to play the lecture in real time.

State 2: " Fast Forward/Rewind " In this state, students fast-rewind or fast-forward the lecture to the part that they want to view. Students are able to view what happens at a lower resolution for navigational purposes. The server delivers data to the viewer's computer at a lower traffic rate due to the lower resolution and dropping of the audio signal. From the traffic perspective, there is no significant difference between the fast forwarding and rewinding. The same amount of information is transferred (low resolution video and no audio), it does not matter if the mode is backwards or forwards. The results of experiments test traffic traces agreed this reasoning. The simple experiments we did are as follows: for a given VOD lecture, we chose a certain part (starting from A and ending at B). We fast-forwarded from point A to point B and then rewind from point B to pint A. Then compared the traffic histograms of the two traces. The results showed very little differences.

State 3: " Pause " This state models, a student stopping the lecture and taking the time to think about the course material, or to look something up. The server stops delivering data to the viewer's computer.

In order to examine the traffic involved in such a VOD lecture, we captured and analyzed the traffic from the viewer's side (the traffic information from the source end is not available) when a tester was viewing a VOD sample lecture. The sample lecture was one lecture of the VOD course CSS244a offered by Stanford University. This course was

prepared with Microsoft NetShow (it was called Vxtreme at that time) and accessible though the Vxtreme's home page (It is not available anymore after the acquisition). The traffic trace was collected by a LANalyzer, which is a tool provided by Novell to monitor the traffic of a station in a Ethernet. In the trace files there are such information as the source address, destination address, packet length, time stamp for all the packets received. With this information, we are able to calculate the traffic (rate) for a VOD session from the viewer's side. The obtained data rate may have variability due to network reasons (as the packets traverse the network to reach the viewer's site). We assume this is not big enough to have a significant impact. More accurate models (ideally from information on the source end) should be built for higher quality formats. At present, we assume that the VOD sessions are separate from the VU course. Models should be obtained for the situation when a VOD session is part of a VU conference based course and stored together with other VU course materials; however, we leave this for future work.

We captured separate traces (about 15 minutes each) "playing" and rewinding/fast-forwarding the lecture; then calculated the traffic (in bits/s) and approximated them with statistical models. By eyeballing, we found that the traffic histograms of the two traces can be "reasonably" fitted by Normal distributions. We now discuss the traffic in each of the three states:

State 1: "Play" - The measured traffic is shown in Figure 3.8. By eyeballing, we found the histogram curve has a bell shape and that the characteristic is close to a Normal distribution with μ=56 kbits/s and σ= 5 kbits/s. When we looked at several other VOD traces

prepared by NetShow, we found that their traffic distribution also had bell shapes.

State 2: "Fast Forward/Backward" - Similarly to state 2, we approximate the traffic shown in Figure 3.9 with a Normal distribution; however, this time the parameters are μ=18 kbits/s and σ= 5kbits/s.

State 3: "Pause" - This state is an idle state and involves no traffic.

By analyzing the data, we found that our sample user "plays" the video for around 10 minutes , "rewinds/fast-forwards" for 1 minutes to the target; and the "pause" time is 5-10 minutes. So we set the duration time[7] of the State1, 2, and 3 as 10 minutes, 1 minute and 10 minutes respectively. When a state reaches its duration time, a state transition happens according to the state transition probability matrix. The transition probabilities were estimated by analyzing the same tester taking the same course (CSS244a). The results are presented in Table 3.3 below.

[7] Please do not get confused by the missing of "state duration time" in the VU model. The definition of the states in the VU model is based on the user's action, i.e. one state has one action. So in fact, the "state duration time" is the same as the "request interarrival time".

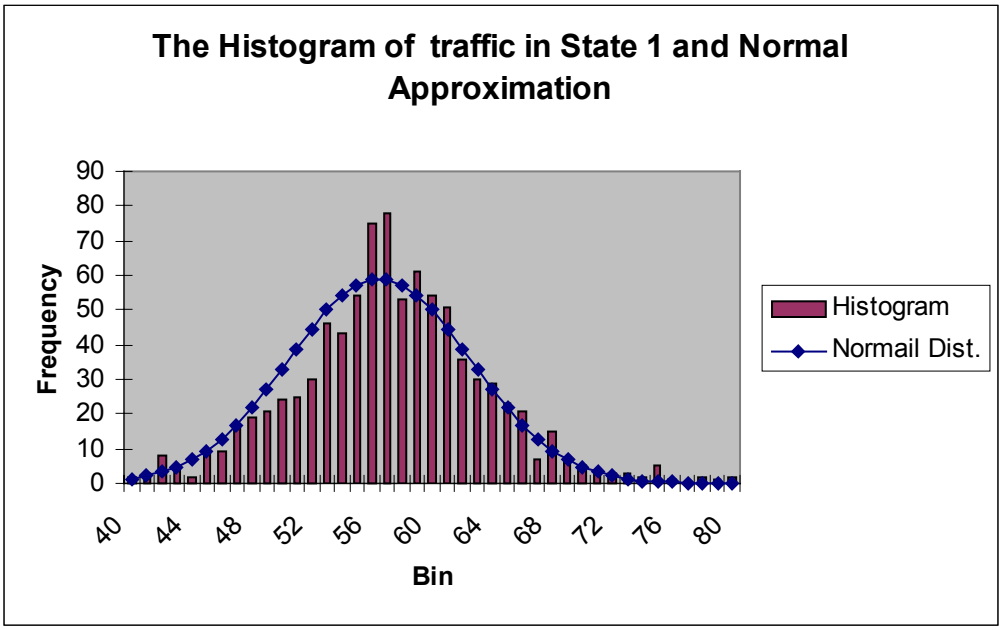

Figure 3.8 The Traffic of State 1 in VOD

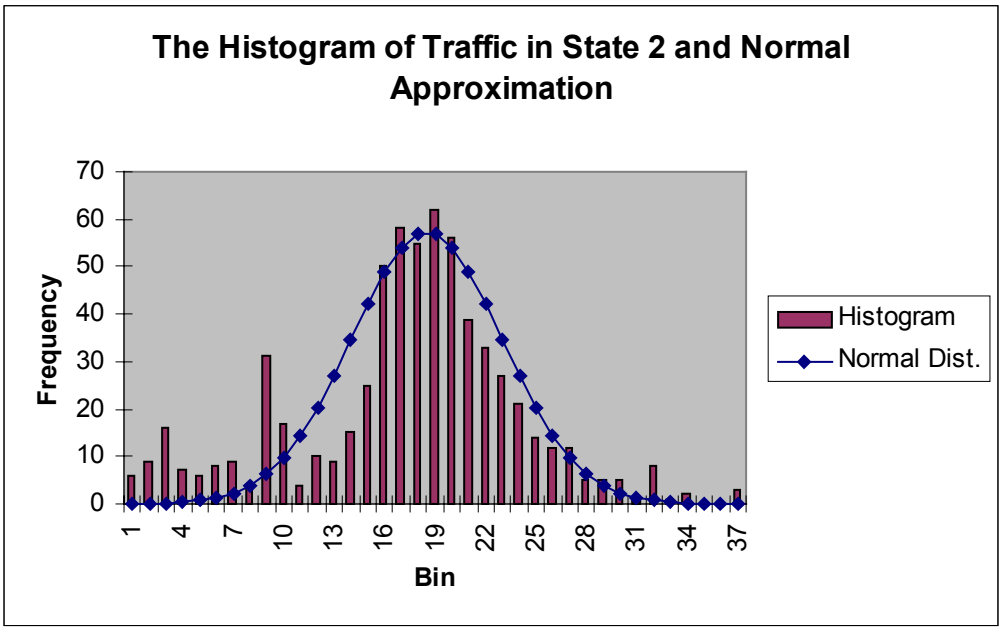
The Histogram of Traffic in State 2 and Normal Approximation
Frequency
70
60
50
40
30
20
10
0
1
4
7
10
13
16
19
22
25
28
31
34
37
Bin
Histogram
Normal Dist.

Figure 3.9 The Traffic of State 2 in VOD

Table 3.3: Transition Probability For VOD [8]

	state 1	*state 2*	*state 3*
state 1	40%	20%	40%
state 2	30%	20%	50%
state 3	10%	5%	85%

The Simulations

The models discussed above were implemented in OPNET. In the OPNET package, there are many standard network components and functions built that we and use; while for the applications, OPNET provides some examples to show how the lower layer models can be used. In our network model, all that we are using are standard network equipment (e.g. bridges, hubs) and network protocols. We can choose the existing models and configure the parameters according to our requirements. But for the application (as opposed to the network) implementations, e.g. our user models, we have to construct our own models. In the OPNET example models, there is one called "GNA" that simulates basic network applications such as Email , FTP et cetera. We chose to build our models based the "GNA" model because it uses the same client-server mechanism as what we are using and some basic functions can also be used in our programs. Meanwhile we do not have

[8] The data here is based on the analysis of one tested user; it may not be valid for all other users and for all the VOD courses.

to redesign the application-network interfaces. All these can speed up the development of our system.

In the “GNA” model, a control program “app-mgr” calls the program “gna” to generate different application processes and control them. The program “gna” has 2 states, one is “burst”, which simulates traffic generation , and the other is “idle”, where no traffic is generated. Inside the “burst” state, traffic is generated according to the “packet size” and “packet interarrival time” parameters, which are defined according to the type of application. The states are switched according to the predefined time of each state according to the type of application.

In our programs, the “app-mgr” is modified to distinguish applications and call the appropriate programs, i.e. “vugna” for VU conference based courses, “cygna” for VOD streaming video, and “odgna” for the applications defined in the original “GNA” models. The program “vugna” is developed to implement the 6-state model of a student taking VU courses. The program “cygna” is developed to implement the 3-state model of a student viewing the VOD lectures. The program “odgna” is a modified version of the “gna” which can work well with other programs we developed.

In each of the states, traffic is generated according to the models we defined: the main parameters we use are the “packet size” and “packet interarrival time”. A set of state-leaving conditions are defined and the decision making mechanism for state transition are also developed. In our programs, the states switch according to the defined “states transition probabilities”. To achieve this, we add a random number generation mechanism and the transition decision is made based on this random number. Some new functions are also developed to

facilitate the model construction, such as a new traffic sending function that allows each state to determine the traffic size it will send and also the responding traffic size. In the streaming video "VOD" program ("cygna"), a user play buffer is implemented so that we can use play buffer's overflow and underflow as a measure of video QoS.

As our server model is not complicated at this stage of the simulation development, we simply adopt the provided "GNA" model. Along with the model development , a set of "Probes" are defined to collect data during simulations, such as the VOD buffer's over and underflow, the VU courses' response time and the throuput of each network segment et cetera.

The models discussed in the above sections were used in the following way to create our simulations. Ideally, each student would be simulated by running the user model appropriate to the course being tested on a "user station", as defined when we discussed the network model. Unfortunately, however, we cannot always afford to have separate user station for each synthetic student. Each user station has a lot of components which require computing resources such as the memory and simulation running time. We do not have unlimited memory and simulation time to run the simulations with arbitrary numbers of components. To resolve this situation, we can simulate multiple (n) users with one station by making the requests generated at n times the frequency as a single user (i.e. shorten the average interarrival time by a factor of n). As we discussed earlier in the section 2.3, the superposition of independent Poisson processes results in a new Poisson process whose rate is the sum of the competent rates. Since we use a stochastic process to generate user requests in our user model and each request is

generated independently, the “lumping” approach is able to simulate multiple independent users by having a single station interact with the Virtual-U system. The error caused by “lumping” depends on the “lumping rate” (i.e. the number of users lumped in one station) and the number of individual stations. As long as there are sufficient number of individual stations defined, the error caused by “lumping” should be minimum. Mostly, the number of simulated users in our experiments is not more than 300 and there are 150 stations defined, so there are sufficient number of stations in our simulations. A spot-checking test was executed to compare the results of 150 “lumped” VU users (by lumping to 50 individual user-stations) with 150 individual VU users. All the tests configurations (details refer to Section 4.2) and measurement methods are identical for the two cases. From the results shown in Table 3.4, we can find that the difference is not significant. The “lumping error” in our simulations should not be worse than that in this spot-checking test, because our lumping rate is lower (2 compared to 3)and there are more individual stations (150 compared to 50) there.

Table 3.4: ‘Lumping’ Effect

# of VU users	Request Response Time (s)				QoS Parameters	
	Max.	Min.	Mean	Standard Deviation	"Not good"	"Bad"
150	21.9	0.0158	0.492	5.09	14.7%	4.2%
150 (lumped)	21.1	0.0158	0.483	5.02	13.6%	4.5%

It should also be noted that the VOD traffic models come from the traces captured from the network, which naturally include network overhead. When information such as the data of a VOD frame is being transmitted across the network, it is put into special formats form according to the network protocols. The information is split and encapsulated into the "protocol data unit", which is defined according to a standard format and is recognizable by all the network entities. Besides this payload information, a protocol data unit also has a header containing such data as the address and control information, which will be used for delivering the payload information. At the receiver end, all the overheads are striped off and the original information is reformed to what was in the sender and then forwarded to the end user. So before we can use the "network captured" traffic model to generate traffic at the application layer, the network overhead should be taken off. However, since we can not accurately estimate how much it accounts for, it is safer to leave it on and be conservative.

3.3 QoS Measurements

In the previous sections, we described the construction of the simulation system. We are now able to execute the simulation and study the system capacity and performance. To make sense of these results, the QoS measures must be properly defined.

As mentioned earlier, when we evaluate the system performance, both statistical and subjective QoS measurements should be used. As we discussed in Chapter 1, we set 3 levels of subjective quality levels and they are defined as:

- "*Good*": Very good quality.

- "*Not good*": The quality is not good, but may be acceptable in some cases.
- "*Bad*": The quality is bad and is unacceptable.

For the VU conference based courses, the request response time, T, which is defined as the time from when a user sends out the request until the requested information is received, reflects the quality directly experienced by the users. Unlike other measures such as the number of requests processed by the server, the packet error rate in the network connecting the server and the user, the throuput of the user's station in the LAN segment, this measure takes in to account the impacts from almost all of the components in the system. We are thus interested various statistical parameters based upon T, such as its "maximum value", "minimum value", "mean" and "standard deviation". The three subjective QoS levels for T itself are thus intuitively set as:

"Good"	T< 5 sec.
"Not Good"	5 sec. ≤ T < 20 sec.
"Bad"	T ≥ 20 sec.

For the VOD course, the actual delay is not critical because there is not much interaction which is very sensitive to the delay. On the other hand, "delay jitter" is important, since too much jitter will cause the play buffer at the client station to overflow and underflow, which will greatly degrade the video quality through frame loss. We use the rate of play-buffer's underflow/overflow, R, as the QoS measurement for VOD course. Note that R is also dependent upon the size of the play-buffer, a bigger

buffer can smooth larger delay variation and make a smaller R; on the other hand a bigger buffer need longer time to fill during the initialization (pre-fill the buffer at the beginning) which means longer initialization delay. It is a trade off between the delay jitter soothing capability and the initialization delay while the size of play buffer is determined. Since our video source transmitted at an average rate of 56kb/s, we assumed a play-buffer size of 128kbits, corresponding to 2-seconds of video. The three quality levels were then defined based on R as is shown below, with the actual values being heuristically chosen.

"Good"	R< 1.0 times/min.
"Not Good"	1.0 times/min. ≤ R < 1.5 times/min.
"Bad"	R>1.5 times/min

3.4 Summary

In this chapter we described the method used to analyze a real telelearning system and to construct a simulation model of it. We also presented our simulation system for the Virtual-U system (based on its 1997 design) as an example. The simulation models developed were a network model, a background traffic model, a server model and a user model. These models were derived from the analysis of the real system structure and usage records (e.g. log files) obtained from real telelearning courses. In each of the models, we also defined several parameters that can be changed according to the resources available to the system, the course content and user's usage pattern. All these models were implemented using OPNET. The simulation system is able to simulate

students taking conference based courses and/or “Video On Demand” streaming video courses. We also defined a set of QoS measurements that represent the system performance both objectively and subjectively. With this simulation system, we are able to carry out research on system capacity and performance issues for telelearning applications. In next chapter, we will give some examples about how we used the simulation system to predict the system capacity and analyze the effects of the key factors. Some of the interesting results we have obtained will also be discussed.

Chapter IV
Experiments and Results

4.1 General Description

In this chapter, we describe how to use the simulation system to analyze the performance and capacity of telelearning applications using Virtual-U and "streaming video" examples. We also present a series of results in order to demonstrate the power of our approach. The following questions are examined:

1) What is the capacity of Virtual-U in offering conference based courses and how many streaming video sessions can be added in the same system?
2) How will the factors such as the server's processing power, the average course file-size, the frequency of image downloads and the background traffic level in the network affect the system performance and capacity?
3) When multiple types (categories) of courses are offered on the same system, how do they affect each other?

The system configurations are based on the models we described in Chapter 3, which include a network model, a user model, a server model and a background traffic model.

The network model that we will use in the simulation is the "SFU-like" two-layered network model as described in Chapter 3 (a 100Mbps high-speed backbone with seven 10Mbps 10Base_T subnets connected to it). A server is located in one of the seven subnets. 150 individual

stations are evenly distributed in the other six subnets; however, we are able to simulate more than 150 users in the system using the technique described in Section 3.2.4.

The user model used in the simulation is the 6-state model as described in Section 3.2.4. For Virtual-U conference based courses (VU), without streaming video, we vary the following parameters in the simulations:

1) The number of server processing units consumed by each user request, N. As mentioned in Chapter 3, we approximate the "N" by a single constant for each type of courses, i.e. the value of "N" is the same for all states of the model for a course.
2) The frequency of loading images, F. This is the probability that a "plain message" event will be followed by a transition to state 6.
3) The mean and standard deviation, (μ, σ), of the Normal distribution that models the sizes of the "images" down-loaded to the users.

As we described in Section 3.2.4, some parameters of the user model are modeled by a "pessimistic" constant which is worse than 90% of the samples. Increasing the "pessimistic level" will lead to user model with heavier traffic , thus a more conservation prediction of the system capacity. Some experiments will be done with a "99% pessimistic level", i.e. we replace the "pessimistic" constants in the former described VU user model with new values which are worse than 99% of the data samples. The values got changed are:

Traffic in State 2 "List conferences" = 17 KB;

Traffic in State 3 "List unread messages" = 61 KB;

Traffic in State 4 "Read a message" = 13 KB;

Traffic in State 5 "Put a messages" = 13 KB;

Since we think "90% pessimistic level" is reasonable to be used in building a user model, for most of our experiments, we use the model described in Section 3.2.4.(It is used as default if there is no specific note).

The server model and background traffic models used in the simulation are as described in Chapter 3. The server processing power, P, is set to a nominal 1500 jobs/sec.; however, recall that this parameter can be adjusted using the "N" parameter described above. The background traffic level, B, is defined as the average percentage of the total bandwidth used by the non-telelearning applications. It is set to zero when we study factors other than background traffic, but we vary this parameter to analyze how background traffic affect system performance. According to our analysis of VU courses, the mean message size is 2.5kB and the interarrival time is 168s on average; this makes the average traffic generated by a user in state 4 equal to 120bps. On the other hand, if an image is loaded, the traffic rate may tens of kilobytes per second for a short period of time. (The real rate depends on many factors as the state of the network, the throuput of the server and the user's access speed et cetera.) Therefore the traffic is bursty and the sustained level is low. For this type of traffic pattern, it is difficult to predict the performance with a "paper and pencil" approach and the effect of multiplexing can only be evaluated by simulation. On the other hand, the traffic pattern of streaming video applications is less bursty and analytically tractable. The average rate of the VOD session we studied is 56kbps[9]. Please note that the VOD rate is not limited to this

[9] We assume the user will only "play" the video to make a conservative estimate

one. The rate of VOD sessions depend on the required video quality and the available bandwidth, it could be 128bps, 256bps et cetera, or as high as 1Mbps.

The simulation results will be discussed in the following sections.

4.2 System Capacity for VU Courses

A good estimate of Virtual-U system's capacity, which is defined as the number of users can be supported at the same time, is important for both the system designers and the educators who use VU to provide courses. In order to get this estimate, our approach is to simulate a given system configuration and different number of users in the system. By examining the system's performance with a given QoS criterion, we are able to find the maximum number of users that can be. In the following, we vary N, the available server processing power.

***System Configuration**:*

Server's processing power consumed by a user's request (N)	1, 10, 15, 20
Average (loaded) image size and standard deviation (μ,σ)	128kB, 40kB
Loading image frequency (F)	25%
Server processing power (P)	1500 jobs/s
Background traffic level (B)	0

The user model used

As the parameters of "image loading", we chose numbers appropriate to "heavy" image use because we are most interested in

exercising the system to find where the performance breaks down and we also wish to get a conservative estimate of the system capacity. These numbers will, of course, need to be tuned for specific courses once this data is available.

Simulation Results:

In the following tables of the simulation results, the "QoS Parameters" (e.g. "Bad") are as defined in Section 3.3. The system is regarded from the users' perspective as "slightly congested" if the end to end response time shows a slow but observable increasing trend and the part in "bad" region is below 20%. The system is regarded from the users' perspective as "congested" if the end to end response time shows a quick increasing trend and the part in "bad" region is over 20%.

The simulation results for "N=1, 10, 15, 20" are listed in Tables 4.1-4.5, and the QoS measurements for "N=10" are shown in Figures 4.1-4.4.

Table 4.1: VU Course Performance for Different Number of Users (N=1)

# of users	Request Response Time (s)				QoS Parameters		Note
	Max.	Min.	Mean	Standard Deviation	"Not good"	"Bad"	
150	21.9	0.0158	0.492	5.09	14.7%	4.2%	
300	22.1	0.0158	0.863	4.99	15.6%	3.9%	
450	28.5	0.0158	2.037	5.76	24.0%	5.2%	
600	62.7	0.0158	4.559	8.53	29.7%	11.6%	
675	126.3	0.0182	10.548	19.45	34.2%	16.1%	SC
750	153.4	0.0255	16.476	23.36	46.8%	25.8%	C

SC: Slightly congested.

C: Congested.

Table 4.2: VU Course Performance for Different Number of Users (N=10)

# of users	Request Response Time (s)				QoS Parameters		Note
	Max.	Min.	Mean	Standard Deviation	"Not good"	"Bad"	
10	10.074	0.0217	0.988	1.379	3.9%	0.0%	
20	10.744	0.0217	1.011	1.842	5.9%	0.0%	
30	23.085	0.0217	1.313	3.183	9.9%	1.2%	
40	33.047	0.0217	1.587	3.778	12.9%	2.6%	
50	54.162	0.0217	2.980	19.312	25.5%	7.9%	
55	123.470	0.0217	6.086	15.444	35.2%	16.2%	SC
60	179.388	0.0217	25.673	50.762	59.8%	46.3%	C

SC: Slightly congested.

C: Congested.

Table 4.3: VU Performance the “99% Pessimistic User Model” (N=10)

# of users	Request Response Time (s)				QoS Parameters		Note
	Max.	Min.	Mean	Standard Deviation	"Not good"	"Bad"	
10	15.822	0.0283	2.122	2.936	16.9%	0.0%	
20	32.730	0.0286	3.457	5.746	31.4%	3.5%	
30	51.596	0.0282	5.665	8.225	41.2%	9.8%	
40	101.664	0.0281	10.304	16.743	53.7%	19.5%	C
50	167.768	0.0281	25.203	28.720	77.1%	49.4%	SC

SC: Slightly congested.
C: Congested.

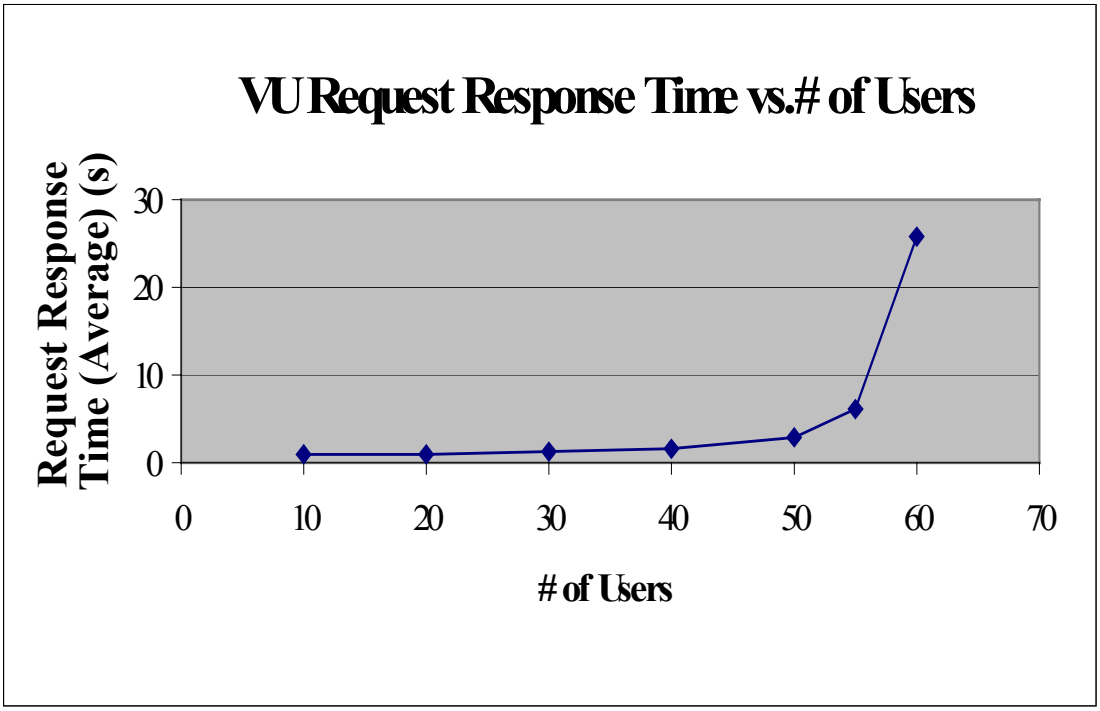

Figure 4.1 VU Response Time vs. # of Users (N=10)

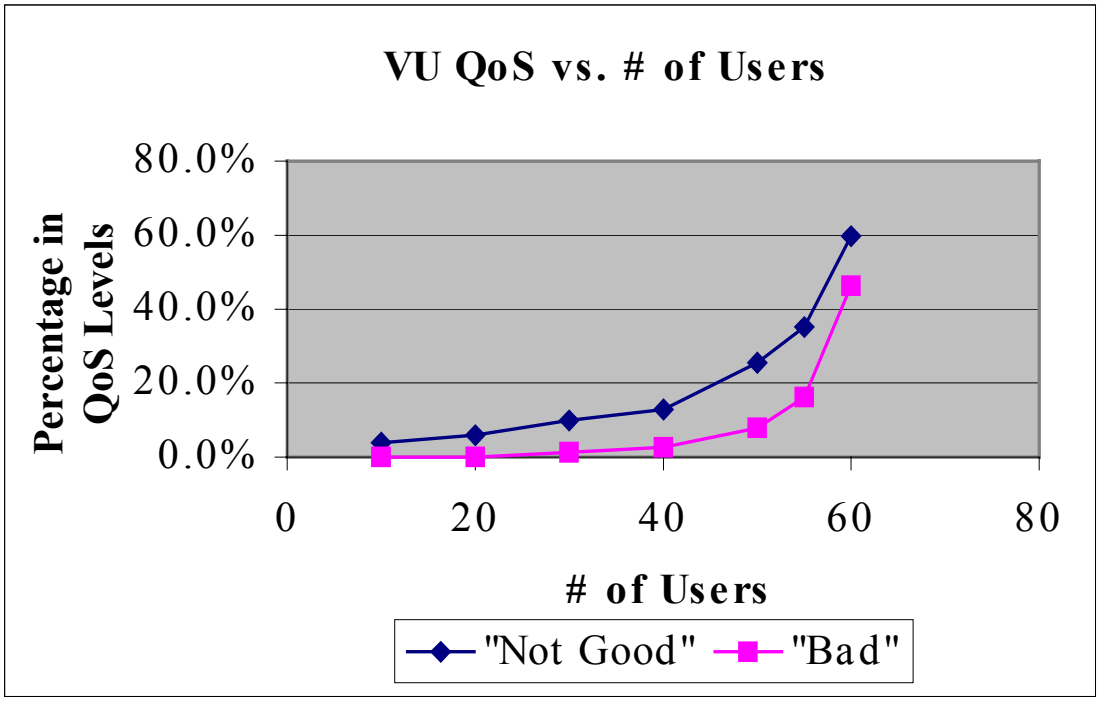

Figure 4.2 VU Subjective QoS vs. # of Users (N=10)

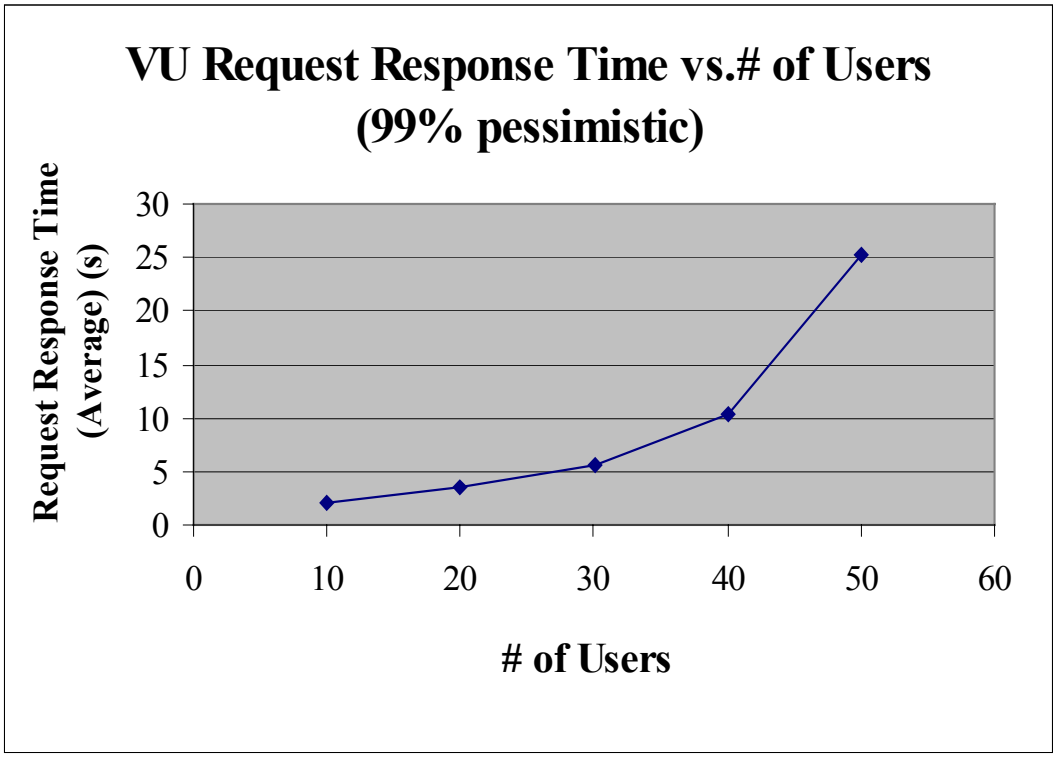

Figure 4.3 VU Response Time(99% pessimistic, N=10)

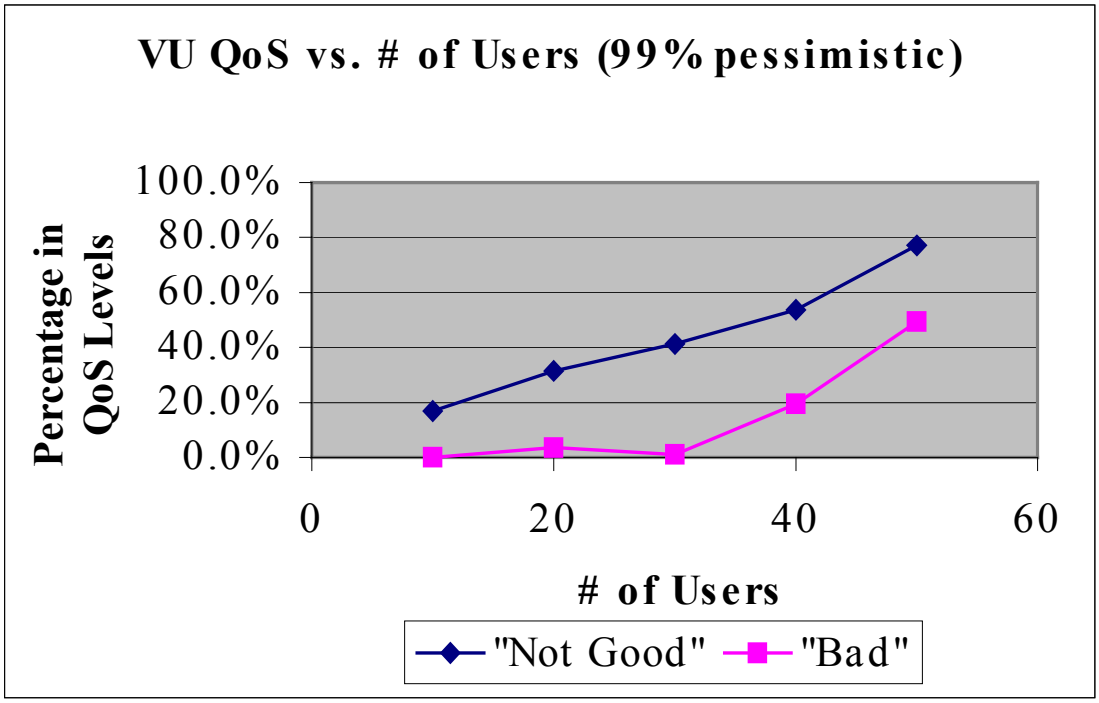

Figure 4.4 VU Subjective QoS vs. # of Users (99% pessimistic, N=10)

Table 4.4: VU Course Performance for Different Number of Users (N=15)

	Request Response Time (s)	QoS Parameters	

# of users	Max.	Min.	Mean	Standard Deviation	"Not good"	"Bad"	Note
10	11.719	0.0231	1.318	1.887	7.4%	0.0%	
20	18.676	0.0174	1.809	3.346	12.8%	0.0%	
30	33.935	0.0227	2.095	4.874	17.4%	2.1%	
35	86.686	0.0227	4.970	12.929	36.4%	14.4%	
40	102.037	0.0227	7.241	16.463	42.6%	17.4%	SC
50	270.632	0.0227	33.712	48.351	67.4%	46.9%	C

SC: Slightly congested.

C: Congested.

Table 4.5: VU Course Performance for Different Number of Users (N=20)

# of users	Request Response Time (s)				QoS Parameters		Note
	Max.	Min.	Mean	Standard Deviation	"Not good"	"Bad"	
10	33.473	0.0238	2.143	4.075	10.5%	1.7%	
15	52.475	0.0241	2.759	6.208	26.8%	3.0%	
20	60.214	0.0238	4.175	10.782	34.1%	13.4%	
25	98.665	0.0238	5.964	14.825	39.7%	15.3%	
30	193.302	0.0238	16.672	35.895	47.6%	28.9%	C

C: Congested.

Observations:

As the number of users increases, the overall quality of service degrades both objectively and subjectively. For the above system configuration with N=1, if the number of users is more than 675, the system starts to get congested (due to starvation of the server's processing power); when there are over 750 users, the system starts to get heavily congested. If we use the "less than 10% of the service is bad" quality as the criterion for determining the system capacity, then we should keep the number of users below 600. But these kind of performance is not what we have seen from the real VU system which has a much less capacity. The reason is that we overestimated the server's ability in processing users' request. The "N=1" is not the case in the real VU. The VU is implemented with CGI scripts, as we mentioned earlier in Chapter 1, when a CGI task is executed, it may spawn many "sub-tasks" which will all consume the server's processing resource. So in the real VU, the "N" is greater than 1 and the overall effective server's processing power is less than 1500 jobs/s.

With the same system configuration, different values of "N" make the server's "effective power" vary, which also changes the capacity; however, the shapes of the "Performance vs. the number of students" curves are similar. As the number of students increases within the system's capacity, the quality of service degrades gradually; however, when system is near its capacity, the quality of service degrades dramatically in both objective and subjective measurements. The system capacities are estimated and listed in Table 4.6 for different values of "N".

We also estimated the number of VU courses could be offered at the same time. Based on the observation that there are around 50 students in the test courses, we assume each VU course will have 50 students registered, among these, we assume that approximately 50% of the users are active during peak usage times. We are most interested in the quality under peak conditions, since it is the worst case that is most important.

Table 4.6: Capacity of the System Offering VU Courses

N	# of users	# of courses
1	600	24
10	50	2
15	30	1[10]
20	20	1[11]

The observed performance of real Virtual-U system is very close to "N= 10" case in simulation. That is, the "P=1500 jobs/s and N = 10" can represent the situation in the real system. However this performance is not very satisfactory. The capacity needs improvement. Our projection for a commercialized Virtual-U that is able to bring us an "on-line university" is a system capable of offering 20~50 (or even more) courses

[10] This is a rounded number.

[11] This is a rounded number.

simultaneously with fairly good quality. This goal should be achievable in the near future, depending on the type of courses that the system must support. The improvement of VU is expected come from using more powerful hardware (e.g. processor, memory, I/O equipment et cetera.) and from improvements in the software implementation (such as replacing the CGI interface with a more efficient system and by making better use of databases).

For the same system configurations, the QoS curves of the user model with "99% pessimistic" are similar to that of user model with "90% pessimistic". However the number of VU users can be supported is only 30 compared to 50 in "90% pessimistic" case, if we use the save criterion "less than 10% of the service is bad". A more pessimistic model leads to a more conservative prediction of the system capacity.

4.3 How Server's Processing Power affects QoS of VU courses

When analyzing how the server's processing power affects the QoS of VU service, we did experiments to derive two characteristics: 1) The capacity vs. the effective server processing power for a given QoS criterion. 2) The QoS vs. effective server processing power for a given number of users in the system.

4.3.1 Capacity vs. the Effective Processing Power

First, we define the "effective processing power" to be the number of user requests that can be processed by the server in one second. This number can be obtained by dividing the server's (absolute) processing power (jobs/s) by the factor "N". Based on the results of Section 1, we estimated the system capacity for different values of the "effective

server's processing power" (Refer to Table 4.7). Note that we are using the same QoS criterion ("less than 10% of the service is in bad quality") as in Section 1.

Table 4.7: System Capacity vs. the Effective Server Processing Power

Effective server processing power	# of users
75	20
100	30
150	50
1500	600

Observations:

As the server's effective processing power increases, the system capacity increases. The plot in Figure 4.5 shows a linear relationship between the system capacity and the server's effective processing. This result is due to our approximation of "N" as being a constant. We do not know exactly what will happen if we generalize N to be function of the system and request parameters, but we do know that the system capacity is determined by the server's effective processing power, which accounts for the key factors of both hardware and software.

The linearity in the plot tells us that the performance bottleneck of the simulated system, and indeed Virtual-U, is in the server rather than the network because all of the variation in the server's processing power is reflected in the system capacity. If the network has a significant effect on the system, the curve will be "flat" after some certain point when the performance is throttled by the limited network bandwidth. From a network's perspective, it is no problem to support up to 600 students. The server's limitation is reached before that of the network.

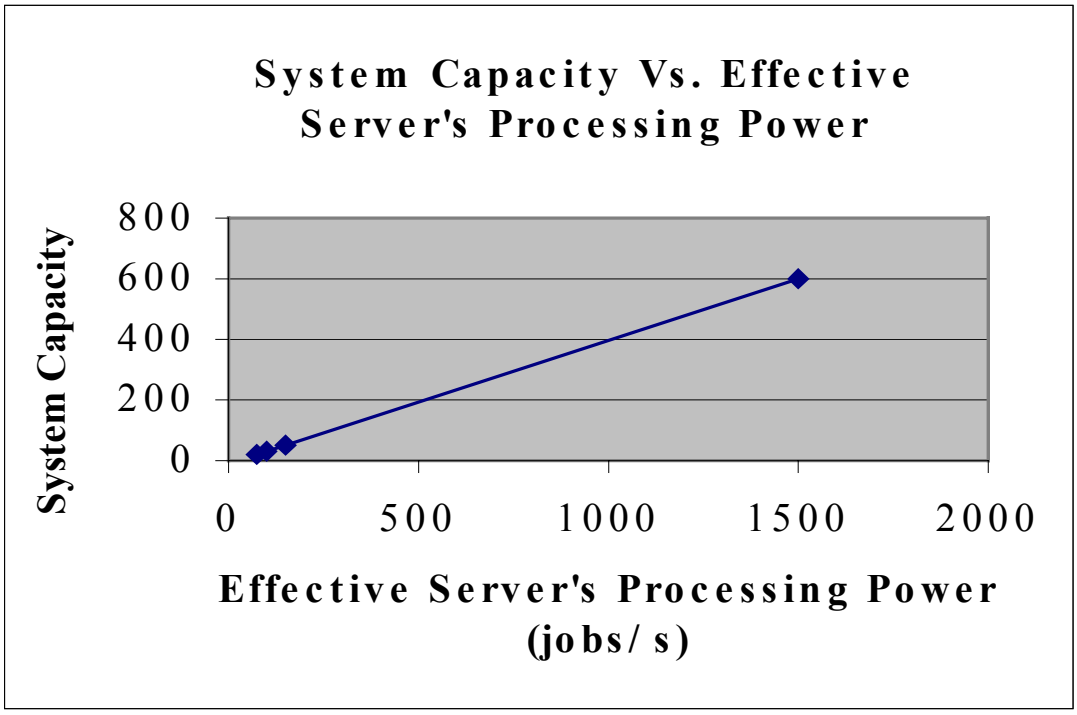

Figure 4.5 System Capacity vs. the Effective Server's Processing Power

4.3.2 QoS vs. the Effective Server's Processing Power

In this experiment, we study how a server's processing ability affects the system performance by varying the server's processing power. We will try to determine both the minimum required amount of server processing power for a desired performance, and also the effective range of the server processing power.

***System Configuration**:*

Server's processing power consumed by a user's request (N)	1
Number of students taking VU courses	300
Average (loaded) image size and standard deviation (μ,σ)	64kB, 20kB
Loading image frequency (F)	25%
Server processing power (P)	375~30000 jobs/s
Background traffic level (B)	0

We chose "N =1" to make the server have an "effective power" the same as the "server processing power" (P), because that is what we projected for the "final" Virtual-U that can be fully used (as we discussed in Section 1). We assume that there are 300 users in the system, because we want to investigate estimated a system that is generally half loaded. (The system capacity is 600 when we are at the nominal P of 1500 job/s).

The “image loading” parameters were chosen to match what we consider to be a “typical” course, as described in Chapter 3.

Simulation Results:

The results of the simulations based on the above parameters are shown in Table 4.8-4.9 and Figs. 4.6 - 4.9.

Table 4.8: VU Request Response Time vs. Server’s Processing Power

Server’s processing power (Jobs/s)	Request response Time (s)				QoS Parameters		Note
	Max.	Min.	Mean	Standard Deviation	"Not good"	"Bad"	
375	544.1	0.1071	122.300	108.50	96.1%	85.4%	C
750	242.3	0.0345	23.812	33.16	74.3%	36.0%	C
1000	123.1	0.0199	5.861	15.37	47.2%	15.1%	
1200	47.8	0.0158	1.874	7.28	31.1%	7.6%	
1500	22.1	0.0158	0.863	4.99	15.6%	3.9%	
7500	11.2	0.0092	0.198	1.18	3.5%	0.0%	
30000	9.4	0.0079	0.154	0.78	0.2%	0.0%	

C: Congested

Table 4.9: VU QoS vs. Server's Processing Power (99% pessimistic)

Server's processing power (Jobs/s)	Request response Time (s)				QoS Parameters		Note
	Max.	Min.	Mean	Standard Deviation	"Not good"	"Bad"	
750	851.4	0.0967	192.700	162.50	95.1%	86.2%	C
1200	250.5	0.0197	26.956	44.89	55.3%	29.3%	C
1500	103.3	0.0235	7.334	18.17	29.1%	16.5%	
1800	60.1	0.0177	3.103	10.33	18.5%	9.3%	
2000	32.1	0.0232	1.369	4.89	11.8%	2.6%	
3000	5.54	0.0235	0.224	0.617	4.5%	0.0%	
7500	7.71	0.0169	0.211	0.063	2.1%	0.0%	

C: Congested

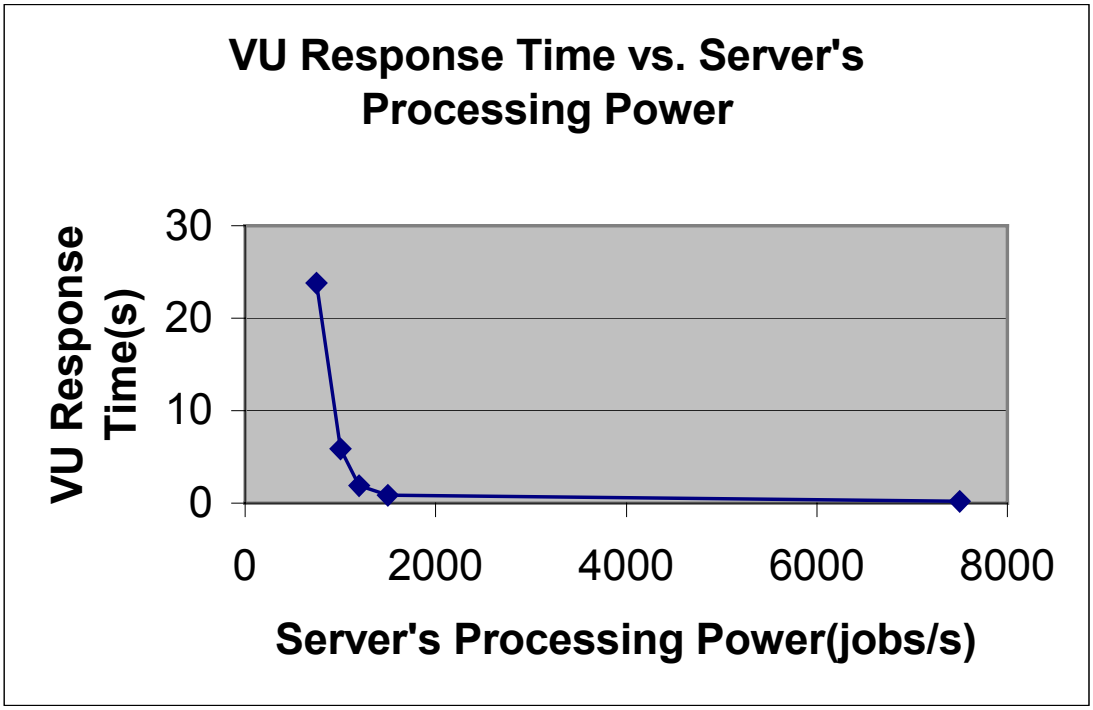

Figure 4.6 VU Response Time vs. Server's Processing Power

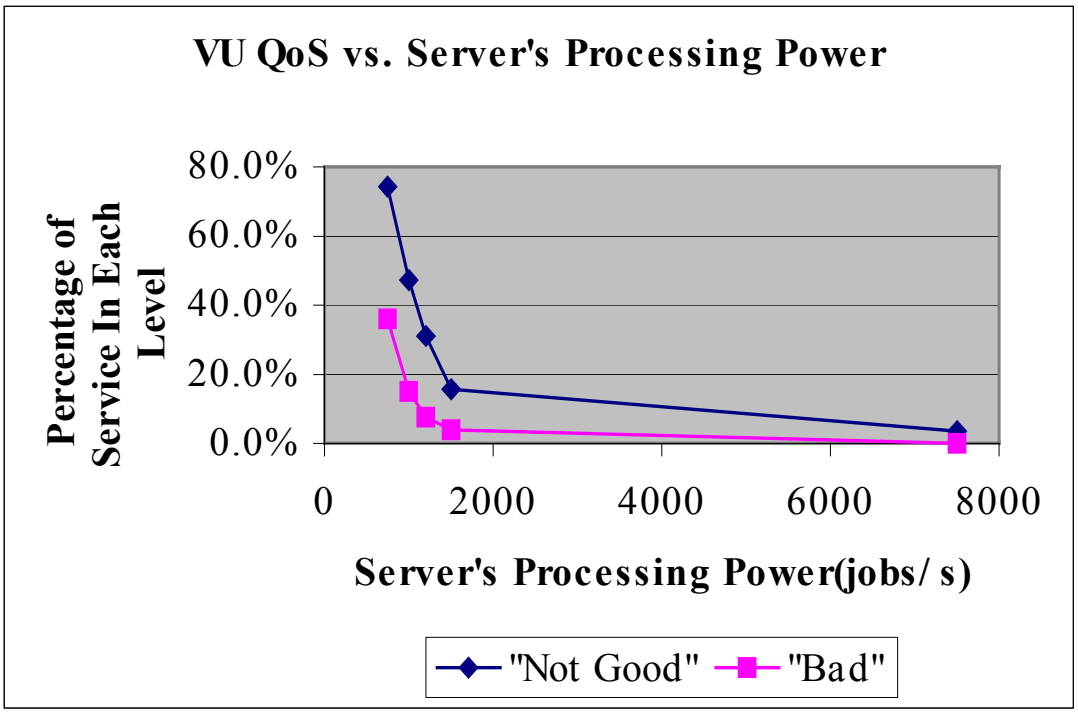

Figure 4.7 VU QoS vs. Server's Processing Power

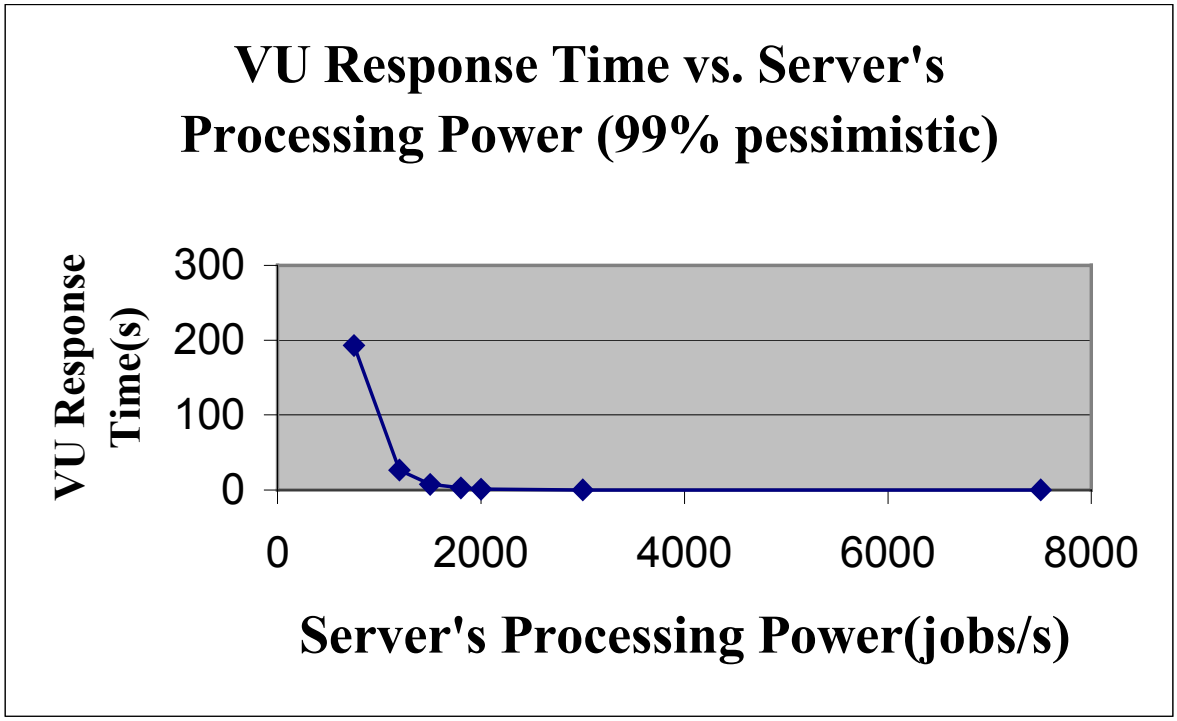

Figure 4.8 VU Response Time vs. Server's Processing Power (99% pessimistic)

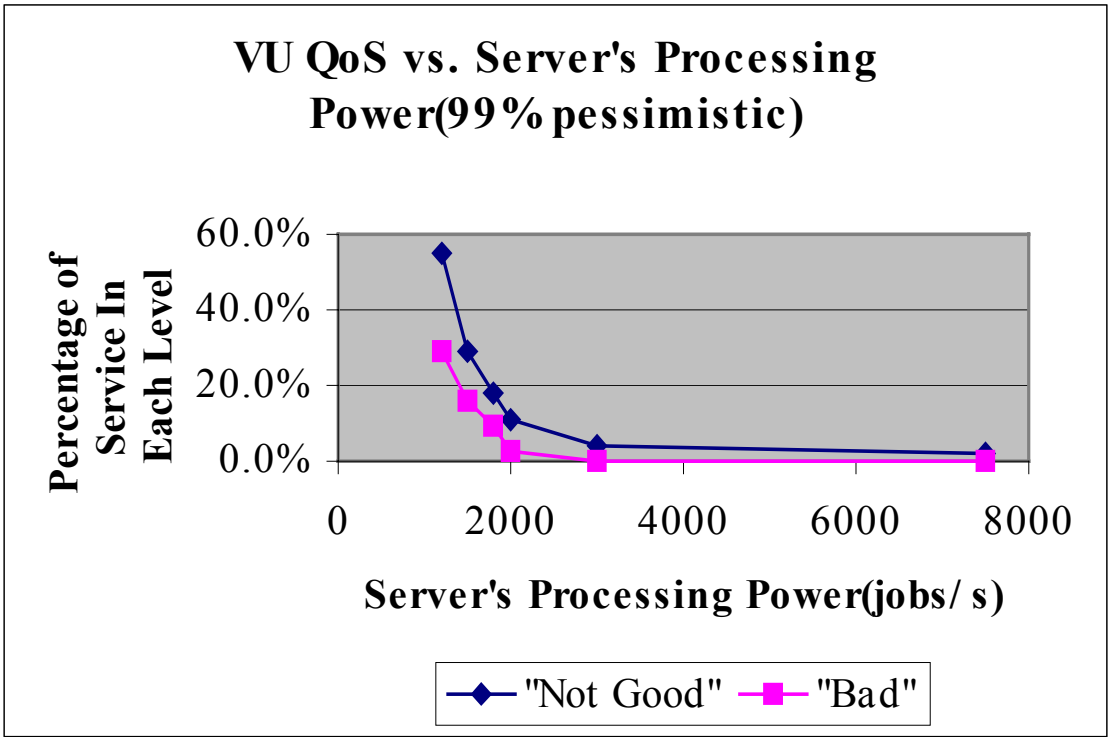

Figure 4.9 VU QoS vs. Server's Processing Power

Observations:

As shown in Table 4.8 and Figures 4.6-4.7, the overall system quality degrades when the server's processing power decreases. When there are 300 students taking VU courses and if the server's processing power is only 375 jobs/s, the whole system is congested. If we use "less than 10% of the service is bad quality" as a criterion for estimating the system capacity, then we should use a server whose processing power better than 1200 jobs/s. On the other hand, for a system of this scale, a server more powerful than 7500 jobs/s will not gain much in quality of service. The "knee" shapes of the curves in Figures 4.6~4.7 show that there exists a "critical" region (375,1500) for the server's processing power in which varying the server's processing power has a great impact on the whole system's performance; and a "flat" region (7500, infinity) in which upgrading the server will gain little for the given QoS criterion. This suggests a method for evaluating the importance of upgrading the server. The shapes of the curves of QoS vs. server processing power are also valid for the servers with more or less processing powers. The main difference is that the "turning" point will vary. For the VU server (1997) which has "N=10" (the effective server processing power is less than the "absolute" processing power), the curve will looks like the above "N=1" curve shift to the left.

For the experiments with the user model of "99% pessimistic", as shown in Table 4.9 and Figures 4.8-4.9. The QoS curves show the similar shape. The main differences is for the same configuration, the "99% pessimistic" model always have worse performance compare to the "905 pessimistic" model because it generate more traffic. If we use "less

than 10% of the service is bad quality" as a QoS criterion , then we should use a server whose processing power better than 1800 jobs/s. For the same condition "90% pessimistic" model requires 1200 jobs/s. The more conservative estimation of the user model leads to requirements of more powerful server for the same QoS criterion.

4.4 How 'Loading Image' Affects QoS of VU Courses

As we know, multimedia content such as images and rich-text will cause much more traffic than simple ASCII text. The "loading images" parameters thus play an important role in determining the system performance. Better quality images (such as better resolution, more colors et cetera.) will cost more bits and make for larger files, while using more images in a course may make the course more "vivid" and easily understood. It is thus useful to determine how the loaded image size and frequency will affect the system's performance. In addition, we want to be able to estimate the largest image files (and loading frequency) that we can safely use in a course for a given system configuration and QoS criterion. These parameters are important to course designers, who want their courses to "work".

4.4.1 The Loaded Image Size

System Configuration:

Server's processing power consumed by a user's request (N)	1

Number of students taking VU courses	300
Average (loaded) image size and standard deviation (μ,σ)	μ=64kB - 1MB
Loading image frequency (F)	50 %
Server processing power (P)	1500 jobs/s
Background traffic level (B)	0

For the image size, we use a Normal distribution, as explained previously. The average image size (mean value) varies from 64kB to 1MB and for these initial experiments, we assume that the standard deviation is simply 1/3 of the mean value. This value was chosen for the variance, since it matched that of the image testset used in the creation of the models (see Section 3.2.4). We assume that this variance will generalize to different courses using different ranges of image size. In practice these distributions should be tailored to the course being modeled.

Although our prediction for a "reasonable" frequency of image loads is 25%, we set the number to be 50% here. Because we want to exercise the worst case and produce a conservative estimate for the maximum supportable image size.

***Simulation Results**:*

The results of the simulations described above are given in Table 4.10.

Table 4.10: VU Response Time vs. Loaded Image Size.

Image	Request Response Time (s)	QoS Parameters	Note

Size (kB)	Max.	Min.	Mean	Standard Deviation	"Not good"	"Bad"	
64	22.2	0.0158	0.354	4.93	12.5%	3.7%	
256	22.1	0.0158	0.863	4.99	15.6%	3.9%	
512	37.2	0.0159	3.148	6.96	25.2%	7.5%	
1024	471.7	0.0200	46.124	85.27	47.2%	25.4%	C

C: Congested

Observations:

As shown in Table 4.10, the overall system QoS degrades as the image size increases. For the current system, we should keep the image size at or below 512kB if we use the "less than 10% of the service is bad quality" as the QoS criterion. As we know, most compressed images (MPEG, GIF et cetera.) are below this level; however, many WORD files and special application images (medical and geographical, for example) are larger.

4.4.2 The Frequency of Loading Images

System Configuration:

Server's processing power consumed by a user's request (N)	1
Number of students taking VU courses	300
Average (loaded) image size and	256kB, 85kB

standard deviation (μ,σ)	
Loading image frequency (F)	25 %, 50%, 70%
Server processing power (P)	1500 jobs/s
Background traffic level (B)	0

Although our prediction for the “reasonable” image size is below the level of 128kB, we set the number to be 256kB here. Because we want to use the worst case to produce a conservative estimate of the image loading frequency.

Simulation Results:

The results of the experiments described above are presented below in Table 4.11 and Figure 4.10.

Table 4.11: VU Response Time vs. Image-loading Frequency.

Image Frequency	Request Response Time (s)				QoS Parameters		Note
	Max.	Min.	Mean	Standard Deviation	"Not good"	"Bad"	
25%	13.9	0.0145	0.636	2.69	7.9%	0.0%	
50%	22.1	0.0158	0.863	4.99	15.6%	3.9%	
70%	26.9	0.0158	1.207	3.36	14.7%	6.5%	

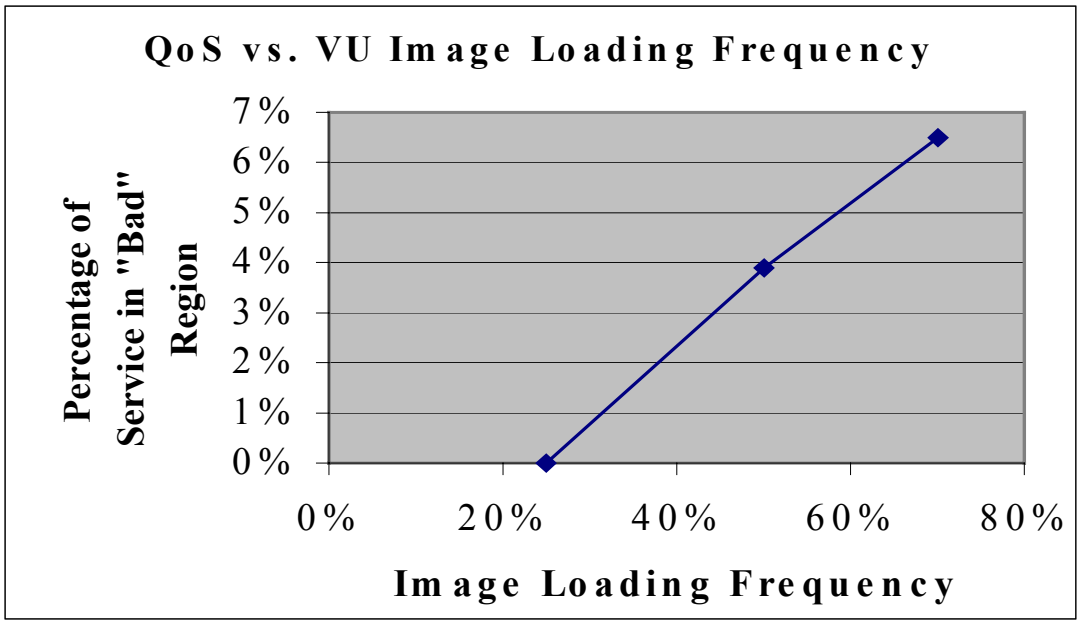

Figure 4.10 QoS vs. VU Image Loading Frequency

Observations:

As shown in Table 4.11 and Figure 4.10, the overall system QoS degrades with the increase of the image loading frequency. If we look at the percentage of time service is in the "bad" region, the performance degrades linearly, as shown in Figure 4.6. For the system configuration above, if we use the "less than 10% of the service is bad quality" as a QoS criterion, the quality is still acceptable even when the loading image frequency is set to 70%. Therefore "image downloads" are unlikely to cause serious problems to VU courses, as long as they are kept to a reasonable level.

4.5 Capacity for Adding VOD Sessions

The effort in this section is to see how many streaming video, or VOD, sessions can be added if the system is half loaded with other VU courses (300 users). The configuration of the system is based on our prediction for the "final" Virtual-U system rather than the 1997's version. Because VU itself is evolving and we believe it makes more sense to provide VOD service when the system is "mature" in providing conference-based courses.

At present, we do not have detailed information about the VOD server of the sample VOD lectures we captured; however, we do not have to use the same server in the telelearning environment, though we may provide the same type of video lectures. In our simulation, we assume the server has dedicated RAM, Disk arrays, I/O et cetera for video processing and that the software for video processing is well designed. In a word, the video processing is efficient at the server site and the server

has enough power to handle the traffic compared to the network. In some situations, it may be desirable to have a separate video server and this is a topic to be explored in future research.

***System Configuration**:*

Server's processing power consumed by a VU user's request (N)	1
Number of students taking VU courses	300
Average (loaded) image size and standard deviation (μ,σ)	64kB, 20kB
Loading image frequency (F)	25 %
Server processing power (P)	1500 jobs/s
Background traffic level (B)	0

Simulation Results:

For the VU course, we continue to use the "request response time" as the QoS measurement; however, for the streaming video session, we measured the rate of the play buffer's under/over flow (times/minute) instead, since this is much more important from the viewers point of view. The results are shown in Table 4.12 and in Figure 4.11.

Table 4.12: Service Quality When Adding VOD Users

# of VOD users	VU					VOD	Note
	Request Response Time (s)			QoS Parameters			
	Max.	Mean	Standard Deviation	"Not good"	"Bad"	Buffer[12] O/U Flow	
0	1.487	0.104	0.17	0.0%	0.0%	N/A	
30	1.517	0.104	0.16	0.0%	0.0%	0.706	
60	1.522	0.110	0.16	0.0%	0.0%	0.784	
90	2.348	0.156	0.24	0.0%	0.0%	0.841	
105	5.032	0.328	0.37	0.0%	0.1%	1.030	
120	9.793	0.596	0.85	12.3%	1.2%	1.416	
135	35.266	3.168	6.91	33.6%	14.5%	6.452	C

C: Congested

[12] The unit is times/minute

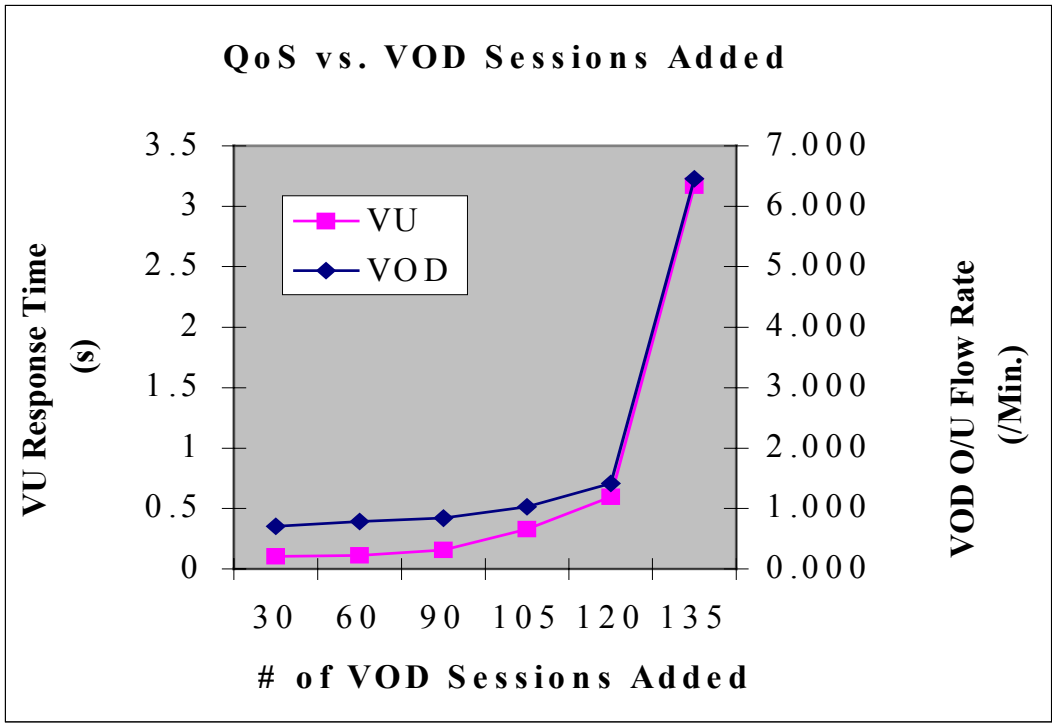

Figure 4.11 QoS vs. VOD Sessions.

Observations:

As the number of VOD user increases, the overall quality of service degrades for both VU and VOD users. The QoS curves have "knee" shapes. When the number of VOD sessions is below 120, the overall QoS does not degrade dramatically because the VU and VOD services are consuming different critical resources. For the VU users, the critical resource is the server's processing power, while the critical resource for the VOD is network bandwidth. Adding more VOD sessions does not greatly degrade the QoS of VOD users as long as there is enough network bandwidth. When the number of VOD sessions is greater than 135, the

system gets congested due to the starvation of network bandwidth. Note that this number will drop to about 7 if 1Mb/s video streams are used. These results agree with our analytical predictions. Theoretically, the maximal bandwidth in the system is throttled by the 10Mbps Ethernet. The practical bandwidth of the Ethernet is around 8Mbps as discussed in Chapter 2. From the network bandwidth perspective, the effect of VU users is minimal. Since each VOD session consumes 56kbps bandwidth, the total supportable VOD sessions will be less than 140 (56kbps * 140 = 8Mbps).

From our simulation experiment results, we can see that when there are 300 VU users in the system, the capacity of the above 56kbps VOD sessions is 120. This number also agrees with simple analytical calculations. For the system with 300 students, up to 120 of them can simultaneously view the VOD sessions (as part of the course material).

The QoS curves for VU and VOD have the same shape, and they "jump" at the same point. The reason for this is that in the non-prioritized network protocols being used, there are no reservations or preferences made for the different applications when the network resources are allocated. This lack is one of the characteristics of the classic (as opposed to the new generation) "best-effort" TCP/IP protocols, which were not designed to carry mixtures of different types of traffic, each with its own QoS requirements. For our telelearning applications, if the network can distinguish the "delay sensitive" VOD sessions from other traffic, and process them with higher priority, the QoS/Capacity of the VOD sessions will be improved. The QoS of VOD will then be sustained even when the network starts to get congested; i.e. the VOD courses' slight slope region in Figure 4.7 will be extended. This would

happen at the cost of degrading the QoS of VU (and other applications) and in refusing new video connections. However, many services as VU are less "delay sensitive" than video, and it is possible to find a balancing point good for the overall system performance. There are many researchers working on possible new protocols [Moh, 1996] [Johnson, 1996], but this topic is outside the scope of this thesis.

4.6 How Background Traffic Affects System Performance

The background traffic of the network greatly affects the system's performance. In our network structure, background traffic exists in three places: 1) The subnet where the server is located. 2) The backbone network. 3) The subnet where the user is located. These cases will now be analyzed individually and we will see what is most likely to be the bottleneck in the system.

We will use "background traffic level", B, to represent the intensity of the background traffic on the network segment being studied. Since one of our main interests here is to find the maximum tolerable background traffic level in each of the specific network segment, we will vary the background traffic level for the segment being studied and set the background traffic in other parts to be zero.

4.6.1 Background Traffic in the Server's Subnet

***System Configuration**:*

Server's processing power consumed by a VU user's request (N)	1
Number of students taking VU courses	300
Number of VOD sessions	60
Average (loaded) image size and standard deviation (μ,σ)	64kB, 20kB
Loading image frequency (F)	25 %
Server processing power (P)	1500 jobs/s
Background traffic level in this network segment (B)	0% ~ 40%

As in previous simulations we assume that the system is "half loaded" with 300 VU users; however, we have added 60 streaming-video sessions because we believe that the Video service will be integrated into the VU system as an important part. In addition, video services are more "network bandwidth sensitive" and can be heavily impacted by the background traffic in the network environment. (Note that 60 is also chosen by the "half loaded" consideration since the capacity is 120).

***Simulation Results**:*

The results of the above simulations are shown below in Table 4.13 and in Figure 4.12.

Table 4.13: QoS vs. Background Traffic in the Server's Subnet

Background Traffic Level	VU					VOD	Note
	Request Response Time (s)			QoS Parameters			
	Max.	Mean	Standard Deviation	"Not good"	"Bad"	Buffer O/U Flow	
0%	1.52	0.111	0.16	0.0%	0.0%	0.784	
10%	1.58	0.123	0.18	0.0%	0.0%	0.815	
20%	2.10	0.194	0.29	0.0%	0.0%	0.806	
30%	5.32	0.564	0.61	0.4%	0.0%	1.232	
40%	15.92	1.137	1.25	7.4%	7.0%	2.992	C

C: Congested.

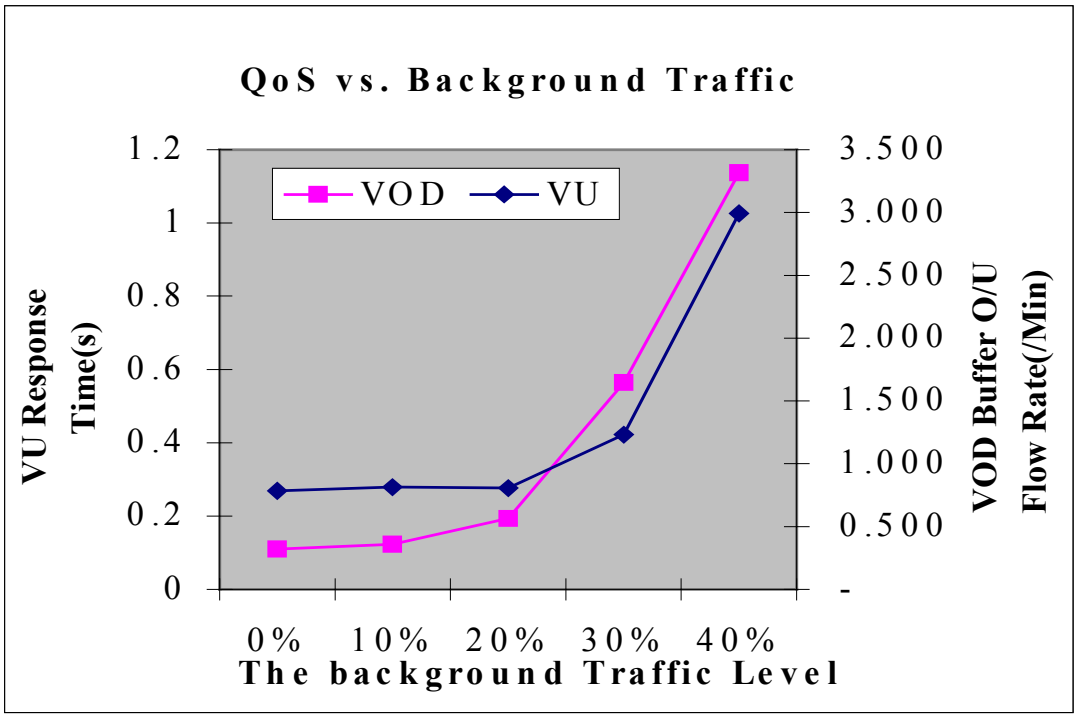

Figure 4.12 QoS vs. Background Traffic in Server's Subnet

Observations:

As shown in Table 4.13 and Figure 4.12, the overall system quality degrades as the level of background traffic on the server's subnet increases. Video is very sensitive to the background traffic of the server's subnet. If we use "play buffer's over/under flow rate is less than 1.5 times/min." as the criterion for estimating the system capacity, we should keep the background traffic level of the server's subnet at less than 30% of the available bandwidth. If the background traffic level is greater than 40%, the network gets congested. This can also be understood using an analytical approach. The total practical bandwidth of the Ethernet is 8Mbps and the 60 VOD sessions need 3.4Mbps (56kbps*60) of bandwidth. Even the 300 VU users do not require such a large amount of network bandwidth; a background traffic level of 40% will definitely congest the Ethernet.

4.6.2 Background Traffic on the Backbone

***System Configuration**:*

Server's processing power consumed by a VU user's request (N)	1
Number of students taking VU courses	300
Number of VOD sessions	60
Average (loaded) image size and standard deviation (μ,σ)	64kB, 20kB
Loading image frequency (F)	25 %
Server processing power (P)	1500 jobs/s
Background traffic level in this network segment (B)	20% ~ 90%

Simulation Results:

The results of this work are shown below in Table 4.14.

Table 4.14: QoS vs. Background Traffic in the Backbone

Background Traffic Level	VU					VOD
	Request response Time(s)			QoS Parameters		Buffer O/U Flow
	Max.	Mean	Standard Deviation	"Not good"	"Bad"	
20%	22.3	1.533	6.02	22.9%	4.1%	0.697
40%	23.3	1.642	6.07	23.6%	5.6%	0.712
60%	21.1	1.278	6.42	23.3%	5.9%	0.768
75%	23.7	1.644	5.94	22.8%	3.6%	0.755
80%	29.3	1.753	6.29	24.1%	4.8%	0.748
90%	30.9	3.256	7.56	31.4%	9.6%	3.485

Observations*:*

As shown in Table 4.12, the overall system quality does not degrade appreciably when the background traffic on the backbone is under 80%. This shows that the backbone network is able to handle the traffic of the above configuration even with a large quantity of background traffic.

4.6.3 Background Traffic in the User's Subnet

We assume that the 300 VU users and 60 VOD sessions mentioned in the former sections are evenly distributed in the user's subnets. That is, each user's subnet has 50 VU users and 10 video sessions. The background traffic on each of the user subnets is identical. Both simulation and "back of the envelope" calculation (as used above) show that the user's subnet gets congested when the background traffic reaches approximately 75%of the capacity in each of the users subnets; an acceptable QoS can be maintained as long as the background traffic level in the user's subnet is below 65%.

4.6.4 Conclusions

From the results above, we found that for the current system configuration, the effect of congestion on the server's subnet is much more severe than elsewhere (30% tolerable background traffic compared to 80% and 65% in the backbone and the user's subnets). From viewpoint of telelearning applications, the critical link is thus the server's subnet. The server shares the 10M bandwidth with other stations, and this bandwidth is used for subnet internal communication and communication to the backbone as well. Thus, making the server's subnet dedicated to the server is important. Historically, Virtual-U testing system experienced bad performance due to the background traffic caused by a name-server located in the same subnet. An upgrade to the server's subnet will improve the performance of the whole system. If we keep the server's subnet unchanged and upgrade only the backbone network or the user's subnet, it will not improve the

performance of the entire system, as the bottlenecks are not there. Another possible approach is to move the server so that it placed directly on the backbone network.

4.7 How Streaming Video and VU Affects Each Other

As we discussed earlier, VU and streaming video have different critical resources to consume. Within the system capacity, one might expect that adding more VU or video users would not greatly degrade the QoS of the other type of user. However, the QoS of streaming video is sensitive to the traffic pattern on the network (i.e. the burstiness), since this may result in delay jitter. Therefore, some characteristics VU courses may affect the QoS of video sessions. For example, the frequency of downloading images in VU is a strong candidate for doing this.

***System Configuration**:*

Server's processing power consumed by a VU user's request (N)	1
Number of students taking VU courses	300
Number of VOD sessions	60
Average (loaded) image size and standard deviation (μ,σ)	64kB, 20kB
Loading image frequency (F)	10% ~ 50%
Server processing power (P)	1500 jobs/s
Background traffic level (B)	0%

Simulation Results:

The simulation results for the above configuration are shown below in Table 4.15 and in Fig. 4.13

Table 4.15: QoS vs. Image Loading Frequency of VU

VU Image Loading Frequency	VU					VOD
	Request Reopens Time(s)			QoS Parameters		Buffer O/U Flow
	Max.	Mean	Standard Deviation	"Not good"	"Bad"	
10%	2.829	0.131	0.22	0.0%	0.0%	0.771
25%	2.348	0.156	0.24	0.0%	0.0%	0.816
50%	2.715	0.242	0.35	0.0%	0.0%	1.018

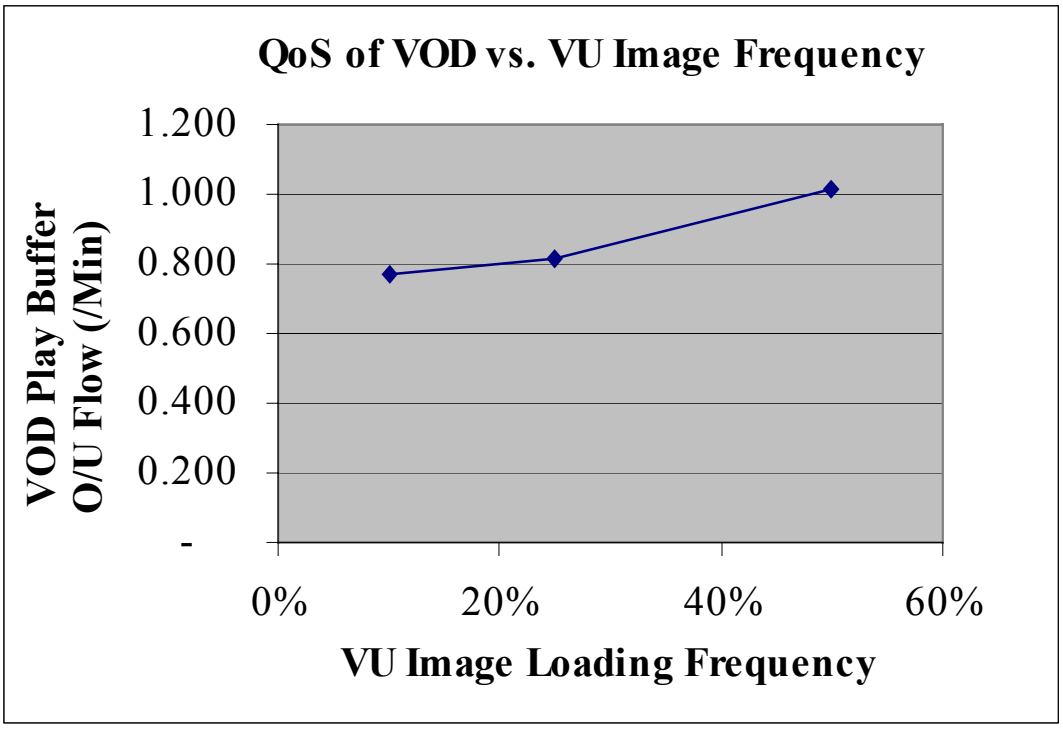

Figure 4.13 QoS of VOD vs. VU Image Frequency

Observations:

As the frequency of loading images in the VU courses increases, the QoS of video sessions degrades. The plot of Figure 4.13 shows a linear shape which means the QoS degrades gradually. From the results of former sections, we know the system is not congested in this configuration. The reason for the QoS degradation can be explained in this way: the VU traffic gets more bursty when more images are downloaded, this causes more underflows and overflows in the VOD play buffers, which in turn downgrades the perceived quality of service. This experiment thus shows that different "modes" of telelearning may affect each other.

Chapter V
Conclusions

In this thesis, we have presented a computer-based simulation method for carrying out performance research on networked multimedia systems such as telelearning. A real system can be decomposed into several functional components, each of which can then be modeled separately with the expert knowledge in that specific field. When the conceptual models are implemented into computer programs, a simulation system that represents the real system's structure and functionality will be available. We used the Virtual-U system as an example to describe how to build such a simulation system. We also presented some results we obtained when we used the simulation system to predict Virtual-U's capacity and analyze key factors' effects on system performance.

Through the work described in this thesis, we have demonstrated that our approach is valid and that it can be fruitful in the future. A simulation system with reasonable complexity can be obtained to represent the real system from the perspective of our research interests (we do not have to repeat everything that exists in the real system). How good the simulation system is depends on the accuracy we can achieve for the models.

We built a simulation system (tool) with OPNET that can simulate students taking courses through the Virtual-U. Although the models used are somewhat coarse grained (leaving room for future students), the simulation system includes almost all of the key components in the real system (the user, the server, the network environment et cetera.) and

is able to help evaluate the overall system performance. The work of building such a simulation system is twofold. Firstly we need to take into account as many of the systems "key factors" as possible. Secondly we should accurately represent how each factor works. Since our main research interests are in the overall system performance and how each factor contributes to it, a framework that includes all the key components is the "Number 1" target of building the first version of the simulation system. The models developed so far all need work for the simulation to be a better guide to what happens in the real world; however, all of the key components of the system are present and working.

We also obtained some interesting results from the computer simulation tool we developed and gained a better understanding of the system capacity and how the key factors affect the system performance. For the system configuration of 1997, the system bottleneck is not the network, but the server. The capacity of the system depends on the server's processing ability, which is determined by both hardware power and software design. From the network bandwidth perspective, the server subnet is most likely to suffer bandwidth starvation. The users activity and the contents of the courses all have effects on the system performance. For example, adding more multimedia content to a course will increase the load of the system. But if we control the amount of the material to a reasonable level, the degradation of QoS is not dramatic. VU and VOD traffic consume different "critical resources" (server power and network bandwidth respectively) and could be used in "mixture". If the mixing is controlled in a good level, both good QoS and system efficiency can be achieved.

Analytical analysis agrees with computer simulation if an application's traffic is analytically tractable. Although our assumption that the video server has enough power to ignore its effect on the performance needs re-evaluation in future work, we have been able to see how the network bandwidth imposes limitation on the system capacity.

As for the accuracy obtained from the simulation, we can not get a good answer until some experiments are made real systems that have significant multimedia content, since otherwise we are just exercising the server. Although the numbers in our results may not be totally accurate, some clues are provided about the overall behavior of the system (e.g. the shape of curve) and our results will help further investigations.

According to our results, in the existing VU system that offers text-based conference courses, the server is the bottleneck and needs improvement. When multimedia courses (e.g. streaming video lectures) are added, QoS oriented new protocols should be considered to make better network support.

The new QoS protocols provide better support to multimedia services through two ways. One is to reserve bandwidth for real time traffic and guarantee the availability of required resources. The other is to label the traffic streams and process them according to their priorities. During the evaluation of these newly designed protocols, we should keep in mind both the protocols' performance improvement and their backward compatibility with existing hardware and software. For the existing Virtual-U network environment, new QoS oriented protocols can be adopted in two levels. One is in the local network; the other is across the Internet.

For the local network, there are many QoS oriented LAN protocols providing better support to multimedia traffic than the 802.3. Among them are IEEE 802.12[Watson, 1995] and Ethernet++ [Edwarts, 1995].

IEEE 802.12 retains the frame format from the original Ethernet. Instead of CSMA/CD, it uses a deterministic protocol called demand priority MAC. Stations need to ask for permissions from the hub before transmitting any frame. The protocol is efficient because it avoids the collision problem in CSMA/CD and there is no token propagation around the network as in Token Rings. The demand priority protocol can support two priorities by providing two request signals: a standard priority (SP) request for the old applications as file transfer and a high priority (HP) request for the delay-sensitive traffic such as voice and video. A hub will always serve a HP request before a SP request. It serves each priority level in a round-robin order in which the available bandwidth will automatically be shared evenly among all stations currently active at the highest priority. A SP request may be promoted to HP if it has waited longer than some fixed amount of time (such as 250 ms). With two priority levels the network can provide a service that guarantees bandwidth and bounds the access delay for real-time applications. It is observed that while 802.12 can keep high priority (HP) request's delay very small, it may also result in the standard priority requests (SP) suffering long delay [Moh, 1996]. This protocol is standardized and has many vendors in the market.

Ethernet++ employs a reserved cyclic access scheme with call admission control to provide high priority (HP) services to real-time traffic. The traditional CSMA/CD is used for standard priority (SP) access that supports non-real-time traffic. It operates a dual protocol

network. This protocol does not provide absolute preemptive channel access for HP stations, HP traffic delay is dependent on SP traffic conditions. The non-preemptive nature provides a reasonable inter-priority fairness. This is very important for telelearning systems as VU, because there will be traffic of both real time applications and non-real time applications.

On the Internet side, IP version 6 [Huitema, 1998] will provide better support for multimedia services. IP version 6 is also called IPv6 as opposed to IP version 4 that is being used at present. IPv6 specification defines flows. “A flow is a sequence of packets sent from a particular source to a particular (unicast or multicast) destination for which the source desires special handling by the intervening routers”. Flow labels will be used if the transmission needs some special treatment. This enables prioritized processing. Reservation is also available.

In the Internet, one reservation protocol of choice is RSVP (ReSerVation Protocol) [Braden, 1996] which is designed for multicast applications such as high-speed video transmission. These applications have some peculiar characteristics, like a large number of receivers that may be experiencing very different transmission conditions and that may also belong to different network domains. They do resource reservation by sending RSVP messages to the network with a specific QoS requirement. The RSVP daemon sets parameters in a packet classifier and packet scheduler to obtain the desired QoS. The packet classifier determines the QoS class for each packet and the scheduler orders packet transmission to achieve the promised QoS for each stream. RSVP carries the request through the network, visiting each node the network uses to carry the stream. At each node, RSVP attempts to make a

resource reservation for the stream. Although the RSVP protocol is designed specifically for multicast applications, it can make unicast reservations. RSVP is not the only IP reservation protocol that has been designed for this purpose, but RSVP currently has the most industry support.

Further work on the simulation system could be in three areas:

1) We should improve the accuracy of the models in representing the real system. For example, in the server model, the server's processing power parameters needs to be estimated better, with different types of requests using different amounts of processing power. This re-evaluation may come from a thorough analysis of the server along with some experiments to get real measurements. One of the difficulties we met in building an accurate user model comes from the "unpredictability" of human thinking habits. One approach to overcome this may be to increase the granularity of the simulation by dividing users into several groups according to their behaviors/habits, so that a statistical model could be found for each of the groups. Efforts should also be made for a user model without the system dependency. One approach may be analyzing the log data of a set of users work with a "congestion free" environment, i.e. the network is isolated and has enough resources for the users; the server is also powerful enough and will not be a bottleneck. For the background traffic model, we may try how a "self-similar" background traffic model affect the system performance. We are not expecting significant different for the VU users since the VU application is not very sensitive to background traffic itself; however, the VOD service may got impacted, because the "self-similar"

background traffic is more bursty, this will cause more underflow or overflow of the viewer's play buffer which in turn will degrade the quality of service of the VOD sessions..

2) More work needs to be done in the validation of our simulation models. The best way to do this it is through experiments on a real system. This approach is the most reliable but some times difficult to do because of the lack of data available regarding real courses – especially ones that push the technological envelope. Another alternative for validation would be to build a log-file collection mechanism right into our simulation. Some level of validation could then be obtained by comparing the simulation's log-file with the one produced by the course that we are trying to model. This second method may be easier, but it should be used with care since we need to validate the "log data collecting system" in the simulation first. It is also not clear that similar log-files would mean that the QoS is being predicted correctly.

3) Finally, more features need to be built into the simulation to make it more "versatile". For example, we could include such applications as video/audio conferencing as one of the tools available in a course. in order to increase the levels of student-student interaction. We could also add new protocols (such as IPv6, RSVP, 802.12 et cetera.) to the network model and evaluate the performance of different types of networking technologies in supporting telelearning applications. This work will provide very useful guidelines to the telelearning system designers for taking complete advantages of today's network environment and optimizing telelearning systems.

Appendix A
OPNET Simulation Package[13]

"OPNET is a comprehensive software environment for modeling, simulating, and analyzing the performance of communications networks, computer systems and applications, and distributed systems."

OPNET is used to analyze the performance and behavior of existing or proposed networks, systems, and processes (as shown in Figure A.1). A set of tools are included with the package that assist users through the following phases of the modeling and simulation cycle:

1) Model Building and Configuration
    - Network Editor - define or change network topology models
    - Node Editor - define or change node level (system architecture) models
    - Process Editor - define or change process level (behavioral logic) models
2) Running Simulations
    - Simulation Tool - define and run simulations using models constructed with the OPNET editing tools.
    - Interactive Debugging Tool - interact with running simulations
3) Analyzing Results
    - Analysis Tool - display and compare statistical results
    - Animation Viewer - watch dynamic behavior of models during simulation runs

[13] The information in the appendix is based on the OPNET web site www.mil3.com.

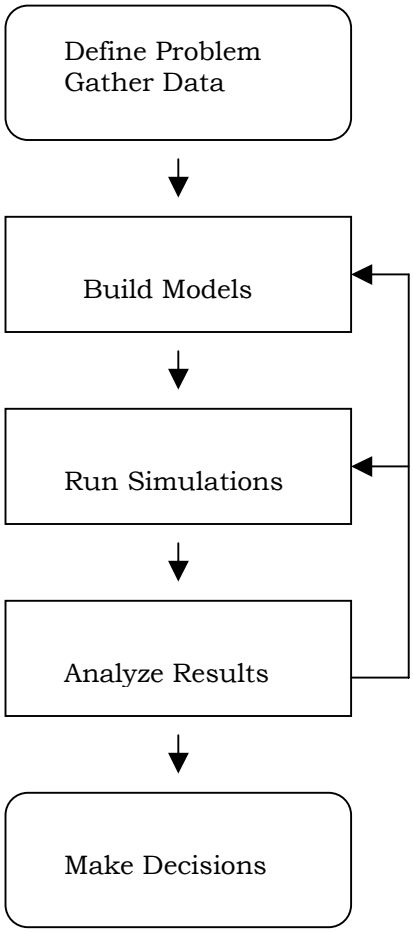

Figure A.1 OPNET work flow [OPNET Web]

"To lend structure and discipline to the overall model, OPNET models fit together in a hierarchical fashion." [OPNET Web] OPNET Network Models define the position and interconnection of communicating entities, or nodes. Each node is described by a block structured data flow diagram, or OPNET Node Model, which depicts the interrelation of processes. Each programmable block in a Node Model has its functionality defined by an OPNET Process Model with a standard

programming language and a broad library of pre-defined modeling functions.

The OPNET Network Editor graphically represents the physical topology of a communication network. Networks are made up of nodes and links objects, which are graphically assembled and parameterized via pop-up dialog boxes. To create node objects, users select node types from a library of example and user-defined models. Each OPNET Node Model has a specific set of attributes that are used to configure it.

The OPNET Node Editor graphically represents node architectures, which are diagrams of data flow between modules typically representing hardware and software subsystems. Module types include processors, queues, and traffic generators. Processors are general modules that defined according to protocol and algorithm specification. The functionality of processor and queue objects is defined using OPNET Process Models. Instances of OPNET Node Models are used to populate OPNET Network Models.

The OPNET Process Editor uses a state-transition diagram approach to support specification of any type of protocol, resource, application, algorithm. States and transitions graphically define the progression of a process in response to simulated events. Within each state, general logic can be specified using a library of pre-defined functions. The full flexibility of the C programming language is also accessible. As with other OPNET editors, users can construct entirely new process models.

After a set of OPNET Network, Node, and Process Models are fully defined, users can run simulations based on them via the OPNET

Simulation Tool, and plot statistical performance measurements based on simulation studies in the OPNET Analysis Tool.

"The OPNET Analysis Tool provides a graphical environment that allows users to view and manipulate data collected during simulation runs." [OPNET Web] Standard and user-specified probes can be inserted into a model to collect statistics. Simulation output collected by probes can be displayed in numbers or figures, or exported to other software packages for analysis. First and second order statistics on each trace along with the confidence intervals can be automatically calculated and displayed. "OPNET supports the display of data traces as time-series plots, histograms, probability density and cumulative distribution functions, and scatter grams." [OPNET Web] All these are helpful in getting results and discoveries.

Appendix B
Program Description

Network Model

In the network model, the backbone network is an eight-port bridge. The seven subnets are: subnet_1, subnet2, subnet_3, subnet4, subnet_5, subnet6 and the server net.

In each of the user's subnet, there are 25 user stations and 2 background traffic generators connected to a 32 port Ethernet hub. In the server's subnet, there are a server and 2 background traffic generating stations.

The multi-ports bridge is defined by the file "ethernet8_bridge.nd.m". The key parameter is the "Bridge frame service rate (bps)".

The Ethernet hub is defined by the model "ethernet32_hub.nd.m".

The background traffic generating station is defined by "ethernet_station_base.nd.m and the key parameters are:

1) Application data size: This is the packet size for the background traffic. (bits)
2) Application traffic generator rate PDF: This is the distribution of the background traffic packets.
3) Application traffic generating rate : This is the average value of the background traffic generating rate.
4) Ethernet address: This is the address of the background traffic generator’s address.

5) Destination highest address: This is the upper bound of the destination address for the background traffic.
6) Destination lowestest address: This is the lowest bound of the destination address for the background traffic.

Server Model

The server is defined by "cyeth10T_server_base.nd.m". The key parameters are :

1) Server's processing power: This number is the server's processing power (jobs/s)
2) Server configuration table: This table lists all the services the server is providing.
3) Tpal address: This is the server's identification/address.

User Model

The user is defined by the node file "vunode1.nd.m". The user process model are implemented by 4 files. The main control file is "vunet_app_mgr.pr.m". It will spawn the appropriate processes for different types of applications. These applications are: "vucli_cli.pr.m" for VU courses, "cygna_cli.pr.m" for VOD course and "odgna_cli.pr.m" for other applications such as Email, FTP et cetera.

The key parameters for the VU course are:

1) Login Rate: This is the hourly session rate.
2) Terminal Traffic: This is the average traffic size from the user to the server.
3) Host Traffic: This is the average traffic size from the server to the user.

4) Server: The address of the VU server.
5) Login Duration: The average time for a login session.
6) Duration Time: The average time between 2 user's request to the server.

The key parameters for the streaming video (VOD) course are:

1) Command Rate: The average rate of users request data from the server.
2) Terminal Traffic: The Traffic size in the direction of user --> server.
3) Host Traffic: The Traffic size in the direction of server--> user.
4) Login Duration: The average time of a VOD session.
5) Server: The address of the VOD server.
6) Duration Time: The duration time in each of the state
7) Login Rate: The average hourly rate of VOD session.

The transition probabilities matrix of a user model is defined in the header part of the relative model file (e.g. cygna_cli.pr.m). The program will determine the state transition according to the conditions defined.

In the VU model, we took a “micro” view of the user’s activity and modeled the requests. In the VOD model, our main interest is in the traffic rate other than the atomic requests. And as we discussed earlier, we got the trace from the viewer’s end, the traffic rate of the flow is still good, but the packets Interarrival time are distorted by the network. So we modeled the rate rather than the request interarrival time in the VOD.

Each state has two parameters, the request interarrival time and the “state duration time”. These parameters are defined in the program to provide better modeling of real problem, but they do not have to all be used at the same time. For example, for VU , we used a so in each state,

the user only send one request, thus the “state duration time” in fact used to model the request interarrival time. In the later, for the more complex models, each state has multiple requests, both the two parameters can be used.

Probes

In the Probe file “probeSS.pb.m”, some useful probes are defined:

"client VOD resp.3": The VOD response time of client1in subnet3.

"client VU resp.3": The VU response time of client1in subnet3.

"client VU resp.6": The VU response time of client1 in subnet6.

"server Eth throuput": The throuput of the Ethernet subnet where the server is located.

"client31 Eth throuput": The Ethernet throuput of sunet3

"BK-sta Eth throuput": The Ethernet throuput of the background traffic station in the server's net.

"VOD playbuffer 3.1": The Play buffer's over/underflow of client1 of the subnet3.

"VOD playbuffer 6.1": The Play buffer's over/underflow of client1 of the subnet6.

"server service time": The service time for the server of the request

"global VOD response time": The VOD response time of all the users.

"global VU response time": The VU response time of all the users.

"global application response time": The response time of all the users for all the applications.

"global ETH throuput": The Ethernet throughput of all the subnets.

"global play buffer over/underflow": The Play buffer's over/underflow of all the users.

References

[Beran, 1995] J. Beran, R. Sherman, M. Taqqu, and W. Willinger, "Long range dependence in variable-bit-rate video traffic", IEEE Transactions Communications, Vol. 43 1995.

[Beranetal, 1992] J. Beranetal, "Variable-Bit-Rate Video Traffic and Long-Range Dependence," accepted for publication in IEEE Trans. On Commun., 1992

[Braden, 1996] R. Braden et al., "resource Reservation Protocol (RSVP) – Version 1 Functional Specification," Aug.12,1996. Available via http://www.ietf.org/html. Charters/intserv-charter.html.

[Crovella, 1996] Mark E. Crovella and Azer Bestavros, "Self-Similarity in World Wide traffic Evidence and Possible Causes" . In proceedings of the 1996 ACM SIGMETRCS International Conference on measurement and Modeling of Computer Systems. Pp.160-169, May 1996.

[Edwarts, 1995] F. Edwards and M.Schulz, "A priority media access control protocol for video communication support on CSMA/CD LANs", ACM Multimedia Systems, Springer-Verlag and ACM, 1995

[Freeman, 1995] Roger L. Freeman, *Practical data communications*, John wiley &Sons, 1995

[Frost, 1994] Victor S. Frost and Benjamin Melamed, "Traffic Modeling for Telecommunications Networks", IEEE Communication Magazine, March 1994

[Hogg,1997] Robert V. Hogg and Elliot A. Tanis, *Probability and statistical inference,* Prentice–Hall, Inc, 1997

[Huitema, 1998] Christian Huitema, *IPv6- the new Internet protocol*, Prentice–Hall, Inc, 1998

[Johnson, 1996] Howard W. Johnson, *Fast Ethernet: dawn of new network,* Prentice–Hall Inc, 1998

[Larson, 1979] H.O. Larson and B. O. Shubert, *Probabilistic Models for Engineering Sciences*, John Wiley and Sons, 1979

[MacDougall, 1987] M. H. MacDougall, *Simulating computer systems: techniques and tools*, MIT press 1987.

[Maglaris, 1988] B. Maglaris, D. Anastassiou, P.Sen, G. Karlsson, and J. D. Roberts "Performance models of statistical multiplesing in packet video communications," IEEE Trans. Commun., vol. COM-36, no. 7, July 1998

[Mandelbrot,1968] B. Bl Mandelbrot and J. W. Van Ness, "Fractional Brownian Motions, Fractional Noises and Applications," SIAM Review, vol. 10, 1968.

[Michiel, 1997] Herman Michiel & Koen Laevens, "Teletraffic Engineering in a Broad-band Era", Proceedings of the IEEE. Vol. 85. No.12 December 1997

[Miller, 1992] Mark A. Miller, *Internetwork: a guild to network communiactions* by M&T public Inc 1992.

[Moh,1996] W. Melody Moh et al, "Evaluation of High Speed LAN Protocols as Multimedia Carriers". In 1996 IEEE International Conference on Computer Design

[NetShow Web] http://www.microsoft.com/product/

[OPNET Web] http://www.mil3.com

[Paxon, 1994] Vern Paxon, "Empirically-Derived Analytic Models of Wide-Area TCP Connections". In IEEE/ACM Transactions on Networking, 2(4), pp. 316-336, August 1994.

[Paxon,1995] Vern Paxon and Sally Floyd, "Wide-Area Traffic: The failure of Poisson Modeling", In IEEE/ACM Transactions on Network, 3(3), pp. 226-224, June 1995.

[Schwatz, 1987] Mischa Schwartz, *Telecommunication networks : protocols, modeling, and analysis*, Addison-Wesley, 1987

[Schwartz, 1993] Schwartz R.L and Tom Christiansen, *Learning Perl,* O'Reilly & Associates 1993

[Schwartz, 1996] Mischa Schwartz, *Broadband integrated networks,* Prentice Hall 1996.

[Tanenbaum,1996] Andrew S. Tanenbaum, *Computer networks*, Prentice Hall 1996

[Telelearning Web] http://WWW.telelearn.ca

[VU Web] http://virtual-u.cs.sfu.ca/vuexchange/

[Watson, 1995] G. Watson, et al. “The demand priority MAC protocol,” IEEE Network Magazine, Jan/Feb 1995

[Zaiane , 1998) Osmar R. Zaïane, Man Xin, Jiawei Han, “Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs”. In Proc. ADL'98 (Advances in Digital Libraries), Santa Barbara, April 1998.