DYNAMIC PROTOCOL SELECTION FOR COMMUNICATION SYSTEMS

By

RAJASEKARAN KANDASWAMI

Bachelor of Engineering in Computer Science

Annamalai University

Chidambaram, India

2001

Submitted to the faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May 2006

DYNAMIC PROTOCOL SELECTION FOR
COMMUNICATION SYSTEMS

Thesis Approved:

Dr. JOHNSON P THOMAS
Thesis Advisor

Dr. DEBAO CHEN

Dr. VENKATESH SARANGAN

Dr. GORDON EMSLIE
Dean of the Graduate College

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor, Dr. Johnson Thomas, whose directions, and guidance helped in completion of this thesis. I respectfully thank Dr. Debao Chen and Dr. Venkatesh Sarangan for being in my thesis committee and for their suggestions. Also I would like to extend my gratitude to Dr. George Hedrick and all other computer science department faculty & staff for providing the facilities needed for the research.

Also warmest thanks must go to my parents, Kandaswami and Sakunthala, and members of my family whose love, support, and encouragement allowed me to write this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

CHAPTER I

INTRODUCTION

Due to the rapid growth and development of distributed network applications, efficiency in data communication between communicating systems is important for determining the performance of the distributed applications (client/server). The three levels of end to end QoS service for a particular network application considering the delay, loss, and bandwidth are Best Effort Service, Differentiated Service and Guaranteed Service.

Traditional internet offers “best effort” delivery model which is an end to end delivery service. This model sees that data is delivered to the end system as quickly as possible without considering the Quality of Service (QoS) parameters such as bandwidth, jitter, and latency. Quality of Service (QoS) refers to the capability of a network to provide better experience a user or application may receive on selected network traffic over various technologies and techniques.

Quality of Service (QoS) requirements of distributed applications depends on three aspects [1] :(1)application class like isochronous, burst, low delay, (2) type of media like text, audio, video and (3) application and user specific requirements. For network applications to function efficiently it requires minimum amount of QoS from the under lying network in terms of throughput, delay, packet loss and jitter. Protocols such

as TCP which provide reliable data delivery can be used for applications such as FTP and Telnet, but are not ideal for application requiring timeliness. An application using UDP transport protocol keeps sending packets when congestion is encountered in the network (no congestion control in UDP). This can starve of the allocated bandwidth of an application using TCP transport protocol in the network. Guarantees in packet delivery helps only when you don't have enough bandwidth. Mechanisms such as resource reservation and scheduling are proposed to solve this problem, but they have their own drawbacks. One such drawback is that reservation of resources is made with the initial availability when the connection is made and it doesn't take into consideration the changes in resource availability during the course of data transmission. Generally in many network applications, when data transmission is sustained for a long duration, this can cause undesired results due to changing network conditions.

To solve this problem we propose a flexible mechanism by constructing multiple protocol stacks by which the application can select end system protocols taking into consideration the user preference and the QoS parameters provided by the underlying network. The protocol stack comprises of different transport layer protocols with differing characteristics. The appropriate protocol for the given current conditions is selected from the stack based on user preference and QoS provided by the network during data communication. The QoS parameters that can be measured and monitored from the underlying network are bandwidth, delay, jitter, loss. The proposed approach for protocol adaptation is based on the Analytic Hierarchical Approach.

In chapter II we consider the different methods used by previous researchers. An overview of our system and the proposed approach used is given in chapter III. The implementation of the approach and the system architecture is explained in chapter IV. Chapter V discusses results obtained.

CHAPTER II

LITERATURE REVIEW

2.1 DEFINITIONS

2.1.1. TRANSPORT LAYER

The fourth layer of the seven layer OSI model is the transport layer which makes client/server applications a reality. Using the service provided by the network layer, the transport layer provides quality of service to the session entity. Transport protocols in this layer such as Transmission Control Protocol/ Internet Protocol (TCP/IP), Open Systems Interconnection (OSI), Novell's IPX/SPX provide mechanisms for moving packets between the client/server. This layer not only ensures data transfer between end users but also responds to service requests from the session layer and issues service requests to the network layer. The type of service provided by this layer is determined during the connection establishment. Message delivery is a type of service which can be in the order by which it is sent or with no guarantee about the order in which it is delivered. The data is divided into smaller packets and dispatched by the transport layer. At the receiving end it also makes sure that the pieces arrive correctly and are reassembled according to the sequence number.

IP addressing is used by the client and the server since the protocols used in our dynamic stack run over the IP protocol. Since it is a source to destination layer it uses the message headers and control messages to communicate between the host machine and its

peers. The transport header is used to differentiate between which message belongs to which connection as multiple connections arrive and exit the host. Along with the IP header the TCP/IP protocol suite requires the transport protocol to be listed in the header along with the source and the destination address. Port numbers help in the identification of the application to the TCP/IP protocol.

Numbers in the range of 1 to1023 are called as well known port numbers, which is used by FTP,HTTP,SMTP,TELNET etc., . Ephemeral port numbers in the range 1024 to 5000 is used by the client instead of well known ports for making a connection with the server.IP address, port number, transport protocol are the three components that form the socket in the TCP/IP suite. Client-Server protocols such as Sockets, Advanced Program-to-Program Communication (APCC), Transport Level Interface (TLI), Sequenced Packet Exchange (SPX), RPC and NetBIOS provides mechanisms by which client request information and services from the server and the server responds to that request.

Protocols	**Layer**
FTP, HTTP, HTTPS, IMAP, IRC, NTP, POP3, SIP, SMTP, SNMP, SSH, Telnet, Bit Torrent, Websphere MQ, ...	Application
DCCP, SCTP, TCP, RTP, UDP, IL, RUDP, ...	Transport
IPv4, IPv6, ...	Network
Ethernet, Wi-Fi, Token ring, FDDI, PPP, ...	Data link
RS-232, EIA-422, RS-449, EIA-485, 10BASE2, 10BASE-T, ...	Physical

Table 1: TCP/IP protocol suite

2.2 PREVIOUS RESEARCH

Various works have been carried in the development of configurable protocols by identifying and combining functionalities of different communication protocols that are tailored to the needs of the application.

2.2.1 Da CaPo PROJECT

The Da CaPo (Dynamic Configuration of Protocols) project [2, 3] is aimed at overcoming communication system bottlenecks by configuring light weight protocols. The Da CaPo system is based on a three layer model that splits communication systems into the layers Application layer (A), Communication Layer (C), and Transport Layer (T). Developing protocols in Da CaPo is done through hybrid methods. Based on the available transport layer service and application requirement a configuration process selects the most appropriate functionality for the communication service. Layer C is composed of protocol functions instead of sub layers. Protocol tasks like error detection, acknowledgment, flow control, decryption and encryption are embedded in each of the protocol function. Here different protocol configurations support different Quality of Service requirements. Network applications give their service requirements within a service request which is then passed on to the C layer. Based on the underlying network services and amount of available resources Da CaPo configures on the fly communication layer protocols. These protocols are optimally configured to adapt according to the application requirements. In the communication layer, four active dependent entities are performing their respective tasks such as:

- QoS negotiation and appropriate protocol configuration is identified using a search based heuristic, Configuration and Resource Allocation (CoRA). Finding appropriate configurations under real time constraints is important. This is achieved by dividing the modules accordingly and having a measure for the resource usage. The modules and resource usage are combined in a structured search enabling CoRA.
- Assuring end-end systems use the same protocol for communication layer connection, negotiating with end system for common configuration, handling errors. Coordinating reconfiguration of protocols if application requirements are not satisfied is carried out by the connection manager.
- Linking and module initialization, packet forwarding, synchronization, avoiding unnecessary copy operation, reducing the context switches to low level are done by the runtime environment.
- The monitoring component, based on the application requirement triggers the connection manager to aid in protocol reconfiguration. It also ensures the availability of resources.

The Da CaPo project does not make assumptions about underlying hardware and also does not assume that the transport layer (T layer) maps directly to the network layer. The T layer software such as Ethernet, protocol stacks such as IP provides services which helps in the configuration of the C layer. Thus, decreasing protocol complexity by configuring appropriate protocols distinctly increases protocol performance.

2.2.2 UNIVERSAL TRANSPORT SYSTEM

The Universal Transport System (UTS) [4] demonstrates the merits of using adaptable protocol for high speed, multimedia, mobile networks where the QoS and application requirement varies and the less overhead produced by its implementation. Here the end-to-end protocols are fragmented into smaller protocol functions. By combining these smaller protocol functions complex protocols are developed. A set of protocol functions is developed by mapping various application classes. The adaptability in terms of selecting various protocol functions during the data transfer is imparted changing TCP, which uses dynamic linking (DLD) to make the protocol very flexible.

A generic adaptive framework which supports protocol implementation concepts such as Integrated Level Planning (ILP) and Application Level Framing (ALF) is given that uses the atomic functions to implement general purpose protocol. ALF is a mechanism for improving protocol functionality which allows packets that are out of order to be processed. Hence this makes data transfer meaningful to the application. During the connection initialization/establishing time and operation an application specifies its requirements to a functionality decider. A profile, namely a set of appropriate protocols is developed from this requirement specification. Based on the profile needed, functions are pulled from the library. A run time protocol is developed by the synthesis engine while also maintaining the optimizations such as ILP. Connection establishment is a mode defined for the synthesis engine through which a connection is established using protocol functions familiar to the end systems. Later it switches to the adaptive mode, by which protocol adaptation is done based on user requirements, end system and Quality of Service provided by the network.

The functionalities provided by the UTS protocol server are TCP/UDP/IP with the BSD socket interface, ARP. The application communicates with the protocol server with the IPC BSD interface. The main aim of UTS is to show how the transport layer protocols can be tailored to the user needs and make them to adapt according to the changing application requirement and network conditions.

2.2.3. ADAPTIVE

A Dynamically Assembled protocol Transformation, Integration, and Validation Environment (ADAPTIVE) [5], is developed to support multimedia applications running on high performance networks. For prototyping, experimentation, and diversity a flexible transport system design is essential. It has mechanism and policies based on the object oriented design concepts to automatically specify and synthesize a more flexible, lightweight, and adaptive transport protocol configuration. ADAPTIVE responds to the feedback changes in application requirements by supporting run-time adaptive reconfiguration. The four areas this project aims to solve are [5]

1. Designing a very simple application interface.
2. Giving a high available throughput to the application.
3. Compatible with a wide variety of underlying networks.
4. Compatible with all the functionality requirements of the multimedia applications.

The design of ADAPTIVE includes three main subsystems that are [5]

1. MANTTS (Map Application and Networks To Transport Systems), helps in selecting appropriate set of policies and mechanisms in order to meet an application's quality-of-service (QoS) requirements and it also communicates with the end systems for considering the dynamically changing network environment.
2. TKO (Transport Kernel Objects), helps in developing a customized lightweight transport system session configuration. The session is composed of reusable objects protocol mechanism library
3. UNITES (UNIform Transport Evaluation Subsystems) helps network traffic monitoring, metric selection/collection/analysis/presentation, and performance measurement. The UNITES delivers feedback to the MANTTS and TKO that assists in evaluation/determining the right time to dynamically change particular mechanisms in a session.

The communication requirements of both the application and high-performance networks are met by tailoring the services of transport systems to meet the requirements of next generation multimedia applications.

CHAPTER III

METHODOLOGY

3.1 APPROACH

This thesis presents the construction and implementation of a framework for adaptable communication systems. The framework contains different transport layer protocols with differing characteristics in a protocol stack. The appropriate transport protocol is selected dynamically according to the changes in the application's QoS requirements and underlying network conditions during the data communication. This selection of suitable protocols helps improve performance compared to end-end fixed protocols.

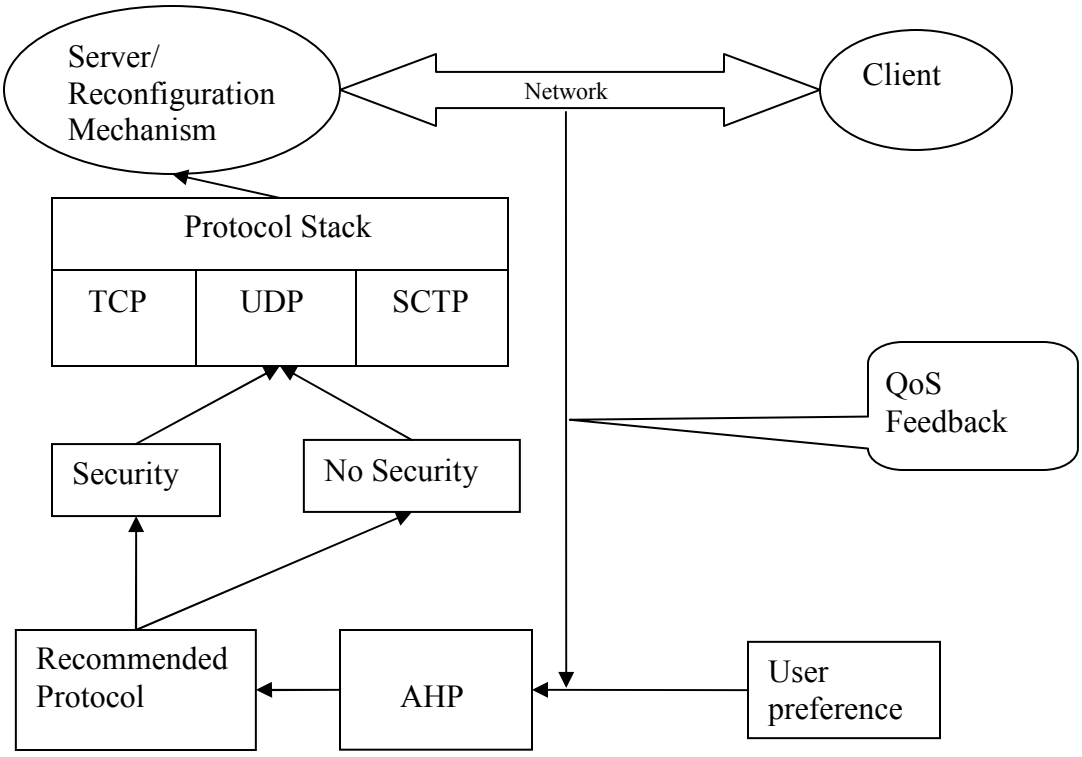

Figure 1: Approach

3.2 CONSTRUCTION OF THE FRAMEWORK

The framework addresses the following

3.2.1 FLEXIBILITY

The protocol stack consists of different transport layer protocols each with different characteristics that are needed for the applications. The protocol stack offers flexibility for expanding/reducing transport layer protocols as needed from the stack since they are managed at the application level.

3.2.2. SYSTEM COMPATIBILITY

For inter process communication (IPC), industry standard BSD socket library with bi-directional data stream is used. This gives the compatibility to port the application code to UNIX/LINUX operating systems. The transport layer protocols involved in the stack are TCP, UDP, and SCTP. If security is needed a security layer is added above these protocols. The management of protocols is done at the user level, whereas they exist at the kernel level including the SCTP kernel patch. Before starting the framework SCTP library (lksctp) is added as a kernel module kernel.

3.2.3. ADAPTATION

Based on the user preferences and measured network QoS parameters, the best transport layer protocol is selected using the Analytical Hierarchy Process. The application need not involve itself directly in the adaptation issues.

3.3 TRANSPORT LAYER PROTOCOLS IN THE STACK

3.3.1 TCP

Transmission Control Protocol (TCP) is a connection-oriented, reliable-delivery byte-stream transport layer communication protocol. The unreliable packets provided by the Internet Protocol are made reliable by using streams between applications in TCP. The three phases in TCP connections are connection establishment, data transfer and connection termination. A 3-way handshake and 4-way handshake is used for establishing a connection and terminating a connection respectively. The ordered delivery of data is made possible due to the initialization of sequence numbers during connection establishment. TCP is not suitable for certain applications because it waits for the lost

packet to be retransmitted before getting the previously send packets. For some real-time applications like streaming multimedia, it is important to receive more data in a timely fashion rather than get all the data in sequence.

3.3.2 UDP

User Datagram Protocol (UDP) provides a minimal transport service with no guarantee of message delivery, no connection establishment, no connection state, small segment header overhead, and unregulated send rate. When the level of service offered by TCP is not required by the communicating application or when the application uses the service that is not in TCP (multicast or broadcast delivery) UDP is preferred. As UDP lacks in reliability, the application using UDP should be able to accept the errors, loss, or duplication. Here there is no handshaking like TCP between sending and receiving transport layer entities before sending a segment of data.

3.3.3 SCTP

Stream Control Transmission Protocol is another IP protocol which provides reliable stream oriented, in sequence transport of message services with congestion control. This protocol can be used in application scenarios where reliability and near real time aspects are important. TCP is byte oriented, where as SCTP deals with framed messages.

Advantages of SCTP are:

- Multi-homing: In this support either in one or both ends of the association, multiple IP address is provided. The association is done by one endpoint providing multiple IP addresses with the combination of SCTP port numbers to the other endpoint. These addresses are used in sending and receiving SCTP

packets. This will help in recovering from network level failure between hosts or network cards. In case of TCP only a single point of association exists.

- Head-of-the-line blocking: This happens when retransmission of lost data affects the timely delivery of other data in unrelated sequences as in the case of TCP byte stream delivery. In case of SCTP the delivery of data is done within independent streams in chunks, which takes care of the Head-of-the-line blocking problem.
- Path selection: One IP address from the list is selected as the primary path for sending the data chunks, whereas during the retransmission of the data another active path is selected from the pool if available. The SCTP users are notified about the change in the path and will be asked to use the new path.
- Gives protection against flooding attacks, notifies about lost data chunks or duplication in date.

3.4 ADDED SECURITY

In addition to the above protocols in the stack, we added a security layer above TCP, UDP, and SCTP protocols. This gives a wide option in selecting protocols based on QoS requirements. Security is provided by means of encrypting at the server side and decrypting the data at the client side. The Data Encryption Standard (DES) [10] is used as the security algorithm. Adding security via encryption and decryption during data transfer is an overhead in terms of processing power.

Data Encryption Standard (DES):

One of the widely used encryption algorithm in the world is DES. This algorithm takes fixed length blocks of plain text bits and transforms it into a cipher text of the same fixed length blocks after some operations. The fixed length block is 64-bit long

and it also uses a 64-bit key. This key is used to perform transformation of plain text to cipher text and vice versa. In order to do the decryption at the client side, the client should know the particular key that was used to encrypt in the server side. Though the key is 64 bits long only 56 bits are used effectively and the remaining 8 bits are used for checking parity [10].

3.5 MEASURING QoS PARAMETERS AND GETTING USER PREFERENCE

The underlying network is constantly monitored and measured to check the changes in the QoS parameters such as bandwidth, jitter, and delay. The level of reliability (based on file type) and security needed for that particular application is given by the user. Low reliability, medium reliability, and high reliability is given for file types video, audio, and text respectively. Based on the file type given by the user, the reliability factor is decided. The underlying network conditions and user preferred QoS parameters are given as feedback to the protocol decision making model. Here the Analytical Hierarchy Process (AHP) is used as our decision making model. Based on the feedback, the decision making model selects appropriate protocols from the protocol stack.

3.6 ANALYTICAL HIERARCHY PROCESS

Developed by Thomas Saaty, the Analytic Hierarchy Process (AHP) [6] is a proven and effective complex decision making process that aids people make best decision and set priorities among alternatives, when quantitative and qualitative aspects of a decision needs to considered. The AHP [7] concept has been successfully applied in finding a solution for real time problems in the field of medicine, sports, computer science. The decision making can be multiple criteria or multi-attribute decision making. The alternatives are ranked by developing numerical value to each alternative based on

how good each alternative satisfies the criteria. Let's assume that n items are taken into account with the aim of providing judgments. These judgments are made by comparing each item with respect to all other items based on the relative weight like priority, size, and importance.

The steps involved are,

- Design phase:

  This is the first step where setting up the problem as a hierarchy takes place. The hierarchy consists of goal, criteria, and alternative as layers. The root node is set as the overall objective of the decision. The branches to the root comprises of the criteria used in deriving the decision. The lowermost layer in the hierarchy is alternatives from which the choices are to be made. (n items that is to be compared).

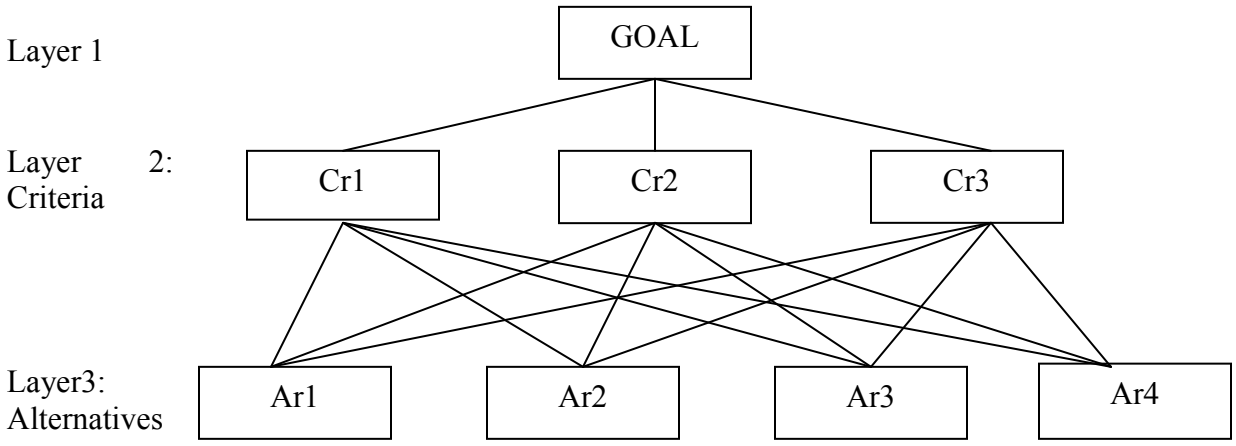

Figure 2: Analytical Hierarchy Process.

- Evaluation Phase:

In the second step on a given hierarchy level, pair-wise comparisons are made among each of the two items. Comparison between items is done with respect to their contribution towards the factor from the level immediately on top of them. The pair-wise comparison is graded using table 1 by raising the questions such as which is more important for the given factor. The rated number point out the importance of the item and it helps to differentiate between them. This pair-wise comparison leads to a reciprocal n X n matrix A, where $a_{ii}=1$ indicates elements are equal and $a_{ji} = 1/a_{ij}$ indicates it is reciprocal. The matrix A is completed with relative weights by making comparison and transitivity of the relative importance between elements. Multiplying the matrix obtained with the criteria matrix weight w (user preference and network conditions) we get:

$$Aw=\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ . & . & & . \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}\begin{pmatrix} w_1 \\ w_2 \\ . \\ w_n \end{pmatrix}=\begin{bmatrix} w_1/w_1 & w_1/w_2 & \dots & w_1/w_n \\ w_2/w_1 & w_2/w_2 & \dots & w_2/w_n \\ . & . & & . \\ w_n/w_1 & w_n/w_2 & \dots & w_n/w_n \end{bmatrix}\begin{pmatrix} w_1 \\ w_2 \\ . \\ w_n \end{pmatrix}=n\begin{pmatrix} w_1 \\ w_2 \\ . \\ w_n \end{pmatrix}$$

so, we have

$$Aw=nw, \quad \text{or} \quad (A-nI)=0.$$

RATING	LEVEL OF IMPORTANCE
1	Equally Important
2	Equally to moderately more Important
3	Moderately more important
4	Moderately to strongly more important
5	Strongly more important
6	Strongly to very strongly more important
7	Very strongly more important
8	Very strongly to extremely more important
9	Extremely more important

Table 2: Rating Vs Importance Level

- Construction of Matrix

With the M number of alternatives and N number of criteria, a M x N matrix is constructed in the final step. The element a_{ij} in the M x N matrix represents the relative weight of the i^{th} alternative in terms of j^{th} criterion. From the pair-wise comparison mentioned in the second step, the vector $V_i = (a_{i1}, a_{i2}, \ldots, a_{iN})$ is the eigenvector of the N x N reciprocal matrix [8] for the i^{th} alternative (i =1,2,3…,M). When the elements in the each vector are added up it comes to one. The AHP can be used in single- and multidimensional decision making, since they use relative values instead of actual ones.

The best alternative, A*, in AHP is calculated using the following expression [8]

$$A^*_{AHP} = \max_{M \geq i \geq 1} \sum_{i=1}^{N} a_{ij} W_j$$

The best alternative in our approach is the best transport layer protocol. This protocol preference is given to the protocol stack and corresponding protocol is selected for data communication.

CHAPTER IV

IMPLEMENTATION

4.1 OPERATING SYSTEM

Fedora core 3 (LINUX environment), with a 2.6.9 version of the kernel is used to develop and implement the code. C language and GNU complier gcc is being used in the development. We installed the LKSCTP library and the kernel patch for Stream Control Transfer Protocol (SCTP), whereas for the TCP and UDP protocols the header files comes bundled in the user libraries. SCTP kernel module is loaded each time after rebooting using the modprobe SCTP command. The security layer consists of transport layer protocols with encryption and decryption algorithms over it.

4.2 MEASUREMENT OF BANDWIDTH, JITTER AND DELAY

Network measurement tools are classified into active tools and passive tools. The passive tools create very little additional bandwidth to the network, whereas the active tools measure maximum performance in the network by introducing congestion [9]. Some active tools are netpref, iperf, ping. In our implementation we used ipref to determine the maximum bandwidth and jitter available between the end systems. To measure the bandwidth and jitter, ipref should be run in server mode and client mode separately on two different machines. The volumes of data sent through the iperf to the network can be managed by the user. The data flows can be a UDP flow or TCP/IP flow. In our implementation TCP/IP flows are used to send data for the iperf client. The

command for running a simple iperf server is: iperf -s -i 2. This is done at the command prompt after installing iperf. Here s stands for server and i for reporting the result status every 2 seconds. By default iperf runs for 10 seconds. An increase in the running time of iperf to 20 seconds can be done by including the option -t in the command: iperf -s i 2 -t 20. The command for running iperf on the client side is: iperf -c ip-address-of-server -i 2 -t 20. The results displayed on the console by running the iperf are transferred to a text file at the server side. A function checks the bandwidth text file and jitter text file at regular intervals for changes in the available bandwidth and jitter respectively. In order to measure the network delay iperf is made to run for extended periods of time (950 seconds) and ping requests are made to the client from the server side.

A Ping request gives the delay between the end systems. If it is initiated constantly it will be considered as an active tool. The outputs from the ping request are directed to a delay text file. A function is used to read the bandwidth text file and delay text file as feedback from the network. A network protocol analyzer named Ethereal is used to examine the kind of packets (TCP, UDP, and SCTP packets) in the network. It is made to run in promiscuous mode, which allows the adapter to intercept and read all packets rather than just packets addressed to it.

4.3 SECURITY USING DES

To implement the DES we use function encrypt, which takes two parameters, buffer and flag. It modifies the passed buffer into fixed length bits (plain text or cipher text). If the flag is set to 1 it modifies the buffer to cipher text and if the flag is set 0 it modifies the buffer to plain text. Since this function encrypts and decrypts 64 bit messages, the data that is send and received is converted in to 64 bit by passing 8

characters at a time to another function encode. The key parameter is also stored in an array of 64 bytes as 0's and 1's by passing the key to the encode function. This 64 bytes of 0's and 1's are passed to the setkey function which is in turn by the encrypt function. The crypt.h header gives the required prototypes for the setkey() and encrypt() functions. When compiling it required linking with –lcrypt.

4.4 ADAPTATION USING SAATY'S ANALYTICAL HIERARCHY PROCESS

The Analytical Hierarchy Process (AHP) is used for dynamically selecting the appropriate transport layer protocol based on the feedback from the network and user preference. The monitored QoS parameters (bandwidth, jitter, and delay) from the network and user preference are passed on to the AHP as feedback, which comes up with an appropriate protocol. The protocol preference is then passed on to the protocol stack where the corresponding transport layer protocol is selected for communication. The first step in AHP is to define the goal, criteria, and alternatives. The hierarchy of our approach is given in the below table,

GOAL	Selecting best transport layer protocol
CRITERIA	Bandwidth, Delay, Reliability, Jitter
ALTERNATIVES	TCP, UDP, SCTP; TCP/ UDP/ SCTP with Security.

Table 3: Setting up the Hierarchy.

By applying Saaty's AHP, we will be getting 5 matrices of which four matrices give the relative importance of the alternatives with respect to the criteria. The last matrix gives relative importance among the criteria. The relative importance among

them is found using pair-wise comparisons. The last matrix i.e. the criteria matrix is the one where the pair-wise comparison importance dynamically changes due to the feedback from the network.

4.4.1 AHP AND FEEDBACK FROM THE NETWORK

By opening three text files, one for bandwidth, one for jitter, and one for delay the feedback from the network are written to files. The bandwidth, delay, and jitter values based on network conditions are read by a function. The various possible network conditions are given in Table 5. Consider a case where the available bandwidth is high and delay, jitter between the host machines is very low. Since it is mentioned above that the criteria matrix is the only matrix that changes according to the network conditions, we give high importance to the bandwidth row and very low importance to the delay row in the criteria matrix. Based on the file type given by the user, the reliability factor is decided. Reliability of high importance, medium importance, and low importance is given to the file type text, audio, and video respectively in the reliability row. If the user preferred security for the file transfer, a layer of security is provided using DES. Based on jitter value, the importance level for the fourth row is decided. Hence the importance level of the criteria matrix is determined. The bandwidth, delay, and jitter are the varying values in the criteria matrix, whereas the reliability is a constant factor.

Bandwidth (Kbps)			Delay (ms)			Jitter (ms)		
High	Medium	Low	High	Medium	Low	High	Medium	Low
>60	25-60	<25	>15	5-10	<5	>1.5	0.5-1.5	<0.5

Table 4: High, Medium, and Low Values for measured QoS

Steps involved for calculating criteria matrix priority:

1. Entering pair-wise response into the matrix and the reciprocal values are calculated.

I .Pair-wise response for Criteria:

Criteria	BW	Delay	Reliability	Jitter
BW	1	6	3	3
Delay	1/6	1	½	½
Reliability	1/3	2	1	1
Jitter	1/3	2	1	1

Here Bandwidth is given more importance compared to delay, reliability, and jitter.

II. Reciprocal values:

Criteria	BW	Delay	Reliability	Jitter
BW	1	6	3	3
Delay	0.1667	1	0.5	0.5
Reliability	0.3333	2	1	1
Jitter	0.3333	2	1	1

2. Column values are added together.

Criteria	BW	Delay	Reliability	Jitter
BW	1	6	3	3
Delay	0.1667	1	0.5	0.5
Reliability	0.3333	2	1	1
Jitter	0.3333	2	1	1
	1.8333	11	5.5	5.5

3. Normalization is done by dividing the column sums.

Criteria	BW	Delay	Reliability	Jitter
BW	0.5455	0.5454	0.5454	0.5454
Delay	0.0909	0.0909	0.0909	0.0909
Reliability	0.1818	0.1818	0.1818	0.1818
Jitter	0.1818	0.1818	0.1818	0.1818
	1	1	1	1

4. Normalized row values are added.

Criteria	BW	Delay	Reliability	Jitter	
BW	0.5455	0.5454	0.5454	0.5454	2.1817
Delay	0.0909	0.0909	0.0909	0.0909	0.3636
Reliability	0.1818	0.1818	0.1818	0.1818	0.7272
Jitter	0.1818	0.1818	0.1818	0.1818	0.7272
	1	1	1	1	

Priorities for the overall goal are calculated by taking the average:

Criteria	Priority
BW	0.5454
Delay	0.0909
Reliability	0.1818
Jitter	0.1818

The same steps are followed in determining the matrix for the alternatives. The pair-wise comparison is developed by comparing the alternatives in terms of each criterion.

1. Delay: Pair-wise comparison:

Delay	TCP	UDP	SCTP	STCP	SUDP	SSCTP
TCP	1	2	1/2	5	7	3
UDP	1/2	1	1/3	3	5	2
SCTP	2	3	1	7	9	5
STCP	1/5	1/3	1/7	1	2	1/2
SUDP	1/7	1/5	1/9	1/2	1	1/3
SSCTP	1/3	1/2	1/5	2	3	1

	Priority
TCP	0.254133
UDP	0.153683
SCTP	0.41265
STCP	0.054333
SUDP	0.0336
SSCTP	0.090167

The head-of-the-line blocking property of SCTP helps it to perform better in the case of high delay. Hence SCTP is given more importance compared to other protocols in pair-wise comparison matrix. Following the pair-wise comparison steps mentioned for criteria matrix, SCTP gets high priority.

2. Bandwidth: Pair-wise comparison

Bandwidth	STCP	SUDP	SSCTP	TCP	UDP	SCTP
STCP	1	1/7	1/3	1/2	1/9	1/4
SUDP	7	1	3	5	1/2	2
SSCTP	3	1/3	1	2	1/5	1/2
TCP	2	1/5	1/2	1	1/7	1/3
UDP	9	2	5	7	1	3
SCTP	4	1/2	2	3	1/3	1

	Priority
TCP	0.0551
UDP	0.4244
SCTP	0.1385
STCP	0.0352
SUDP	0.2555
SSCTP	0.0909

UDP performs better when the available bandwidth is high. Hence UDP is given more importance compared to other protocols in pair-wise comparison matrix. Following the pair-wise comparison steps mentioned for criteria matrix, UDP gets high priority.

2. Jitter: Pair-wise Comparison.

Jitter	TCP	UDP	SCTP	STCP	SUDP	SSCTP
TCP	1	1/9	1/3	2	1/7	1/3
UDP	9	1	7	9	2	9
SCTP	3	1/7	1	3	1/5	3
STCP	1/2	1/9	1/3	1	1/5	1/3
SUDP	7	1/2	5	5	1	5
SSCTP	3	1/9	1/3	3	1/5	1

Jitter	Priority
TCP	0.0443
UDP	0.4661
SCTP	0.1027
STCP	0.036
SUDP	0.2767
SSCTP	0.0743

UDP is given more importance in pair-wise comparison matrix, when the jitter is high. High jitter will be caused when the protocols TCP and SCTP are used for real-time streaming, due to retransmission of lost packets. Hence using UDP in these cases will avoid the jitter in real-time streaming. Following the pair-wise comparison steps mentioned for criteria matrix, UDP gets high priority.

4. Reliability: Pair-wise Comparison

Reliability

	TCP	UDP	SCTP	STCP	SUDP	SSCTP
TCP	2	7	1	5	9	3
UDP	1/5	1	1/7	1/2	2	1/3
SCTP	1	5	1/2	3	7	2
TCP	1/2	3	1/3	2	5	1
SUDP	1/7	1/2	1/9	1/3	1	1/5
SSCTP	1/3	2	1/5	1	3	1/2

	Priority
TCP	0.4126
UDP	0.0544
SCTP	0.2541
STCP	0.1539
SUDP	0.0332
SSCTP	0.0905

Since TCP is more reliable compared to UDP and SCTP, it is given more importance. Following the pair-wise comparison steps mentioned for criteria matrix, TCP gets high priority.

4.4.2 SELECTION OF PROTOCOL:

Criteria

Alternative	BW	Delay	Reliability	Jitter
TCP	0.0551	0.2541	0.4126	0.0443
UDP	0.4244	0.1537	0.0544	0.4661
SCTP	0.1385	0.4127	0.2541	0.1027
STCP	0.0352	0.0543	0.1539	0.036
SUDP	0.2555	0.0336	0.0332	0.2767
SSCTP	0.0909	0.0902	0.0905	0.0743

Criteria	Priority
BW	0.5454
Delay	0.0909
Reliability	0.1818
Jitter	0.1818

$$A^*_{AHP} = \max_{M \geq i \geq 1} \sum_{i=1}^{N} a_{ij} W_j \qquad (1)$$

a_{MN} represents the above 6x4 matrix, where M and N represents the alternatives criteria respectively. W represents the criteria priority in 4x1 matrixes. By using equation (1) the best transport layer protocol is selected. The calculation is given below,

Overall Priority TCP:
(0.5454*0.0551)+(0.0909*0.2541)+(0.1818*0.4126)+(0.1818*0.0443)= 0.1362
Overall Priority UDP:
(0.5454*0.4244)+(0.0909*0.1536)+(0.1818*0.0544)+(0.1818*0.4661)= 0.3400
Overall Priority SCTP:
(0.5454*0.1385)+(0.0909*0.4126)+(0.1818*0.2541)+(0.1818*0.1027)= 0.1779
Overall Priority STCP:
(0.5454*0.0352)+(0.0909*0.0543)+(0.1818*0.1539)+(0.1818*0.036)= 0.0586
Overall Priority SUDP:
(0.5454*0.2555)+(0.0909*0.0336)+(0.1818*0.0332)+(0.1818*0.2767)= 0.1987
Overall Priority SSCTP:
(0.5454*0.0909)+(0.09090*0.0901)+(0.1818*0.0905)+(0.1818*0.0743)= 0.0877

Based on the given conditions, UDP gets high priority and it is selected as the best transport layer protocol by the AHP.

4.5 SYSTEM ARCHITECTURE

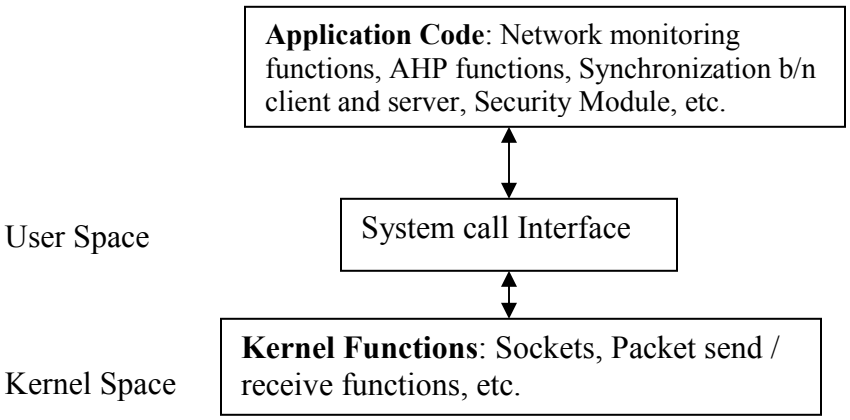

Figure 3: System Architecture

Sockets have different properties such as connection-oriented or connectionless. In the case of connection-oriented sockets the data is allowed to go to and fro as needed, whereas in the case of connectionless there will no open connection and only one message is allowed to pass through at a time. AF_NET and AF_UNIX is the most common socket families used. The AF_NET is used for internet connections and AF_UNIX is used for UNIX inter-process communication [11]. If the socket is unconnected then send(), read() cannot be used to send and receive data, since there won't be any destination address. The socket can be made connected just by calling connect().There are three types of sockets which we have used in this thesis: UDP, TCP and SCTP. These sockets are created and are kept open till the end of the program and switching is done by using one of these sockets. The sockets are created in the main module and passed as parameters between various functions that use them. The port number associated with one of the sockets is taken from the user; both on the client and

server side. The other two sockets are associated with port numbers one higher than the previous port number. The three parameters for the socket function are address family (AF_NET/AF_UNIX/ AF_NS), semantics of communication (SOCK_STREAM**,** SOCK_DGRAM**,** SOCK_RAW)**,** protocol to use. The SOCK_DGRAM provides datagram sockets with unreliable, not sequenced, and duplicated bidirectional data flow. The SOCK_STREAM provides stream sockets with bidirectional, sequenced, unduplicated data flow. In case of SCTP socket creation IPPROTO_SCTP is used as the third parameter [11]. The socket function upon successful creation returns an integer.

The sockets created are of the AF_INET family and the client binds the server address taken as an input at the start of the program. The sockets are closed when the client/user issues a quit message to the server indicating request for a proper close of all connections. Errors arising during creation of sockets and their binding are checked at both client and server sides. UDP sockets sometimes appear to have opened without an error, even if the remote host is not reachable. The error becomes apparent when we try to read/write data from/to the socket. The reason for this is because UDP is a connectionless protocol, which means that the operating system does not establish a link for the socket with the other end until it actually needs to send or receive data. This error is taken care of by having a timeout variable set for the UDP connection on the client side and using a select command on the UDP socket. The select () system call takes four arguments: three 'lists' of sockets, and a timeout. The three socket lists indicate interest in read, write, and exception events for the socket. The function will return whenever the indicated socket fires one of these events and if there is no event of read or write within the timeout period, the function simply returns. When the user space calls the socket ()

system call it gets trapped in the kernel through an interrupt. The control is then transferred to the kernel space. The kernel in addition to error checking, saves the register contents, finds a file descriptor and passes an integer value back to the user space.

The user space contains the Analytical Hierarchy Process module, network monitoring functions which gets the feedback from the network, security module, and module for synchronization between client and server. The user space module executes system command such as, socket(), listen() or sendto(). The sys_socketcall() function located in the /usr/src/linux/net/socket.c file helps in passing system command to the kernel side. The kernel side equivalent function which the user requested is called by this function. When socket () function is called by the user to create a new socket, the sys_socketcall() will pass control to the kernel side, which in turn selects the kernel side equivalent function sys_socket () .

CHAPTER V

RESULTS

Based on the measured bandwidth and the delay the network conditions are classified into different possible types. The table below indicates the preference for each condition from 1 to 9. Value of 1 in the preference column means that network condition is excellent for data transfer, whereas preference of 9 if not ideal for data transfer.

Possible Network Conditions	Preference
Delay Low, Bandwidth High	1
Delay Low, Bandwidth Medium	2
Delay Low, Bandwidth Low	3
Delay Medium, Bandwidth High	4
Delay Medium, Bandwidth Medium	5
Delay Medium, Bandwidth Low	6
Delay High, Bandwidth High	7
Delay High, Bandwidth Medium	8
Delay High, Bandwidth Low	9

Table 5: Possible Network Conditions

5.1 FILE TRANSFER IN FIXED NETWORK CONDITIONS

Here file transfer is done using different transport layer protocols including the proposed dynamic protocols under three different conditions. The conditions are excellent (Delay Low, Bandwidth Low), moderate (Delay Medium, Bandwidth Medium),

and bad (Delay High, Bandwidth Low).At the client side the throughput for the transfer of a 5MB file is measured. TCP, UDP, SCTP are used in transferring the 5MB file under different network conditions and their throughput values are recorded. According to the user preference about the type of the file (text, audio, and video) and level of jitter, the dynamic protocol is used in transferring the file under the three network conditions. The network conditions are changed during the file transfer. The achieved throughput and protocol used for transfer is recorded.

Case 1: Text File Transfer.

The graph below shows the throughput achieved using TCP, SCTP, and Dynamic protocols under excellent, moderate, and bad network conditions. Since a text file is being transferred, the comparison of Dynamic protocol is between TCP and SCTP. From the graph, SCTP has better throughput compared to TCP under all network conditions. Dynamic protocol transfers the text file using TCP protocol in excellent condition, as the network conditions changes to moderate from excellent condition the protocol switches to SCTP and remains in SCTP for the bad condition also.

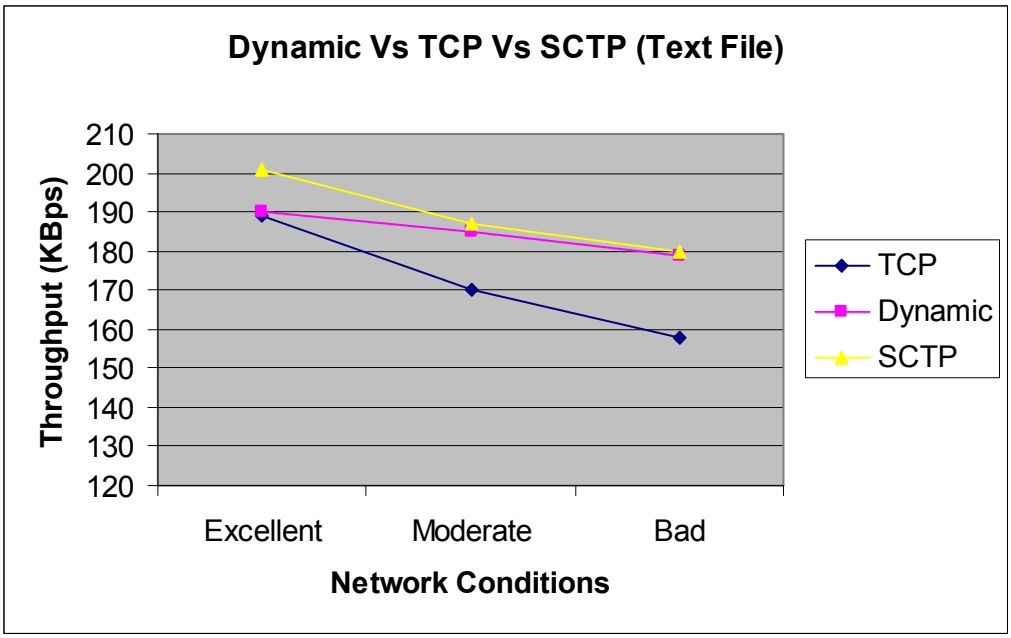

Figure 4: Throughput Comparison for Text File

Case 2: Audio File Transfer.

The graph below shows the throughput achieved using UDP, SCTP, and Dynamic protocols under excellent, moderate, and bad network conditions. As it is an audio file us being transferred, the comparison of Dynamic protocol is drawn between UDP and SCTP. From the graph, UDP has better throughput for the excellent and moderate conditions, and as the condition degrades, SCTP has better throughput compared to UDP. The Dynamic protocol transfers the audio file using the UDP protocol under excellent conditions as packet loss will be minimal. When the network conditions change to moderate, UDP experiences more packet loss. As it is an audio file the system provides reliability by switching to SCTP under moderate and bad network conditions, thus minimizing packet loss.

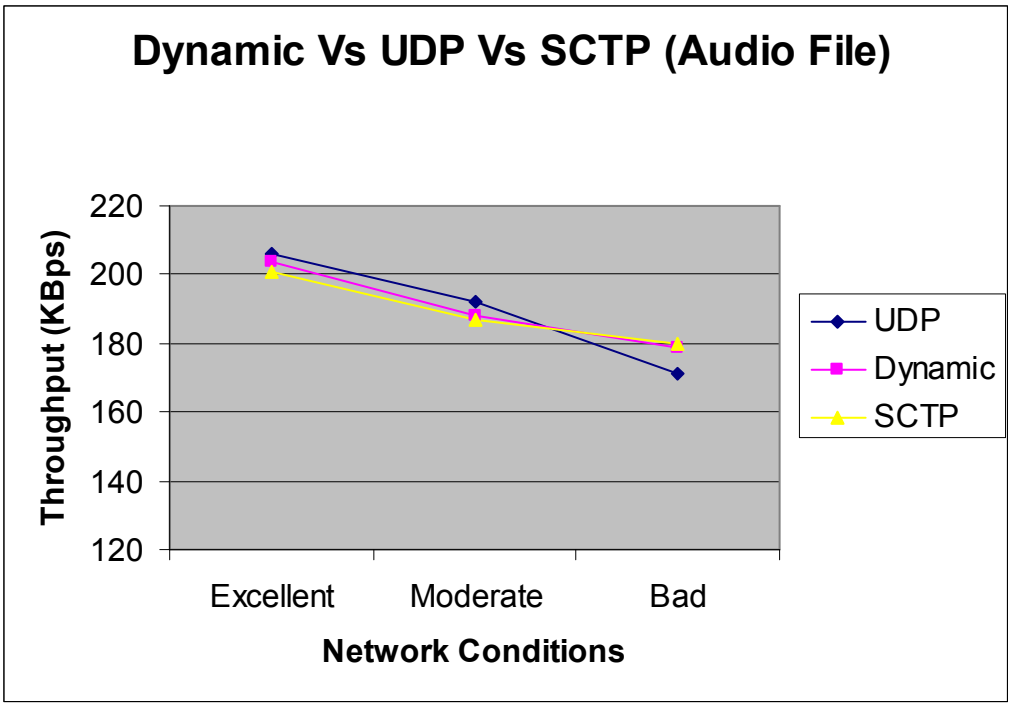

Figure 5: Throughput Comparison for Audio File

Selecting UDP instead of SCTP (worst network condition and high jitter): Since SCTP uses congestion control; high levels of delay will be introduced when packet loss occurs. Retransmission of lost packets will cause high jitter and using UDP in these cases will avoid the jitter in real-time streaming. In real-time streaming we need continuous flow of data, whereas retransmission of lost packets is not needed as it will cause unnecessary disturbance in the audio. In such a scenario the quality of perception decreases. So when we are experiencing high jitter in the worst case scenario, switching to UDP from SCTP is the best option.

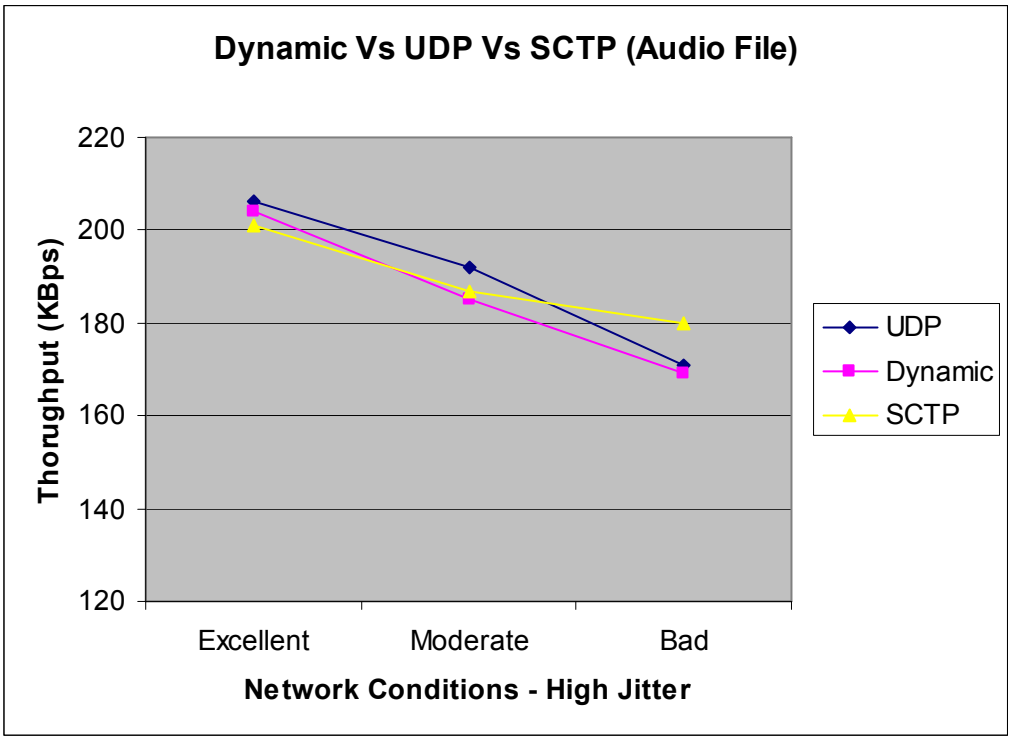

Figure 6: Audio File Transfer – High Jitter

Case 3: Video File Transfer.

The graph below shows the throughput achieved using UDP, SCTP, and Dynamic protocols under excellent, moderate, and bad network conditions. Since a video file is being transferred, the comparison of the Dynamic protocol is between UDP and SCTP. From the graph, UDP has better throughput for excellent and moderate conditions. As the condition degrades the SCTP has better throughput compared to UDP. Since it is a video file, the dynamic protocol transfers the video file using UDP protocol under excellent and moderate network conditions, as reliability is given least importance. When the network conditions changes from moderate to worse, UDP experiences higher packet loss and to minimize that we switch to SCTP.

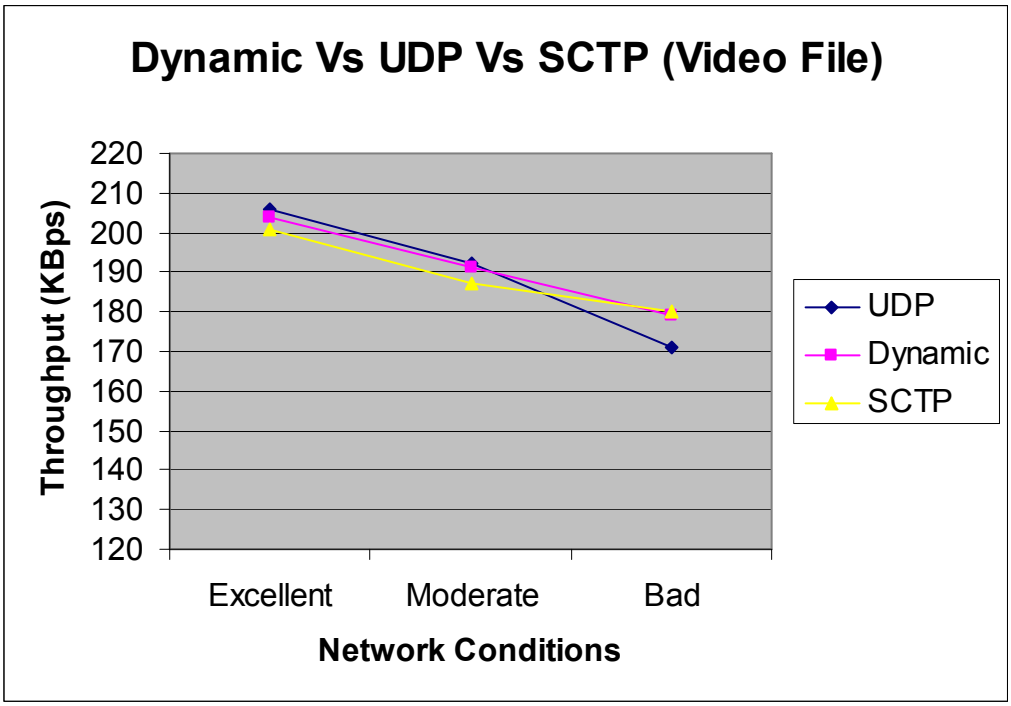

Figure 7: Throughput Comparison for Video File

Selecting UDP instead of SCTP (worst network condition and high jitter):

Since SCTP uses congestion control; high levels of delay will be introduced when packet loss occurs. Retransmission of lost packets will cause high jitter and using UDP in these cases will avoid the jitter in real-time streaming. In real-time streaming we need continuous flow of data, whereas retransmission of lost packets is not needed as it will cause unnecessary disturbance in the video. In such a scenario the quality of perception decreases. So when we are experiencing high jitter in the worst case scenario, switching to UDP from SCTP is the best option.

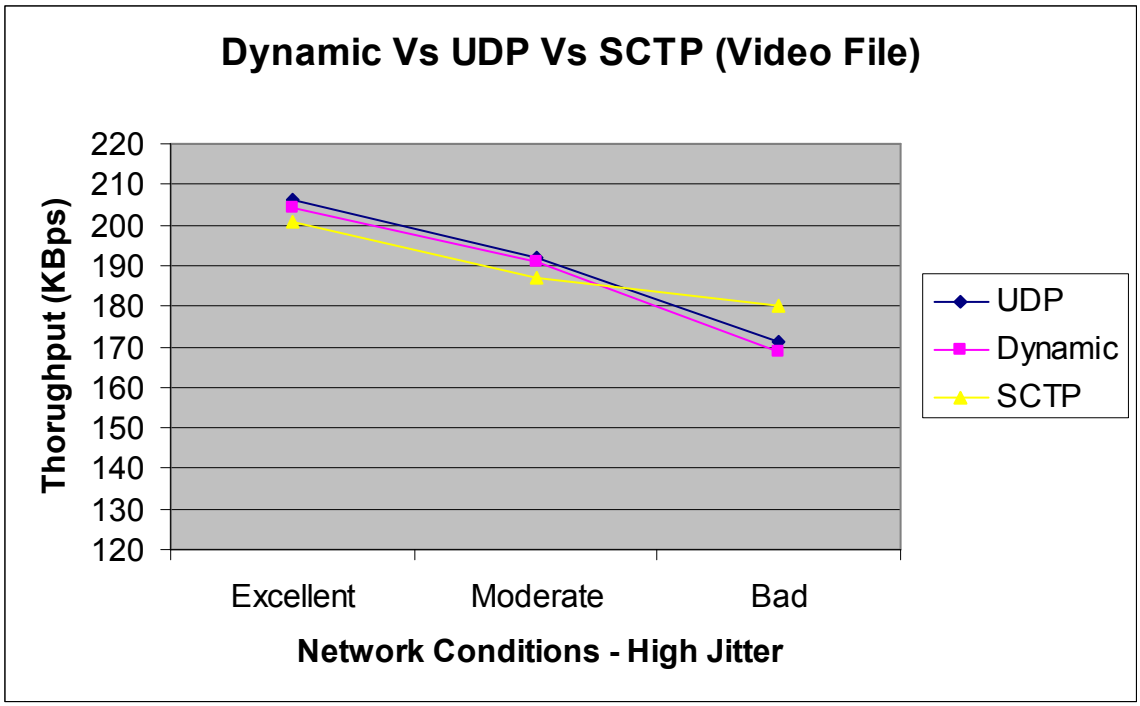

Figure 8: Video File Transfer – High Jitter

5.2 PACKET LOSS

Case 1: Video File Transfer

The packet loss for dynamic and UDP is same in the excellent and moderate network condition, as dynamic follows UDP. But since dynamic follows SCTP in the bad condition, it makes sure every packet is received at the client side without packet loss. For measuring the packet loss a 5MB file is used in the transfer.

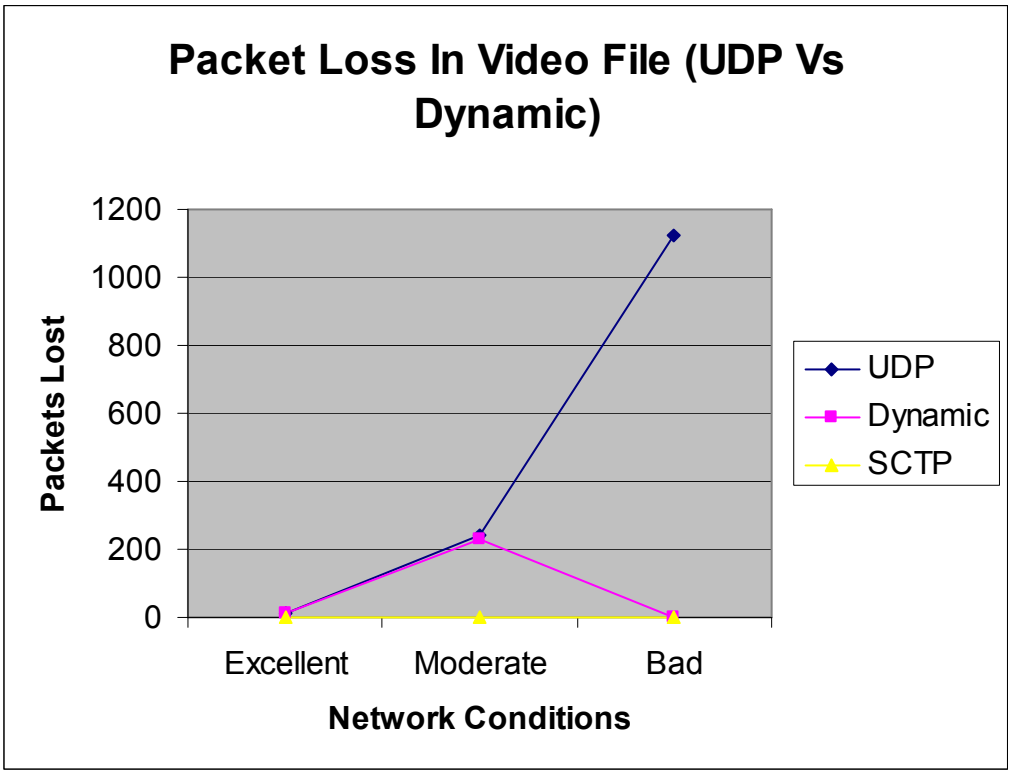

Figure 9: Packet Loss-Video File Transfer

Case 2: Audio File Transfer

The dynamic protocol follows UDP in the excellent network condition. The packet loss is same for both of them. But since dynamic follows SCTP in moderate and bad conditions, it makes sure every packet is received at the client side without packet loss.

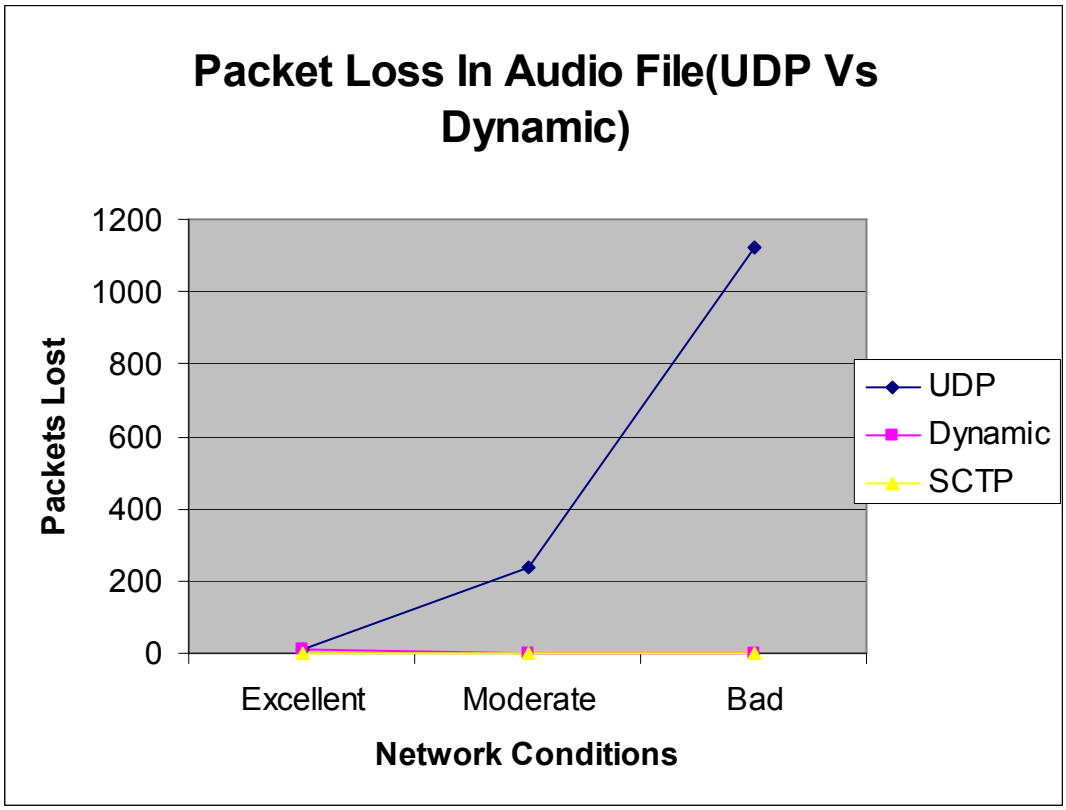

Figure 10: Packet Loss-Audio File Transfer

5.3 FILE TRANSFER IN RANDOM NETWORK CONDITIONS

Here the file transfer is done using different transport layer protocols including dynamic protocol under all the nine network conditions as mentioned in the Table 3. The network conditions are randomly changed during the file transfer. The achieved throughput and protocol used for transfer is recorded.

Case 1: Random n/w Conditions- Text File Transfer.

As we can see from the graph, the dynamic framework uses the TCP protocol when conditions are better i.e. when the delay conditions are low. Whereas when network conditions are not good, the TCP throughput drops drastically compared to SCTP. In that case, it switches to SCTP for the file transfer. Hence it achieves better throughput during the entire file transfer.

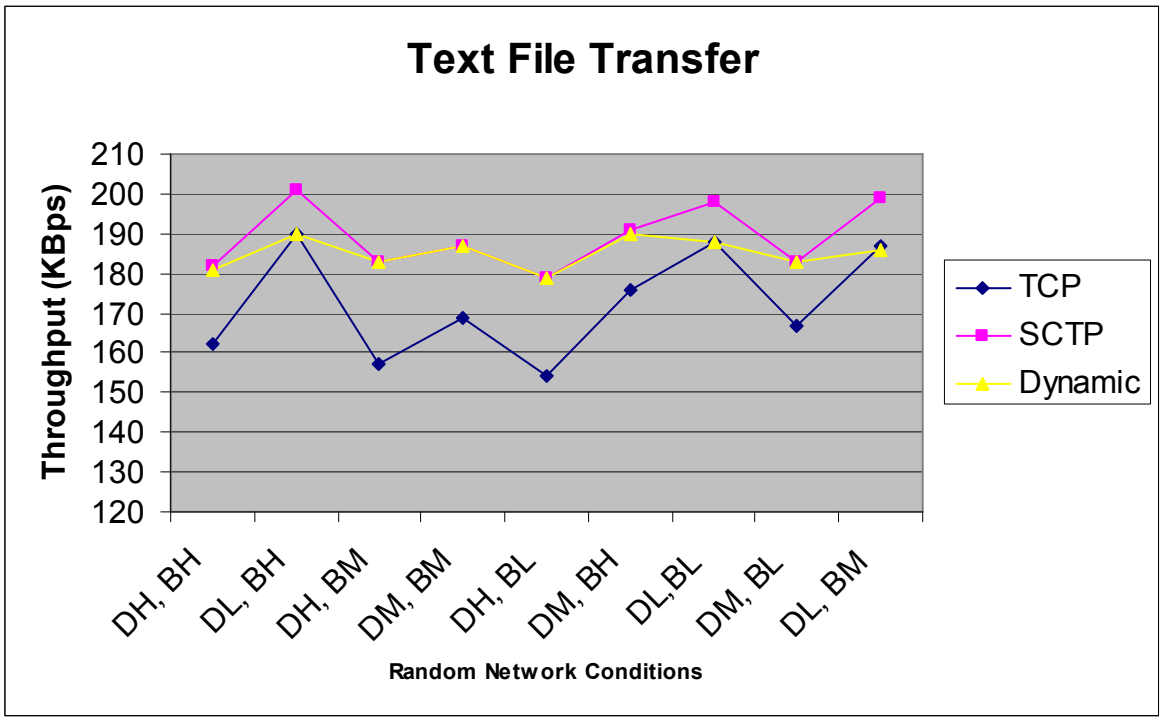

Figure 11: Random n/w Conditions- Text File Transfer.

Case 2: Random n/w Conditions- Audio File Transfer.

As we can see from the graph the dynamic framework uses the UDP protocol when conditions are better i.e. when the delay conditions are low. UDP has better throughput only in some cases, when network conditions are not good (when the

delay is medium or higher). In those cases packet loss occurs in large amounts. Since audio file transfer cannot tolerate large packet losses it switches to SCTP for audio file transfer. Hence it achieves better throughput and minimum packet loss during the entire file transfer.

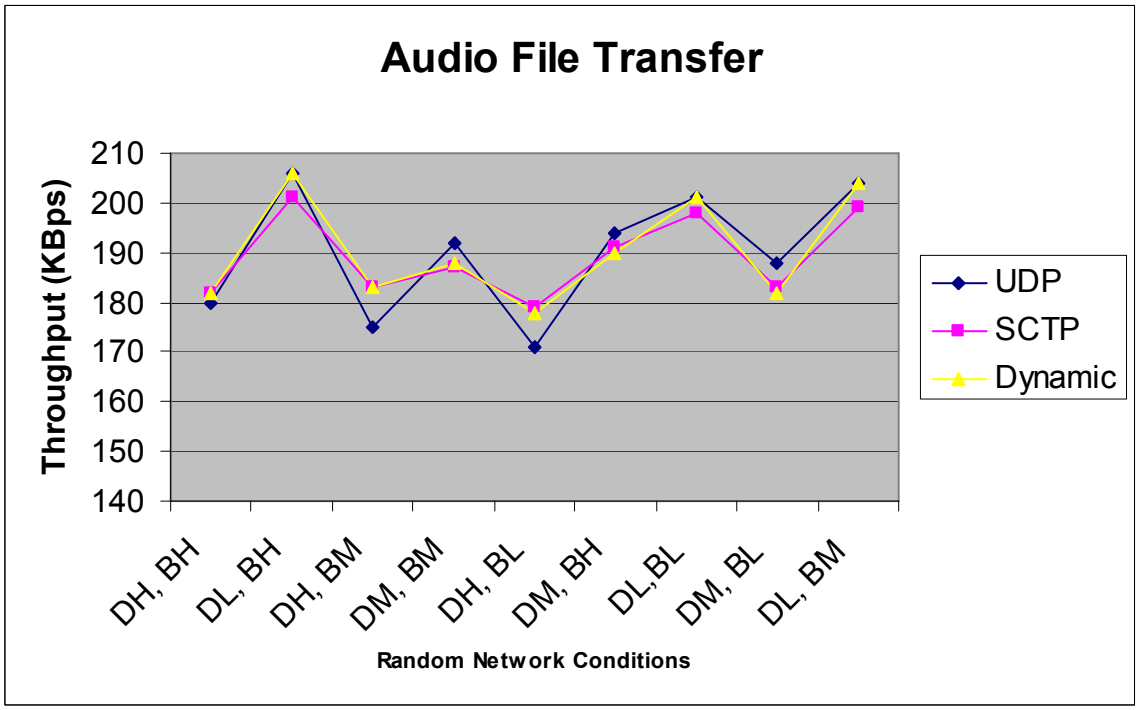

Figure 12: Random n/w Conditions- Audio File Transfer.

Case 3: Random n/w Conditions- Video File Transfer.

As we can see from the graph, the dynamic framework uses the UDP protocol when conditions are better and medium i.e. when the delay conditions are low. The UDP has low throughput and high packet loss when network conditions are not favorable (when the delay is higher). It switches to SCTP only when conditions are not favorable.

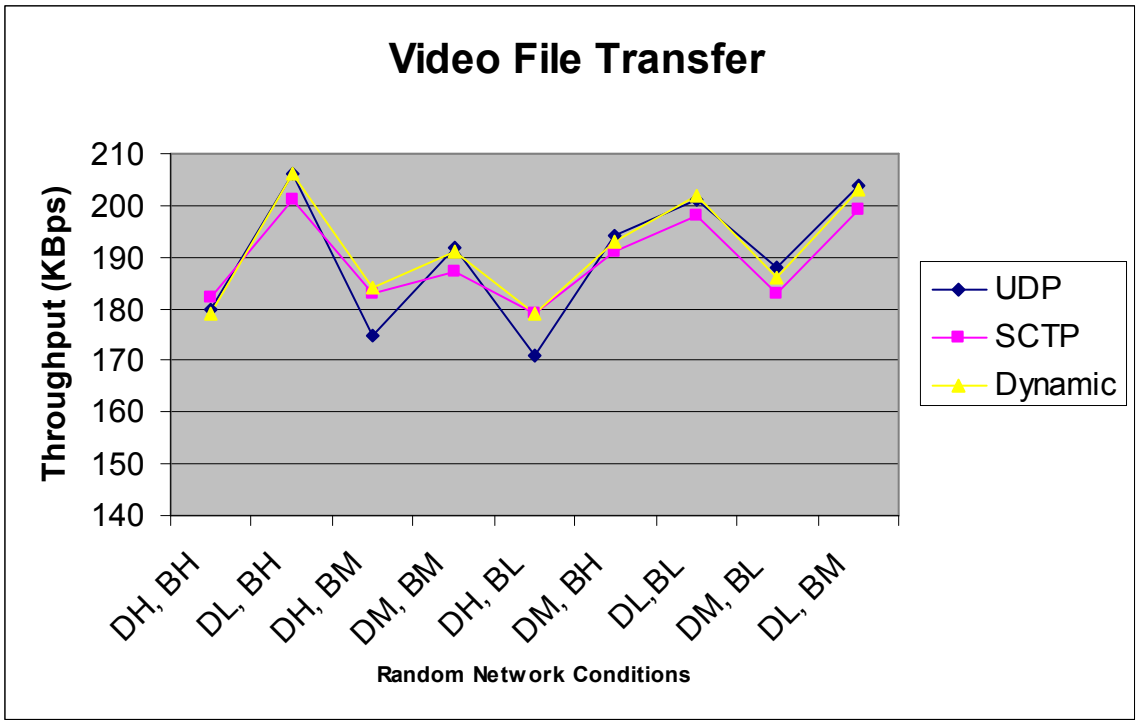

Figure 13: Random n/w Conditions- Video File Transfer.

5.4 OVERHEAD IN USING THE SYSTEM

There is an overhead of 8.1 seconds for 5MB file for message encryption and decryption. Machines used for the file transfer had the following hardware characteristics: Processor - AMD Athlon XP 2000+ with speed of 1666.6 MHz, Processor Cache size- 256 KB, and with 1GB of RAM. While exclusively giving security for the entire file transfer, the throughput considerably reduces. Another overhead in the system is the file reading. The framework at regular intervals opens the two text files to check for the changes in the bandwidth and delay values. This overhead is negligible.

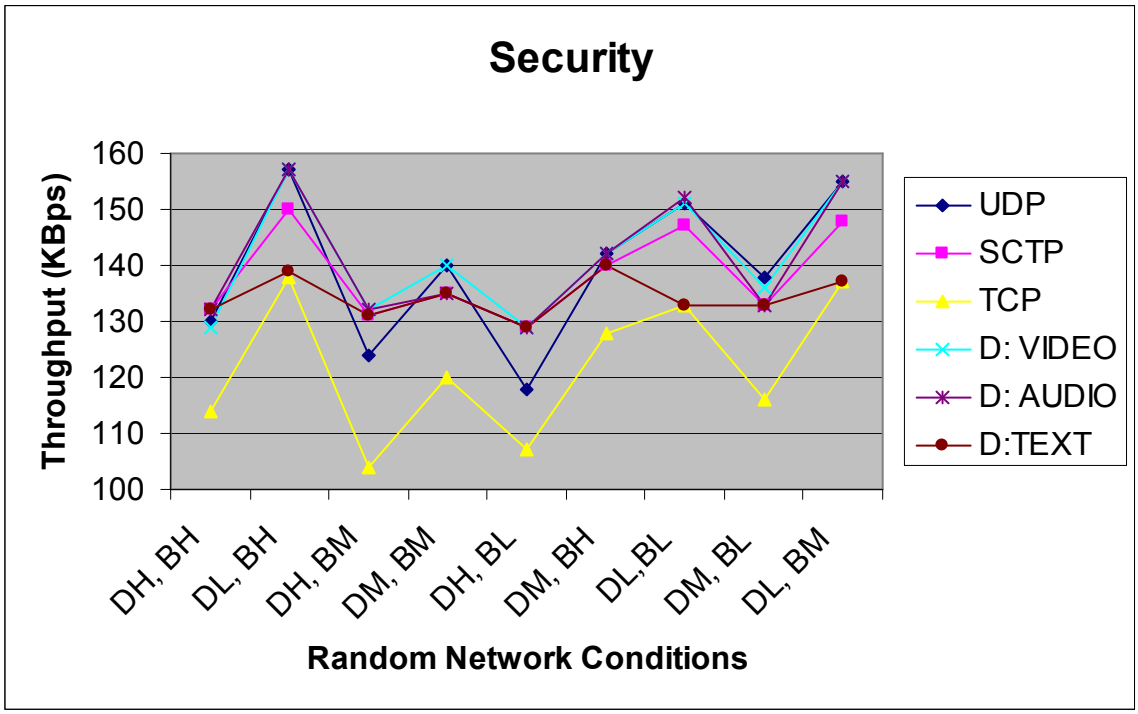

Figure 14: Security overhead

5.5 OVERHEAD CAUSED BY Iperf TOOL

To estimate the overhead caused by running Iperf tool in the background to measure network bandwidth we have taken an average of 30 runs for the transfer of a 1.2MB file with and without running an instance if iperf tool.

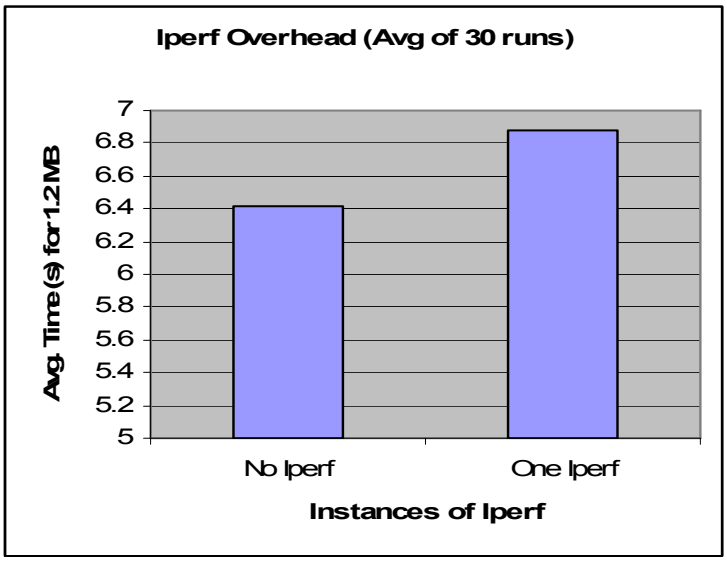

Figure 15: Overhead Graph for running Iperf

The overhead caused was in the range of 0.4 seconds on a machine running the iperf client, with the following hardware characteristics: Processor - AMD Athlon XP 2000+ with speed of 1666.6 MHz, Processor Cache size- 256 KB, and with 1GB of RAM.

CHAPTER VI

Conclusion

This thesis has successfully shown that using the Analytical Hierarchy process as decision making model, transport layer protocols can be switched according to the underlying network conditions and the QoS preferred by the user. The AHP employed here make the decision process very transparent and easy. The result shows that better bandwidth utilization and less packet loss is achieved, using the dynamic framework in transferring a file. As we have seen there are two overheads, adaptability and security. The adaptability overhead does not make an impact on the throughput, whereas the security overhead affects the throughput value by bigger margin. The message encryption acts as an advantage to the application in spite of the overhead.

Future work will focus on increasing the number of QoS parameters monitored from the network and giving feedback to Analytical Hierarchy Process. This will give a wider range of criteria for the AHP and therefore provide a more accurate prediction. Adding new functionalities including other layers such as the network and data link layer to the protocol stack in addition to the existing ones, will also ensures better communication under different network conditions.

REFERENCES

1. 1. Plagemann, T., Saethre, K. A., Goebel, V.: "Application Requirements and QoS Negotiation in Multimedia Systems", in: Proceedings of Second Workshop on Protocols for Multimedia Systems, Salzburg Austria, October 1995
2. B. Stiller, An application framework for the DaCaPo++ project, Proc. 5th Openwork-shop for High Speed Networks, Paris, pp 4-24, March 1996.
3. B. Stiller, D. Bauer, G. Caronni, C. Class, C. Conrad, B. Plattner, M. Vogt and M. Waldvogel, DaCaPo++: Communication Support for Distributed Applications, Technical report, Computer Engineering and Networks Laboratory (TIK), ETH Zrich.
4. A. Richards, The Universal Transport System: An end to end protocol analysis and design, PhD. Thesis, The University of Technology Sydney, 1996.
5. Douglas C. Schmidt, Donald F. Box, and Tatsuya Suda, A Flexible and Adaptive Transport System Architecture to Support Lightweight Protocols for Multimedia Applications on High-Performance Networks.
6. Saaty TL, 1980, The Analytic Hierarchy Process, NY, McGraw Hill.
7. Saaty, T.L., A scaling method for priorities in hierarchical structures, Journal of Mathematical Psychology 15 (1977).

8. Evangelos Triantaphyllou and Chi-Tun Lin, Development and Evaluation of Five Fuzzy Multiattribute Decision-Making Methods, International Journal of Approximate Reasoning, Volume 14, Issue 4, May 1996.
9. T. Charles Yun and Internet2, End to End Performance Tools and Instructions- A Primer, Version: 0.2.12, 04 November 2002. (http://pcp05306333pcs.wanarb01.mi.comcast.net/rev4/sections/reference/primers/e2e%20primer-012.pdf)
10. Data Encryption Standard (DES). Federal Information Processing Standards Publication, Reaffirmed 1999 October. (http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf)
11. W. Richard Stevens, UNIX Network Programming, Volume 1 Second Edition: Networking APIs: Sockets and XTI, Prentice Hall, 1998.

VITA

Rajasekaran Kandaswami

Candidate for the Degree of

Master of Science

Thesis: DYNAMIC PROTOCOL SELECTION FOR COMMUNICATION SYSTEMS

Major Field: Computer Science

Biographical:

Personal Data: Born in Namakkal, TamilNadu, India, on July 4, 1979, the son of Mr. E. Kandaswami and Mrs. Sakunthala Kandaswami.

Education: Obtained Senior High School Diploma from Malco Vidyalaya, India in May 1997. Received Bachelor of Engineering in Computer Science & Engineering from Annamalai University, Chidambaram, India in October 2001. Completed the requirements for the Master of Science Degree at Oklahoma State University in May 2006.

Experience: May 2005 – December 2005: ColdFusion developer, Dr. Donald French, Zoology Department, Oklahoma State University, Stillwater.

Name: Rajasekaran Kandaswami Date of Degree: May 2006

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: Dynamic Protocol Selection for Communication Systems

Pages in Study: 50 Candidate for the Degree of Master of Science

Major Field: Computer Science

Abstract:

Advances in networking are leading towards intelligent communication protocols, particularly with respect to Quality of Service. Traditional communication protocols are static which results in lower performance in the network. Such protocols concentrate on reliability related Quality of Service aspects and ignore aspects such as synchronization, security, and negotiation of Quality of Service. In this thesis we propose dynamically changing communication protocols based on the user requirements and best available Quality of Service provided by the underlying network.

The dynamically changing protocol system is built with differing transport characteristics. These include connection oriented, connectionless, error-free, limited error reduction, delivery of messages in the order in which they were sent or messages with no guarantee about the order of delivery, security, no security compared to the traditional fixed end-system protocols. This will meet fast changing user requirements such as request-response, bulk transfer, security, and real-time data transfers. The dynamically changing communication protocol proposed here begins the communication with the best available protocol and then changes if needed to the best protocol based on measured network Quality of Service parameters such as bandwidth, jitter, and delay. Hence optimal performance is achieved in data communication when loss, errors or congestion is encountered in the network. The Analytical Hierarchy Process, a powerful and flexible decision making process is used for selecting the appropriate communication protocol from the protocol stack based on Quality of Service parameters. Results demonstrate that the proposed dynamically changing protocol achieves better throughput and less packets loss compared to traditional systems under changing network conditions.

Advisor's Approval: Dr. Johnson P Thomas