



**DEVELOPING  
AN EFFICIENT STREAM CONTROL TRANSMISSION PROTOCOL  
(SCTP) MULTI-STREAMING USING PLUGGABLE PRIORITY  
ALGORITHM FOR NETWORK OPTIMIZATION**

**ABDULBAQI KHASHEA AL-HADEETHI**

**JULY 2014**

**DEVELOPING  
AN EFFICIENT STREAM CONTROL TRANSMISSION PROTOCOL  
(SCTP) MULTI-STREAMING USING PLUGGABLE PRIORITY  
ALGORITHM FOR NETWORK OPTIMIZATION**

**A THESIS SUBMITTED TO  
THE GRADUATE SCHOOL OF NATURAL AND APPLIED  
SCIENCES OF  
ÇANKAYA UNIVERSITY**

**BY  
ABDULBAQI KHASHEA AL-HADEETHI**

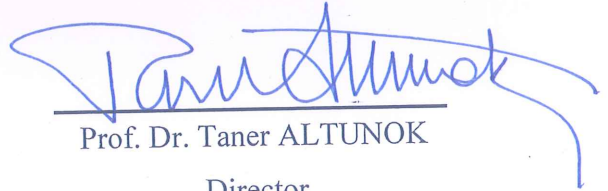
**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE  
DEGREE OF  
MASTER OF SCIENCE  
IN  
THE DEPARTMENT OF  
MATHEMATICS AND COMPUTER SCIENCE**

**JULY 2014**


Title of thesis: **Developing An Efficient Stream Control Transmission Protocol (SCTP) Multi-Streaming Using Pluggable Priority Algorithm for Network Optimization .**

Submitted by: **Abdulbaqi Khashea AL-HADEETHI**

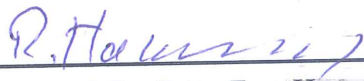
Approval of the Graduate School of Natural and Applied Science, Çankaya University.

  
Prof. Dr. Taner ALTUNOK  
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

  
Prof. Dr. Billur KAYMAKCALAN  
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

  
Assist. Prof. Dr. Reza Zare Hassanpour  
Supervisor

**Examination Date: 31.07.2014**

**Examination Committee Members:**

Assist. Prof. Dr. Ö. Tolga PUSATLI (Çankaya Univ.) .....

Assist. Prof. Dr. Reza Zare HASSANPOUR (Çankaya Univ.) .....

Assoc. Prof. Dr. Fahd JARAD (THK University) .....

## STATEMENT OF NON PLAGIARISM PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct, I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Abdulbaqi Khashea AL-HADEETHI

Signature



Date

: 31/07/2014

**ABSTRACT**

**DEVELOPING**

**AN EFFICIENT STREAM CONTROL TRANSMISSION PROTOCOL**

**(SCTP) MULTI-STREAMING USING PLUGGABLE PRIORITY**

**ALGORITHM FOR NETWORK OPTIMIZATION**

Abdulbaqi Khashea AL-HADEETHI

M.Sc., Department of Mathematics and Computer Science

Supervisor: Assist. Prof. Dr. Reza Zare Hassanpour

July 2014, 60 pages

This thesis introduces a pluggable priorities algorithm for the SCTP protocol(PP-SCTP) as a method of reducing delay in the transmission of main data during periods of low bandwidth access. The SCTP protocol in the transport layer uses a Round-Robin or First-Come, First-Served method to transfer the data. These methods do not provide efficient and flexible choices. We set the priority of the streams in accordance with client demand. The PP-SCTP is useful for applications that send different types of data, such as still images, video, text, and documents. The goal of this thesis is to obtain a more flexible and more efficient choice for the end user or customer. The problem germane to this thesis lies in the transport layer, examples of which include the SCTP protocol with multi-streaming. Coding was written to test the various combinations and permutations of algorithms and protocols. In Chapter 5, we discuss the results of our work with the PP-SCTP priority algorithm, most notably its greater efficiency and the reduction of average latency in comparison to other algorithms and protocols in various

combinations and permutations, including the First-Come, First-Serve data transfer method.

**Keywords:** Transport Layer, SCTP Protocol, Multi-homing, Multi-Streaming, Round-Robin Algorithm, First-Come, First-Served Algorithm, Pluggable Priorities Algorithm, Netbeans Interface Application.

## ÖZET

### **AĞ OPTİMİZASYONU İÇİN TAKILABİLİR ÖNCELİKLİ ALGORİTMASINI KULLANARAK VERİMLİ BİR AKIŞ KONTROL İLETİŞİM PROTOKOLÜ (SCTP) ÇOKLU AKIŞIN GELİŞTİRİLMESİ**

Abdulkaki Khashea AL-HADEETHI

Matematik ve Bilgisayar Bilimi Bölümü Yüksek Lisansı

Danışman: Doçent Dr. Reza Zare Hassanpour

Temmuz 2014, 60 sayfa

Bu tezde düşük bant genişlikli erişim dönemleri sırasında ana verilerin iletimindeki gecikmeyi azaltma yöntemi olarak SCTP protokolü için TAKILABİLİR öncelikleri (PP-SCTP) algoritması sunulmaktadır. İletim katmanındaki SCTP Protokolü verilerin transferi için bir Çevrimsel Sıralı veya İlk Gelen İlk Hizmeti Alır yöntemini kullanır. Bu yöntemler verimli ve esnek seçenekler sunmamaktadır. Biz ise akışların önceliğini istemci talebine göre belirliyoruz. PP-SCTP örneğin hareketsiz görüntüler, video, metin ve belgeler gibi farklı veri türleri gönderen uygulamalar için faydalıdır. Bu tezin amacı son kullanıcı veya müşteri için daha esnek ve daha verimli bir seçenek elde etmektir. Bu tez ile ilgili problem iletim katmanında yer alır ki bunun örnekleri arasında çoklu akışlı SCTP protokolü vardır. Algoritmaların ve protokollerin çeşitli kombinasyonlarını ve permütasyonlarını test etmek için program yazılmıştır. Bölüm 5'te PP-SCTP öncelik algoritması ile yaptığımız çalışmanın sonuçlarını, en dikkate değer sonucu olarak çeşitli kombinasyonlarda ve permütasyonlarda diğer algoritmalara ve protokollere kıyasla daha büyük verimliliğe sahip oluşunu ve ortalama gecikmenin azalmasını tartışıyoruz.

**Anahtar kelimeler:** İletim Katmanı, SCTP Protokolü, Multihoming (çoklu özgüdüm), Çoklu Akış, Çevrimsel Sıralı Algoritma, İlk Gelen İlk Hizmeti Alır Algoritması, TAKILABİLİR Öncelikleri Algoritması, Netbeans Arayüz Uygulaması.



## **ACKNOWLEDGEMENTS**

First of all, my thanks, gratitude and praise to (ALLAH)(Almighty God) for His helpful kindness and guidance through my journey and its final destination: for all this, thank you, my God, all thanks.

I should also wish to say my thanks and deepest gratitude to my lovely country for financially supporting me in my studies.

I should also wish to say my thanks and deepest gratitude to Dr.Abdulbasset Turkey Said, president of the Board of Supreme Audit, for giving me the opportunity to complete my academic development, and as a result, for enhancing my employment prospects.

I should also wish to say my thanks and express deepest gratitude to my family (my mother, my wife, my brother Mustafa) for supporting me and helping me to complete my work.

Finally, I would like to thank, and express my deepest gratitude to my supervisor Assist.Prof. Dr. Reza Zar Hassanpourfor his advice, support and assistance.

## TABLE OF CONTENTS

STATEMENT OF NON PLAGIARISM.....	iii
ABSTRACT.....	iv
KEYWORDS.....	V
ÖZET.....	Vi
Anahtar Kelimeler.....	Vii
ACKNOWLEDGEMENTS.....	Viii
TABLE OF CONTENTS.....	Ix
LIST OF FIGURES.....	Xii
LIST OF TABLES.....	Xiii
LIST OF ABBREVIATIONS.....	Xiv

### CHAPTERS:

1. RESEARCH OVERVIEW.....	1
1.1. Introduction.....	1
1.2. Problem Statement and Motivation.....	3
1.2.1. Problem Definition.....	3
1.2.2. Motivation.....	3
1.3. Scope of the Thesis.....	4
1.4. Organization of the Thesis.....	5
2. BACKGROUND INFORMATION.....	6
2.1. Introduction to the Computer Network.....	6
2.2. Internet.....	7

2.3.	TCP, UDP, and SCTP/IP Protocols.....	7
2.4.	Networks Layers.....	9
2.5.	Properties of SCTP, TCP, UDP.....	11
2.6.	Significant Feature of SCTP.....	12
2.6.1.	Multi-Homing.....	12
2.6.2.	Multi-Streaming.....	13
2.6.3.	Allow Half-Closed Connections.....	14
2.6.4.	Preservation of Message Boundaries.....	15
2.6.5.	Protection Against SYN Flooding Attacks.....	16
2.6.6.	Selective Acknowledgements.....	17
3.	LITERATURE SURVEY.....	18
3.1.	SCTP Overview.....	18
3.2.	SCTP Base Protocol.....	18
3.3.	Algorithms Used with SCTP.....	19
3.3.1.	Using Round Robin Algorithm With The Original SCTP.....	19
3.3.2.	Using the First-Come, First-Served Algorithm.....	20
3.3.3.	A Fair Bandwidth Scheduler.....	20
3.4.	Related Work.....	20
4.	PROPOSED METHOD.....	28
4.1.	Definition of Per-Stream Priority .....	28
4.2.	Specification.....	29
4.3.	System Flowchart.....	32
4.4.	Stream Control Transmission Protocol Design.....	33
4.5.	Data Transfer.....	34
4.6.	Streams.....	35

4.7.	Sender Scheduling.....	35
4.8.	Explanation of Algorithm with Multi-Streaming.....	38
4.9.	General Example.....	39
5.	PROJECT IMPLEMENTATION AND DISCUSSION OF RESULTS.....	41
5.1.	Softwares and Tools.....	41
5.2.	Project Implementation.....	42
5.3.	Simulation.....	44
5.4.	Details of Running.....	45
5.5.	Running The Simulation.....	46
5.5.1.	First Step: Assigning the Number of the Files.....	46
5.5.2.	Second Step: Selecting the Files .....	47
5.5.3.	Third Step: Selecting the Protocol.....	48
5.5.4.	Forth Step: Selecting the Algorithm.....	49
5.5.5.	Fifth Step: Assigning Priority.....	50
5.6.	Comparison of Results.....	54
5.7.	Discussion of Results.....	55
5.7.1.	Results	55
6.	CONCLUSION AND FUTURE WORK.....	59
6.1.	Conclusion.....	59
6.2.	Future Work.....	60
	REFERENCES.....	R1

## LIST OF FIGURES

### FIGURES

<b>Figure 1</b>	Stream Control Transmission Protocol (SCTP)[35] .....	2
<b>Figure 2</b>	SCTP structure [36].....	4
<b>Figure 3</b>	An example of a local area network.....	6
<b>Figure 4</b>	Overview of SCTP position and association.....	12
<b>Figure 5</b>	Multi- homing .....	13
<b>Figure 6</b>	Multi- streaming .....	14
<b>Figure 7</b>	TCP, and SCTP connection termination illustration.....	15
<b>Figure 8</b>	TCP, and SCTP Preservation of message boundaries .....	15
<b>Figure 9</b>	SYN flooding attacks .....	16
<b>Figure 10</b>	TCP, and SCTP connection initiation illustration.....	17
<b>Figure 11</b>	Designed system interaction.....	32
<b>Figure 12</b>	SCTP Packet Format with Common Header and Chunks.....	34
<b>Figure 13</b>	SCTP Data Chunk Format.....	35
<b>Figure 14</b>	SCTP Sender Stream Scheduling.....	36
<b>Figure 15</b>	An SCTP association consisting of four streams carrying data from one upper layer application.....	37
<b>Figure 16</b>	An illustration Showing HOL blocking of individual Stream.....	38
<b>Figure 17</b>	SCTP multi-streaming with priority Algorithm.....	40
<b>Figure 18</b>	Illustration of the NetBeans interface.....	41
<b>Figure 19</b>	assignment file number.....	46
<b>Figure 20</b>	file selection.....	47
<b>Figure 21</b>	protocol selection.....	48
<b>Figure 22</b>	algorithm selection.....	49
<b>Figure 23</b>	priority value assignment.....	50
<b>Figure 24</b>	select file for sending interface.....	51

<b>Figure 25</b>	priority algorithm interface.....	52
<b>Figure 26</b>	selecting priority or FCFS algorithm interface.....	53
<b>Figure 27</b>	select TCP or SCTP protocol interface.....	54
<b>Figure 28</b>	average latency of SCTP using FCFS and PRIORITY respectively.....	55
<b>Figure 29</b>	Average throughput of TCP, SCTP using both FCFS and PRIORITY respectively.....	56
<b>Figure 30</b>	The average waiting time of PP-SCTP interface.....	57
<b>Figure 31</b>	The average waiting time of FCFS-SCTP interface.....	58

## LIST OF TABLES

### TABLES

<b>Table 1</b>	Comparison between OSI and TCP/IP.....	8
<b>Table 2</b>	SCTP, TCP, and UDP Compression.....	11

## LIST OF ABBREVIATIONS

SCTP	.....Stream Control Transport Protocol
TCP	.....Transport Control Protocol
UDP	.....user datagram protocol
TCP/IP	..... internet protocol
OSI	.....Open Systems Interconnection model
FCFS	.....First Come First Served algorithm
PP-SCTP	.....pluggable priority algorithm within SCTP
LAN	.....local area network
DoD	..... US Department of defense
IETF	.....The Internet Engineering Task Force
RFC	..... Request for Comments
CSMA/CD	.....Carrier Sense Multiple Access With Collision Detection
MAC	.....media access control address
SNMP	.....Simple Network Management Protocol
DHCP	.....Dynamic Host Configuration Protocol
HTTP	..... Hypertext Transfer Protocol
ICMP	.....Internet Control Message Protocol
ARP	..... Address Resolution Protocol
DNS	.....Domain Name System
SMTP	.....Simple Mail Transfer Protocol
RARP	.....Reverse Address Resolution Protocol
IMS	..... IP Multimedia Subsystem

RTT	.....	round-trip delay time
TTN	.....	Traffic Transmission Number
SS7	.....	Signaling System No. 7, a set of telephone signaling protocols
PSTN	.....	public switched telephone network
QoS	.....	Quality of service
C++	.....	...(pronounced cee plus plus) is a general purpose programming language
JAVA	.....	is a computer programming language
IDE	.....	integrated development environment
DLL	.....	Dynamic-link library
SMTP	.....	simple mail transfer protocol
API	.....	application programming interface
SS7	.....	signaling system no.7
HOL	.....	head-of-line



## CHAPTER 1

### RESEARCH OVERVIEW

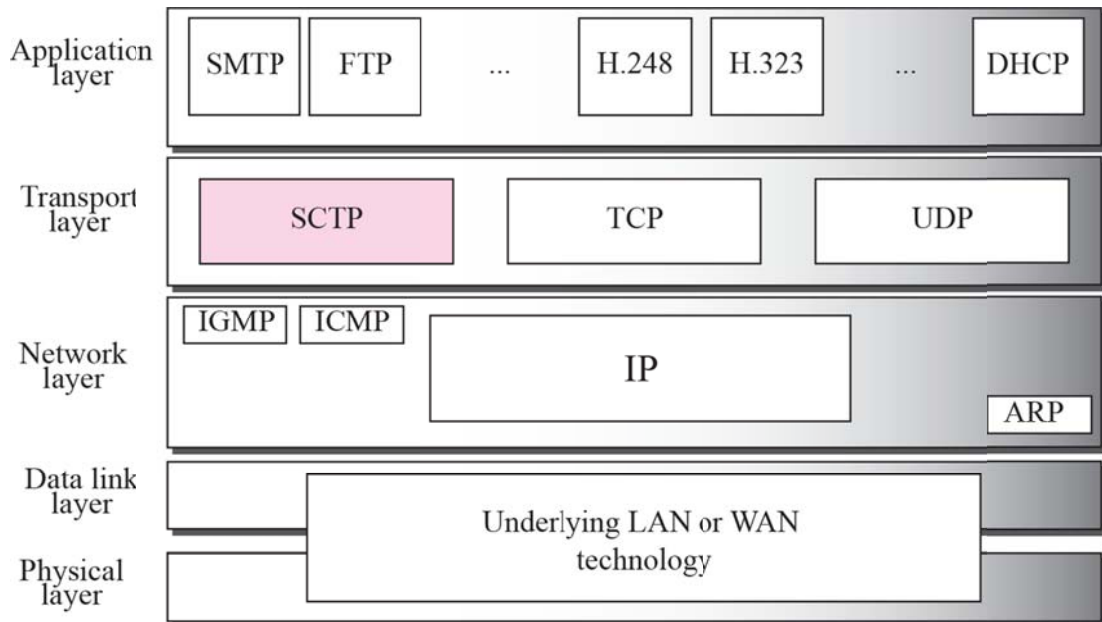
#### 1.1. Introduction

The Stream Control Transmission Protocol (SCTP) is a fairly new transport protocol. It is connection oriented and message oriented and uses a four-way handshake. Many features were inherited from the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP); nevertheless, it has distinct new features such as multi-homing and multiple streams.

Multi-homing is the use of multiple addresses for fast failover. Multiple streams are unidirectional logical channels within an SCTP connection as will be described in detail in the following section. SCTP also has an extensible packet format consisting of a common header and chunks. Multiple smaller chunks can be bundled within a packet. Nagle's algorithm, known from TCP, is used to bundle as many messages as possible. DATA chunks carry user data and control chunks are used to transfer SCTP-related control information between the SCTP endpoints, such as for association setup and teardown.

SCTP is mainly used for telephone signaling, which was originally developed for monitoring systems in addition to other purposes. Implementations are available for recent versions of many modern operating systems. It is not mentioned with regard to multiple streams how scheduling should be carried out. Different implementations use different strategies; e.g., some systems use a Round-Robin algorithm, while Linux and Solaris use the First-Come, First-Served algorithm.

From this it follows that there are several degrees of freedom to achieve optimization. In this dissertation, we will address the possibilities. [16]



**Figure 1:** Stream Control Transmission Protocol (SCTP) [34]

It is a new typical for general-function transportation proposed by the Internet Engineering Task Force (IETF). Stream Control Transmission Protocol (SCTP) addresses application and safety gaps left open by its predecessors, namely the Transport Control Protocol (TCP) and User Datagram Protocol (UDP).

Internet protocol based networks mainly use either the Transmission Control protocol (TCP) or the User Datagram Protocol (UDP) for data transport. On the other hand, these two general-function protocols supply disorganized services and do not ideally satisfy all application requests.

The general function Stream Control Transmission Protocol is considered to develop the scope further than TCP and UDP. SCTP was developed from a telephony signaling protocol for IP networks. Today, this protocol is a planned Internet Engineering Task

Force standard (RFC 2960). Both TCP and SCTP provide reliable, full-duplex connections and a means to control network overcrowding. However, unlike either TCP or UDP, SCTP offers new access options that are mostly preferred for telephony signaling and multimedia applications.[16]

## **1.2.Problem Statement and Motivation**

### **1.2.1.Problem Definition**

As mentioned above, SCTP has multiple streams per association as its key feature in contrast to TCP. However, the inherent problem with this feature is that the algorithm used in SCTP interface chunks assigns a Traffic Transmission Number (TTN) to each SCTP chunk.

The SCTP protocol description has not mentioned how to implement multiple streams scheduling on different across platforms. Normally the standard SCTP uses Round-Robin or perhaps First-Come, First-Served. Similarly, this specific pattern does not present an efficient and flexible choice and is not suitable for different applications. [15] In our thesis, we proposed to optimize the multi-streaming feature of SCTP with pluggable scheduling. Therefore, users will be able to customize the priority of the multi-stream scheduling algorithm of SCTP depending on the particular application at time of use rather than Round-Robin or First-Come, First-Served algorithms.[16]

This proposal will add a new priority to SCTP, thereby making it more efficient and attractive. Moreover, the scheduling algorithm can be loaded or unloaded at run time.

### **1.2.2.Motivation**

Throughout the previous discussion, we mentioned that SCTP has been developed to handle various text, conversation, or multimedia applications between endpoints during one connection. Each connection consists of several streams and each stream would carry any type of data packet. In this thesis, we concentrate on giving order rank for each application of relative importance.

Additionally, we will apply a pluggable priority scheduling to the SCTP socket interface.

Traditionally, transmitting different types of data from an SCTP user application to the SCTP chunk level in parallel between endpoints has depended on inefficient approaches. The following figure illustrates the SCTP architecture concept:

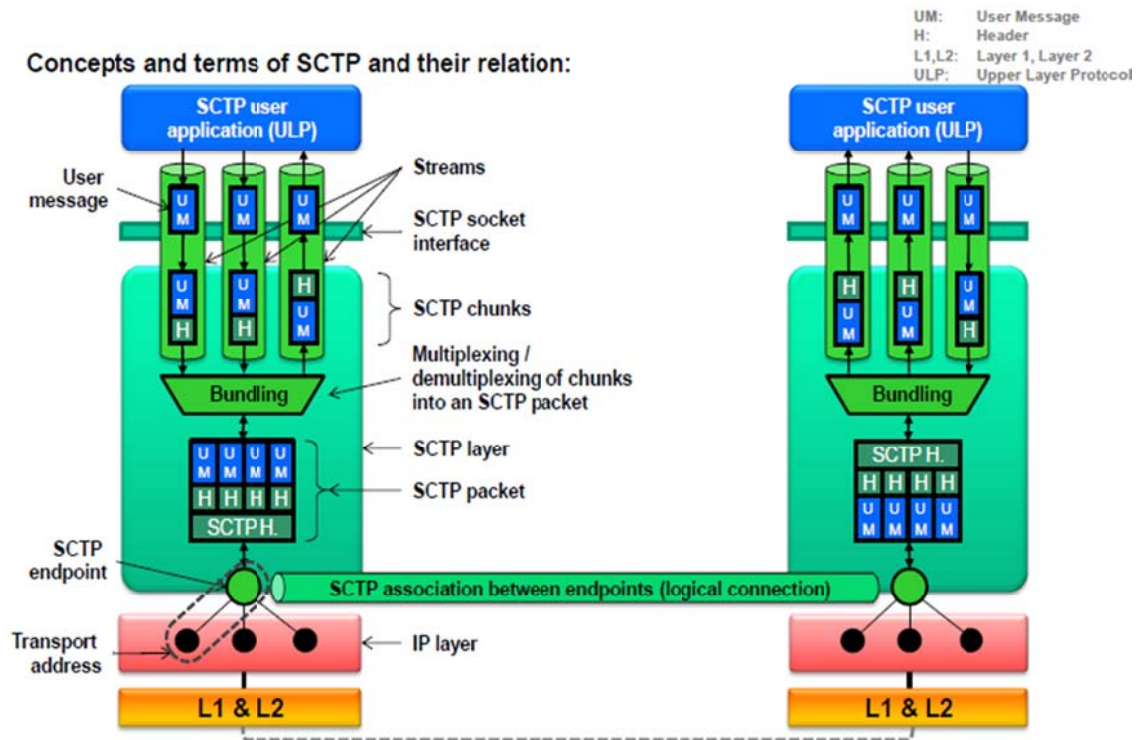


Figure 2: SCTP structure[35]

An SCTP stream is a logic connection and unidirectional channel which produces an end-to-end association. An SCTP connection consists of multiple streams during association setup. Each stream has an autonomous send and receive chunk buffer. The stream and chunk buffers exist while the SCTP association is active.

### 1.3.Scope of the Thesis

We are going to look inside the Stream Control Transport protocol(SCTP)with the intention of enhancing this protocol. SCTP was developed by the IETF committee – Signal Transmission working groups’ efforts (SIGTRAN). They strove to produce a

model identical to the switching network which interacts with IP networks. This produced an SCTP which is responsible for call control signals using IP networks. Initially, SCTP was used solely within large telecommunications companies. The User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) were not sufficient; however, they provide inspiration for the SCTP model. The TCP and UDP protocols lack multi-homing. Furthermore, they are not able to send information to alternate addresses when a primary address becomes unavailable. The SCTP was concealed behind the veil of the support-plane networks; therefore, it did not become public as was the case for TCP. Now, SCTP has become exceedingly important for many Internet applications.

#### **1.4. Organization of the Thesis**

In this section, we will divide the phases of our work into chapters thus:

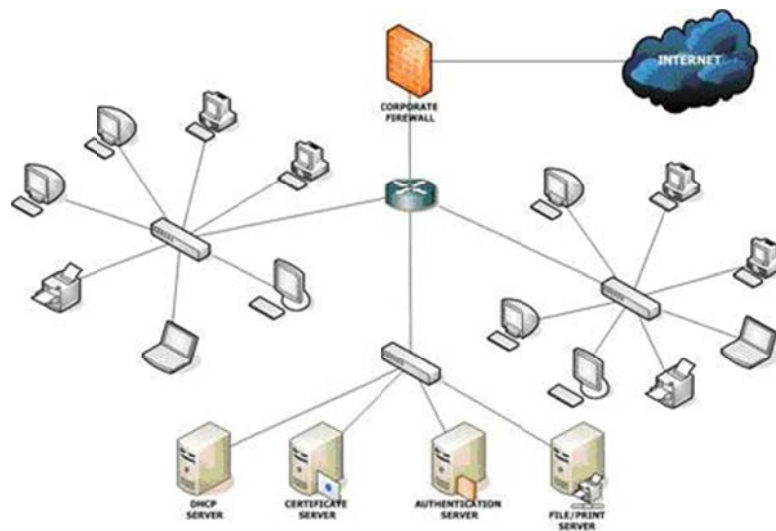
- ❖ In Chapter One, we start with an overview about The Stream Control Transmission Protocol (SCTP) and specify our work on this protocol, after which we specify the problem and motivation, which is then put into the scope of the thesis.
- ❖ In Chapter Two, we discuss background information about network types, network layers and protocols (i.e., overviews of TCP, UDP and SCTP).
- ❖ In Chapter Three, we show related works from other people.
- ❖ In Chapter Four, we propose methods to enhance the network through the Stream Control Transmission Protocol (SCTP) by writing code priority algorithms for network optimization.
- ❖ In Chapter Five, we present the tools and values used, simulation details, a snapshot of the simulation, the results of the simulation and the results of our designed project. Then we implement and collect the results and compare and contrast them with the results of other people by a snapshot of simulation.
- ❖ In Chapter Six, we discuss conclusions and future works.

## CHAPTER 2

### BACKGROUND INFORMATION

#### 2.1. Introduction to the Computer Network

A computer network is a cluster or group of devices such as computers, printers and other devices connected together so that they can communicate with each other for different applications. Figure 3 gives an example of a type of network known as a local area network (LAN).[1,2]



**Figure 3:** Example of a local area network

Actually, we can categorize network configuration to two styles of network: the client-server network and the peer-to-peer network.

For larger networks, client-server networks are more apposite, as there is a server in the center which works as a large storage device for documents, files and other applications on the network. In other cases, this server may also be private.

Normally universities use a client-server style of network. Computers for students, staff, visitors, etc. function as client machines with administrator access to the server.

For small or medium-sized local networks, the peer-to-peer pattern is more appropriate. Here, computers interact with each other and not necessarily with a server. This type of network presents more challenges in terms of security.

In these types of network, all computers have the same class, but in some applications there may be a hierarchy or clustering diffusing.

Both types have two networking styles: wired networks and wireless networks. However, both are applied as one of the main computer networks.

A network protocol is a set of defined actions and conventions for communication between network devices. Network protocols typically use packet switching techniques to receive and send data as packets or chunks. Network protocols define the mechanisms of how communication occurs for devices to connect with each other. They also define the formatting of the data packages to be sent and received. A protocol might support reliability and security. Many variable computer network protocols have been developed and designed for specific purposes. The most well known modules used in networks are Open Systems. The primary architecture model of the Open Systems Interconnection (OSI) protocol consists of seven layers for both inter-computing and inter-networking communications.

## **2.2. Internet**

The Internet is the network of networks consisting of countless devices (millions of devices in total) connected to each other at any given moment. These networks might be small domestic, academic, business and government networks, or any other local network, which together exchange information and interact with different applications.

## **2.3. TCP, UDP and SCTP/IP Protocols**

These protocols are suitable for Internet techniques. The US Department of Defense (DOD) has promoted the TCP/IP protocol as a military project. Nowadays, most Internet protocols are designed and evolved by the IETF committee. IETF was initially financed

by the US government, but now it is an independent organization. The Internet Architecture Board (IAB) harmonizes TCP/IP protocols and guides Internet growth. The Request for Comments (RFC) equips drafts as the best documentation for TCP/IP protocols, which are discussed and accepted by the IETF. All drafts are accessible online free of charge and each draft has a reference number. [3]

Both OSI and TCP/IP network models were separate network protocols and TCP/IP was on an evolution stage. There were relations between the designers of both models (OSI, TCP/IP) when the old standard model (OSI) was published. OSI consists of seven normal layers, where as TCP/IP consists of four normal layers. The OSI model has been penetrative in TCP/IP; therefore, there are similarities between the old standard model (OSI) and new developed model (TCP/IP) in terms of terminology. The following table compares the TCP/IP and OSI network models.

OSI Model		TCP/IP Model	
1.	APPLICATION LAYER.	1.	APPLICATION LAYER.
2.	PRESENTATION LAYER.		
3.	SESSION LAYER.		
4.	TRANSPORT LAYER.	2.	TRANSPORT LAYER.
5.	NETWORK LAYER.	3.	INTERNET LAYER.
6.	DATALINK LAYER.	4.	NETWORK ACCESS LAYER.
7.	PHYSICAL LAYER.		

**Table 1:** Comparison between OSI and TCP/IP

As we can see from the table above, the first three layers of the OSI model are represented in the TCP/IP model in one layer. Similarly, the Data Link Layer and the Physical Layer of OSI are represented by the Network Access Layer in the TCP/IP model.



## **2.4. Network Layers**

We are going to take a look of the TCP/IP layers, as we are going to work in this thesis on the transport layer. [4,5]

### **1. Layer 1: Network Access Layer**

This layer describes and explains how data is physically represented in order to be sent via the network: this includes bits of data converted to signals. There is an ability to exchange data between nodes of local network depending on the MAC address, which explains its being called a network layer. There are different technologies such as Ethernet, Token Ring, FDDI, X.25, Frame Relay etc. included in network layer. Ethernet, the most popular LAN, is a sample to describe the network access layer. There are some methods used with Ethernet, one of which is named CSMA/CD, in which every node has the same priority to access the physical medium and which can use only the free wire channel to send data. When a node wishes to put data on the wire, it first checks whether the channel is engaged by another node or whether any of the network nodes are active. When it detects that there is traffic on the node, it waits until the channel is free, after which it places its signal onto the medium. In the Collision Detection Method, the sender node continues to list the channel state after sending its signal since if two nodes place those signals onto the channel at the same time, they will collide with each other and destroy the data. Thus, the node will retransmit its data when it detects a collision.[1,2,4,5]

### **2. Layer 2: Internet Layer**

This is the second layer of the model (TCP/IP), and the location of this layer is between the Network Access Layer and the Transport layer. The Internet Layer puts data into packets known as IP data grams. The header of each packet contains the logical addresses of both the source and destination nodes and other control information, but the body of the packets contains the user data which is intended to be routed to the destination. The Internet Layer is also responsible for forwarding the IP datagram between intermediate nodes. The packet switching network does not depend upon a

connection between networks. This layer is known as the Internet Layer. The main function of this layer is to give hosts the ability to send packets to any destination node independent of the connection. At the destination node, the packets maybe received in the wrong order. Reordering the received packet is the task of the higher layer known as the Application Layer. There are many different protocols for this layer; however, the main protocols are ICMP, IP, ARP and so on.

### **3. Layer 3: Transport Layer**

The Transport Layer is the third layer of the four layer TCP/IP model. The location of this layer is between the Application Layer and the Internet Layer. The main function of this layer is to grant access between two devices (source and destination nodes)in order to carry on a conversation. The main protocols enclosed at the Transport Layer are TCP (Transmission Control Protocol), UDP (User Datagram Protocol) and SCTP(Stream Control Transport Layer).

### **4. Layer 4: Application Layer**

The fourth layer of the TCP/IP model is the Application Layer at the top. There are relations between Application Layer and Transport Layer. This layer defines TCP/IP application protocols, and all higher level protocols founded in this layer.

In this layer, there is an interface between the following layer (Transport Layer) services to use the network.

The higher level protocols in this layer are DNS,HTTP, TELNET, FTP, SSH, TFTP, DHCP, SMTP, and RDP, etc.[4,5]

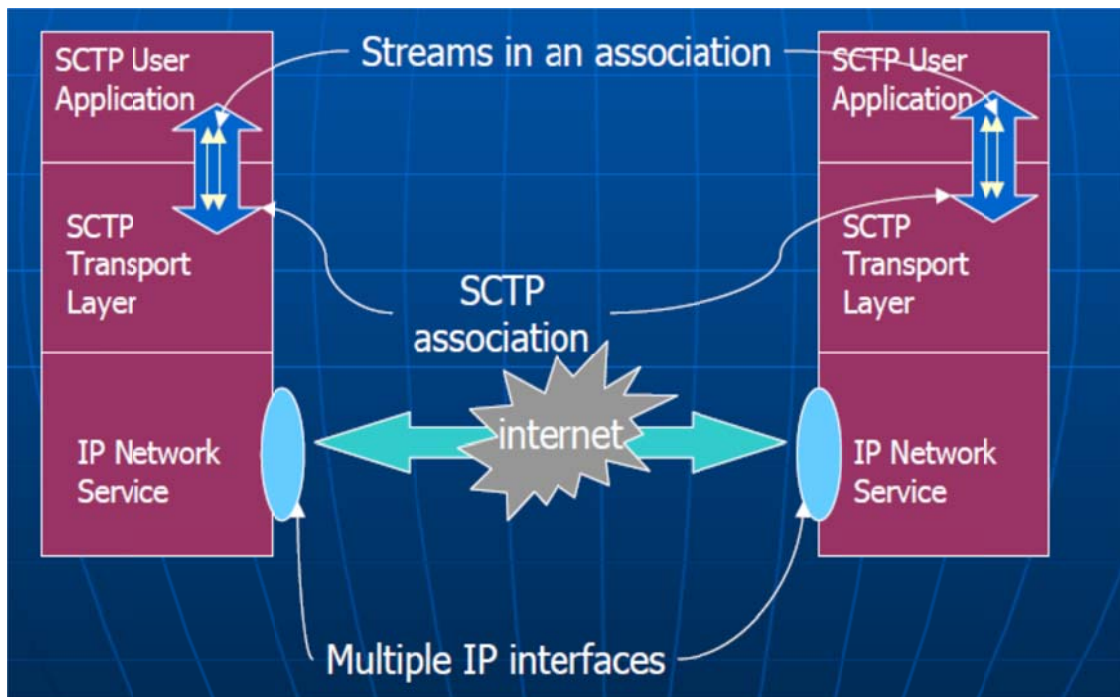
## 2.5.Properties of SCTP, TCP and UDP

The table below provides a comparison between the main transport layer protocols; so we will overlook the key features of SCTP, where SCTP prevails over TCP and UDP by merge advantages on each.[3,6]

Feature/Service	SCTP	TCP	UDP
Allow half-closed connections	No	yes	N/A
Application PDU bundling	Yes	Yes	No
Application PDU fragmentation	Yes	Yes	No
Congestion control	Yes	Yes	No
Connection-oriented	Yes	Yes	No
ECN capable	Yes	Yes	No
Flow control	Yes	Yes	No
Full duplex	Yes	yes	Yes
Multi-homing	Yes	No	No
Multi-streaming	Yes	No	No
Ordered data delivery	Yes	Yes	No
Partial-reliable data transfer	Optional	No	No
Path MTU discovery	Yes	Yes	No
Preserve message boundaries	Yes	No	Yes
Protect against SYN flooding attacks	Yes	No	N/A
Pseudo-header for checksum	Uses vtags	Yes	Yes
Reach ability check	Yes	Yes	No
Reliable data transfer	Yes	Yes	No
Selective acknowledgements	Yes	Optional	No
Time wait state	For vtags	For 4-tuple	N/A
Unordered data delivery	Yes	No	Yes

N/A means not applicable

**Table 2:** SCTP, TCP, and UDP comparison

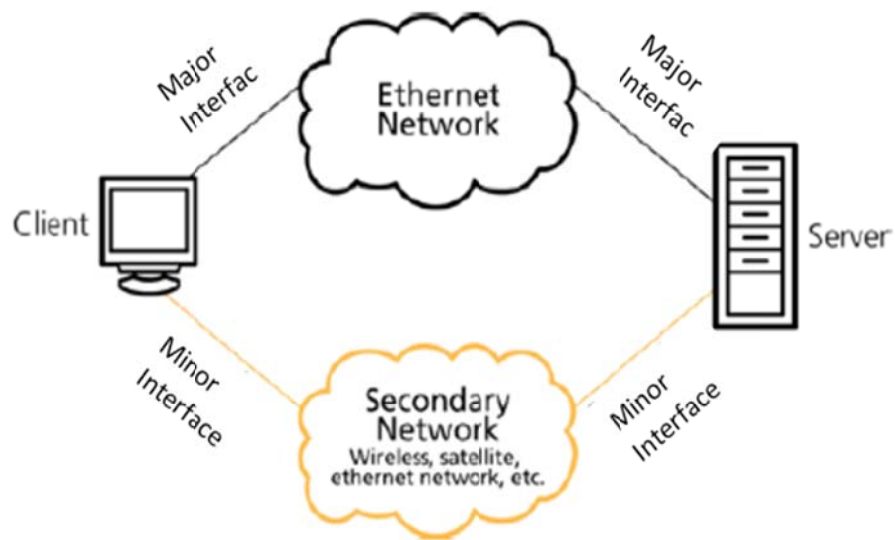


**Figure 4:** Overview of SCTP position and association.

## 2.6. Significant Features of SCTP

### 2.6.1. Multi-homing

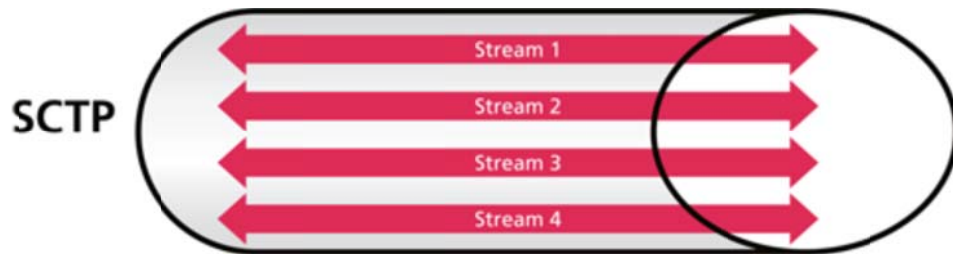
SCTP was essentially developed to address and manage the call establishing of telecommunications over IP as telecommunications services are too sensitive and have limited tolerance to time delays. The most important feature which SCTP advances is multi-homing. This feature gives the network system an ability to establish multi interfaces. One of these interfaces is a major interface and the remaining interfaces are minor interfaces. All, or some, minor interfaces may be used at any moment. When the major interface fails, one of the minor interfaces will take over. Thus, communications will progress without delay to establish a new interface as in TCP and continue transferring to the interface. There are different algorithms which have been applied to select the major connection interface and other minor interfaces. The RTT is used for all interfaces. When the RTT determines that the major interface is slower than a minor interface, it might exchange the state of the interfaces.[7,8,9]



**Figure 5:** Multi-homing.

### **2.6.2. Multi-streaming**

The second important feature of the SCTP model is allowing establishment of simultaneous multi-streams through one session. Each stream might have different destinations or multiple streams might have the same destination. However, it is of utmost importance to preserve data chunk boundaries. For instance, a system will not send pieces of the same data chunk through more than one stream. In fact, each message has to travel through one stream. This is contrary to TCP, in which only one stream is established within a connection. When one of the streams in SCTP is blocked for some reason, the other streams can continue and use another stream to handle the connection of the blocked stream.[7,9,10]

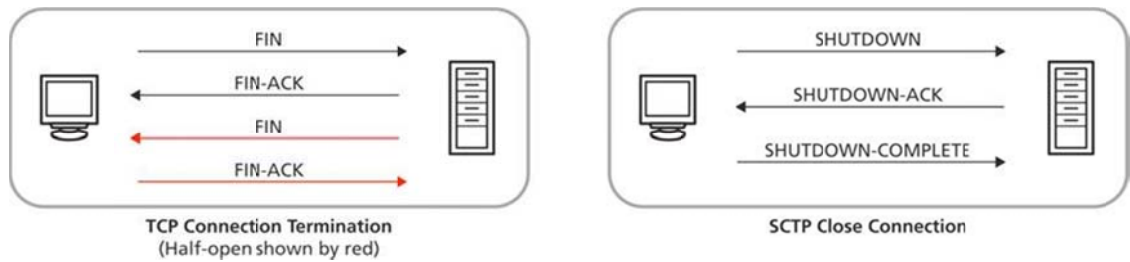


**Figure 6:** Multi-streaming illustration.

While handling multi-streaming within SCTP, only the operating system of the sender is responsible for using the best algorithm for multiplexing and de-multiplexing chunks into STP packets.[7]

### **2.6.3. Allowing half-closed connections**

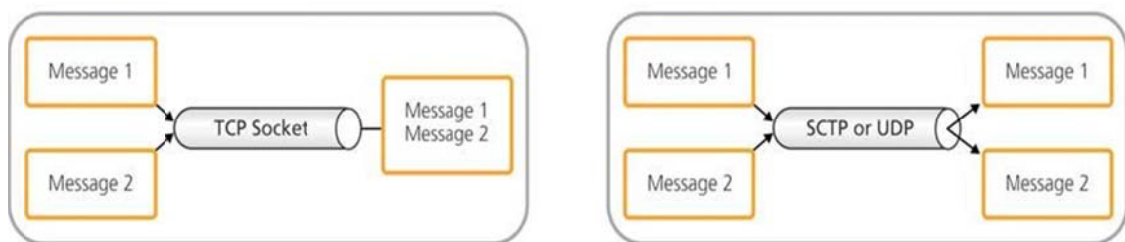
One of the new features that SCTP provides is the half-closed connections which occur when only one of the connected pairs believes that the connection has shut down. This is contrary to the TCP model, which uses a four-way termination decree. A four-way termination decree consists of bidirectional exchange final messages of the “FINAL REQUEST” message, “FINAL ACKNOWLEDGEMENT” message, “FINAL ACCEPTANCE” message, and the last, but not least, “FINAL ACKNOWLEDGEMENT” message. The half-open connection occurs just while acquiring the “FINAL ACKNOWLEDGEMENT” message and in this case, one of the connection sides will assume that the connection is still active while the second side has suspended it. SCTP eliminates this confusion by applying a three-way shutdown comprised of a SHUTDOWN MESSAGE, a SHUTDOWN ACKNOWLEDGEMENT MESSAGE, and a SHUTDOWN COMPLETION MESSAGE.[11]



**Figure 7:** Illustration of connection termination of TCP and SCTP.

#### 2.6.4.Preservation of message boundaries

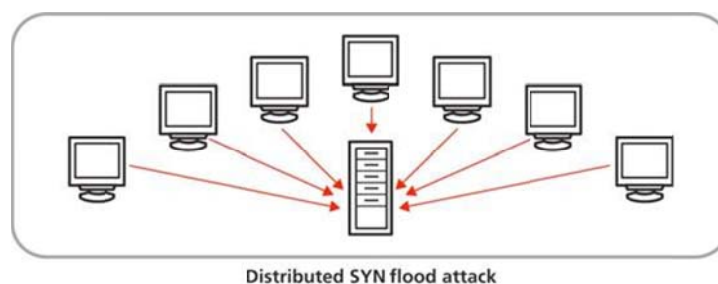
Whenever a sender attempts to send messages, the received information offers to reserve it in a different manner depending on the uses of the transport layer protocol. Both UDP and SCTP keep the boundaries of the messages, contrary to TCP, which does not maintain it. For instance, when a client attempts to send two messages (the first message is 200 bytes, and second message is 100 bytes),the server will receive both messages as an original format in case SCTP or UDP is used. However, both messages would be sent and received as one message with a total size of 300 bytes in TCP. Therefore, the application layer should be involved to acquire the original messages in TCP.[12, 13]



**Figure 8:** Illustration of TCP's and SCTP's feature of preserving the message boundaries.

### 2.6.5. Protection Against SYN Flooding Attacks

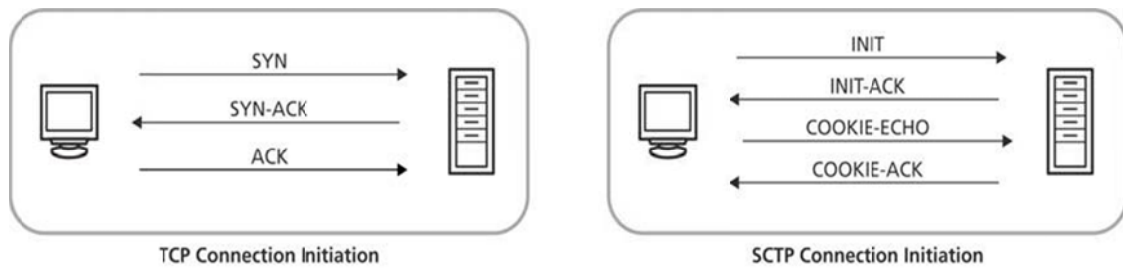
Before starting to explain the SYN-flooding attack, we will preview the transport layer connection establishment. Within TCP, it is called a “three-way handshake” as it requires exchange of three messages between both sides of the connection. For example, we can deal with the typical connection between client and server: The first client sends a SYNCHRONOUS message so as to request a resource from a receiver.



**Figure 9:** Illustration of SYN flooding attacks.

On the other hand, the second one will reserve the requested resource and port its response by SYNCHRONOUS ACKNOWLEDGEMENT. Finally the client will send ACKNOWLEDGEMENT and initiate the resource to send data. In opposition, SCTP uses a *four-way handshake*. Firstly; the client initiates communication by sending an INITIATE message. Secondly; the server reserves an appropriate port for cookies to identify the connection and send the INITIATE ACKNOWLEDGEMENT without reserving a resource. Thirdly, the client sends the COOKIE-ECHO message. And finally, the server will reserve resources and send a COOKIE ACKNOWLEDGEMENT.[12, 13, 14]





**Figure 10:** Illustration of TCP's and SCTP's initiating connection.

As seen above, the server may be susceptible to shut down by a synchronous flood attack due to the manner of establishing a connection in the TCP model. A synchronous flood attack appears when a sender or multiple sender machines send a flood of SYNCHRONOUS messages to a server. The TCP model is susceptible to this attack because a receiver machine and CPU processing are required for each coming SYNCHRONOUS message. However, in SCTP, the servers do not assign any resources to the coming INITIATE message until it receives the COOKIE ACKNOWLEDGEMENT message.

### 2.6.6. Selective acknowledgements

The SCTP's use of the acknowledge receipt mechanism depends on the nature of the application. SCTP will not use an acknowledgement when the application has a tolerance for missing, disordering or duplicating packets such as chat conversation, online football matches, etc. However, when the application has not tolerance for such situations, the acknowledgement is used to send a document. The jitter is doubled when using acknowledgement, so this may be considered to be the disadvantage of acknowledgement.[14]

## **CHAPTER 3**

### **LITERATURE REVIEW**

#### **3.1.SCTP Overview**

In this section, we are going to take a look at some main networks using SCTP as an efficient and reliable transport layer protocol.

Originally, the SCTP protocol was developed for Public Switched Telephone Networks (PSTN) while the TCP/IP model was, and is, the base protocol for Internet applications. Convergence technology endeavors to merge these two separate technologies to work together. Thus, they needed to transport SS7 over IP. Since TCP is not sufficient for reliable actions for these tasks, the Internet Engineering Task Force (IETF) developed the Stream Control Transmission Protocol (SCTP) in 2000. SCTP is a common objective transport protocol layer which provides a set of advanced features relating to multi-homing, multi-streaming, security and partial reliability in addition to the same benefits as TCP.[17,18]

#### **3.2. SCTP Base Protocol**

Initially, SCTP was specified in RFC 2960 in 2000. Then, SCTP was updated within RFC 4960 in 2007 based on the research. An SCTP packet is not unlike any traditional network data packet consisting of a packet header and a data user space. The packet header also contains a checksum number, which is a 32-bit cyclic redundancy check to obtain a reliable connection. The checksum number of SCTP is more powerful than the checksum used for TCP and UDP. Contemporary Ethernet cards are equipped with hardware supporting the CRC32C in SCTP packets.

SCTP uses three messages to terminate an association between two endpoints to ensure that all sent messages were received which are the SHUTDOWN message, the SHUTDOWN ACKNOWLEDGEMENT message, and the SHUTDOWN COMPLETE

message. In some cases, SCTP uses the ABORT message to cut off the association which causes to loss of some sent messages. [19, 20]

### **3.3. Algorithms Used with SCTP**

The algorithms used with SCTP help to assign the transmission sequence number (TSN) and then buffer SCTP chunks of multiple streams. Each stream might carry chunks of different application types.

#### **3.3.1.Using Round-Robin algorithm with the original SCTP [28, 29]**

Initially, standard SCTP with Request for Comments number 4460 used an algorithm which first ensured that no user message would be deferred from for assigned a transmission sequence number (TSN). The algorithms to assign TSN's include:

- (a) Using a Round-Robin order algorithm to assign a transmission sequence number(TSN) over all streams with waiting data.
- (b) Keeping the linear order in which the user messages are submitted to the SCTP association.

When the network layer is ready to read data from an SCTP association, the SCTP layer selects the message with the lowest TSN.

Drawbacks: An easy Round-Robin approach to schedule packet transmissions over multiple methods can cause lower throughput, primarily as a result of the fact that out of order arrivals will be queued at the receive buffer. Even with the absence of loss or a strained buffer, CMT requires intelligent scheduling to extend throughput and reduce receiver-side queuing.

When we use this technique without priority, the priority is given throughout the scheduling process there by making its execution insignificant. An unfortunate consequence of naive scheduling, however, is the fact that aggregated performance reduces as path characteristics become increasingly disparate. In the end, no destination, regardless of delay, can deliver packets faster than the speed of the slowest path.

### **3.3.2.Using the First-Come, First-Served Algorithm**

This algorithm is already used by existing implementations. The enhanced SCTP is applied in the First-Come, First-Served (FCFS) algorithm. FCFS is simple as it only passes the messages to the network layer in the order in which they have been delivered by the application. No modification of the order occurs at all.

Drawbacks: Both FCFS and Round-Robin algorithms have no efficiency when dealing with different types of data as they deal with applications with delay tolerances. [30, 31]

### **3.3.3.A fair bandwidth scheduler[29, 31]**

A fair bandwidth scheduler uses a fair division of the available bandwidth for all current streams of the particular association. Thus, all streams of the particular connection use the same bandwidth. The length of a message is taking calculation in this algorithm for scheduler of each stream. This approach provides benefits as it maintains an equal amount of the available bandwidth for every used stream.

Drawbacks: This method causes a transmission delay overhead while doing bandwidth scaling. Moreover, QoS (Quality of Service), which may not be necessary for all applications, may be required to apply this approach.

The main benefit which SCTP provides is to minimize the association setup delay time. SCTP provides powerful properties to applications without increasing application complexity. However, available bandwidth fluctuating with calculations can introduce delays in communication between endpoints across all streams due to the manner of the bandwidth scheduling algorithm. This algorithm is useful for applications requiring high QoS, such as military applications.

### **3.4.Related Work**

There are many applications simultaneously transferring different types of data between a source and destination of the same type. These applications need to assign resources among different data types depending on demand. Within the frame of TCP and SCTP, varying solutions have been provided for this problem.

One can always create multiple connections between a source and destination of the same type in TCP. However, these simultaneous connections may share the available resources equally because of the TCP's congestion control algorithm, which may be unwanted for the application.

Parallel TCP (pTCP) [36] strips data to different micro-flows in order to offer service differentiation for these TCP connections and reassembles them at the receiver side. pTCP procures end-to-end service differentiation via control of the number of micro-flows. Similar effects are achieved in a different behavior in multiple TCP [37]: AIMD parameters of the TCP connections are manipulated so that a proportional share is obtained for multiple TCP connections.

There are two main issues with these TCP-based techniques:

- 1) Allowing multiple connections and manipulating the resource assignment are unwieldy burdens for application programmers.
- 2) Amending the TCP's congestion control style may lead to fairness problems. These problems are overcome by SCTP thanks to its unmatched feature of multi-streaming. Applications have opportunities to maintain multiple streams in a single association and their aggregate behavior is in compliance with TCP-friendliness.

Nevertheless, the present SCTP standard does not specify the algorithm for multi-stream scheduling, which makes it difficult to offer service differentiation for the streams. SF-SCTP [38] groups SCTP streams into sub-flows and allows independent flow and congestion control for any sub-flow to indicate this issue. In this vein, it becomes possible to implement service differentiation at the sub-flow level. As the DATA chunk header of SCTP and its mechanism for congestion control are changed, this technique can provide not only interoperability but also fairness. [39] discusses SCTP's multi-stream scheduling problem within the scope of Concurrent Multiple Transfer (CMT). These authors prove with simulations that a better performance compared to a basic Round-Robin scheme can be achieved by mapping each stream to a definite path. Nevertheless, the present SCTP specifications have not provided standards for CMT yet; therefore, it is not possible instantly to implement this technique. [40] Seggelmann et al. discusses the advantages of using different algorithms for SCTP multi-stream scheduling by using various scenarios. They suggest the method of per packet scheduling at the end.

However, this suggestion has only been confirmed via simulations. We suggest a light-weight but efficient solution to this question. We do not change SCTP's congestion control approach or its packet structure; thus, in the present Internet environment, it will be safer to implement.

G. Heinz's *Priorities in SCTP Multi-streaming* [41] offers the solution which resembles our idea most. Nevertheless, he only deploys a queue scheduling algorithm of single priority in the ns-2 SCTP module, while our extension offers a general framework for implementation of any scheduling algorithm within the SCTP stack in the Linux Kernel, which is confirmed by test-bed experiments.

i. Although scheduling is used in many different areas, stream scheduling of SCTP deserves further analysis because of its interaction with other protocol mechanisms, such as bundling.

This affects the behavior of SCTP on the wire, so it can be used to optimize this behavior in certain scenarios. Scheduling algorithms can also be used for the SCTP's stream scheduling. Standard algorithms such as First-Come, First-Served are already used by existing implementations, as well as Round-Robin, which provides predictable behavior on the wire decoupled from the behavior of the application. Other algorithms can also provide benefits.

A fair bandwidth scheduler can maintain an equal amount of the available bandwidth for every used stream, and a priority scheduler can be used to have preference for a certain stream or set of streams over others.

They suggested using these algorithms in specific scenarios for optimization. This can be the fair bandwidth algorithm when tunneling multiple connections over different streams of a single SCTP association to treat every tunneled connection fairly. Monitoring applications can benefit from priority scheduling by sending warnings with a higher priority than informational messages.

Priority scheduling can also be used to realize a flow control per stream. For its simplicity, the First-Come, First-Served protocol is suitable to minimize the end-to-end delay because it just passes the messages in the order provided by the application. The end-to-end delay can also be reduced by avoiding head-of-line blocking. A simple model

to calculate the end-to-end delay with multiple streams scheduled with the Round-Robin algorithm is given. However, this model has several limitations. [40]

ii. In this environment, they proposed a protocol known as parallel TCP (pTCP). This is an end-to-end transport layer protocol that effectively supports striped connections. Although the pTCP design does not require any specific behavior from the component flows of the striped connection, in this paper they focus only on the case where a component flow exhibits the same behavior as that of a regular TCP connection. pTCP achieves effective aggregation of bandwidth for a striped connection through a combination of unique mechanisms including: (a) decoupling functionalities pertaining to the aggregate connection from those that pertain to an individual path; (b) effective striping across the multiple paths based on the instantaneous bandwidths; and (c) appropriate re-striping and redundant striping of packets during periods of fluctuation in path characteristics. As pointed out earlier, a protocol such as pTCP can have applications in several different settings.

iii. One of the most significant features of Internet data flow is fairness. Fairness means that each flow will pass through bottlenecks by receiving a fair share from the bandwidth available in cases of congestion. By making most of the data flows on the Internet, TCP flows at least reach a proximate fairness via the use of congestion control units adapting any TCP's throughput as one function of the congestion.

Max-min fairness is the most common form of fairness. All connections will receive the same share of a bottleneck in max-min fairness. In case a connection is unable to use its share due to reasons such as a slower rate in another bottleneck, the reserve capacity will be shared equally among the other connections. In other words, a source which cannot use more than an  $N$ th of the bandwidth of a bottleneck will always have the ability to send at its highest ratio.

Proportional fairness is another type of fairness. If any change related to the distribution of the rates causes a negative proportional changed sum. This means that this system is proportionally fair. In case a source cannot benefit from the  $N$ th of the bottleneck, less than its maximum may still be allocated.

Another issue is that of weighted proportional fairness. In this concept, association with a price for each connection is on the table. In such a case, paid amount per rate is that which is proportionally fair and not the rates as discussed above. Therefore, there would be no difference between the two connections with a price of one and one connection with the price of two.

Exciting results concerning weighted proportional fairness have been published recently. One of the results is that rate control based on additive increase and multiplicative decrease, as in TCP, achieves proportional fairness. The other result is that in a weighted proportionally fair system where the weights are the prices the users pay per time unit, when each user chooses the price that maximizes the utility that is received from the network, the system evolves to a state where the total utility of the network is maximized. It is a typical example of local optimizations leading to a global optimum. This property even holds when the exact function relating utility to the bandwidth received by a user is unknown and different for each user. The only constraint on that function is that the utility has to be an increasing, concave and differentiable function of the bandwidth, which happens to be one of the definitions of elastic traffic. [37]

iv. Multiple streams within a connection allow the separation of logically independent data. The application assigns each message to a stream where in messages belonging together are assigned to the same stream. In the case of SCTP, this is done with an identifier for each message indicating the stream.

With this identifier, the protocol only needs to restore the sequence of messages belonging together, i.e. those of the same stream, while messages of the affected streams can arrive unordered. Therefore, after a packet loss, only messages of the affected streams need to be delayed in order to restore the sequence, while on other streams the transmission can continue. These results are a reduced average delay compared to other reliable protocols without multi-streaming, such as TCP, in which all proceeding messages are delayed after a loss, resulting in a so-called head-of-line blocking.

Any message passed by the sending application is added into the matching stream buffer. Then, these messages are grouped into packets for sending. A single stream's message order is given, but the order of messages belonging to different streams will be



defined by a scheduler. The messages are classified into stream buffers again subsequent to reception in order to restore their order. Another scheduler is required that decides the order in which the messages of different streams will be delivered while passing these messages to the receiver application.

### **Transmission Scheduling**

Sender and receiver schedulers are required for multi-streaming. The sender scheduler determines the sending order of the messages and the receiver scheduler will define the order for delivery of data to the application. There is no standard as to how these schedules are realized in the SCTP specifications, which depends on implementation. Nowadays, the implementations utilize the generic algorithms of Round-Robin and First-Come, First-Served for sender scheduling.

Nevertheless, it may be useful to select a specific scheduler for some cases. The receiver scheduler is only concerned with determining the sending order of the messages and thus, it is not very useful to vary the algorithm. On the other hand, the sender scheduler may have an influence on the behavior on the wire and so varying algorithms may be utilized as the means of optimization.

One can also utilize a certain scheduler for optimization in the case of multi-path transfer. The greatest problem of multiple paths is that varying delays on the path may lead to reordering of messages. As the messages are reordered, the restoration of the receiver becomes more complex and thus, more buffer space is needed. The buffer space available will at least match the bandwidth delay product. In the contrary case, the data transfer speed is reduced. The most basic method for mitigation of reordering is to allocate streams to the paths and to send messages in a stream on its own path.

This approach will prevent some messages from being sent in order from passing the messages onto a slower path.

However, only allocating streams to the paths may not be the ideal solution if the delays in the paths or the amount of data in the streams varies greatly. In such situations, it is possible that the faster paths remain unchallenged while the slower paths are overloaded. Thus, one should assign different streams to a path or stream while a great amount of

data are divided between multiple paths depending on the situation. This may necessitate use of an exceedingly complex and resource-intensive scheduler.

We will examine the potential performance advantages of an optimized scheduler and check it against standard implementation in order to find the potential of a scheduler taking multiple and diverse paths into consideration. According to the situation, this scheduler will always select the ideal combination of streams and paths in order to determine the maximum optimization. [39]

v. We work on theoretical and practical results of annexing priorities to SCTP streams in order to provide an ability of addressing periods of poor network conditions for multimedia applications. We see stream priorities as a supplementary service for applications. It is allowed by a stream priority scheme to define the relative significance of the data. Transmission of significant data will prevail; thus it will decrease any perceived delays for significant data in due course during low quality service periods. In the situation exemplified here, less significant data will be transmitted according to available bandwidth.

The addition of stream priorities is an extension to SCTP's existing *sender-side* API and scheduling algorithm implementation only. Priorities do not change the on-the-wire SCTP protocol and thereby do not change the SCTP's current packet format (i.e. there is no addition of a new control chunk). By avoiding such modifications, stream priorities do not require the SCTP's receiver-side to be aware that prioritization is occurring at the sender's side. This transparency maintains backward compatibility with non-priority enhanced endpoints, thereby allowing any SCTP receiver to operate with both priority and non-priority enhanced SCTP senders. In addition, this transparency allows for easier Internet deployment.

Originally, we considered a strict priority scheme for SCTP. In such a scheme, items on stream  $j$  always have priority over items on stream  $k$ , with  $j < k$ . However, a strict priority scheme has an obvious weakness: in cases where an SCTP sender has bandwidth sufficient only to transmit data for streams 0-3, data on stream 4 will be indefinitely postponed. In some applications (such as SS7 signaling and stereo audio streaming applications), multiple data streams are considered to be of equal importance.

A priority scheme must have a method of addressing this situation. By assigning the same priority to two or more streams, our priority scheme will treat the data of those streams equally. [41]

vi. During this thesis, they give the required modifications to the SCTP specification to support discriminatory treatment of the SCTP stream and they outlined the thought of sub-flow. During this modification, they labeled every sub-flow possessing its own flow and congestion management and consist of SCTP streams that need an equivalent form of QoS from the network.

The SCTP association can have many sub-flows that serve as autonomous transmission channels depending on the necessary QoS.

Their style sidesteps fake sharing while congestion info is only shared by SCTP which needs same QoS.

In this thesis, SF-SCTP was designed to introduce servers to different applications with different requirements for its data.

The SF-SCTP behavior introduced is similar to the aggregation of multiple parallel original SCTP associations. This behavior has certain advantages and disadvantages. In one aspect, similar to parallel TCP flows but without the overhead of maintaining multiple connections, SF-SCTP can be used to improve the utilization of a network with high bandwidth and a delay product or non-ignorable non-congestion loss. [38][42]

## **CHAPTER 4**

### **PROPOSED METHOD**

Normally the standard SCTP uses Round-Robin or perhaps the First-Come, First-Served protocol. This, however, does not provide an efficient and flexible choice and is not suitable for some applications. [15]

In our thesis, we proposed to optimize the multi-streaming feature of SCTP with pluggable scheduling. Therefore, users can customize the priority of the multi-stream scheduling algorithm of SCTP depending on the particular application on the instance rather than Round-Robin or First-Come, First-Served algorithms.

This proposal will add a new priority to SCTP such that it is made more efficient and attractive. Moreover, the scheduling algorithm can be loaded or unloaded at run time.

#### **4.1. Definition of Per-Stream Priority**

To obtain diverse applications with the ability to assign periods within weak network conditions, we examine the theoretical and practical influence of assigning a priority to SCTP streams. We add pluggable priorities as a further service available to the SCTP protocol. A pluggable stream priority scheme allows an application to specify a priority according to the sender. We also developed an algorithm to calculate the current status of streams. In this manner, we reduced the delays for critical data during periods of low quality in the network at the moment. We define an SCTP stream priority scheme as:

A stream M including data having priority more than or equal to the other data within N stream.

The pluggable priorities algorithm is an addendum to the present application program interface at the sender's side. This modification does not cause modification of the

packet format; thus, it requires no change in the application interface at the receiver's side. Our algorithm allows the sender to operate with both FCFS and PRIORITIES algorithms. Furthermore, the receiver will be unaware for which algorithm is used with the SCTP association.

This feature results in better deployment and optimization for the Internet. In the beginning, we worked on a pluggable priority scheme for SCTP. The items on stream  $j$  shall have not priority over items on stream  $k$  all the time in such a scheme but we do not keep a strict priority in sight as such a strict priority scheme has a clear disadvantage: in situations in which a SCTP sender's bandwidth is only sufficient to transmit data for streams 0-3, data on stream for shall be postponed for an indefinite period. Yet, the pluggable scheme provides better flexibility for the sender to fix its priorities depending on its considerations.

## 4.2.SPECIFICATION

We should firstly add a new area to the data structure of the SCTP stream in order to realize the scheme discussed in Section 4.1. This integer field (priority) maintains a positive value that corresponds to the stream's relative priority which the user has adjusted. The sender is initialized priority values for all streams depending on its needs subsequent to the association setup. In this vein, priority-enhanced SCTP will behave as basic SCTP unless the values are modified. Secondly, the following Interfaces are added to the SCTP Sockets API:

```
sctp-priority ()
{
  for (M = 0; M < num_streams; M++)
  {
    streams[M].priority = M;
  }
}
```

To adjust priorities and rank the streams of equal importance, uses:

sctp\_setplugpriority sets all streams depending on the desire between (and including) the streams of startStream and endStream. Then, the interface resets the priorities on all streams greater than the endStream.

```

sctp_setplugpriority (int startStream, int endStream)
{
    int p = streams[startStream].priority;
    for (M = startStream; M <= endStream; M++)
    {
        Streams_[M].priority = p;
    }
    for (M = endStream + 1; M < num_streams; M++)
    {
        p++;
        streams[M].priority = p;
    }
}

```

To disable SCTP priorities completely, use:

```

1. int stream_num; // current stream number
2. int current_priority // current priority
3. boolean priority_class_has_data // does the current priority
// have any streams that have data to be sent?
4. int priority_class_base_stream // the first stream with the current priority
5. stream_num = 0;
6. current_priority = 0;
7. priority_class_has_data = false;
8. priority_class_base_stream = 0;
9. while (space exists in SCTP packet)
10. {
11. if (streams[stream_num] has data chunks to transmit)
12. {
13. assign TSN to a chunk from streams[stream_num]
14. bundle chunk into SCTP packet
15. priority_class_has_data = true;
16. }
17. stream_num++;
18. if ((priority_class_has_data) &&
(streams[stream_num].priorities > current_priority))
19. {
20. stream_num = priority_class_base_stream;
21. priority_class_has_data = false;
22. }
23. else if (streams[stream_num].priorities > current_priority)
24. {
25. current_priority++;
26. priority_class_base_stream = stream_num;
27. priority_class_has_data = false;
28. }
29. }

```

Finally, when priorities are enabled, the following algorithm should be used to assign the Transmission Sequence Numbers (TSN) to data among the stream queues:

The algorithm above is added subsequent to implementation checks for space in the congestion and receiver windows. The algorithm executes a Round-Robin type function between the streams after initialization (Lines 1-8) when space exists in the SCTP packet (Line 9). Initially, the current stream is checked for data (Line 11). In case the stream has data chunks to transmit, a TSN is allocated to that chunk at the head of the stream's queue (Lines 13-14). Since the stream carries data to be sent, the `priority_class_has_data` flag is adjusted to `true` (Line 15). This flag allows the algorithm to follow when the level of priority is increased.

The algorithm then considers the next stream number (Lines 17-28). If this stream is of the same priority observed before, then we loop to process the new stream's data.

If this stream is of a lower priority, we check the `priority_class_has_data` flag. If the flag is set to `true`, then the algorithm resets the stream number to the first stream number with the current priority (Line 26). This ensures that all data of greater priority takes precedence over all lower priority data. Upon initial observation of the algorithm, we might conclude that a number of factors influence the running time: namely, size of the SCTP packet, number of streams, amount of data to transmit, and number of priority classes. However, when analyzing the worst case running time, we can limit these factors to only the number of streams and the amount of data to be transmitted.

We do not consider the packet size since, at any instance of time, either:

1. the packet size is too small for all of the available stream data – this condition will restrict the running time, since the algorithm terminates once a packet is full  
or
2. the packet size is greater than the amount of available data. In this case, the amount of available data will ultimately determine the running time.

In theory, a worst-case scenario occurs when the amount of packet space remaining is infinite. However, in practice the packet space is finite.

The running time is not affected by the priority classes. While executing the algorithm, these classes only have an effect on the order of transmission of packets and they do not increase the number of loop iterations. Thus, we believe that the worst conditions appear when the  $d$  data chunks are sent over  $n$  streams. We adjust the size of SCTP packet to

infinity in order to obtain a maximum running time. Under these conditions, the running time (T) for our algorithm will be:

$$T = n + d$$

For any stream and piece of data in the association, the while loop will be executed at least once. Thus, we reach a conclusion that this algorithm is order  $O(n + d)$ .

#### 4.3.SYSTEM FLOWCHART

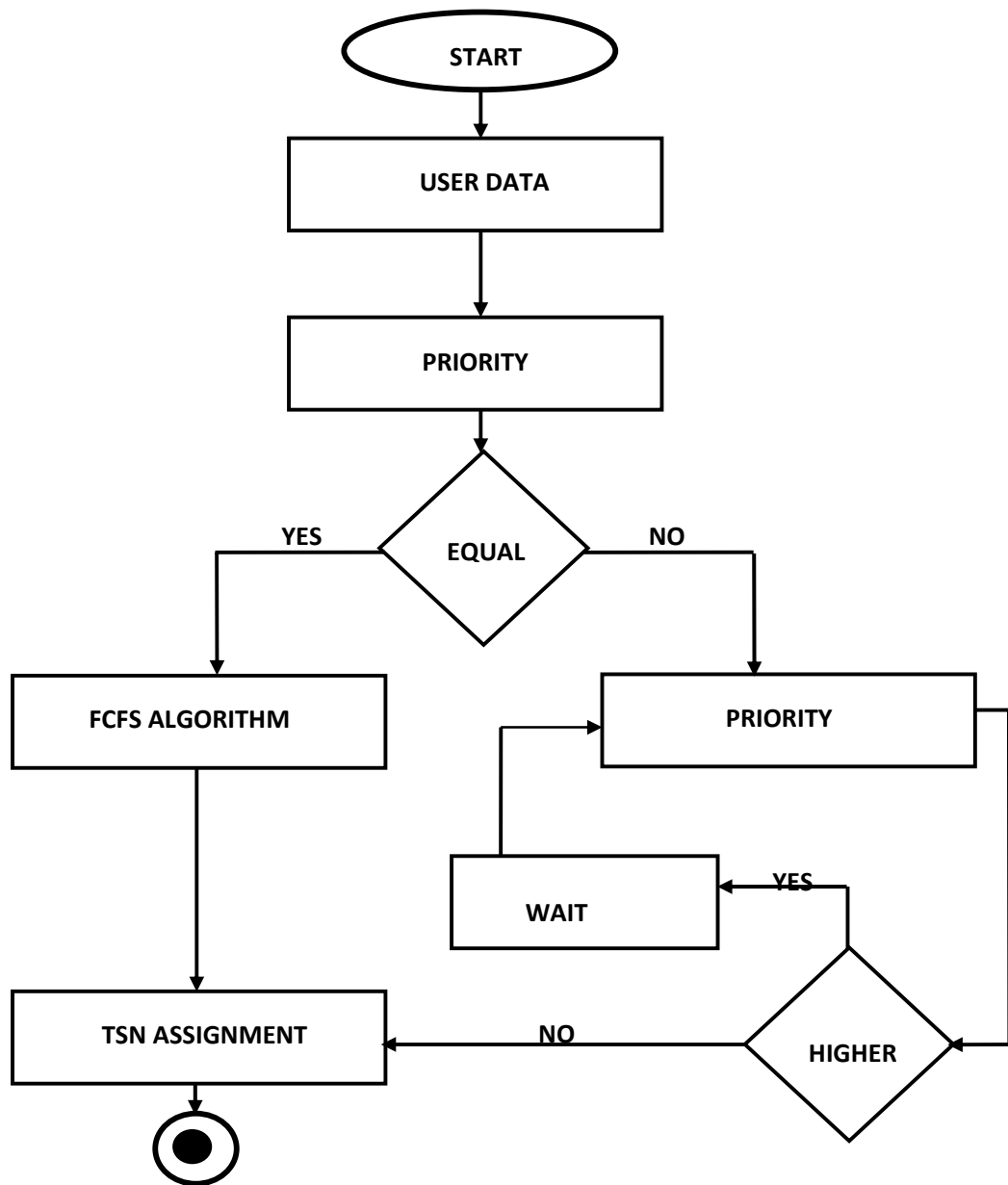


Figure 11: Designed system interaction.



In this chart:

**Start:** Start of system

**User data:** How many streams, types of stream?

**Priority:** This integer field will store a positive value corresponding to the relative priority of the stream which the user has set.

**Equal:** If the values are equal, unless the values are changed, priority-enhanced SCTP behaves as basic SCTP.

**FCFS Algorithm:** First-Come, First-Served algorithms

**PRIORITY Algorithm:** A pluggable stream priority algorithm

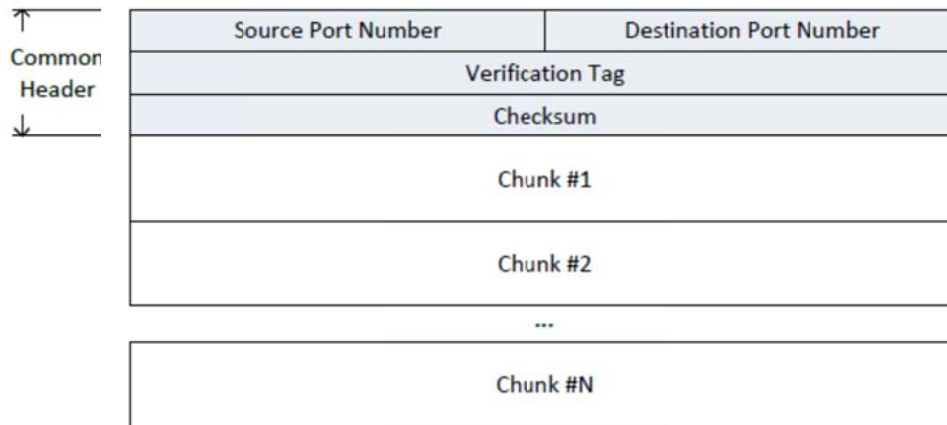
**Higher:** Condition box, if priority is high or low

**Wait:** When priority is high, it will enter a loop

**TSN assignment:** Transmission sequence number

#### **4.4.Stream Control Transmission Protocol Design**

Any SCTP packet begins with a Common Header as indicated in Figure 12. This header is comprised of the port numbers of the source and destination, a 32-bit Verification Tag, and a Checksum in order to identify any corrupted packets. A random value which is unique per direction is assigned to the Verification Tag and it is exchanged in due course of the connection establishment. The tag selected for each direction is used for any packets sent for the duration of a connection. This mitigates the blind attack risk in which an attacker finds the port numbers of a connection and tries to insert tags. An attacker will also have to find the Verification Tag with this implementation



**Figure 12:** Sctp Packet Format with Common Header and Chunks

#### 4.5.Data Transfer

A safe data transfer is on the table for Sctp. All DATA chunks, including user data, have an assigned TSN, as illustrated in Figure 13. This number is utilized for acknowledgement. The SACK chunks report the highest TSN of continuous data and TSNs of out-of-order data chunks which have already been received in gap reports. As an option, selective acknowledgements are granted in order to reduce the retransmission level. There is limited space for TCP options, thus only a few out-of-order packets may be acknowledged. Conversely, until a whole packet is filled, Sctp supports an arbitrary number of gap reports as this quality has been integrated from the first moment. All user messages are sent in their own DATA chunks for as long as possible thanks to the message orientation facility of Sctp.

Large (in terms of data content) messages exceeding the highest possible packet are divided and sent with different DATA chunks so that each will be in its own packet. The receiver reassembles the message by restoring the order using the TSNs. Multiple smaller messages may be delivered in a single packet (known as bundling) in order to decrease the overhead, which will be necessary otherwise.

Chunk Type	Reserved	Flags	Length
TSN			
Stream Identifier		Stream Sequence Number	
Payload Protocol Identifier			
User Data			

**Figure 13:** SCTP Data Chunk Format

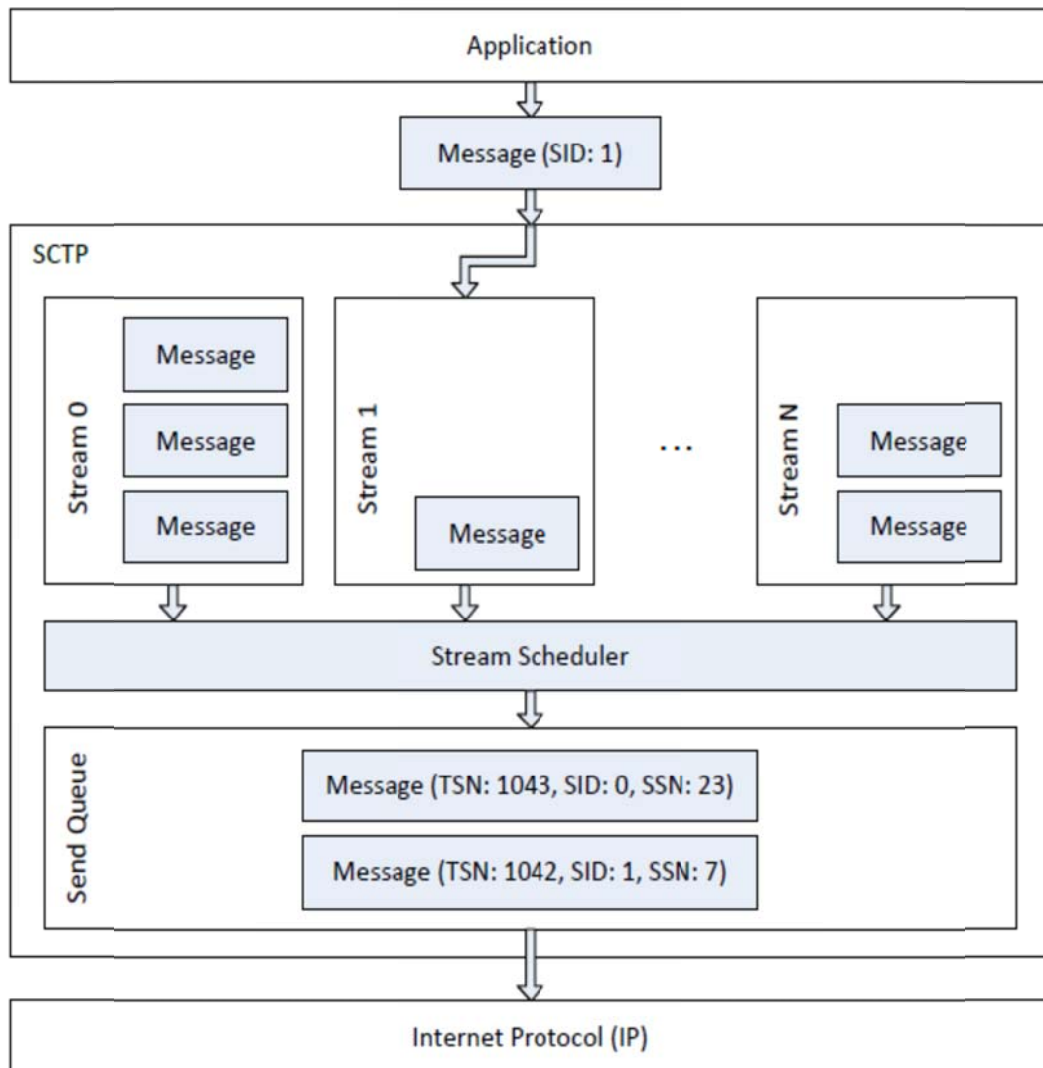
#### 4.6.Streams :

As indicated by its name, a salient feature of SCTP is multi-streaming. Within the data transfer process, the streams are unidirectional logical channels. By adjusting the Stream Identifier (SID), the user allocates a message to a stream. The SID is 16 bit, thus  $2^{16}$  or 65,536 possible streams are available. The message order is kept within only a single stream; therefore, when a message is lost, only the following messages of that stream are delayed until retransmission is received. In the contrary case, the original message order cannot be restored.

In TCP, delays of message are a common issue (called head-of-line blocking) and the situation is the same without streams. A Stream Sequence Number (SSN) maintained per stream and increased for any message is utilized in order to restore the message order (if needed). However, the user may opt to leave messages unordered within a stream and in this case, the SSN is ignored and generally adjusted to 0.

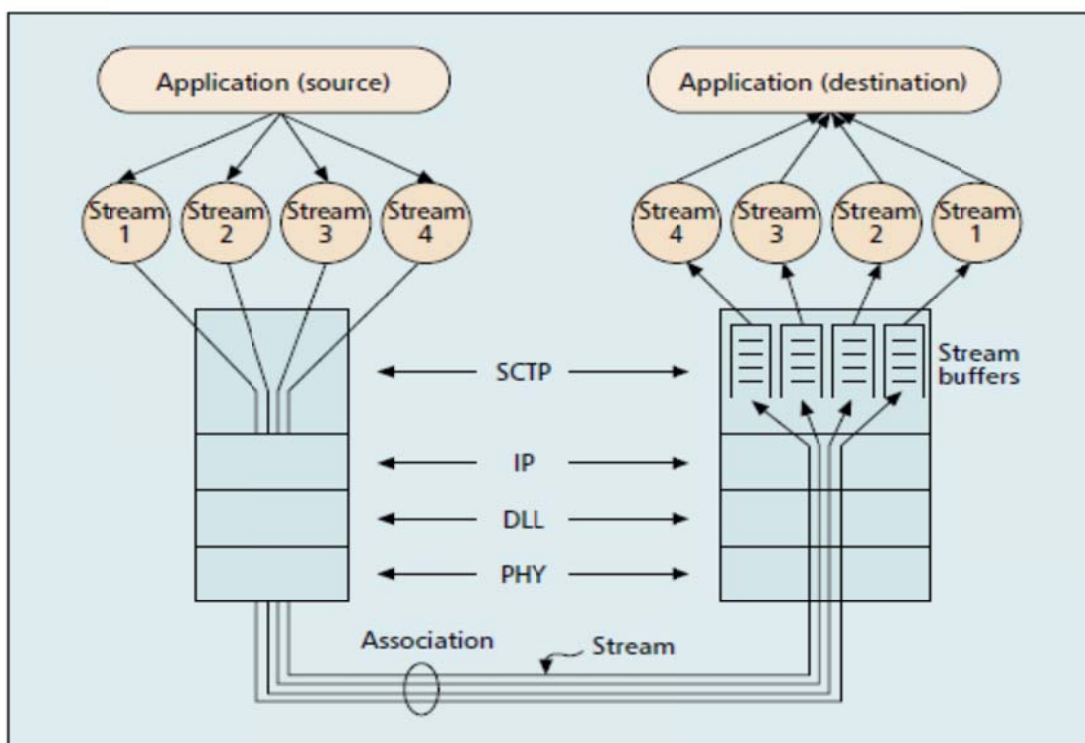
#### 4.7.Sender Scheduling

After assigning different messages to varying streams, all messages are transmitted over a single association. It requires a scheduler determining the order for messages of various stream queues as described in Figure 14. The SCTP specifications do not indicate such a requirement.



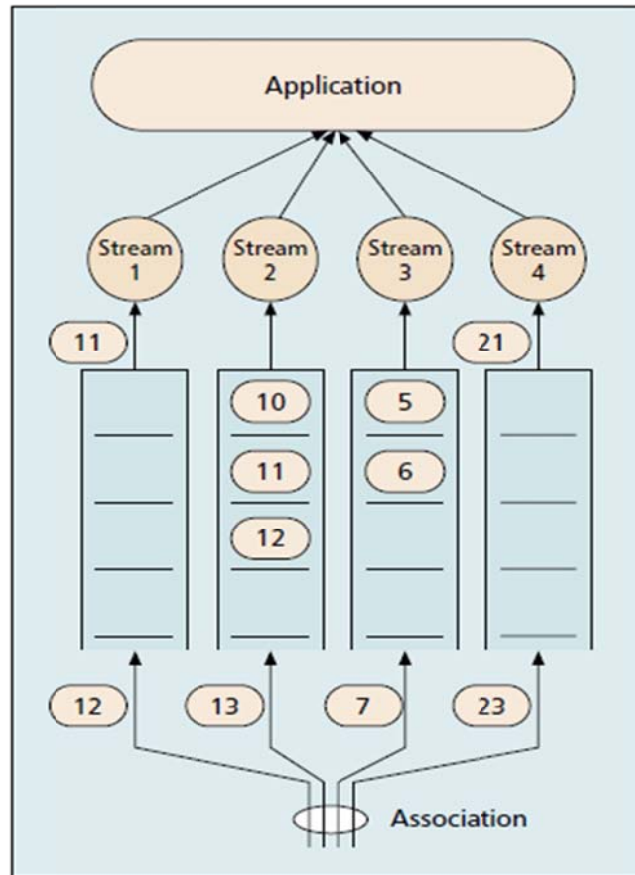
**Figure 14:** Sctp Receiver Stream Scheduling

Multi-streaming allows the data of the upper layer application to be multiplexed onto one channel as indicated in Figure 15. Data sequencing is performed within a stream; in the case of a segment of a definite stream being lost, the following segments are kept in the receiver’s stream buffer until such time that the loss segment is retransmitted from the source. However, it is still possible to pass data from other streams to the upper-layer application. This avoids the HOL blocking in which a single stream carries data from all the upper-layer applications. In this vein, the HOL effect is limited within the scope of individual streams as it does not have an effect on the whole association.



**Figure 15:** An SCTP association consisting of four streams carrying data from one upper layer application

An SCTP association which is comprised of four streams is illustrated in Figure 16 and this figure shows the multi-streaming and HOL structure. The segments are defined by stream sequence numbers (SSNs) which are unique within a stream; however, it is possible for different streams to have the same SSN. In Figure 16, SSN 11 in stream 1 has been delivered to the upper-layer application, SSN 9 of the second stream is lost in the network; SSNs 10, 11, 12 are therefore thus queued in the buffer of the second stream waiting for retransmitted SSN 9 to arrive. Arriving SSN 13 at stream 2 will also be queued. Similarly, SSN 4 of stream 3 is missing during transmission resulting in the blocking of SSNs 5, 6, and 7. It is necessary to take into consideration the fact that SSN 1 can be delivered immediately even if the other streams are blocked when SSN 12 arrives at the buffer of stream 1. This indicates that the segments arriving on stream 1 can still be delivered to the upper layer application in spite of the fact that streams 2 and 3 are (and stream 4 will be) blocked due to lost segments.



**Figure 16:** An illustration showing HOL blocking of individual streams

#### 4.8.Explanationof Algorithm with multi-streaming

One of the most significant features of SCTP is multi-streaming, in which one association can collect various independent streams. The TCP's head of line (HOL) blocking problem is avoided thanks to this feature and it makes SCTP an appropriate means of transportation for signaling messages. A stream is a unidirectional, logical channel which is formed of one to another associated SCTP endpoints in SCTP. In this vein, user messages are not retained across multiple streams; they are retained within each stream. Nonetheless, the whole association is subjected to safe data transfer and congestion control. SCTP separates the phases of transmission and delivery of data in order to fulfill these requirements. Each DATA chunk has two independent sequence

numbers, namely the Transmission Sequence Number (TSN) and Stream Sequence Number (SSN). The SSN (with STREAM ID) is utilized for data delivery, whereas the TSN is used for data transmission so as to account for loss recovery, flow control and congestion control. This process is indicated in Figure 16.

#### **4.9. General Example**

We will give an example to illustrate how to implement the algorithm below.

We assume that we have four messages from the application (text, audio, video and image). All of these messages will be assigned to streams depending on the stream identifier (SID), and then we will enter the SCTP protocol through multi-streaming. Every message will take a specific stream; however, here we need to prioritize the video message above other messages (to demonstrate an example).

When the messages are distributed onto all streams, they will be divided into many chunks inside each stream depending on the size of the packet. All chunks will take a transmission sequence number (TSN) and a stream sequence number (SSN).

All streams will enter the stream scheduler (pluggable priority algorithm) before entering the sender queue. Our algorithm will give priority to video; therefore, the DATA chunks of stream 2 will enter the sender queue first and then they will enter a chunk that has already had a transmission sequence number (TSN) assigned to it.

At the other side, the receiver will determine depending on the order of chunks by using the First-Come, First-Served(FCFS) algorithm, as seen in Figure 17.

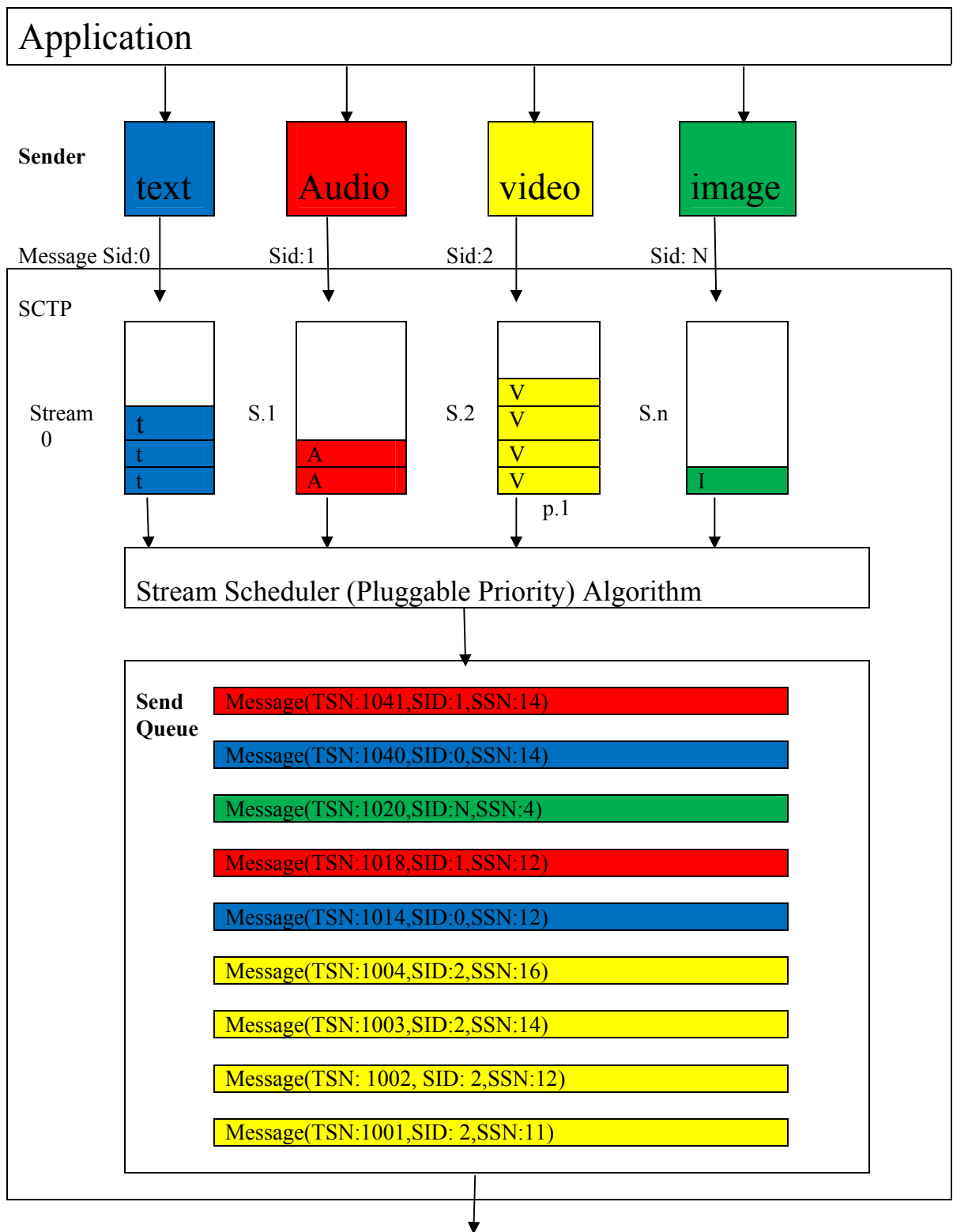


Figure 17: Sctp multi-streaming with priority algorithm



## CHAPTER 5

### PROJECT IMPLEMENTATION AND DISCUSSION OF RESULTS

#### 5.1.SOFTWARE AND TOOLS

We developed our project by using the Java programming language due to its simplicity. Java has dispensed with many of the complex features of C++ and C, resulting in a simpler language (no pointers, no unions and no enumerations). Furthermore, Java is an object-oriented, single-rooted programming language. [33]

Our code was executed in an application named NetBeans, which is mainly used with the Java language; however, it can be used with other languages such as C,PHP,C++, HTML, etc. The NetBeans IDE is written in Java and can run on Windows, OS X, Linux, Solaris and other platforms supporting a compatible JVM.

#### NETBEANS IDE INTERFACE

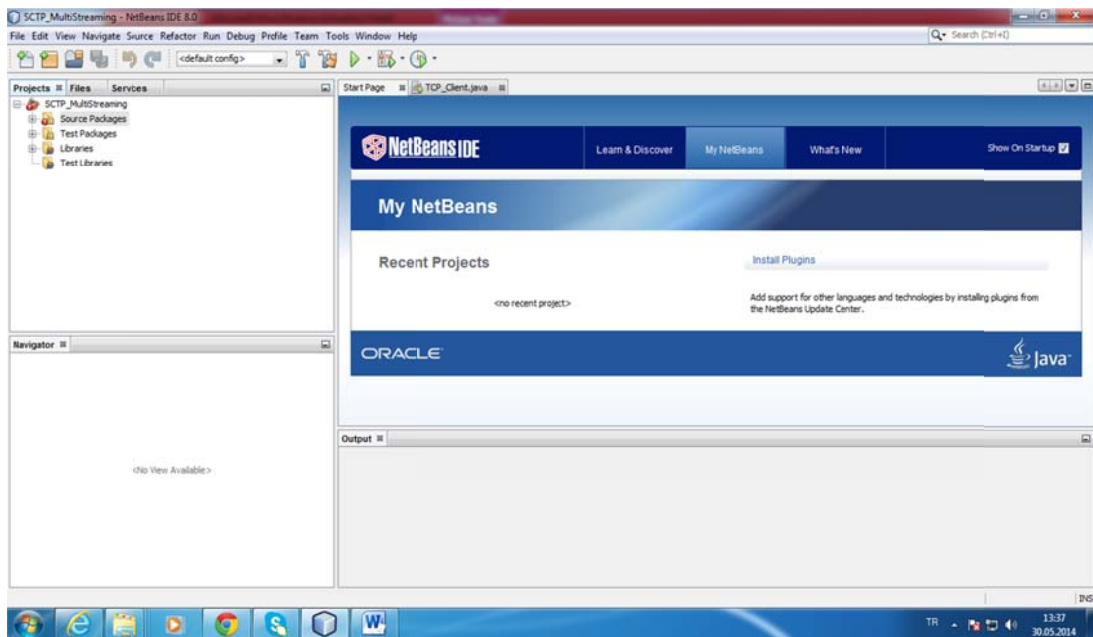


Figure 18: Illustration of the NetBeans interface

## 5.2. Project Implementation

We have designed a pluggable priority scheduling of incoming data segments in the multi-stream transport protocol in the socket. What we mean by the term “pluggable” is selecting the priority depending on the client that is responsible for which type of file would be assigned by that priority level. This can be changed. We proposed PP-SCTP as a name to our proposed SCTP with pluggable priority. This modification will be applied only at the sender’s side, where the receiver will not make any modifications to its operating system and socket code. Thus, the receiver will be able to deal with any type of algorithm that the sender uses.

// The main function of priority class:

```
public class SCTP_Priority {
    ArrayList files = new ArrayList();

    if(n==1)
    {
        namef1 = f1.getName();

        //System.out.println("Name of file1::"+namef1);

        mf1 = new MimetypesFileTypeMap().getContentType(f1);

        // System.out.println("Type of f1::"+mf1);

        size1 = f1.length();

        name_size1 = namef1 + "+" + size1 + "+" + mf1;

        files.add(name_size1);

        } public static String copy_list[] = new String[4];
    static int flag = 0;
    long starttime = 0, stoptime = 0;
    double MegaBitsPerSec=0; //holds bandwidth in megabits/sec
    double BytesPerMiliSec=0; //holds bandwidth in bytes/milise
    double MegaBits= 0; //holds calculate value for bytes to megabits
    double MilisecToSecond=0;
    double time=0;
```

```

double final_size=0;
int n=0;
public String priority()
    StringBuilder sb=new StringBuilder();
    // Sender s=new Sender();
    MainGui mg=new MainGui();
    //complete_list= s.send_path();
int n=mg.gcount;
    complete_list= mg.send_path();
    File f1=null,f2=null,f3=null,f4=null;
int flag1=0;
starttime=System.currentTimeMillis();
System.out.println("starttime"+starttime);
    File dir = new File(System.getProperty("user.dir") + "\\test");
final File[] w_files = dir.listFiles();
for (File fn: w_files) fn.delete();
if (!dir.isDirectory())
    { dir.mkdir(); }
    String InputFileLocation=System.getProperty("user.dir")+"\\test\\test.txt";
try {
    File wri_file = new File(InputFileLocation);
    BufferedWriter output = new BufferedWriter(new FileWriter(wri_file));
for(int b=0;b<complete_list.size();b++)
    {
if(flag1==0)
    {
        f1=new File(complete_list.get(b).toString());
flag1++;
    }
    else
    if(flag1==1)
    {
        f2=new File(complete_list.get(b).toString());
flag1++;
    }
    else
    if(flag1==2)
    {
        f3=new File(complete_list.get(b).toString());
flag1++;
    }
}
}

```

```

else
if(flag1==3)
    {
        f4=new File(complete_list.get(b).toString());
flag1++;
    }
    }
    String namef1=null;
    String mf1=null;
long size1=0;
    String namef2=null;
    String mf2=null;
long size2=0;
    String namef3=null;
    String mf3=null;
long size3=0;
    String namef4=null;
    String mf4=null;
long size4=0;
    String name_size1=null;
    String name_size2=null;
    String name_size3=null;
    String name_size4=null;

```

### 5.3.SIMULATION

As mentioned at the beginning of this chapter, application of our algorithm was carried out using the Java programming language, and by using NetBeans as the IDE. We divided the code of the project into several separable DDL files, such as the file code of SCTP at the client side referred to as the DDL file requiring implementation of the Pluggable-Priority algorithm or the First-Come, First-Served algorithm each structured in separable files. In addition to implementing the SCTP protocol at the receiver side, the SCTP client information file determines SCTP client channels, SCTP receiver channels and the establishment streams on the client and receiver sides respectively in a separate DDL file.

In our implementation, we applied a different scheme in which we were able to select the number of files to be sent. It was assumed that each file would be sent on a one-

stream channel in the case of using SCTP. However, while applying the TCP protocol for each channel, it was necessary to establish a new association with the same particular receiver by applying the three-way hand shaking approach.

We have designed a transport layer for both the SCTP model and TCP model to transfer the files from a server to a client. We named the server file MAIN GUI ddl and the client file CLIENT GUI ddl. The IPs of both client and server were assigned to a local IP.

#### **5.4. DETAILS OF RUNNING**

When the MAIN GUI file in the NETBEANS IDE is executed, several actions are covered as indicated below:

- 1- Select the file desired to be sent to the client.
- 2- Send the files to the application layer (first layer of network protocol) so as to be sent via the network.
- 3- Select the transport layer protocol (SCTP or TCP). In this layer, there are two main actions: hand shaking, and then the multiplexing of data packets into chunks.
- 4- If SCTP is selected, the four hand shaking procedures will be performed before starting to send the actual data. On the other hand, three hand shaking procedures will be performed if TCP is selected.
- 5- While using the SCTP model, it will immediately be established for streams as it is one of the main key features of SCTP. The first stream has been designed to be the primary stream while the others will be secondary streams. In the TCP protocol, we will establish just one stream for that session.
- 6- While selecting an algorithm, such as PRIORITY or FCFS, the data packets will come from the application layer in SCTP;CHUNK will be multiplexed into the chunks of the transport layer. The multiplexing method will depend on the algorithm selected.
- 7- The maximum SCTP-CHUNK size assigned is 50 Kb. Thus, when selecting a file exceeding the 50 Kb limit, it will be divided into more than one chunk.

## 5.5. Running the Simulation

### 5.5.1. First Step: Assigning the Number of the File

The user selects the number of the file to be sent. The following snapshot in Figure 19 illustrates this action.

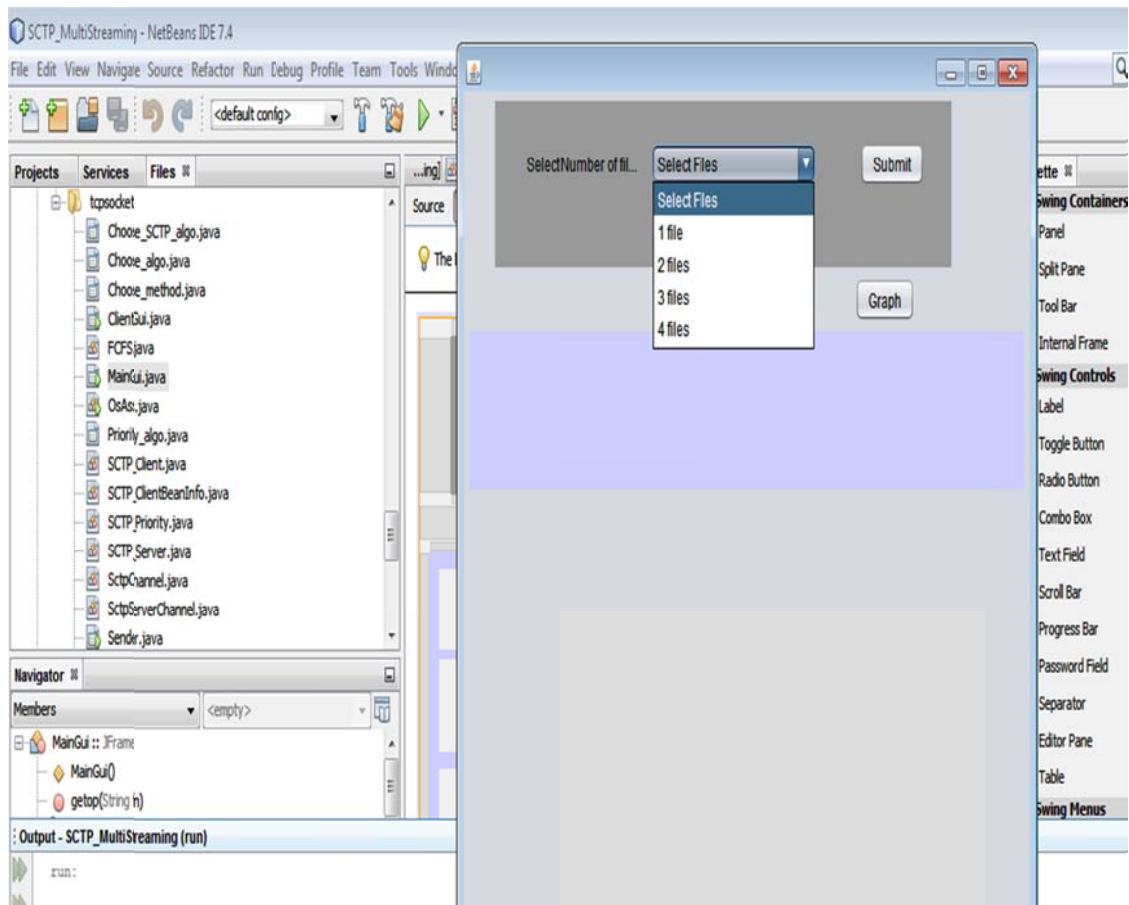


Figure 19: File number assignment

### 5.5.2.SecondStep: Selecting the Files

After determining how many files will be sent, the submit button is clicked, after which the client will select the desired files. A snapshot below illustrates this.

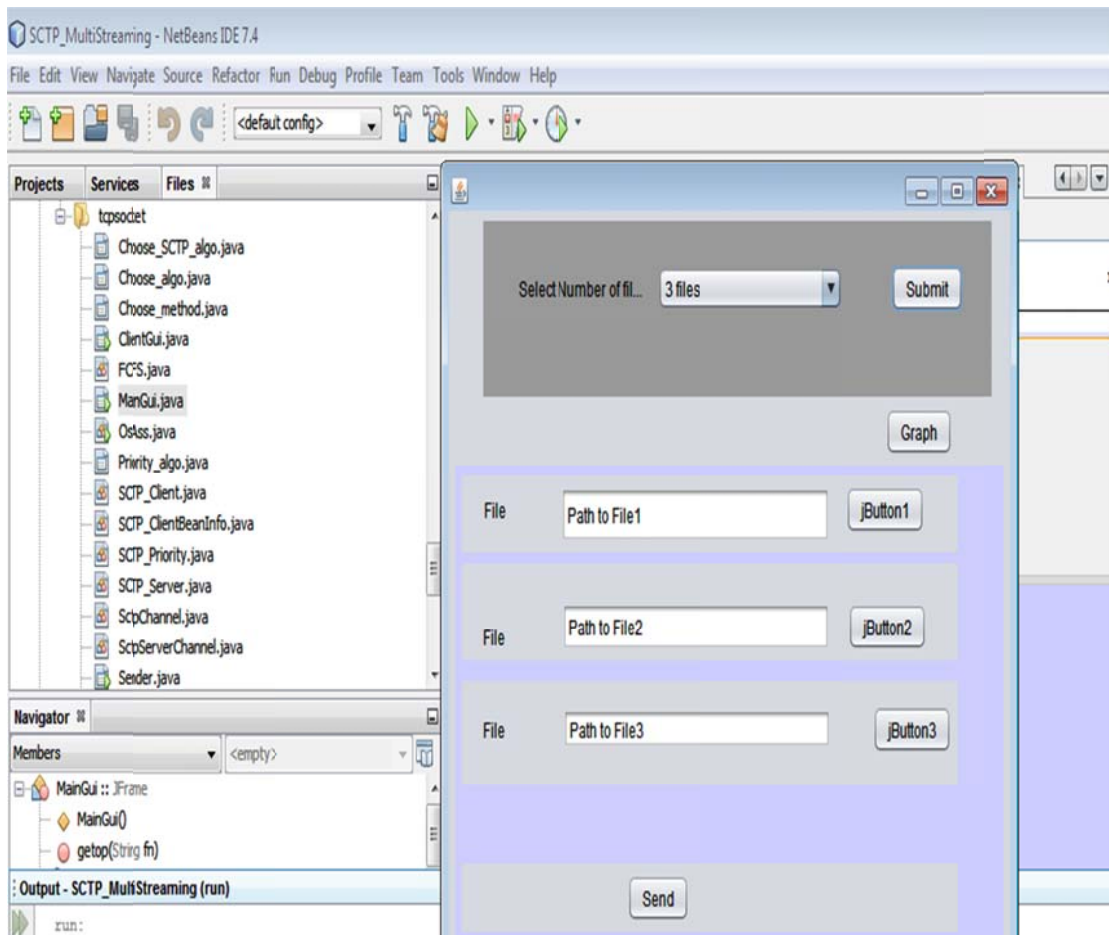
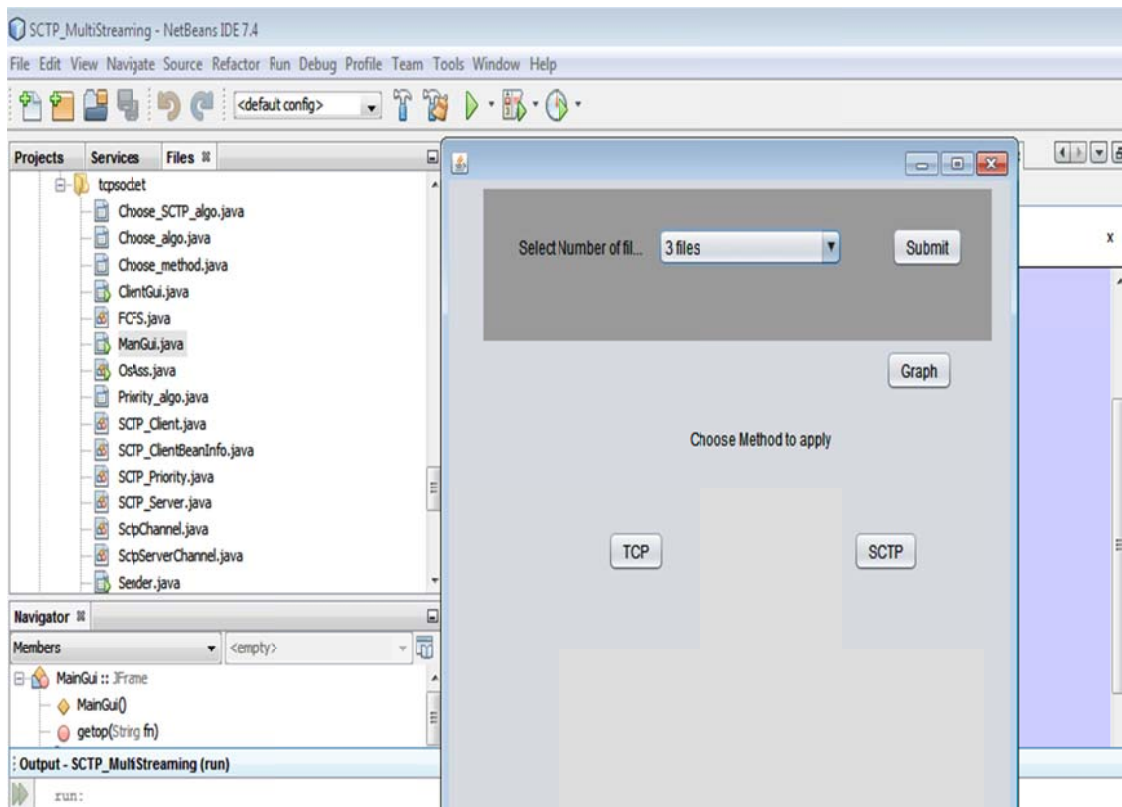


Figure 20:File selection

### 5.5.3.ThirdStep: Selecting the Protocol

We then add a feature to call one of the transport layer protocols, which are the TCP or SCTP protocols. The client may select either. While making the selection, the selected protocol will establish the stream and channel with the server that we have already assigned in our code, as illustrated in the following figure.



**Figure 21:** Protocol selection

When the SCTP protocol is selected, it will establish four streams for each association regardless of the number of files required to be sent. The number of streams we set will be the default, as is the nature of SCTP. However, if the client selects the TCP protocol, it will make an equal number of three hand shaking mechanisms for each file as each establishment contains only one stream.



#### 5.5.4. Fourth Step: Selecting the Algorithm

The client will then select the algorithm required to be applied. Selecting any algorithm means that we will call the DDL file of the particular algorithm selected. The following figure illustrates this situation.

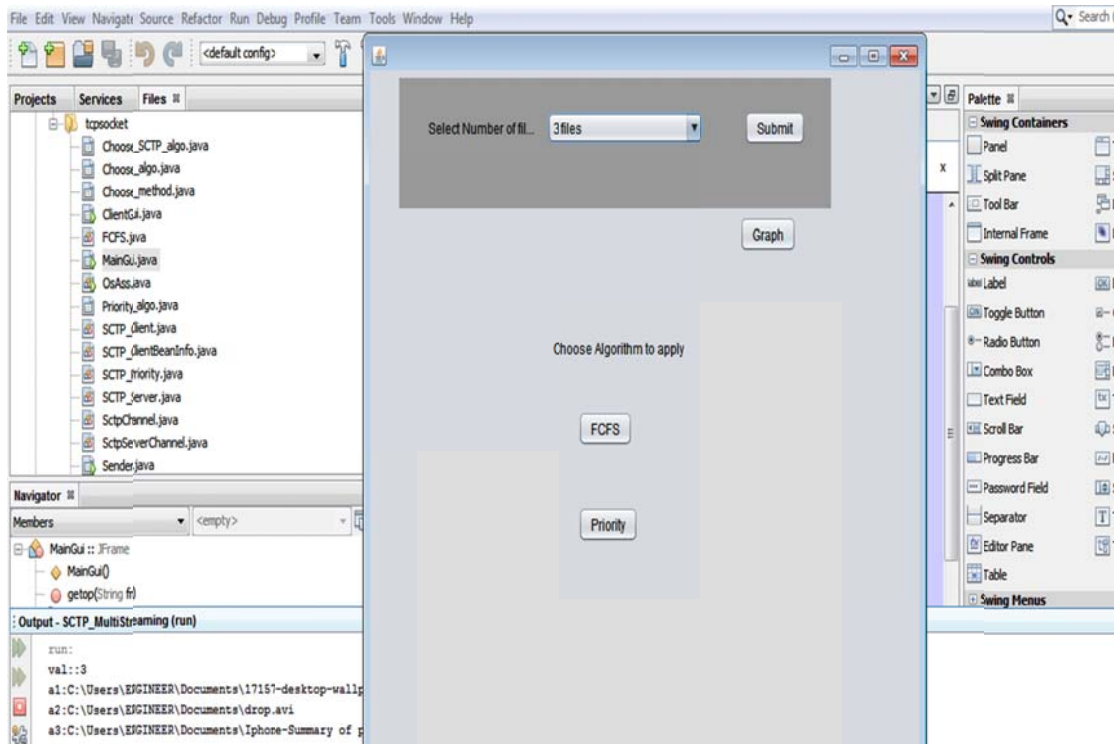
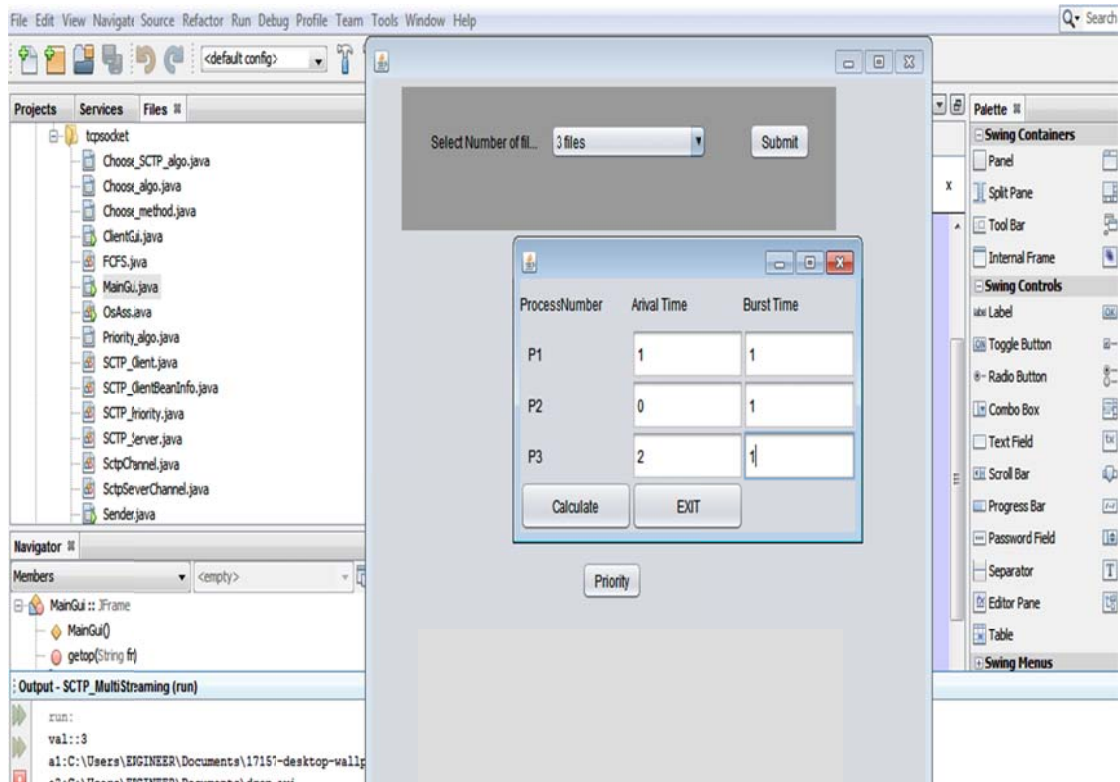


Figure 22: Algorithm selection

### 5.5.5.Fifth Step: Assigning Priority

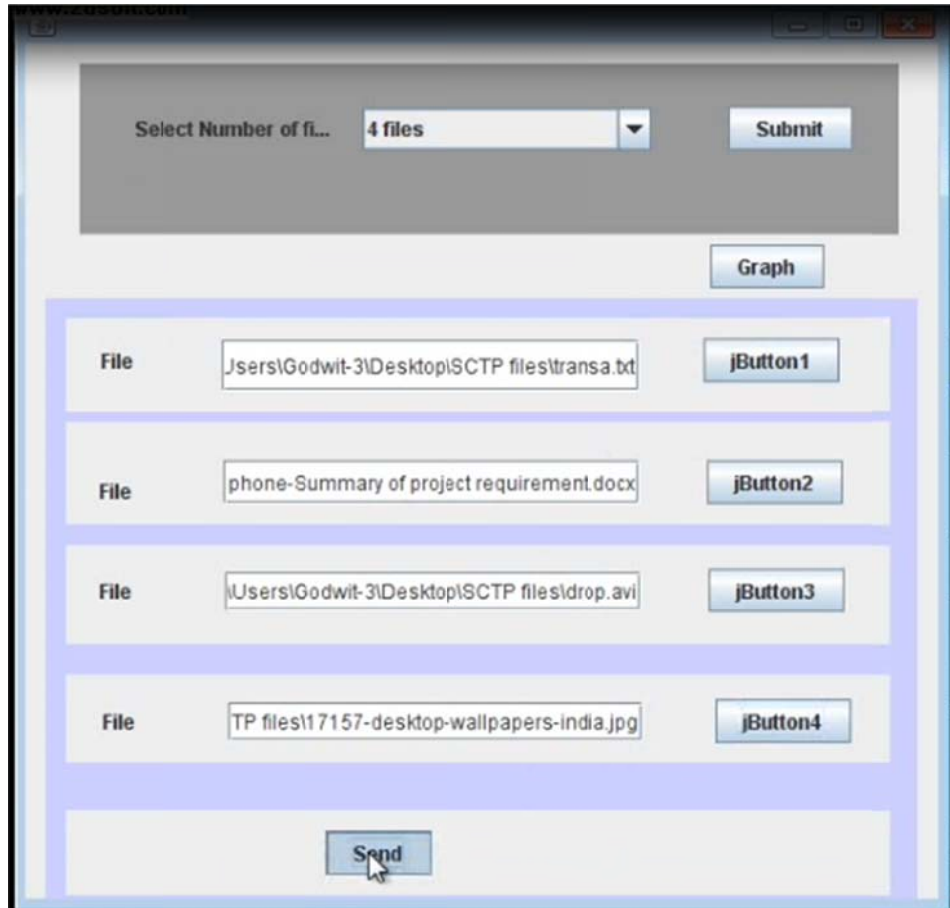
When selecting the priority algorithm, the client should assign the priority of each file. The assigning of priority can be applied in randomly starting from zero to the highest priority, which is one fewer than the number of files.



**Figure 23:** Priority value assignment

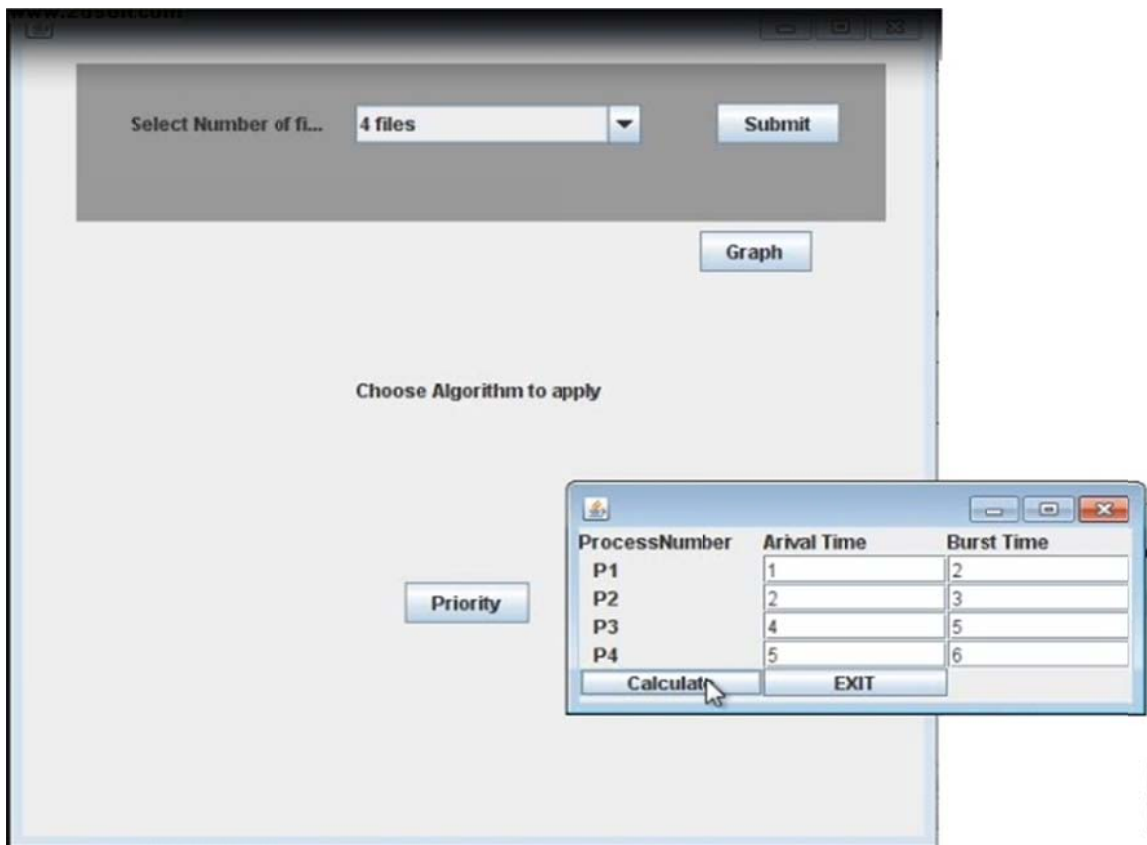
We have made the simulation of sending four files via PP-SCTP. Any type of file can be sent and the sender can assign the priority for each file. If more than one file has the same priority, the socket will deal with them with the FCFS algorithm.

The following figure illustrates how to choose the files:



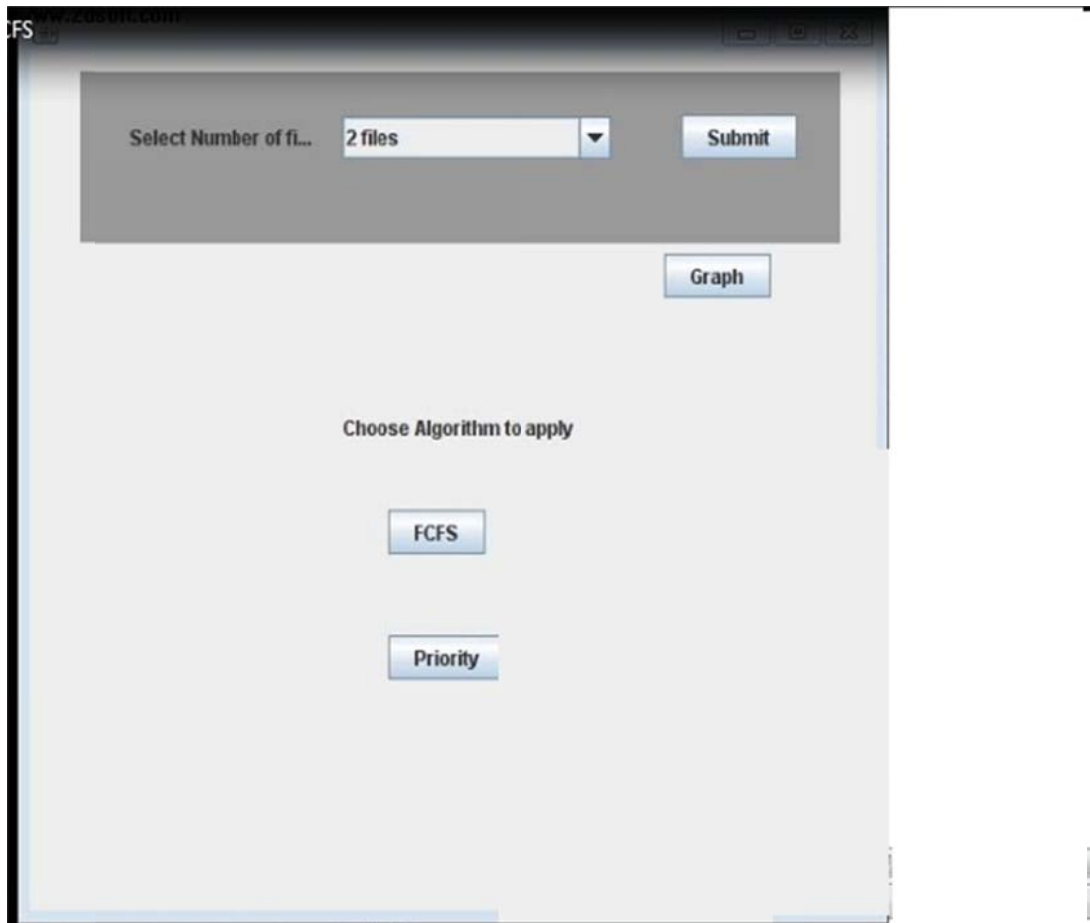
**Figure 24:** Select file for sending interface.

The following figure illustrates how to choose the PP-SCTP:



**Figure 25:** Priority algorithm interface.

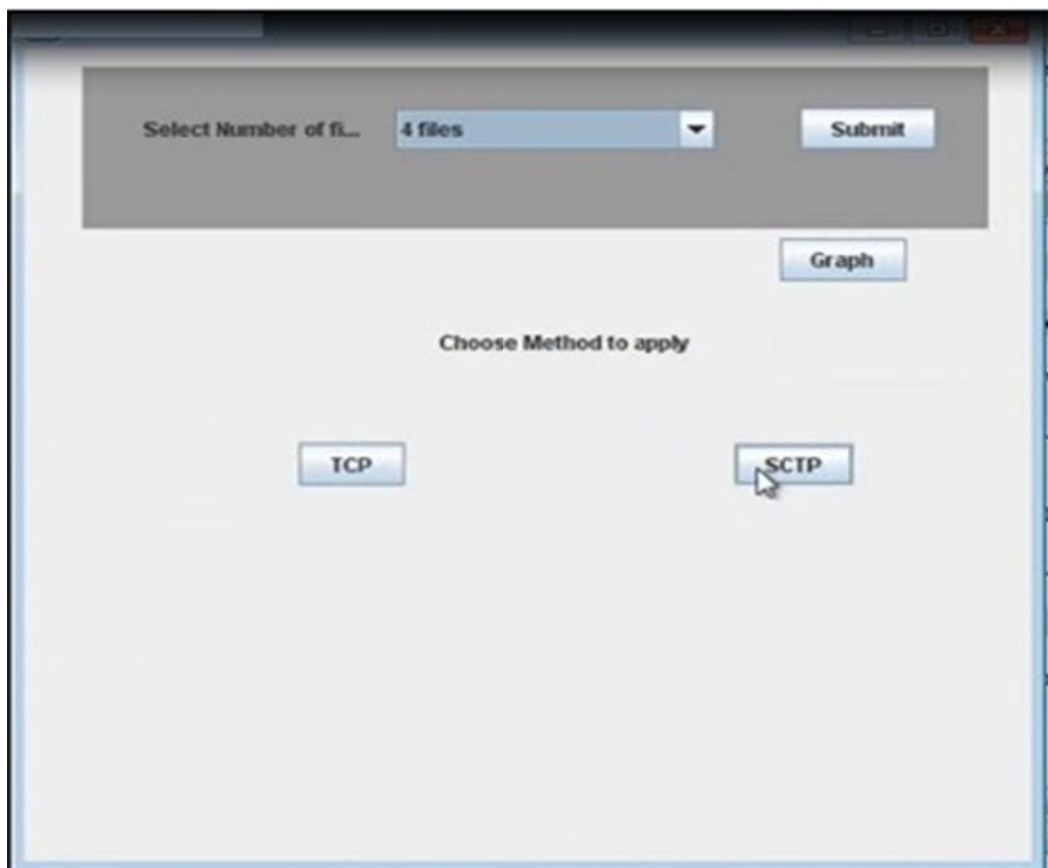
Comparisons with standard SCTP: Then we have applied the standard SCTP with the First-Come, First-Serve algorithm. Therefore, the sender can choose the standard SCTP or the PP-SCTP as indicated below:



**Figure 26:** Selecting priority or FCFS algorithm interface.

## 5.6. Comparison of SCTP to TCP

We also have tested the Transport Control Protocol. Moreover, we have given the sender more flexibility ;therefore, the sender is able to plug into the SCTP or the TCP depending on the network quality or state of service. The following figure illustrates the manner ofchoosing a transport layer protocol:



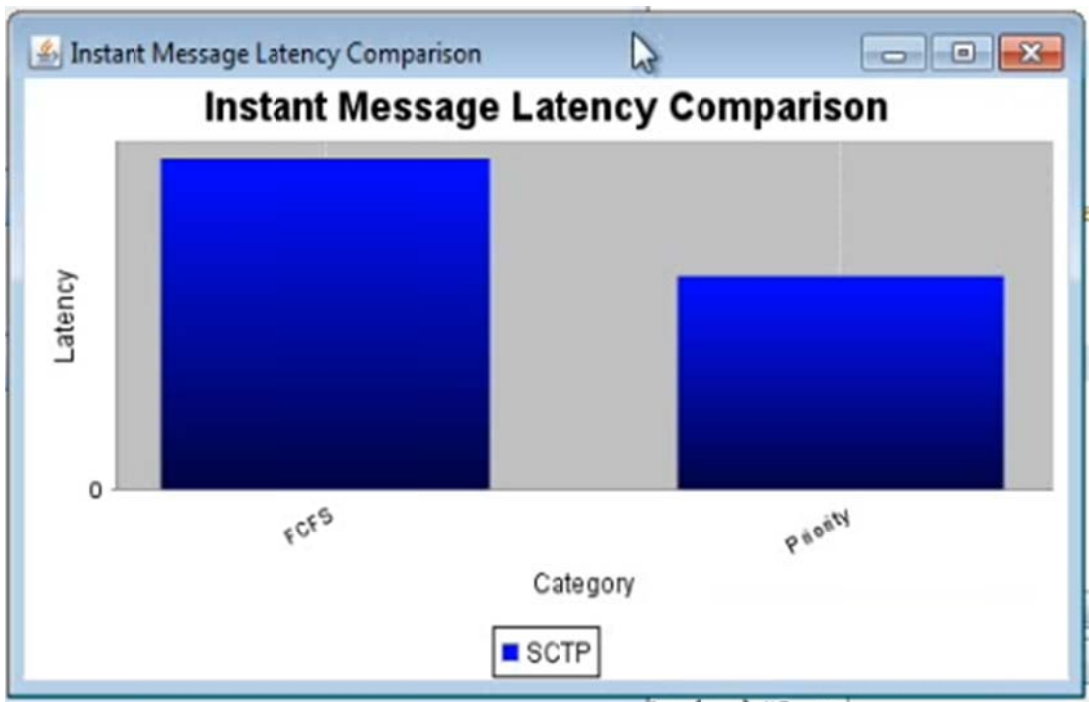
**Figure 27:** Select TCP or SCTP protocol interface.

## 5.7. Discussion of Results

### 5.7.1.RESULTS

First of all, we compared the results of our proposed protocol, PP-SCTP, to the standard SCTP using the FIFS algorithm. Thus, we have seen that an improvement of the SCTP protocol will be efficient. The PP-SCTP gained less latency than the standard protocol.

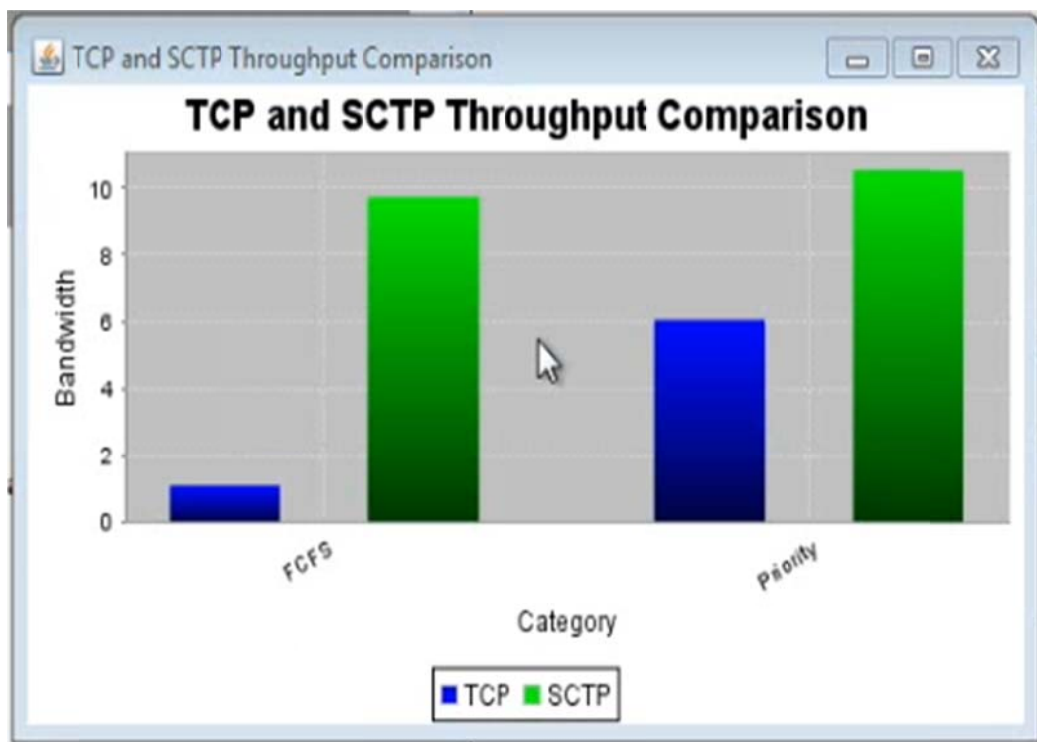
The following figure shows the average latency to send four files in PP-SCTP and the average latency to send the same files in FCFS-SCTP.



**Figure 28:** Average latency of SCTP using FCFS and PRIORITY respectively.

Finally, we have compared the results of our proposed protocol results for the PP-SCTP and the FCFS-SCTP with the TCP with both algorithms. Thus, we have seen that an improvement of the SCTP protocol will also be of benefit.

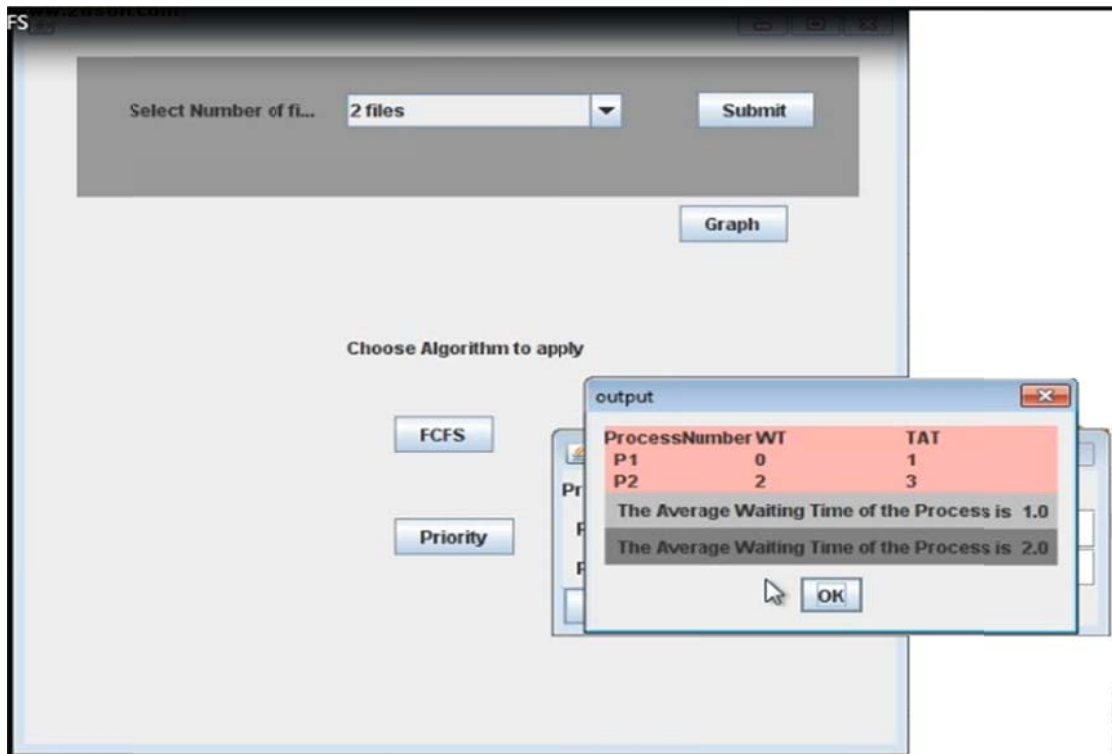
The following figure shows the average throughput to send the same four files using PP-SCTP, FCFS-SCTP, period algorithm with TCP, and FCFS with TCP.



**Figure 29:** Average throughput of TCP and SCTP using FCFS and PRIORITY respectively.

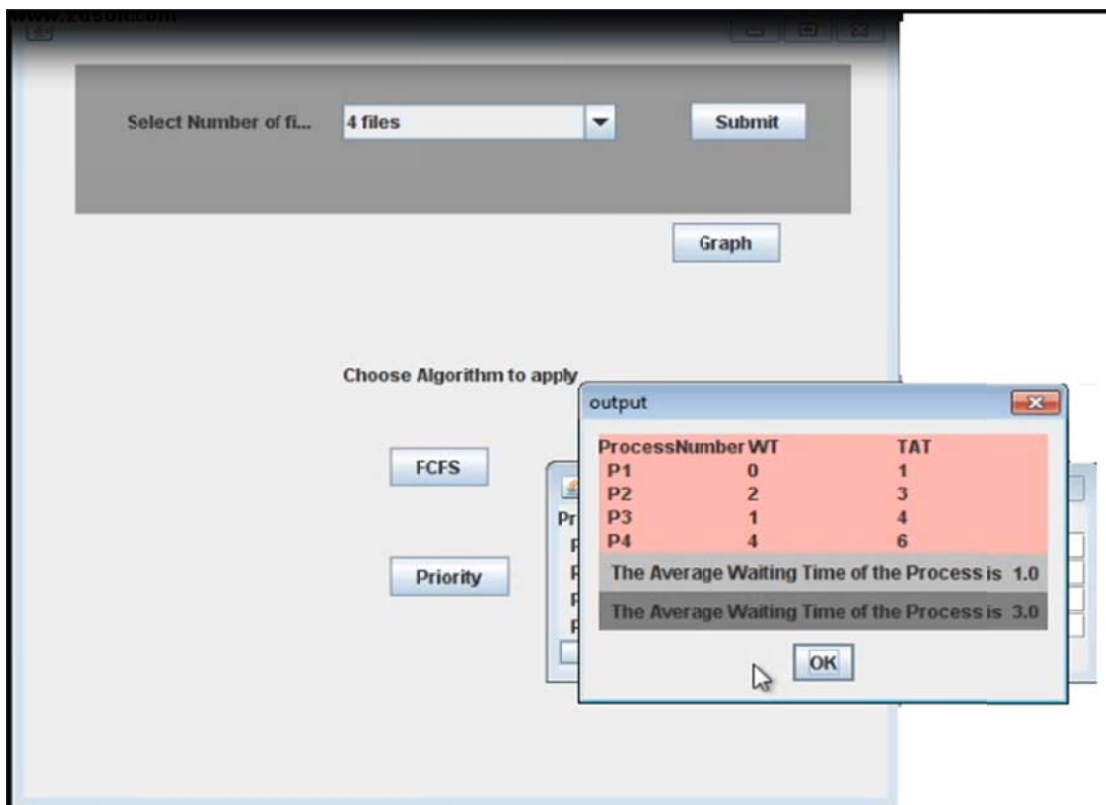


The following figure indicates the average waiting time for the standard SCTP using the PRIORITY algorithm.



**Figure 30:** The average waiting time for PP-SCTP interface.

The following figure indicates the average waiting time for the standard SCTP protocol.



**Figure 31:** Average waiting time for FCFS-SCTP interface.

## CHAPTER 6

### CONCLUSION AND FUTURE WORKS

#### 6.1. CONCLUSION

The intention was to enhance the SCTP protocol as it provides many new options compared to its predecessors, namely the TCP and UDP protocols. The chunks give a more modular and extensible packet format, which is especially advantageous for real-time applications.

The initial motivation for our work was to enhance the SCTP protocol for multimedia applications, the effects of which can be seen on the throughput of the entire network. We defined a per-stream priority scheme as an optional scheduling algorithm. It is debatable as to how a pluggable priority algorithm to the standard SCTP may be added. We show only a sender that is required to modify the socket while a receiver is not required to change his protocol; therefore, the receiver is not aware of the algorithm used at the sender's side. The additional feature of our pluggability is that the sender has the privilege to use the TCP or SCTP protocol sat either priority or standard. The results have shown via simulation that our proposed PP-SCTP protocol has the minimum end-to-end latency under certain conditions.

#### **Technical Limitations**

The Proposed Protocol shows better performance during simulation studies; however, it needs to be tested in a real-time environment.

## **6.2.FUTURE WORKS**

Our work has produced better performance and less end-to-end latency. We recommend working in the future to eliminate the drawbacks of the multi-streaming control protocol such as retransmission delays for thin streams, security and network traffic congestion management. It will be necessary to improve the current algorithms or develop new algorithms to select the best stream in the multi-homing environment.

## REFERENCES

- [1] **Mohd Nazri Ismail and Abdullah Mohd Zin,(2008)**, “A survey on the application of computer network technologies and services over heterogeneous environment”
- [2]**Norman Matloff, (2005)**, “Overview of Computer Networks. Dept. of Computer Science”
- [3] **F. Kurose and Keith W. Ross,(2012)**, “Computer Networking A Top-Down Approach” 6<sup>th</sup> edition
- [4] **International Technical Support Organization Raleigh Centre,(1994)**, “Introduction to Networking Technologies”
- [5] **Randall Stewart,(2001)**, “SCTP New Transport Protocol for TCP/IP”
- [6] **Chonggang Wang, Mahmoud Daneshmand, Bo Li Kazem Sohraby** “A Survey of Transport Protocols for Wireless Sensor Networks”
- [7] **Paul Stalvig,(2007)**, “Introduction to the Stream Control Transmission Protocol (SCTP)”
- [8]**Thomas Dreibholz and Erwin P. Rathgeb**, “Stream Control Transmission Protocol”
- [9] **Thomas Ravier, Rob Brennan and Thomas Curran Teltec,(2011)**, “Experimental studies of SCTP multi-homing”
- [10] **Mohammed Atiquzzaman, and William Ivancic,(2003)**, “Evaluation of SCTP Multi-streaming over Satellite Links”

- [11] **Seok J.Koh, Sang Tae Kim, Jong Shik Ha, (2005)**, “Performance Comparison of SCTP and TCP over Linux Platform”
- [12] **Shaojian Fu and Mohammed Atiquzzaman, (2004)**,“SCTP: State of the Art in Research, Products, and Technical Challenges”
- [13] **Syed Yasmeen Shahdad<sup>1</sup>, Gulshan Amin<sup>2</sup>, Pushpender Sarao<sup>3</sup>, (2014)**, “Multi-Homing and Multi-Streaming Protocol in Computer networks”
- [14] **Yaogong Wang, Injong Rhee, and Sangtae Ha, (2011)**, “Augment SCTP Multi-Streaming with Pluggable Scheduling”
- [15] **Rahul Choudhari and Somanath Tripathy, (2012)**, “SCTP-Sec: A secure Transmission Control Protocol”
- [16] **Robin Seggelmann, Michael Tüxen, Erwin P. Rathgeb, (2010)**, “Stream Scheduling Considerations for SCTP”
- [17] **Klaus D. Gradischnig, and Siemens AG, (2010)**, “Signalling transport over IP-based networks using IETF standards”
- [18] **Irene Rüngeler, Robin Seggelmann and Michael Tüxen, (2011)**, “Stream Control Transmission Protocol”
- [19] **R. Seggelmann<sup>1</sup>, M. Tuxen<sup>2</sup> and E. Rathgeb<sup>3</sup>**, “Design and Implementation of SCTP-aware DTLS”
- [20] **Guanhua Ye Tarek Saadawi Myung, (2011)**, “SCTP Congestion Control Performance in Wireless Multi-Hope Network”
- [21] **Ahmed Abd, Tarek Saadawi, Myung Lee, (2004)**, “LS-SCTP: A bandwidth aggregation technique for stream control transmission protocol”
- [22] **T. Daniel Wallace and Abdallah Shami, (2012)**,“A Review of Multihoming Issues Using the Stream Control Transmission Protocol”

[23]**Ming Zhang, Junwen Lai, Arvind Krishnamurthy, Larry Peterson, Randolph Wang**, “A Transport Layer Approach for Improving End-to-End Performance and Robustness Using Redundant Paths”

[24]**Jianxin Liao, Jingyu Wang, Xiaomin Zhu, (2008)**, “Proceeding cmpSCTP: An Extension of SCTP to Support Concurrent Multi-path Transfer”

[25] **M. Fiore a, C. Casetti a, G. Galante, (2007)**,“Concurrent multipath communication for real-time traffic”

[26] **C. Casetti and R. Greco, and G. Galante, (2011)**, “Load balancing over multipaths using bandwidth-aware source scheduling”

[27] **LukaszBudzisz, Johan Garcia and Anna Brunstrom, (2012)**, “A Taxonomy and Survey of SCTP,” Research Technische Universität Berlin Karlstad University

[28]**Wallace A. (2012)**, “Concurrent Multipath Transfer: Scheduling, Modelling, and Congestion Window Management” Thesis

[29] **Thomas Dreibholz and Erwin P. Rathgeb, (2011)**, “Technologies Stream Control Transmission Protocol,” University of Duisburg-Essen

[30] **Thomas Danil, (2012)**, “Concurrent Multipath Transfer: Scheduling, Modelling, and Congestion Window Management,” (Thesis format: Monograph); Wallace

[31] **Professor Dr. Peter Brucker, (2007)**, “**Scheduling Algorithms**,”Fifth Edition Universität Osnabrück Fachbereich

[32] **C. Casetti and R. Greco, and G. Galante**, “Load balancing over multipaths using bandwidth-aware source scheduling”

[33]**Michael Hammond, (2002)**, “Programming for Linguists”

[34]**Behrouz A. Forouzan, “Stream Control Transmission Protocol (SCTP)”**,Fourth Edition Chapter 16

[35][https://www.google.com.tr/search?q=SCTP+structure+look+figure&=MOWzU7WEI\\_doAT61YKobQ&=0CFQQsAQ&=1246&=564](https://www.google.com.tr/search?q=SCTP+structure+look+figure&=MOWzU7WEI_doAT61YKobQ&=0CFQQsAQ&=1246&=564)

[36]**H.-Y. Hsieh and R. Sivakumar, (2002)**, “pTCP: An End-to-End Transport Layer Protocol for Striped Connections”

[37]**J. Crowcroft and P. Oechslin (1998)**,“Differentiated End-to-End Internet Services Using a Weighted Proportional Fair Sharing TCP”

[38] **J. Zou, (2007)**, “Preferential Treatment of SCTP Streams in a Differentiated Services Environment” Ph.D. dissertation, City University of New York.

[39]**T. Dreibholz, R. Seggelmann, M. Tüxen and E. P. Rathgeb (2010)**,“Transmission Scheduling Optimizations for Concurrent Multipath Transfer”.

[40] **R. Seggelmann, M. Tuexen, and E. P. Rathgeb (2010)**, “Stream Scheduling Considerations for SCTP”

[41] **G. Heinz, (2003)**, “Priorities in SCTP Multistreaming ”, Master’s thesis, University of Delaware

[42]**ProQuest Information and learning Company,(2008)**, " UMI,UMI Microform 328884, book: google books.google.com.tr