1-1-2002

A framework for secure applications with QoS

Ajay Kanagala
University of Nevada, Las Vegas

A FRAMEWORK FOR SECURE APPLICATIONS WITH QoS

by

Ajay Kanagala

Bachelor of Engineering
University of Madras, India
1997

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science Degree
Department of Computer Science
Howard R Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
May 2002

UNLV

Thesis Approval
The Graduate College
University of Nevada, Las Vegas

July 3rd 2001

The Thesis prepared by

Ajay Kanagala

Entitled

A Framework for Secure Applications with QoS

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

Ajoy Kumar Datta

[illegible]

[illegible]

Dean of the Graduate College

[illegible]

Examination Committee Member

[illegible]

Examination Committee Member

[illegible]

Graduate College Faculty Representative

[illegible]

ABSTRACT

A Framework For Secure Applications With QoS

by

Ajay Kanagala

Dr Ajoy K Datta, Examination Committee Chair
Professor of Computer Science
University of Nevada, Las Vegas

In this thesis we propose a solution to support Quality of Service (QoS) network communication transactions modeled on differentiated services and economic principles We propose an enhanced user-level negotiation protocol where transactions are accomplished in a secure mode and where resource allocations are simple and fair With the availability of inexpensive network monitoring hardware, any issue dealing with network security becomes a priority Current networks must support multiple levels of service over a single network connection Differentiated levels of service create new resource management problems for network middleware Just as a first class passenger is willing to pay the premium for superior service, today's customers are ready to pay extra for preferential treatment of their network traffic We cannot have a separate airplane for each first class passenger, similarly we cannot have a separate network connection for each of our customers. Therefore, the bulk of network traffic must flow through shared lines. Resource management is a fundamental concept in the design of operating systems and is quickly becoming one of the most important concepts in network operating system design

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my heartfelt thanks to my advisors Dr Ajoy Datta and Mr. Joe Lombardo They have been a strong guiding force for this work and without their continuous inspiration and support much of this would not have seen the light of day I am lost at words here to express my deepest gratitude and appreciation for Mr Joe Lombardo's help in this thesis. Most or for that matter all the technical discussions that I have had relating to this thesis were with him He was a true guide, so to say who brought me on the track whenever I went astray and was there to dig into his technical expertise to work around problems He is a real taskmaster and a stickler for details and those roles really helped me get to where I am right now I am really thankful to him for this. Valuable insights stemming from his vast experience have helped me in bringing the problem closer to the solution. I would also like to thank Dr Kazem Taghva and Dr Wolfgang Bein for serving on my thesis committee I would be failing in my duty if I did not thank the University of Nevada Las Vegas, and especially the Department of Computer Science, for providing me with an excellent education to assist me in the work place. At this point I would like to reflect on my stay here in Las Vegas. It has been a home away from home, the faculty and staff of the university have made my education a pleasant and enriching one I would like to thank my parents for the tireless effort of instilling in me the values that a person should live by I pray to God for the blessings to help me in future endeavors.

CHAPTER I

INTRODUCTION

Current network frameworks are designed on the principle of best effort There is no guaranteed bandwidth for any particular application that might request secure and preferential treatment to its data As the number of active applications on the same network grows, the smaller is the part that is available for each individual application Indeed, data highways also have their traffic jams and rush hours Since there is no reserved bandwidth, it is difficult to use synchronous applications like video on the Internet and other local intranets Technically it is possible to reserve some bandwidth, but this is not always possible when lines and routers are shared with other organizations Taking this tremendous growth in traffic into account, increasing the bandwidth was seen as a possible solution to the problem But this problem was more than a simple capacity issue It was based on the type of traffic that was flowing through the network. The network traffic had not only increased in volume, it had also changed its nature. New breeds of Internet applications were taking over the network in the areas of multimedia, telephony, etc Each of them not only required sufficient bandwidth, but they are also involved other issues like strict timing requirements, multicasting capabilities etc. These applications require more than just the "best-effort" that the present IP delivers. In effect they require some **"value added information"** in the core of the network *[14]* The need for QoS arises because, until now, the best-effort

Network traffic, as the number of applications and users has increased the rules of the game need to change to accommodate ever-changing scenarios in the area of internetworking. In the past, the success of IP was its simplicity. Its fundamental design principle was derived from "end-to-end argument" which puts "value added information" at the ends of the network, i.e. the source and the destination network hosts thus leaving the core network "dumb". The job of the routers at the various intersection points was to check the destination IP address against a forwarding table to determine the next hop for the data gram. Basically the IP provided a "best-effort" service subject to delays and even data losses

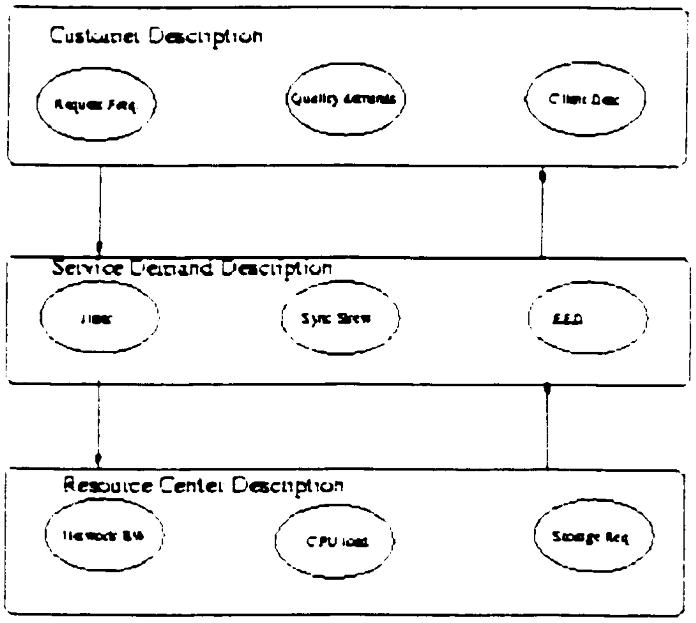

Figure 1-1 QoS Architecture

So QoS is the ability of the network element to have a level of assurance that its traffic and service requirements can be satisfied. This can be accomplished with the complete cooperation of all the layers in the network from top to bottom, as well as the various network elements from end-to-end. Any QoS assurances are only as good as the weakest link in the chain between the sender and the receiver *[14]*. QoS manages the bandwidth according to the application demands and network management setting. Since this involves sharing, it cannot provide everything with certainty. As a result, QoS with a service level requires resource allocation to individual data streams. This bandwidth allocation to an application in a resource reservation is no longer available for use by the "best-effort" applications. Since the bandwidth is a finite commodity, a priority for QoS designers has been to ensure that the best-effort traffic also receives some bandwidth and that it is not starved for resources once the reservations are made. QoS is the capability of the network to provide services suitable to the current network traffic and various underlying technologies. Basically, in plain terminology, it means that the network through this special feature is able to differentiate the network traffic, and based on this differentiation it is able to treat them differently. QoS in simple terms is a Policy, a policy that defines the rules that determines the specifics of how, when and where QoS is applied to network traffic *[17]*.

1.2 Bandwidth vs Prioritization

As noted above, the fundamental trade-off in quality of service has always been between bandwidth on the one hand, and traffic engineering or prioritization on the other. If there is unlimited bandwidth, we have perfect quality of service for all traffic flows. As bandwidth becomes more constrained, traffic engineering and prioritization become more important. In real networks, both are always relevant.

Predictably, therefore, two global QoS strategies - one linked to bandwidth, the other to prioritization - are currently prevalent: Resource Reservation, where network resources are apportioned according to an application's QoS request, and subject to bandwidth management policy, and Differentiated Services (prioritization), where network traffic is classified and network resources are apportioned according to bandwidth management policy criteria. To enable QoS in this latter case, network elements give preferential treatment to classifications identified as having more stringent requirements. These types of QoS can be applied to individual application "flows" or to flow aggregates. A "flow" is a stream of packets - an individual, unidirectional, data stream between two applications (sender and receiver), with the packets all uniquely identified by a "five-tuple" (they share the same transport protocol, source address, source port number, destination address, and destination port number). An "aggregate" here is two or more flows. Comparing two flows will reveal a common parameter, which could be from any of the five-tuple parameters, a label or priority number or authentication information. Different kinds of applications and network topologies dictate which type of quality of service is most appropriate for individual flows or aggregates. The most popular QoS protocols and algorithms are as follows.

1.2.1 ReSerVation Protocol (RSVP)

Provides the signaling to enable network resource reservation (Integrated Services). It is generally used on a per-flow basis, though it's also used to reserve resources for aggregates. RSVP makes a great effort to provide the closest thing to circuit emulation on IP networks. It's been used in videoconferencing for years. But in order for integrated services to work well, the router must maintain state information on each flow; the router determines what flows get what resources based on available capacity. RSVP doesn't have much scalability (processing must be done on every individual flow on the core Internet routers) and it lacks good policy control mechanisms.

1.2.2 Differentiated Services (DiffServ)

A coarse and simple way to categorize and prioritize network traffic (flow) aggregates Instead of maintaining individual flows on all routers, the flows are aggregated into a single flow that receives "treatment" (per class or per service state) Service classes are identified, and each packet is marked to identify it as belonging to a particular service, and then sent through the network Each router in the path must examine the packet's header to determine how it will be treated (detain it in favor of a high priority packet or let it go now)

A minimal DiffServ system demands "admission control" (The network has to be able to refuse customers when demand exceeds capacity), "packet scheduling" (a method for treating data from different customers as needed), "traffic classification" (sorting streams into "sub streams", each of which gets different treatment), and "policies" and "rules" (for allocating the network's resources)

1.2.3 Multi Protocol Labeling Switching (MPLS)

This provides bandwidth management for aggregates via network routing control, according to labels inside (encapsulating) packet headers MPLS routing establishes "fixed bandwidth pipes" similar to ATM or frame relay virtual circuits, but with MPLS it deals with "coarse levels" of QoS - it is not actually as fine grained as the virtual circuits and provisioning of, say, ATM. MPLS resides only in routers and is multi-protocol, so it can be used with network protocols other than IP (ATM, IPX, PPP, or frame relay) or even directly over the data-link layer

1.3 Resource Reservation

Aiming to provide more guaranteed availability of resources for applications, networks and operating systems are incorporating mechanisms for resource reservation. Reservation mechanisms have to keep track of the use of the limited set of resources provided by the system and receive requests from new users interested in using these resources *[24]*. New requests are subject to admission tests based on the usage of the resources and the guarantee levels that satisfy the user. Reservations are then accepted, if there are resources available, or rejected if not. The problem of allocating limited resources becomes even more complex if we consider that current computational systems are basically heterogeneous, subject to mobility and constant reconfiguration, but still have to provide a dependable and accurate service in a limited response time. In the area of computer networks, we can mention the development of ATM [ATM standards] as a significant advance toward the provision of QoS-constrained communication services. Aiming to provide similar behavior, but working at the logical network level, the IETF is proposing a new suite of protocols for the Internet [IETF standards]. The suite of protocols is based on IP version 6 (IPv6) in conjunction with a reservation protocol (RSVP) and a new transport protocol named RTP.

1.4 Differentiated Services

The Internet is powered by many different mechanisms. To give an idea of how things can be pictured, consider yourself as a mechanism, somewhat like an intelligent postman. Your basic task is to transmit and deliver the packets that pass through the service provider and the end user. This service provider has different services based on the price that the user is willing to pay. The most expensive services may require a quicker treatment inside the "post office", as well as along the route and also in the way it is transmitted. Such a network, which is needed to handle this task, should consist of a number of packet handling centers and paths between them [19]. The work involved at one of these centers when a packet is received is based on the information in the header. The decisions are easy because the rules are simple and are small in number.

1. The packet could be delivered immediately, before all the other packets that might have arrived earlier, since this packet has the priority.
2. The packets could be deposited in storage areas, waiting to be delivered.
3. The packets could be totally discarded because of the network conditions.

When Internet was purely an end-to-end service, all that was being done, was to read the address of the packet from the header and, based on the information in the routing table, choose the right direction for forwarding the packet. At that time, most of the packets received the same treatment, independent of the sender or the receiver [8].

1.5 Security as a Service

Another area that computer networks are concentrating their resources on is Security. Computer Networks, with their vast information resources, have traditionally been assumed to be secure. Gone are the days when networks were the provinces of large organizations, and operators in secure buildings then managed computers. Consequently, many common network utilities for remote terminal sessions, file transfer, remote printing, etc. were written based on this assumption [2]. Networks have grown to encompass the whole wide world and the traffic on these networks has also increased several fold. coupled with this is the technology advances on the hardware platform side. thus bringing on to the network. cheap computers which are being used to monitor the traffic thus re-enforcing the need to extend these network utilities to incorporate greater security features These small utilities are part of day-to-day work and the quality of service that they offer. as well as how they actually execute best. will vary with the machine's current capabilities This is to a large extent determined by the current hosting environment at any given time

The fact that these programs are used to transmit vital information across private networks is vulnerable to attack Security must be handled with the utmost care. in order for the utilities to perform to their fullest extent without compromising the quality This security scheme must be flexible and at the same time strong In a setting where denial of service is not tolerated. insufficient resources may dictate that a costly but safe security scheme be traded for a less costly and less safe one Moving on to the wireless world. the scenario also changes dramatically because in roaming environments appropriate mechanisms for solving disputes on roaming bills are not well supported *[5]*. In any security scheme. "trust" is difficult to manage Trust is a feeling and. as such. that feeling cannot be measured or evaluated algorithmically *[18]* Therefore. any decision that relies on trust must and will ultimately be made by a human This trust develops from a variety of sources It could be the experience that the user had with the system that increases the trust on the system or it is just that plain fact that it is difficult to explain This might seem trivial. but trust is an important resource and it is that resource that the system has to provide and gain from the user for the system to really succeed and gain market share This trust in particular is closely tied to the "quality" of any secure system and thus security forms a natural component in a system that is based on the notion of QoS There are many key issues that need to be examined when designing secure systems, such as the issue of providing predictable services in the face of varying operating conditions and user controls. Any system should be able to adapt to the changing scenarios

1 6 Contributions

In this thesis we propose a solution to support Quality of Service (QoS) network communication transactions modeled on differentiated services and economic principles We propose an enhanced user-level negotiation protocol where transactions are accomplished in a secure mode and where resource allocation is simple and fair With the availability of inexpensive network monitoring hardware, any issue that deals with network security becomes a top priority Current networks must support multiple levels of service over a single network connection Differentiated levels of service create new resource management problems for network middleware Just as a first class passenger is willing to pay the premium for superior service, today's customers are ready to pay extra for preferential treatment of their network traffic We cannot have a separate airplane for each first class passenger; similarly we cannot have a separate network connection for each of our customers Therefore, much of the bulk of network traffic must flow through shared lines. Resource management is a fundamental concept in the design of operating systems and is quickly becoming one of the most important concepts in network operating system design. It has been noted that authentication protocols are the basis of security of many distributed systems, and therefore an integral part of these protocols is the need to ensure that they function correctly *[23]*. So every effort was taken to ensure that the goal of authentication, stated simply and rather informally, is that two principals, be it two computers or two people, should be given the assurance beyond reasonable doubt that they are communicating with each other and not with intruders [7].

1.7 Outline of the thesis

The remainder of the thesis is organized as follows Chapter 2 discusses about network services, QoS metrics that are important in the present day networking scenarios and the various resource reservation and adaptation techniques Chapter 3 presents an enhanced user satisfaction and economic framework protocols Conclusion and future research directions are discussed in Chapter 4

CHAPTER 2

BACKGROUND INFORMATION

The promise of networking is sharing networked resources among many users and applications for greater productivity and competitive advantage. Quality of services (QoS) enables complex networks to control and predictably service a variety of applications. Every network needs QoS for optimum efficiency, whether it's a small business, large enterprise, or service provider. This chapter gives an overview to network services, QoS metrics that are involved and the different pricing models that are used.

2.1 Network Services

As described earlier, the dependence on networks by the major corporations has increased by leaps and bounds. Along with this growth, the network itself has evolved over time. Many of the applications such as email, multimedia, and web-based products have contributed to this immensely. As a result, the traffic on the network has also undergone a dramatic shift in content. Along with this the comes the driving need for more complex handling of the data, at the same time making sure that the system works without any hitches because these days they have become mission critical equipment. Each time a new application is deployed on the network to meet a business need, it requires a host of supporting network services *[16]*. With this present scenario involving numerous e-business applications, where the network traffic is rather

of the information that it is carrying, much responsibility falls on the network services to not only provide reliable security to the data transmitted, but also to provide preferential treatment to the data being transmitted because it could, in turn, translate into more customers and more revenue streams The various applications that are created to solve a particular problem in emerging network services create new ones So new applications that are created must also be updated with the changing face of the network This situation results in inordinate amounts of time and money and takes longer time for deployment, thus making it essential to work toward a framework that would not only take advantage of the QoS and other services, but also help in the smooth deployment of these products on the network When analyzing the development of such a framework, let us look at the various parameters that actually need the attention

2.2 QoS Metric

QoS is basically defined as a set of perceivable attributes that are clearly explained in a language that is simple using parameters that are both subjective and objective *[1]* When we say that the parameters are subjective, we are quantifying the opinions expressed by end users When we say that they are objective, we mean that the parameters are related to one particular service that are measurable and also that can be easily verified

This specification and enforcement of QoS in multimedia systems presents many challenges Any application involves parameters, such as size, frame rate, startup delay, reliability and end-to-end delay, which are part of QoS parameters Many of these parameters may also include other subjective parameters, like the degree of importance of these above parameters, cost factor in providing this quality etc. Apart from these application-determined factors, there are also network related factors that include bandwidth, delay, loss, jitter and loss rate. System side parameters include the CPU usage, buffering of data, and memory related parameters.

The basic components that are measurable in order to say that a particular system is providing that quality that the system is supposed to provide are **indent delay, jitter, bandwidth and reliability** *[10]*.

Definition 2.2.1 (Delay) *is the elapsed time for a packet to be passed from the sender, through the network, to the receiver. The higher the delay greater is the stress that is placed on the transport protocol to operate efficiently. For the TCP protocol, higher levels of delay imply greater amounts of data held "in transit" in the network, which in turn places stress on the counters and timers associated with the protocol. It should also be noted that TCP is a "self-clocking" protocol, where the sender's transmission rate is dynamically adjusted to the flow of signal information coming back from the receiver, via the reverse direction acknowledgments (ACK's), which notify the sender of successful reception. The greater the delay between sender and receiver, the more insensitive the feedback loop becomes, and therefore the protocol becomes more insensitive to short thermodynamic changes in network load. For interactive voice and video applications, the introduction of delay causes the system to appear unresponsive.*

Definition 2.2.2 (Jitter) *is the variation in end-to-end transit delay (in mathematical terms, it is measured as the absolute value of the first differential of the sequence of* *individual delay measurements). High levels of jitter cause the TCP protocol to make very conservative estimates of round trip time (RTT), causing the protocol to operate inefficiently when it reverts to timeouts to reestablish a data flow. High levels of jitter in UDP-based applications are unacceptable in situations where the application is real-time based, such as an audio or video signal. In such cases, jitter causes the signal to be distorted, which in turn can only be rectified by increasing the receiver's reassembly playback queue, which effects the delay of the signal, making interactive sessions very cumbersome to maintain.*

Definition 2.2.3 (Bandwidth) *is the maximal data transfer rate that can be sustained between two end points. It should be noted that this is limited not only by the physical infrastructure of the traffic path within the transit networks, which provides an upper bound to available bandwidth, but is also limited by the number of other flows which share common components of this selected end-to-end path.*

Definition 2.2.4 (Reliability) *is commonly conceived as a property of the transmission system, and in this context it can be thought of as the average error rate of the medium. Reliability can also be a byproduct of the switching system, in that a poorly configured or poorly performing switching system can alter the order of packets in transit, delivering packets to the receiver in a different order than that of the original transmission by the sender, or even dropping packets through transient routing loops. Unreliable or error-prone network transit paths can also cause retransmission of the lost packets. TCP cannot distinguish between loss due to packet corruption and loss due* *to congestion and packet loss automatically invokes the same congestion avoidance behavior response from the sender for loss both due to congestion and corruption, causing the sender's transmit rates to be reduced by invoking congestion avoidance algorithms, even though no congestion may have been experienced by the network. In the case of UDP-based voice and video applications, unreliability causes induced distortion in the original analog signal at the receiver's end.*

2.3 Unified Metric for Video QoS

Basically, from the empirical and the workload model that was obtained, it can be inferred that there are two kinds of factors that a QoS metric must represent *[3]* Resource consumption and user satisfaction factors represent the primary metrics of QoS When we talk about resource consumption, we mean the ways to measure what has been incurred by the utilization of the source in the system, which includes but is not limited to the CPU, memory and other network related parameters On the other hand, the user satisfaction, help us to quantify the QoS guarantees that the system can deliver with the desired response quality *[3]*.

In order to determine the achieved quality and the deviation between the actual response and the perceived one, the application QoS related parameters like jitter, frame rate, frame width, color resolution etc are a good measure *[11]* Along with these above parameters, a new metric system that looks at security as a service provided at cost, which in turn helps us to determine if the system under study meets some Quality that the user might request. In order to analyze the system which includes the parameters that define the objectives of QoS, we adopt the following approach User satisfaction

(*US*) is quantified, by this we mean that we assign weights to User satisfaction as well as to the cost incurred as a result of resource consumption (*RC*) when trying to provide the Quality of service that was agreed upon. Using this we can develop a function that consists of the mixture of these objectives that are weighted.

Let the QoS metric that represents service to the user *i* be denoted as M_i

$$M_i \quad \alpha \quad \left(\frac{W_{us} * US_i}{W_{rc} * RC_i} \right) \qquad (1)$$

Where W_{us} denotes the weight of the User Satisfaction component and W_{rc} represents the weight of the Resource Consumed component.

2.4 User Satisfaction

User satisfaction is measured by a number of factors like success, jitter, cost, end-to-end delay, synchronization skew, and startup latency and loss rate.

$$US \quad \alpha \quad {}^{Succ} f(j, c, e, sk, l, lr) \qquad (2)$$

Where success of the system to provide the QoS (*Succ*), jitter (*j*), cost of the operation (*c*), end-to-end delay (*e*), skew (*sk*), latency (*l*) and loss rate (*lr*).

"*Succ*" i.e. Success of a request is defined differently for the various QoS classes. When taking into consideration a deterministic service, we know clearly when the service has been accomplished, because it meets tougher performance requirements, which is an improvement over best-effort traffic management. Let us consider the case where, in order for a service to be provided, there is some sort of admission control In this situation we can easily determine that it was a success if the request has been admitted On the other hand, in a best-effort situation this is not the case since every request is admitted So here the situation changes and we have to see if the request is run to completion in order to evaluate it success rate We are considering the situation where it is assumed that there is some sort of admission policy and evaluating the QoS from the point after the admission process to be the successful completion of the request So we are safe to assume that *Succ* is a constant *[1]*

Analyzing each of these parameters in order to determine and understand the overall impact on User Satisfaction We do know that startup latency does have an impact on user satisfaction It is basically more at the beginning and not a continuous one, i e it is measured only once per session and does not impact the other delays like frame delay etc In order to simplify the equation 2, the parameters loss rate (***lr***) and end-to-end delay (***e***) can be subsumed by the jitter parameter (***j***)

Therefore the simplified equation is

$$US \ \alpha \ \frac{1}{f(j,c,sk)} \qquad (3)$$

Service cost is an important parameter Pricing is one of the key issues in any Differentiated Services model The issues that dominate are how pricing schemes can help maximize User Satisfaction based on ratio of user benefit to service cost When dealing with the following, three options are available

- Influence the user behavior in a manner where the network resources are not wasted

- Provide the service based on the price irrespective of how that the user is using service
- Pricing as a tool to control traffic

2.5 Pricing as a tool to control traffic

The major elements in play include

- Bandwidth
- Quality
- Availability
- Price

When one of the factors is presented as a function of another factor, and the remaining two factors remain a constant, various cases arise. Those that are important in this research include

- Price of Bandwidth
- Price of Quality
- Price of Availability

The standard functions of pricing in a reservation-oriented network include Price of Bandwidth and Price of Quality. The price of the connection is calculated as a function of bandwidth and quality. However the third factor, Price of Availability, also comes into play where there is the issue of availability and when there is demand. The price is higher during busy hours when compared to the lean period. So price could also be said to depend on Availability.

A simple but realistic mathematical model may be helpful The graphs that are shown in this document are all derived from the following simple formula

$$Log\,(P) = CB * log\,(B) + CQ * log\,(Q) + CA * log\,(1 - A) + CP \qquad (4)$$

P Price, B Bandwidth, A Availability, Q Quality

Here the Logarithmic form makes each of the parameters present a more systematic effect

2.5.1 Price of Bandwidth

Figure 4.2.1 depicts one of the fundamental problems associated with pricing network services related to the relationship between bandwidth and price A linear relationship is not practical because bandwidths differ significantly According to the example, if the price of a 56kbps connection were 595, the price for a 1.5Mbps connection would be 1,795, and the price of a 45Mbps connection would be 54,000 In this case the relation is far from linear *[3]*

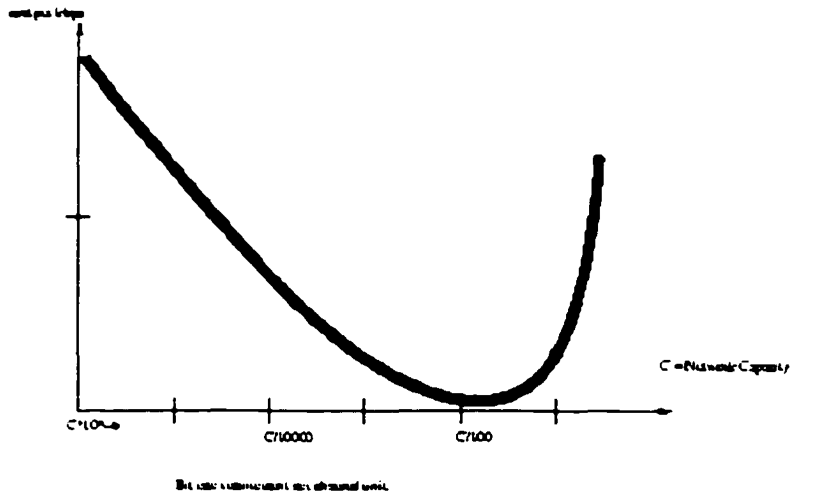

Figure 2-1: An Approximate relationship between charged unit and costs.[3]

These figures support the model that was discussed earlier, where the parameter CB is significantly smaller than 1. Assume that the average bit rate required by a customer is R_{ave}, and the total cost of providing this service for all the customers is $CT * R_{ave}$. So when the average bandwidth requirement is increased by 15 fold in a way that all the other factors are constant, then the total cost of the service provided is evidently more than $CT * R_{ave}$, but most probably significantly less than $15 * CT * R_{ave}$

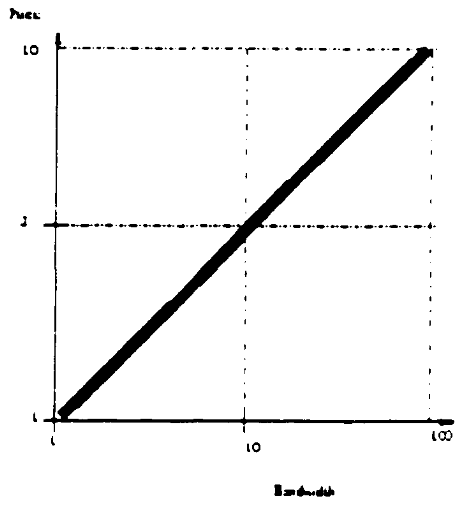

Figure 2-2 A tentative relationship between bandwidth and price [3]

When the dimensioning problem of any given network is considered, a totally different conclusion may be drawn. Basically, one connection with a constant bit rate of 1Mbps consumes the same volume of resources as 10 connections with a constant bit rate of 100Kbps. If numerous small connections are made, the situation is totally different. Management costs will induce the major costs. So here are two opposing phenomena: The total cost tends to be high if the charged unit is very small and the statistical multiplication tends to deteriorate if the number of independent units is small.

Coming to reality, the situation is not static, but highly dynamic. The capacity of the network is updated constantly in relationship to demand. One customer seldom dominates high capacity networks so the effect of statistical multiplexing considerably deteriorates. Therefore, despite the significant opposing arguments, it is possible to tentatively apply the formula (4) with CB smaller than 1. A value of 0.6 is used to generate the figure 2.2 as well as other figures related to pricing.

2.5.2 Price of Quality

From a general and also in a particular viewpoint of pricing, Quality is an ambiguous term. Here the Quality is evaluated on a particular quality aspect, say the delay variation or the skew rate. In a system where the data is transmitted in video, delay play an important aspect to determine the quality of that video transmission. In any real time networks there will be other control mechanisms as well as other additional management actions. It is justified to suppose that price of real-time service should be somewhat higher if the other parameters are constant *[3]*

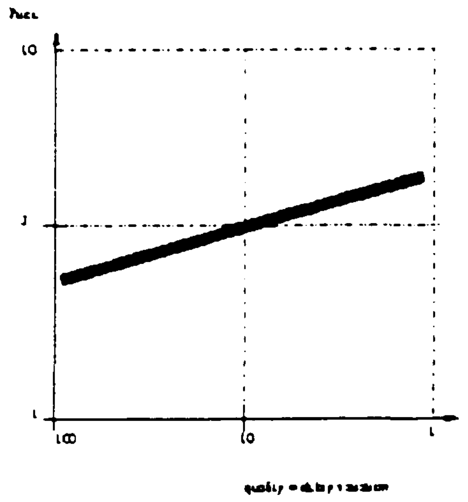

Figure 2-3: A tentative relationship between quality and price.[3]

The main point here is that real-time support makes traffic control complicated It is nearly impossible to give any clear rule for the price difference There are instances where the traffic flow sent by the user is constant, it is not clear whether there is any significant difference from a statistical-multiplexing or traffic control viewpoint as to whether a flow uses real-time or data service */3/*

2 5 3 Price of Availability

When we assume that the availability is a common characteristic of the entire network then the relationship between price and availability may seem a bit artificial */3/* One possible interpretation is that availability is related to availability of a service at a certain price at different times When the availability is high, the service is available even during the busiest times of the year

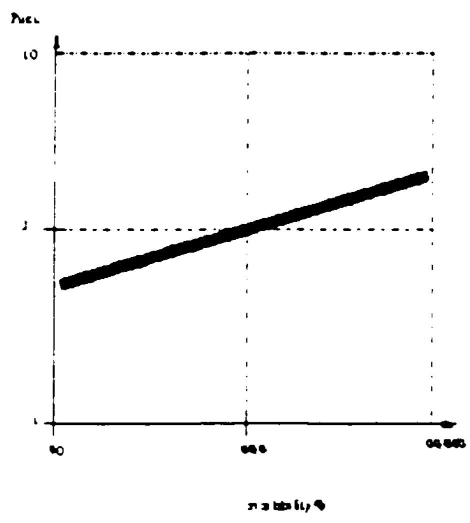

Figure 2-4 A tentative relationship between availability and price [3]

An intermediate availability means that the service is available at a price on a typical busy hour and finally, low availability means that the price is valid for idle hours

2.6 Resource Consumption (RC)

Looking at equation 1. we notice that the metric proposed also depends on resource consumption Basically. there are a couple of factors that go into determining the resource overhead. i e . CPU utilization (CPU). network bandwidth (B^{nw}). buffer utilization (B^{uf}) and storage bandwidth (B^{st}) overhead. Based on the design of the system. different benefit functions can be used to indicate the criticality of the resource *[1]*

$$RC = F_{rc}\left(CPU, B^{nw}, B^{uf}, B^{st}\right)$$

An interesting observation is that the resource cost takes into consideration r the current set of available resources As the resources are used by the current services. they become more critical and resource cost becomes more dependent on the fraction of the available resources utilized. When the resources are low. the admission control mechanism must allocate the resources to the high priority tasks based on differentiated services.

Resource consumption of a server S_i to a request R_j is characterized as a ratio that captures the utilization of resources. similar to the load factor measure *[9]*.

$$\mathrm{RC}\,(S_i, R_j)\ \alpha\ \left(\frac{CPU_i}{CPU_j}, \frac{B^{nw}{}_i}{B^{nw}{}_j}, \frac{B^{uf}{}_i}{B^{uf}{}_j}, \frac{B^{st}{}_i}{B^{st}{}_j}\right)$$

Different requests for service have different resource requirements and the Resource Consumption (RC) factor will vary from one request to another Lower the RC value, greater is the capacity of the system to service additional requests The resource cost component of the QoS metric is essentially the cost of the bottleneck resource, since this constraining resource measures the degree of the QoS delivered

2.7 Resource Adaptation

One of the new trends in the research in the area of resource reservation protocols is resource adaptation *[25]* Some authors consider resource reservation as a strong guaranteed method to deliver a specified quality of service to an application during its entire lifetime Another school of thought proposes the development of adaptive applications to deal with the changes in resource availability during the provision of service A third idea based on resource adaptation, which mixes both approaches mentioned previously, has been considered as a viable and necessary alternative to either

Several drawbacks appear in efforts to provide guaranteed resource reservation services for the following reason *[24]*

- The physical structure of the whole computer system may change due to hardware reconfiguration and computer mobility, and changes in the distribution of resources may become necessary;
- Monitoring services may detect that the QoS requested by an application is not being achieved with the resources allocated for it, then decide to allocate new resources.

- The system may reclaim resources that are not being utilized or that are necessary for more important activities

On the other hand, adaptation is very limited for services with strong requirements, and does not solve the problems faced by this category of application when using the best-effort systems currently available Resource reservation combined with adaptation entails a more relaxed approach for providing QoS to applications According to this new approach, resources can be seen by applications as guaranteed during some time, but their availability can vary over long periods Applications are responsible for estimating their initial resource requirements and for negotiating their reservation with resource providers During run time the application has to be able to adapt its behavior based on feedback received from the system QoS mechanisms have to be aware of the possibility of resource adaptation, making it transparent to the application whenever possible When the agreed QoS is not reachable with the resources available, the application has to be informed that the agreed QoS has to be renegotiated Applications have to be ready to handle this kind of situation without severe disruptions in the service being provided to the user More specifically, applications holding resources subject to changes in their availability because of resource adaptation have to be able to degrade gracefully when it occurs

2.8 Application Adaptation

Providing support to both the applications that require *QoS Assurance* and that are able to perform *QoS Adaptation* could solve most of the resource allocation problems *[15]*. To elaborate in more detail, QoS Assured applications are those applications that are clear in the resource requirements. These applications do not change the resource requirements over time and are basically assured of the resources that were requested and these resources are at guaranteed levels On the other hand, the QoS Adaptive applications are those that are structured in different modes of operations in multiple levels. These levels are used to adapt to the changing needs as the applications approach different scenarios The applications mode of operation is adapted based on the current resource allocation In this class of QoS Adaptive application, there are basically two classes that are distinct They are user-triggered adaptation and cross-resource adaptation *[15]*. Currently a major chunk of the applications adapt in reaction to variation in just a single parameter or resource Most of them try the balancing act by reducing the quality in one dimension to increase a more important aspect of the product on another dimension Moreover, a greater level is attained if the applications handling the adaptation could be done over multiple resources But this will result in choices that are difficult to balance and complicated to implement and could eventually defeat the very essence of the concept

Research in the QoS area has been mostly concentrated in analyzing the network resource consumption and service demands based on the service parameters Consider here the example where the design of the architecture to provide end-to-end QoS architectures is illustrated *[21]* There has been some work in the areas of negotiation for the resources and reservation of these resources for the distributed multimedia applications. Negotiation of QoS for an application session is a balancing process between the QoS specified by a client, the resource capabilities of the distributed system, and the functional capabilities of the distributed application Negotiation requires application level QoS descriptions and an end-to-end view spanning the whole distributed application. XNRP, introduced in this thesis, is a protocol meeting these requirements. XNRP performs negotiation based on client specified QoS value ranges and is independent of application level QoS semantics. It allows QoS negotiation and resource reservation in three phases and supports arbitrarily interconnected flow graphs of application components *[22]*. Congestion Management is a new area where the methods are present in routers to support various signaling protocols and actually provide different classes of service. These are usually implemented at the core routers and involve creating different queues for the different classes of traffic, an algorithm that is used for classifying the packets to the various queues, and a scheduler that handles the actual transmission of these packets to the various queues *[20]*. Along with this is the use of scripts to handle the QoS specifications when it comes to Resource scheduling. Since most of the multimedia presentations are representative of a class of applications that read stored data according to a real-time script. The problem in supporting real-time scripts is to determine their presentation and also a way to identify the resources that satisfy them. Coupled with this problem is the actual choosing of a policy that is allocating the resources among competing presentations *[4]*.

CHAPTER 3

ENHANCEMENT AND ORIGINAL WORK

In order to achieve the desired system performance, QoS mechanisms must guarantee the availability of the shared resources needed to perform the services requested by users The concept of resource reservation provides the predictable system behavior necessary for applications with quality of service (QoS) constraints Coupled with this and the concept of Middleware, where middleware is the term that is used to describe a broad array of tools and data that help applications to use networked resources and services Some tools, such as authentication and directories, are in all categorizations Other services, such as co scheduling of networked resources, secure multicast, and object brokering and messaging, are the major middleware interests of particular communities, such as scientific researchers or business systems vendors One definition that reflects this breadth of meaning is, ***"Middleware is the intersection of the stuff that network engineers don't want to do with the stuff that applications developers don't want to do." [26]***

Middleware has emerged as a critical second level of the enterprise IT infrastructure, between the network and application levels The need for middleware stems from the increasing growth in the number of applications, in the customizations within those applications, and in the number of locations in our environments These and other factors now require that a set of core data and services be moved from their multiple instance into a centralized institutional offering This central provision of service eases application development. increases robustness. assists data management. and provides overall operating efficiencies

The first step when creating the framework where new features are being added will be analyzing user satisfaction Into this framework. security is being added and this is an important feature. which not only increases the confidence of the user but also has the processing overhead of any requests since it involves a lot of number crunching when the authentication and encryption of the data that passes through the application is being processed

3 1 Enhanced User Satisfaction

User satisfaction is measured by a number of factors like success. jitter. cost. end-to-end delay. synchronization skew. startup latency. loss rate and security

$$US \quad \alpha \quad ^{Succ} f(j,c,e,sk,l,lr,\sec) \qquad (5)$$

where Success of the system to provide the QoS *(Succ)*. jitter *(j)*. cost of the operation (*c*). end-to-end delay (*e*). skew (*sk*). latency (*l*) and loss rate *(lr)*.

"*Succ*" i e Success of a request is defined differently for the various QoS classes When taking into consideration a deterministic service then we know clearly when the service has been accomplished. because it meets tougher performance requirements. which is an improvement over best-effort traffic management Let us consider the case where in order for a service to be provided there is some sort of admission control In this situation we can easily determine that it was a success if the request has be admitted on the other hand in a best-effort situation this is not the case since every request is admitted so here the situation changes and we have to see if the request is run till completion in order to evaluate it success rate We here are considering the situation where it is assumed that there is some sort of admission policy and evaluating the QoS from the point after the admission process to he successful completion of the request So we are safe to assume that *Succ* is a constant */1/*

Analyzing each of these parameters in order to determine the overall impact on User Satisfaction We do know that startup latency does have an impact on user satisfaction It is basically more at the beginning at not a continuous one, i e it is measured only once per session and does not impact the other delays like frame delay etc In order to simplify the equation 2, the parameters loss rate (***lr***) and end-to-end delay (***e***) can be subsumed by the jitter parameter (*j*)

Therefore the modified equation is

$$US \;\alpha\; \frac{1}{f(j,c,sk,\mathbf{sec})} \qquad (6)$$

3 2 Economic Framework Protocols

Based on the discussion in the earlier sections, an enhanced design to QoS architecture based on Economic Principles is proposed Looking into the Service Manager side there are four modules that handle delivery of service, they include:

- Tradeoffs Protocol, Rewards Protocol, Security Module
- The Service Monitor monitors the system condition and the user behavior
- Negotiation Protocols facilitate the user and the service manager to a negotiation mechanism for price/quality compromise

3.2.1 Tradeoffs and Rewards Protocol

Trading involves user-level negotiation protocols that involve the concept of rewarding for accepting resource tradeoffs and also for maintaining the behavior that was accepted throughout the session. These Rewards and Penalties are basically a mechanism to encourage and also police good behavior from the users. When a user violates the terms of the negotiated contract various penalties are imposed. The definition of good behavior depends on the degree of freedom the user has to alter the services provided once it has been started. There are basically two possible approaches called the static services

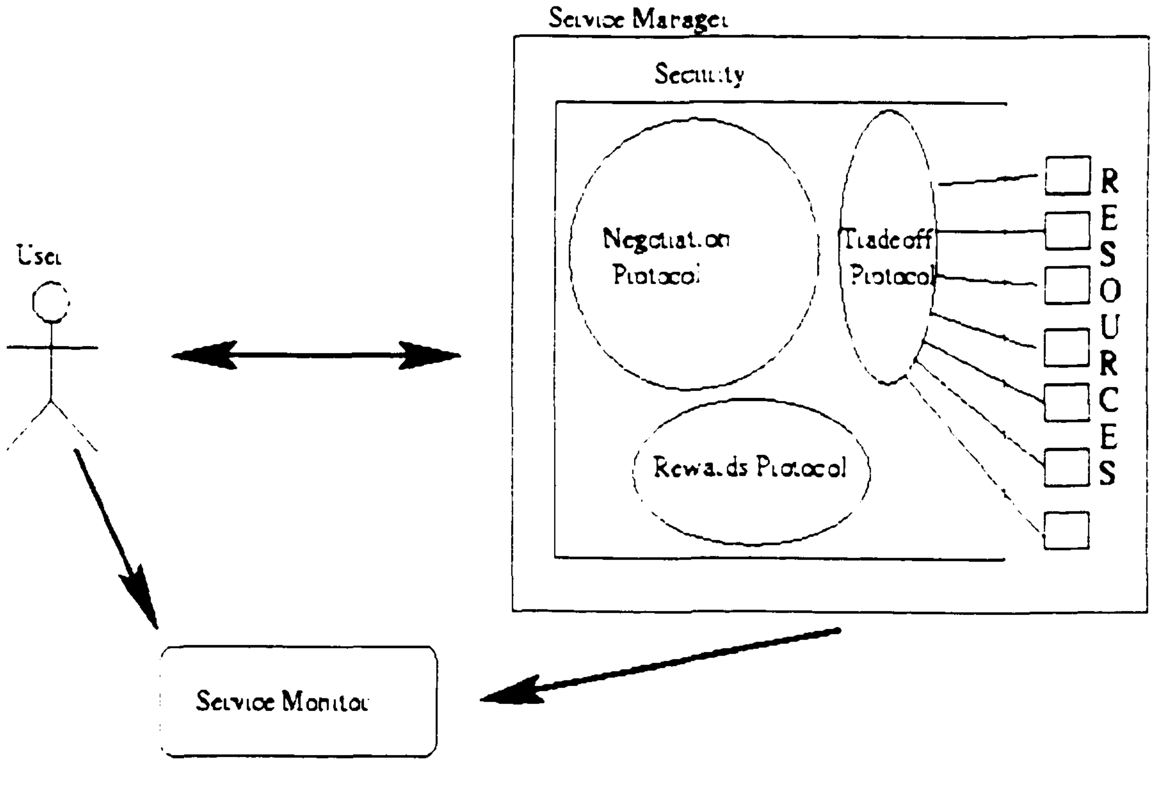

Figure 3-1: Protocol Architecture

Negotiation and dynamic service alteration [1]. As the names clearly define them, a static service negotiation is one where the user and the service manager negotiate the terms of the contract at the start of the session. Rewards and penalties are assessed at the beginning of the session. Therefore, once the session has started, the user has no scope to alter the service except through re-negotiation of the contract with the service manager. On the other hand the Dynamic service alteration, the user and the service manager negotiate the terms of the session. The user is expected to adhere to these terms of the contract, but may change conditions if desired and receive a penalty. In this the rewards and penalties are assessed at the end of the session.

3.2.2 Negotiation Protocol

The Negotiation Protocol interfaces with the User to execute the negotiation protocols and decides on the pricing and the resource usage for that session by that user. The user level protocol that handles resource-price negotiation can be initiated either by the user or by the service manager. The user initiated one could be those who are interested in quality at a cost whereas the service manager could be one where the system is trying to evaluate other service alternatives, so that more requests could be processed.

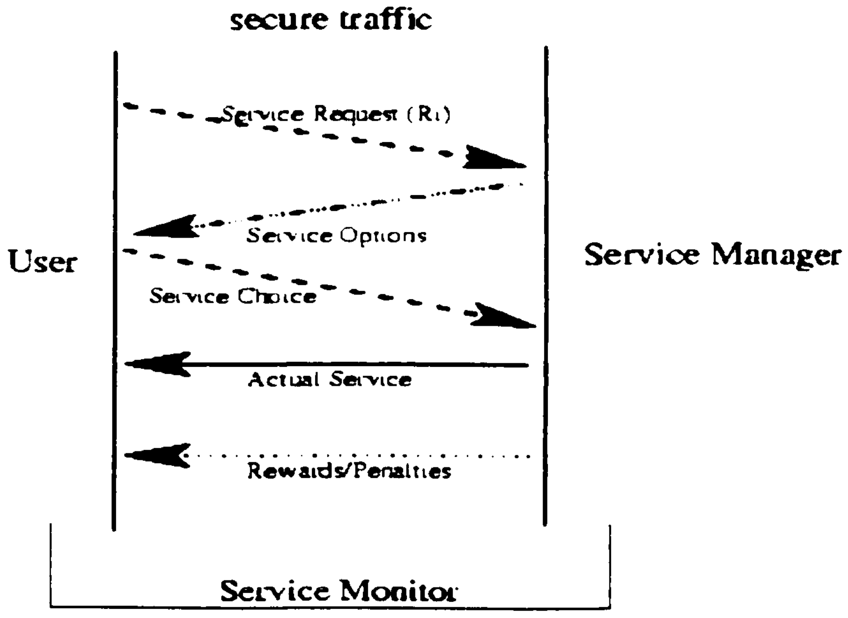

Figure 3-2: Control Flow Diagram.

User level Negotiation Protocol

- ❑ ***User send a request to Service Manager(SM)***
- ❑ ***Service Manager determines.........***
    - ➢ ***Can this request be serviced fully??***
        - ◆ ***If YES, send in various service options that can be provided.***
        - ◆ ***LOCK resources for the best option for a predefined period of time.***
        - ◆ ***NO, send message back that the requested service cannot be met.***
        - ◆ ***ASK for change in service requirements.***
    - ➢ ***Get Resource Tradeoffs.***
    - ➢ ***Call the Rewards Program.***
- ❑ ***User returns with the choice of service within the time frame otherwise the resources are released.***
    - ➢ ***Service is provided based on the selected option***
    - ➢ ***Resources that are not used are released for other services.***
- ❑ ***Monitor service delivery to handle violations and deadlocks.***
    - ➢ ***Slap penalties on the violation of the agreement.***
- ❑ ***Constantly re-negotiate service agreement to handle the optimal usage of resources.***

The user sends in a service request to the Service Manager(SM) stating the intention of performing set task with the resources that are available. Looking at the request the SM determines whether there are enough resources available to complete the task and also determines if there are any other critical applications that have requested for resources. Based on this, the SM will allocate the necessary resources to a particular request. This allocation of resources is not physically done, it is more like and intention that based on the resources that are currently available in the system that the SM checks to see if the tasks that was specified by the user will be handled without any hitches Once the service manager is convinced that such a task can be accomplished. the SM will then acknowledge the user of its intention to provide the resources and then will send to the user a set of options and the costs associated with each of these options and lock the one that might be best for that particular request These resources are locked for a predefined period of time to take into consideration the cases where the connection could have been lost or the user has changed the request Once the user sends in the request based on the on the was selected by the SM. then the resources are allocated to this user for performing the necessary tasks. However the user can return request additional resources or coming up with a totally new request where in the SM could reject the request or could go ahead and evaluate the current resource requirements and based on the rules that are available. present the user with a fresh set of choices This process could go on till the user and the SM comes to an agreement If the SM determines that it does not have the requested resources. it would then send in a request for a modified request based on the availability

CHAPTER 4

CONCLUSION

In this thesis we have proposed a solution to support Quality of Service (QoS) network communication transactions modeled on differentiated services and economic principles. The user level protocol is an enhanced one where the transactions are accomplished in a secure mode. Here security is in itself a service that is being provided. In order to accomplish that all the communication between the user and the service manager has to be routed through a secure layer. This protocol is simple and fair, because the resources are allocated based on the arrival of the request. The economic relationship between the user and the resource consumption are constantly updated to meet the changing face of requirements with incentives wherever needed.

Lot more can be done in this area with respect to bringing in more intelligence to the various areas an application might pass through as it goes through the network. Similarly, upcoming technology changes such as IPv6 will likely make overall network management easier by including QOS as well as IP Security in the core of the TCP/IP protocol. Bandwidth management will be a critical issue next year for companies adopting VOIP (voice over IP) or other technologies that require timely delivery of data. Managers of traditional data-only networks should leave QOS to those on the bleeding edge of VOIP and focus, instead, on providing sufficient capacity to handle the heaviest consistent loads of data traffic.

APPENDIX

PROGRAM LISTING

The LoginManager is the class that handles the login details of the client. This class encapsulates the client details like the HOSTNAME, USER NAME and the PASSWORD. This is the information that is used to connect to the host that is specified. The other classes can also use this information when they want to get information as to who are the clients that are logged in at that moment.

```
/**
 * Title:        LoginManager
 * Description:  Contains the login details of the user
 * Copyright:    Copyright (c) 2001
 * Organization: UNLV - Computer Science Dept
 */

import java.io.*;

public class LoginManager implements Serializable
{
  public String hostName;
  public String userName;
  public String passWord;

  public LoginManager( String hn,String un, String pwd ) {
    hostName = hn;
    userName = un;
    passWord = pwd;
  }

  public String getHostName( ) {
    return( hostName );
  }

  public String getUserName( ) {
    return( userName );
  }

  public String getPassword( ) {
    return( passWord );
  }
}
```

The ServerConnectUI is the class that handles the creation of the login dialog for the server. This code does not have any dependency with the type of IDE is used. All the components and the Layouts are hand generated thus enabling easy maintenance of code as we go on adding other functionality.

```
/**
 * Title:        ServerConnectUI
 * Description:  Creates the Server Connect UI dialog
 * Copyright:    Copyright (c) 2001
 * Organization: UNLV - Computer Science Dept
 */


public class ServerConnectUI extends JFrame {
```

```
//Construct the frame
public ServerConnectUI() {
 enableEvents(AWTEvent.WINDOW_EVENT_MASK);
 try {
   key     = "java.io.tmpdir";
   val     = System.getProperty(key);
   tmpPath = val + "caliperSerTmp";
   Init();
 }
 catch(Exception e) {
  e.printStackTrace();
 }
}

public boolean isServerConnected() {
  return isServerConnected;
}

//Component initialization
private void Init() throws Exception  {
   connectDialogFrame = new JFrame("Server Connect Dialog");
         Dimension screenSize = connectDialogFrame.getToolkit().getScreenSize();

   //This includes the text Filed and the label
   editPane = new JPanel();
   editPane.add(Box.createHorizontalGlue());

   hostNameLabel = new JLabel("Hostname ");
   editPane.add(hostNameBox);

   userLabel = new JLabel("User Name ");
   editPane.add(userNameBox);

   passwordLabel = new JLabel("Password ");
   editPane.add(passwordBox);

   editPane.setBorder(BorderFactory.createEmptyBorder(10,10,10,10));
   contentPane.add(editPane, BorderLayout.CENTER);

   buttonPane = new JPanel();

   checkBox = new JCheckBox("Save Details ");
   checkBox.setSelected(isSaveSelected);
   checkBox.addItemListener(new ItemListener() {
     public void itemStateChanged(ItemEvent e) {
       if (e.getStateChange() == ItemEvent.SELECTED) {
         isSaveSelected = true;
       } else {
         isSaveSelected = false;
       }
     }
   });
   editPane.add(checkBox);

   okButton = new JButton("OK");
   buttonPane.add(okButton);
   buttonPane.add(Box.createRigidArea(new Dimension(10, 0)));
   okButton.addActionListener(new ActionListener() {
     public void actionPerformed(ActionEvent e) {
       try {
         FileOutputStream f = new FileOutputStream(tmpPath);
         ObjectOutputStream  s  =  new  ObjectOutputStream(f);
         LoginManager lm = null;
```

```
            if(isSaveSelected) {
                    lm = new LoginManager( hostNameBox.getText(),
                                userNameBox.getText(),
                                passwordBox.getText() );
            }
            else {
                    lm = new LoginManager("","","");
            }
            s.writeObject( lm );
                s.flush( );
            }
        catch( IOException ex ) {
            JFrame frame = null;
            JOptionPane.showMessageDialog( frame, ex );
        }
        isServerConnected = remoteRequester.connect(
                            hostNameBox.getText( ),
                            userNameBox.getText( ),
                            passwordBox.getText( ));
        connectDialogFrame.dispose( );
      }
    } );


    cancelButton = new JButton( "Cancel" );
    buttonPane.add( cancelButton );
    buttonPane.add( Box.createRigidArea( new Dimension( 10, 0 )));
    cancelButton.addActionListener( new ActionListener( ) {
      public void actionPerformed( ActionEvent e ) {
        connectDialogFrame.dispose( );
      }
    } );

    contentPane.add( buttonPane, BorderLayout.SOUTH );
    connectDialogFrame.pack( );
    connectDialogFrame.setVisible( true );

    setUpGUIWithData( );
    remoteRequester = WorkerRepository.getRemoteRequester( );
  }

  private void setUpGUIWithData( ) throws Exception {
    try {
      FileInputStream in = new FileInputStream( tmpPath );
      ObjectInputStream si = new ObjectInputStream( in );
      LoginManager loginManager = ( LoginManager )si.readObject( );
      hostNameBox.setText( loginManager.getHostName( ));
      userNameBox.setText( loginManager.getUserName( ));
      passwordBox.setText( loginManager.getPassword( ));
    }
    catch( IOException ioe ) {
      System.out.println( ioe );
    }
  }
}

/**
 * Title:        RemoteRequester
 * Description:  This is the class that connects to the server
 * Copyright:    Copyright (c) 2001
 * Organization: UNLV - Computer Science Dept
 */

public class RemoteRequester {
```

```
public RemoteRequester( ) {
}

/**
 * Start up and connect to the server via the rexec daemon
 */
public boolean connect(String hostName, String userName, String userPassword)
{
   long status = 0;
   String caliperReadyLine;

   if (WorkerRepository.getCommandArgs( ).serverCommand != null)
      serverCommand = WorkerRepository.getCommandArgs( ).serverCommand;
   else
      serverCommand = SERVER_CMD;

   try {
      loginSocket = new Socket(hostName, 512 /* rexec daemon */);
      out = new DataOutputStream(loginSocket.getOutputStream( ));
      in = new BufferedReader(new InputStreamReader(loginSocket.getInputStream( )));

      // rexec login protocol
      out.write(0);
      out.write(userName.getBytes( ), 0, userName.length( ));
      out.write(0);
      out.write(userPassword.getBytes( ), 0, userPassword.length( ));
      out.write(0);
      out.write(serverCommand.getBytes( ), 0, serverCommand.length( ));
      out.write(0);
      out.flush( );
      int b = in.read( ); // Read rexec confirmation byte (zero)
      if( b != 0 ) {
         GUIDispatcher.postErrorDialog("Login failed: " + in.readLine( ));
         return false;
      }
      caliperReadyLine = getServerResponseLine( );
      if (getStatus(caliperReadyLine) != REPLY_CONNECTION_COMPLETE) {
         return false;
      }
   } catch (UnknownHostException e) {
      GUIDispatcher.postErrorDialog("Cannot log on: Unknown host: " + hostName);
      return false;
   } catch (IOException ioe) {
      handleIOException(ioe, "Unable to connect to host ");
      return false;
   }
   this.isConnected = true;
   return (true);
}

public boolean isConnected( ) {
   return (this.isConnected);
}

/**
 * getSourceFile( ) retrieves file remotePath and writes it to resultFile
 * Returns true if successful
 */
public boolean getSourceFile(String remotePath,
                   String resultFile) {
   if (! blockTransferToFile("GET_SOURCE_FILE " + remotePath + "\n", resultFile)) {
      return false;
   }
```

```
    return true;
}

/**
 * Set the list of directories used to search for source files
 * searchPath must be a semi-colon separated list of directories
 * Return true if request succeeds
 */
public boolean setSearchPath(String searchPath) {
    try {
        sendRequest("SET_SEARCH_PATH " + searchPath + "\n");
        if (getStatus(getServerResponseLine()) != REPLY_REQUEST_COMPLETE) {
            return false;
        }
    } catch (IOException ioe) {
        handleIOException(ioe, "Unable to set source search path ");
        return false;
    }
    return true;
}

/**
 * Get the source line number associated with the beginning of a function
 * Return -1 on failure
 */
public int getSourcePosition(String moduleName, String function) {
    long count = -1;
    try {
        sendRequest("GET_SOURCE_POS " + moduleName + " " + function + "\n");
        count = getCountResponse(getServerResponseLine());
        if (count < 0) {
            return -1;
        }
    } catch (IOException ioe) {
        handleIOException(ioe, "Unable to retrieve source position ");
        return -1;
    }
    return (int)count;
}

/**
 * Disconnect from server  Return true on success
 */
public boolean disconnect() {
    try {
        sendRequest("DISCONNECT\n");
        if (getStatus(getServerResponseLine()) != REPLY_DISCONNECTED) {
            return false;
        }
        out.close();
        in.close();
        loginSocket.close();
    } catch (IOException ioe) {
        handleIOException(ioe, "Unable to disconnect from server ");
        return false;
    }
    this.isConnected = false;
    out = null;
    in = null;
    loginSocket = null;
    return true;
}
```

```
private void handleIOException( IOException ioe, String msg ) {
   GUIDispatcher.postErrorDialog( msg + "\nIOException ( " + ioe
       + " ) on host " + this.hostName );
}

/**
 * Get a long value from a string. Returns REPLY_BAD_REPLY_CODE if longVal cannot
 * be converted to a long
 */
private long getLong( String longVal ) {
   long val;
   try {
      val = Long.parseLong( longVal );
   } catch ( NumberFormatException e ) {
      GUIDispatcher.postErrorDialog(
         "INTERNAL ERROR: Server returned non-numeric status: " + longVal );
      return this.REPLY_BAD_REPLY_CODE;
   }
   return val;
}

/**
 * Get the status code from a server reply string
 */
private long getStatus( String serverReply ) {
   int startPos = 0;
   int endPos = serverReply.indexOf( ' ', startPos );

   if ( endPos < 0 ) {
      endPos = serverReply.length( ) - 1;
   }
   String codeString = serverReply.substring( startPos, endPos );
   long status = getLong( codeString );
   if ( status == REPLY_ERROR ) {
      GUIDispatcher.postErrorDialog( "Error: " + serverReply );
   }
   return status;
}


private void debugPrint( String debugInfo ) {
   if ( WorkerRepository.getCommandArgs( ).verbose )
      System.out.println( debugInfo );
}

private void sendRequest( String request ) throws IOException {
   debugPrint( "REQUEST: " + request );
   out.write( request.getBytes( ), 0, request.length( ) );
}

/**
 * Get a newline-terminated response from the server
 */
private String getServerResponseLine( ) throws IOException {
   String response = in.readLine( );
   debugPrint( response );
   return response;
}

/**
 * Retrieve a given number of bytes from the server output. Used after
 * server has indicated it is going to send a byteCount-size block of data
 */
private char [] getServerResponseBytes( int byteCount ) throws IOException {
    // Read exactly the number of bytes indicated
    char[] data = new char[byteCount];
    int bytesRead = 0;
    int offset    = 0;

    while (offset < byteCount) {
       bytesRead = in.read(data, offset, (data.length - offset));
       if (bytesRead == -1)
          break;
       offset += bytesRead;
    }
    return data;
  }

  /**
   * Retrieve and check a transfer complete message from server
   * Return true if transfer completed successfully
   */
  private boolean transferComplete( String lineFromServer ) {
    if (lineFromServer == null) {
       GUIDispatcher.postErrorDialog("INTERNAL ERROR: Null transfer-complete status ");
       return false;
    }
    if (getStatus(lineFromServer) != REPLY_BLOCK_TRANSFER_COMPLETE) {
       return false;
    }
    return true;
  }

  /**
   * Retrieve the count portion of a numeric response from the server
   * Return -1 on error
   */
  private long getCountResponse( String countResponse) throws IOException {
    int startPos = 0;

    if (countResponse == null) {
      GUIDispatcher.postErrorDialog("INTERNAL ERROR: null status for count response ");
      return -1;
    }
    if (getStatus(countResponse) != REPLY_NUMERIC_VALUE) {
      return -1;
    }
    startPos = countResponse.indexOf(' ', 0);
    if (startPos < 0) {
       GUIDispatcher.postErrorDialog(
          "INTERNAL ERROR: Expecting count, got: " + countResponse);
       return -1;
    }
    long count = Long.parseLong(countResponse.substring(startPos+1 ));
    debugPrint("\nSERVER: count = " + count);
    return count;
  }

  /**
   * Send a block-transfer request to the server (such as, get source or
   * disassembly), retrieve the block of data, and write it to a file
   */
  private boolean blockTransferToFile( String request, String file) {
    long byteCount = 0;

    try {
       sendRequest(request);
       byteCount = getCountResponse(getServerResponseLine( ));
```

```
            if (byteCount < 0) {
               return false;
            }
            char [] data = getServerResponseBytes( int )byteCount );
            FileWriter fw = new FileWriter( file );
            fw.write( data );
            fw.close( );

            if( !transferComplete( getServerResponseLine( ) ) {
               return false;
            }
         } catch ( IOException ioe ) {
            handleIOException( ioe, "Failed to transfer data " );
            return false;
         }
         debugPrint( "WROTE TRANSFERED DATA TO " + file );
         return true;
      }

   }


/**
 * Title        SecurityManager
 * Description  These are the functions that handle the encryption and decryption
 *              of the data that passes through
 * Copyright    Copyright (c) 2001
 * Organization  UNLV - Computer Science Dept
 */

*************************************************************
//The Encryption code using the DES Algorithm

void des_encrypt( word32 l, word32 r, word32 *output, DESContext *ks,
                  int encrypt )
{
   register word32 t,u;
   register int i;
   register word32 *s;

   s = ks->key_schedule;

   IP( l,r,t );
   t=(r<<1)|(r>>31);
   r=(l<<1)|(l>>31);
   l=t;

   /* I don't know if it is worth the effort of loop unrolling the
    * inner loop */
   if (encrypt)
     {
      for (i=0; i<32; i+=4)
            {
             D_ENCRYPT(l,r,i+0); /*  1 */
             D_ENCRYPT(r,l,i+2); /*  2 */
            }
     }
   else
     {
      for (i=30; i>0; i-=4)
            {
             D_ENCRYPT(l,r,i-0); /* 16 */
             D_ENCRYPT(r,l,i-2); /* 15 */
            }
```

```
  }
 l=(l>>1)|(l<<31);
 r=(r>>1)|(r<<31);

 FP(r,l,t);
 output[0]=l;
 output[1]=r;
}

#define HPERM_OP(a,t,n,m) ((t)=((((a)<<(16-(n)))^(a))&(m)),\
        (a)=(a)^(t)^(t>>(16-(n))))

void des_set_key(unsigned char *key, DESContext *ks)
{
 register word32 c, d, t, s, shifts;
 register int i;
 register word32 *schedule;

 schedule = ks->key_schedule;

 c = GET_32BIT_LSB_FIRST(key);
 d = GET_32BIT_LSB_FIRST(key + 4);

 /* I now do it in 47 simple operations :-)
  * Thanks to John Fletcher (john_fletcher@lccmail.ocf.llnl.gov)
  * for the inspiration. :-) */
 PERM_OP(d,c,t,4,0x0f0f0f0f);
 HPERM_OP(c,t,-2,0xcccc0000);
 HPERM_OP(d,t,-2,0xcccc0000);
 PERM_OP(d,c,t,1,0x55555555);
 PERM_OP(c,d,t,8,0x00ff00ff);
 PERM_OP(d,c,t,1,0x55555555);
 d = ((d & 0xff) << 16) | (d & 0xff00) |
   ((d >> 16) & 0xff) | ((c >> 4) & 0xf000000);
 c&=0x0fffffff;

 shifts = 0x7efc;
 for (i=0; i < 16; i++)
  {
   if (shifts & 1)
        { c=((c>>2)|(c<<26)); d=((d>>2)|(d<<26)); }
   else
        { c=((c>>1)|(c<<27)); d=((d>>1)|(d<<27)); }
   shifts >>= 1;
   c&=0x0fffffff;
   d&=0x0fffffff;

   /* could be a few less shifts but I am to lazy at this
    * point in time to investigate */

   s = des_skb[0][ (c    )&0x3f                ]|
       des_skb[1][((c>> 6)&0x03)|((c>> 7)&0x3c)]|
       des_skb[2][((c>>13)&0x0f)|((c>>14)&0x30)]|
       des_skb[3][((c>>20)&0x01)|((c>>21)&0x06)|((c>>22)&0x38)];

   t = des_skb[4][ (d    )&0x3f                ]|
       des_skb[5][((d>> 7)&0x03)|((d>> 8)&0x3c)]|
       des_skb[6][ (d>>15)&0x3f                ]|
       des_skb[7][((d>>21)&0x0f)|((d>>22)&0x30)];

   /* table contained 0213 4657 */
   *schedule++ = ((t << 16) | (s & 0xffff));
   s = ((s >> 16) | (t & 0xffff0000));
   *schedule++ = (s << 4) | (s >> 28);
```

```
    }
}

void des_cbc_encrypt(DESContext *ks, unsigned char *iv,
                     unsigned char *dest, const unsigned char *src,
                     unsigned int len)
{
  word32 iv0, iv1, out[2];
  unsigned int i;

  assert((len & 7) == 0);

  iv0 = GET_32BIT_LSB_FIRST(iv);
  iv1 = GET_32BIT_LSB_FIRST(iv + 4);

  for (i = 0; i < len; i += 8)
    {
      iv0 ^= GET_32BIT_LSB_FIRST(src + i);
      iv1 ^= GET_32BIT_LSB_FIRST(src + i + 4);
      des_encrypt(iv0, iv1, out, ks, 1);
      iv0 = out[0];
      iv1 = out[1];
      PUT_32BIT_LSB_FIRST(dest + i, iv0);
      PUT_32BIT_LSB_FIRST(dest + i + 4, iv1);
    }
  PUT_32BIT_LSB_FIRST(iv, iv0);
  PUT_32BIT_LSB_FIRST(iv + 4, iv1);
}

void des_cbc_decrypt(DESContext *ks, unsigned char *iv,
                     unsigned char *dest, const unsigned char *src,
                     unsigned int len)
{
  word32 iv0, iv1, d0, d1, out[2];
  unsigned int i;

  assert((len & 7) == 0);

  iv0 = GET_32BIT_LSB_FIRST(iv);
  iv1 = GET_32BIT_LSB_FIRST(iv + 4);

  for (i = 0; i < len; i += 8)
    {
      d0 = GET_32BIT_LSB_FIRST(src + i);
      d1 = GET_32BIT_LSB_FIRST(src + i + 4);
      des_encrypt(d0, d1, out, ks, 0);
      iv0 ^= out[0];
      iv1 ^= out[1];
      PUT_32BIT_LSB_FIRST(dest + i, iv0);
      PUT_32BIT_LSB_FIRST(dest + i + 4, iv1);
      iv0 = d0;
      iv1 = d1;
    }
  PUT_32BIT_LSB_FIRST(iv, iv0);
  PUT_32BIT_LSB_FIRST(iv + 4, iv1);
}

void des_3cbc_encrypt(DESContext *ks1, unsigned char *iv1,
                      DESContext *ks2, unsigned char *iv2,
                      DESContext *ks3, unsigned char *iv3,
                      unsigned char *dest, const unsigned char *src,
                      unsigned int len)
{
  des_cbc_encrypt(ks1, iv1, dest, src, len);
```

```
 des_cbc_decrypt( ks2, iv2, dest, dest, len );
 des_cbc_encrypt( ks3, iv3, dest, dest, len );
}

void des_3cbc_decrypt( DESContext *ks1, unsigned char *iv1,
                        DESContext *ks2, unsigned char *iv2,
                        DESContext *ks3, unsigned char *iv3,
                        unsigned char *dest, const unsigned char *src,
                        unsigned int len )
{
 des_cbc_decrypt( ks3, iv3, dest, src, len );
 des_cbc_encrypt( ks2, iv2, dest, dest, len );
 des_cbc_decrypt( ks1, iv1, dest, dest, len );
}

#ifdef DES_TEST

void des_encrypt_buf( DESContext *ks, unsigned char *out,
                       const unsigned char *in, int encrypt )
{
 word32 in0, in1, output[0];

 in0 = GET_32BIT_LSB_FIRST( in );
 in1 = GET_32BIT_LSB_FIRST( in + 4 );
 des_encrypt( in0, in1, output, ks, encrypt );
 PUT_32BIT_LSB_FIRST( out, output[0] );
 PUT_32BIT_LSB_FIRST( out + 4, output[1] );
}
```

BIBLIOGRAPHY

[1] N Venkatasubramanian and K Nahrstedt. An Integrated Metric for Video QoS. *Proceedings of the fifth ACM International Conference on Multimedia*. Pages 371-380, 1997

[2] L Brown and MGII Jaatun. Secure File Transfer over TCP/IP. *In IEEE Tencon92 - 1992 IEEE Region 10 International Conference, Melbourne*. Pages 404-498, 1992

[3] PC Fishburn and AM Odlyzko. Dynamic Behavior of Differential Pricing and Quality of Service Options for the Internet. *Proceedings of First International Conference on Information and Computation Economics (ICE-98)*, 1998

[4] R Staehli and J Walpole. Using Script-Based QoS Specifications for Resource Scheduling. *In Collected Abstracts from the 4th International Workshop on Network and Operating Systems Support for Digital Audio and Video*. Pages 93-95, 1993

[5] H Lin and L Harn. Authentication Protocols for Personal Communication Systems. *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*. Pages 256 - 261, 1995

[6] J Costello. Security on Computer Networks. *Proceedings of the 19th ACM SIGCCS conference on User Services*. Pages 49 - 52, 1991

[7] M Burrows, M Abadi and R Needham. A Logic of authentication. *ACM Trans Comput. Syst. 8*, Pages 18 - 36, 1990

[8] A Paul. QoS in Data Networks: Protocols and Standards. *This report is available ftp: ftp.netlab.ohiostate.edu pub jain courses cis788899 qos protocols index.html*

[9] N Venaktasubramanian and S Ramanathan. Load Management in distributed video servers. *In International conference on Distributed Computing System*. May 1997

[10] P Ferguson and G Houston. Quality of Service in the Internet: Fact, Fiction, or Compromise? *Submitted for presentation at INET'98, 21-24July 1998, Geneva*

Switzerland. This paper is also available as a technical report from http: www.employees.org:80 ~ferguson inet qos.htm

[11] Ohio State University QoS-Policy-Constraint based routing *This paper is available as a technical report from http: www.tradespeak.com htmldocs 1854.html*

[12] Comer, Douglas *Internetworking with TCP IP* Prentice Hall 1991

[13] Stevens, Richard *UNIX Network Programming, Volume 1, Second Edition: Networking APIs: Sockets and XTI,* Prentice Hall, 1998

[14] The Need for QoS - A White Paper QoSForum.com July 1999 *http: www.qosforum.com white-papers Need_for_QoS-v4.pdf*

[15] An Economic Approach to Adaptive Resource Management Neil Stratford and Richard Mortier *http: www.cl.cam.ac.uk Research SRG netos xeno economics*

[16] Network Interface Services Initiative – A White Paper Intel Corporation http://www.intel.com/network/white_papers/nisi/

[17] Introduction to QoS Policies - A White Paper QoSForum.com 6th July 199 *http: www.qosforum.com white-papers qospol_v11.pdf*

[18] QoS applied to security in mobile computing Terje Fallmyr, Tage Stabell-Kulo 1997 http://www.cs.uit.no/forskning/rapporter/Reports/9729.html

[19] Differentiated Services for Internet2 (draft) John Sikora and Ben Teitelbaum *http: www.internet2.edu qos may98Workshop html diffserv.html*

[20] QoS in Data Networks Protocols and Standards Arindam Paul November 1999 *ftp: ftp.netlab.ohio-state.edu pub jain courses cis788899 qos_protocols index.html*

[21] K Nahrstedt and J M Smith The QoS broker IEEE Multimedia, 2:53 – 67, 1995

[22] QoS Negotiation and Resource Reservation for Distributed Multimedia Applications Kurt Rothermel, Gabriel Dermler and Walter Fiederer July 1996 *http: ncstrl.informatik.unistuttgart.de Dienst Repository 2.0 Body ncstrl.ustuttgart_fi TR-1996-09 pdf*

[23] R M Needham and M D Schroeder Using encryption for authentication in large networks of computers *Communication ACM 21,2 (Feb. 1978), 120 - 126*

[24] The Design of a Generic QoS Architecture for Open Systems Frank Siqueira http://www.cs.su.oz.au/~mingli/QoS_Architecture.html

[25] Jan Gecsel Adaptation in Distributed Multimedia Systems *IEEE Multimedia. Vol. 4, No2, April - June 1997.*

[26] Overview of Middleware, *http: middleware.internet2.edu overview*

VITA

Graduate College
University of Nevada, Las Vegas

Ajay Kanagala

Local Address:
507 W Rincon Ave
Campbell, California 95008

Home Address:
No 1 Amudha Street
Dr.Seethapathy Nagar, Velachery
Chennai, Tamil Nadu, India 600 042

Degree:
Bachelor of Engineering, Computer Science and Engineering, 1997
University of Madras, Chennai

Thesis Title:
A Framework For Secure Applications With QoS

Thesis Examination Committee:
Chairperson, Dr. Ajoy K. Datta, Ph. D.
Committee Member, Dr. Kazem Taghva, Ph. D.
Committee Member, Dr. Wolfgang Bein Ph. D.
Graduate Faculty Representative, Dr. Henry Selvaraj, Ph. D.