ADAMA SCIENCES & TECHNOLOGY UNIVERSITY

School of Electrical Engineering & Computing

Department of Computing

A MASTER'S THESIS

ON

Attempt to optimize Selective Repeat protocol by minimizing the positive and negative acknowledgement

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE DEGREE

OF

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

BY:

DAHLAK DANIEL SOLOMON

GSR/02969/07

Advisor:

Dr. N. S. Kumar

Jan, 2017

Declaration

This thesis is my original work, and it has not been presented for a degree in any other university. All sources of materials used for the thesis have been acknowledged

_____

Dahlak Daniel

This thesis has been submitted for examination with my approval as university advisor

_____

N. S. Kumar (PhD)

January, 2017

ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

SCHOOL OF ELECTRICAL ENGINEERING & COMPUTING

DEPARTMENT OF COMPUTING

Attempt to optimize Selective Repeat protocol by minimizing the positive and negative acknowledgement

BY:

DAHLAK DANIEL SOLOMON

Names and Signature of Members of the Examining Board

_____	_____
Advisor's name	Signature
_____	_____
Chair Person's name	Signature
_____	_____
External Examiner's name	Signature
_____	_____
Internal Examiner's name	Signature
_____	_____
Department head name	Signature
_____	_____
School Dean name	Signature

Acknowledgement

I don’t know how to thank my mother Biskut Teshome and my father Daniel Solomon, if you were not with me along this narrow and long road it is most unlikely I am reaching this position and doing this Master’s thesis, GOD Bless you.

I would like to thank my advisor Dr. N. S. Kumar for his continuous and constructive reviews of this work.

Last but Most, Thank you GOD for everything you did for me to rich to this position.

Abstract

Data Link layer is a layer that is responsible for controlling the flow control, error detection and error correction. There are protocols that are used for implementing Data Link layer; one of the most efficient protocols is Selective Repeat protocol by its ability of re-transmit only the damaged frame in case of transmission error. The receiver sends positive or negative acknowledgment to show whether the frame is received correctly or damaged. When the sender receives the acknowledgement the sender will send new frames or it will re-transmit the damaged frames based on the acknowledgment. Sending acknowledgment for each frame is tedious work and also it consumes bandwidth as well as time.

This study intended to minimize the individual acknowledgment process to cumulative acknowledgment at transport layer. Our designed optimized protocol use cumulative acknowledgment between the sender and the receiver to make the protocol more efficient acknowledgment, time and bandwidth usage to do this we design the protocol on transport layer with packet.

Keywords: Selective Repeat, Stop-and-Wait, Data Link Layer, Positive acknowledgment, Negative Acknowledgment, Siding Window

Table of Contents

List of Tables

List of Figure

Nomenclature

ACK ----- Positive Acknowledgment

ARQ ----- Automatic Repeat Request

EVDO ----- Evolution Data Optimized

Gbp/s ----- Gigabytes per Second

NAK ----- Negative Acknowledgment

OSI ----- Open System Interconnection

PDU ----- Protocol Data Unit

SAK ----- Some Acknowledgment

TCP ----- Transmission Control Protocol

TCP/IP ----- Transmission Control Protocol/Internet Protocol

UDP ----- User Datagram Protocol

CHAPTER ONE

1. The Problem and its Background

1.1 Introduction

The concept of networking is now dominating the world with its amazing idea that can help as to share data to anywhere and anytime without any physical movement required. Beside to that there are also types of networking that can be used to transfer data like Flash Disk, Compact Disk, Hard Disk, etc. These kinds of transferring data are very useful in case of large data but in case of speed they are slow so after many research the invention of networking became more interesting. With the current technology by using fiber optics cable we can send data with the speed of 10Gbp/s.

In order to share data the sender and the receiver must have the same protocol that will allow the devices to communication with each other. This problem became the ground to structuring the networking protocol commonly to create a general network structure that will be used by all computers. The current protocol is called TCP/IP model that is implementing by differentiate the network into layers. Each layer have it is own purpose also communicate with the adjacent layers. Data link layer is the fourth layer which is responsible to deals with flow control, error correction and detection.

The current Data link layer which is widely used is Selective Repeat protocol [1]. This protocol is sliding window based protocol which send frames up to the size of the window size and as the acknowledgments comes from the receiver the sender will shift the window and send the frames from the next window. This protocol can be more efficient in case of noise channels. Because unlike other protocol Selective Repeat protocol sends only the damaged or error frames this save bandwidth that is used to re-transmit the frames that are correctly received by the receiver[2]. The acknowledgment process between the sender and the receiver follow the individual acknowledgment for each frame by sending positive acknowledgement for frames that are received without error and negative acknowledgment for the frames that are damaged or error occurred. Based on the acknowledgement received by the sender will send the frames from the

next window if the received acknowledgment is positive and if the received acknowledgment is negative the sender will send the only the damaged or the error frames.

1.2 Background of the study

TCP/IP reference model is the current architecture that is widely used. There are around seven layers that are used for this model; data link layer is one of the layers. [3] This layer is responsible for providing a well-defined service interface to the network layer, dealing with transmission errors and regulating the flow of data so that slow receivers are not swamped by fast senders. There are two protocols that have been used by data link layer which are Stop-and-Wait and Sliding Windows. Sliding Windows is more preferable that Stop-and-Wait because Stop-and-Wait protocol send only one frame and until the acknowledgment received the sender will not send other frames but in case of Sliding Windows the sender will send as a size of the window size without waiting acknowledgment for the first frame, as the sequence acknowledgment received the sender send the frames from the next window.

Sliding Windows protocol has three implementations the first one is A One-Bit Sliding Window Protocol, the second one is Go-Back-N and the last one is Selective Repeat protocol. The Go-Back-N protocol works well if errors are rare, but if the line is poor it wastes a lot of bandwidth on retransmitted frames. An alternative strategy, the Selective Repeat protocol, is to allow the receiver to accept and buffer the frames following a damaged or lost one.

According to Selective Repeat protocol is more widely using for implementing data link layer. Because Go-Back-N protocol will consume more bandwidth if the channel is noise, due to it re-transmitting all frames in the windows if any frame have errors during transmission. But in case of the Selective Repeat protocol only the frames that have errors will be re-transmitted.

The basic idea behind the Selective Repeat ARQ protocol is that the receiver accepts packets that are out of order and requests only those frames to be transmitted which are in error [4]. Selective Repeat protocol has many advantages that can improve network performance as well as it consume less bandwidth when we compare it with another protocols that are provided to adopt data link layer. But still Selective Repeat protocol has some disadvantages that influence the protocol to consume time and bandwidth. These disadvantages came due to the acknowledgment process that is required as each frame are sent and received. The receiver must send positive or

negative acknowledgments for each frame that has been received to notify the sender the status of the received frame. This is time consuming and it requires bandwidth to send acknowledgments for each frame.

1.3 Statement of the Problem

In Selective Repeat protocol the sender will first set a frame size and then it send each frame up to the window size limit at the receiver side, the receiver will acknowledge each frames receiver one by one. When the acknowledgment is received by the sender the sender send the other windows size. In case of error occurrence on the transmission the receiver send negative acknowledgment to the send to tell the receiver to re-transmit the lost or error frames and the sender re-transmit the lost frames again. This fashion makes the sender to receive positive or negative acknowledgement for each sent frames and also on the receiver side the receiver must send positive and negative acknowledgment for each received frames. This procedure consume more time and also bandwidth because to send positive or negative acknowledgment as well as to receive the acknowledgments. This problem also creates tedious workloads that influence the performance of both parties.

1.4 Purpose of the Study

To fill the gap between senders and receivers depending on the transmission speed, there are several protocols used to maintain the transmission and receiving speed. The existing Selective Repeat protocols are giving more number of positive and negative acknowledgements. In this thesis we successfully implemented the modified Selective Repeat protocol, so that it will produce minimum number of positive and negative acknowledgements. When we test and compare the protocol with other protocols by using the designed software tool, it proves better performance than the other existing protocols. This study motivates the researchers to concentrate in this area.

1.5 Objectives

1.5.1 General Objective

The general objective of this study is to optimize Selective Repeat protocol to more efficient protocol by minimizing the positive and negative acknowledgement.

1.5.2 Specific Objectives

- To analysis how Selective Repeat protocol algorithm works.
- To identify the acknowledgement processes of the sender and the receiver.
- To examine the necessity of why the acknowledgement process has to be minimized.
- To explore how the acknowledgement process of each frame has been affecting the network performances.
- To find a way to minimize the acknowledgement process to provide more efficient protocol.
- To customize the algorithm of the Selective Repeat.

1.6 Research Specific Questions

The following are the research questions for the study that it will try to find the answers.

- Does the current acknowledgment process is efficient?
- Do we need to change individual acknowledgment to cumulative acknowledgment?
- How to transform individual acknowledgment to cumulative acknowledgment?
- Investigate how cumulative acknowledgment improves the current efficiency?

1.7 Theoretical / Conceptual Framework

In general, To provide reliable data transfer the flow control and error control protocols like Stop-and-Wait, Go–Back-N and Selective Repeat are combined together to get effective data link control. Flow control refers to set of procedures used to refine the data which is transmitted by the sender before waiting for ACK from the receiver. Error control refers to correct and detect the errors. Specifically in unreliable networks, the reliable data transfer service was provided by ARQ in a way that the receiver asks the sender to re-transmit the lost data or damaged data that can't be corrected. This will guarantee that the data is received by the destination without any loss data in the middle of the transmission and also the data that are damaged are corrected to provide error free data. The main advantage is that the destination will receive data's in which the order of data transmitted by the sender without error in the data.

1.8 Scope and Limitation of the Study

Due to the limited amount of time available for this thesis research, the scope of this study will be limited on the acknowledgment process of Selective Repeat protocol which is focused on optimizing the protocol in which way that the acknowledgment process can be cumulative rather that individual acknowledgment. Also the study will includes measurement of the performance about the optimized protocol to show how the study improved the protocol to more efferent manner.

The study has been face by the following limitation during the time of analysis, conducting and developing the research

- Limited time for the study
- Finding a simulation software that allow to work at Data Link layer to transmit frame from the sender to the receiver and to test the performance of the proposed protocol at Data Link layer
- Related works that are similar to the idea of changing the individual acknowledgment

1.9 Significance of the Study

This research will make the Selective Repeat protocol to use cumulative acknowledgment rather that individual acknowledgment which is more efficient accordingly to the current acknowledgment process. Sending and receiving acknowledgment for each frame is time and bandwidth consumes process that affects the efficiency of the protocol. Imagine that if the sender sends ten frames the receiver must acknowledge all ten frames, but in the proposed optimized protocol the receiver will acknowledge the tenth frame to show that the reset frames are correctly received in case of all frames are correct. In case of damaged frames the receiver send negative acknowledgment for that frames and acknowledge the reset correctly received frames.

Using cumulative acknowledgment minimize the time and bandwidth that are consumed in order to indicate the sender whether the frames are correctly received or not. So this study plays a great role in the networking technology in terms of providing fast communication.

1.10 Beneficiary of the Research

The study has benefits for the sender, the receiver and the devices that are found between the sender and the receiver. Because the sender will be beneficiary by sending packet will less time and by performing less workload for acknowledgment process, the receiver will be beneficiary be receiving packets will less time and by sending only one acknowledgment for one sliding window rather than one acknowledgment for each packets and lastly the devices occurred between the sender and the receiver will be beneficiary by receiving and forwarding less packets for the packets that sent from the sender to the receiver.

Since the study has a new idea of acknowledgment process this study will make a new researchers that are interested beneficiary by leading them the way and by providing information's what study has been done on this same idea.

In general this research provides benefit on many aspects of networking starting from the current performances of data communications technology. In a way of providing fast and efficient communication with better performance to the users.

1.11 Organization of the thesis

This section describes the organization of the rest of the thesis. Chapter one discusses about background information about Selective Repeat protocol, the statement of problem, objectives, methodologies and related concepts are presented. Chapter two presents the conceptual discussion about Selective Repeat protocol and detailed literature review. What the current acknowledgments look like and how it works in both sender and receiver side.

CHAPTER TWO

2. Conceptual Discussions and Literature Review

2.1 Network Software and Organization

The first computer networks were designed with the hardware as the main concern and the software as an afterthought. This strategy no longer works. Network software is now highly structured.

2.1.1 Protocol Hierarchies

To reduce their design complexity, most networks are organized as a stack of layers or levels, each one built upon the one below it. The number of layers, the name of each layer, the contents of each layer, and the function of each layer differ from network to network. The purpose of each layer is to offer certain services to the higher layers while shielding those layers from the details of how the offered services are actually implemented. In a sense, each layer is a kind of virtual machine, offering certain services to the layer above it.

This concept is actually a familiar one and is used throughout computer science, where it is variously known as information hiding, abstract data types, data encapsulation, and object-oriented programming. The fundamental idea is that a particular piece of software (or hardware) provides a service to its users but keeps the details of its internal state and algorithms hidden from them.

2.1.2 Design Issues for the Layers

Some of the key design issues that occur in computer networks will come up in layer after layer. Below, we will briefly mention the more important ones. Reliability is the design issue of making a network that operates correctly even though it is made up of a collection of components that are themselves unreliable. Think about the bits of a packet traveling through the network.

2.2 Reference Models

As we try to describe on the above section the networking software is organized in to layers. Based on what are organized in to layers there are two important architectures of network reference model which are the OSI reference model and TCP/IP reference model. In the next subsections we will try to describe the reference models in details.

2.2.1 OSI reference model

This model is based on a proposal developed by the International Standards Organization (ISO) which is developed by Day and Zimmermann on 1983. After that it is revised by Day on 1995. The model is called the OSI (Open Systems Interconnection) Reference Model because it deals with connecting open system that is, systems that are open for communication with other systems.

[9] OSI generally specifies the functions required for data communication and the interaction between these functions. The OSI model stipulates that higher-level functions build on lower-level functions according to well-defined rules. The result is therefore a hierarchical concept consisting of layers.

The OSI model has seven layers. The principles that were applied to arrive at the seven layers can be briefly summarized as follows:

- ✓ A layer should be created where a different abstraction is needed.
- ✓ Each layer should perform a well-defined function.
- ✓ The function of each layer should be chosen with an eye toward defining internationally standardized protocols.
- ✓ The layer boundaries should be chosen to minimize the information flow across the interfaces.
- ✓ The number of layers should be large enough that distinct functions need not be thrown together in the same layer out of necessity and small enough that the architecture does not become unwieldy.

Let us describe the seven layers of the [3] OSI reference model each in detail that are shown on Fig 1.

- **The Physical Layer: -** This layer is responsible to transmitting raw bit over the communication channel.
- **The Data Link Layer: -** The main task of the data link layer is to transform a raw transmission facility into a line that appears free of undetected transmission errors. It accomplishes this task by having the sender break up the input data into data frames and transmits the frames sequentially.
- **The Network Layer: -** The network layer controls the operation of the subnet. A key design issue is determining how packets are routed from source to destination.
- **The Transport Layer:-** The basic function of the transport layer is to accept data from above it, split it up into smaller units if need be, pass these to the network layer, and ensure that the pieces all arrive correctly at the other end. Furthermore, all this must be done efficiently and in a way that isolates the upper layers from the inevitable changes in the hardware technology over the course of time. It also determines what type of service to provide to the session layer, and, ultimately, to the users of the network.
- **The Session Layer: -** The session layer allows users on different machines to establish sessions between them. Sessions offer various services, including dialog control, token management, and synchronization.
- **The Presentation Layer: -** The presentation layer is concerned with the syntax and semantics of the information transmitted.
- **The Application Layer: -** The application layer contains a variety of protocols that are commonly needed by users. HTTP, file transfer, electronic mail, and network news are some example of application layer protocol.

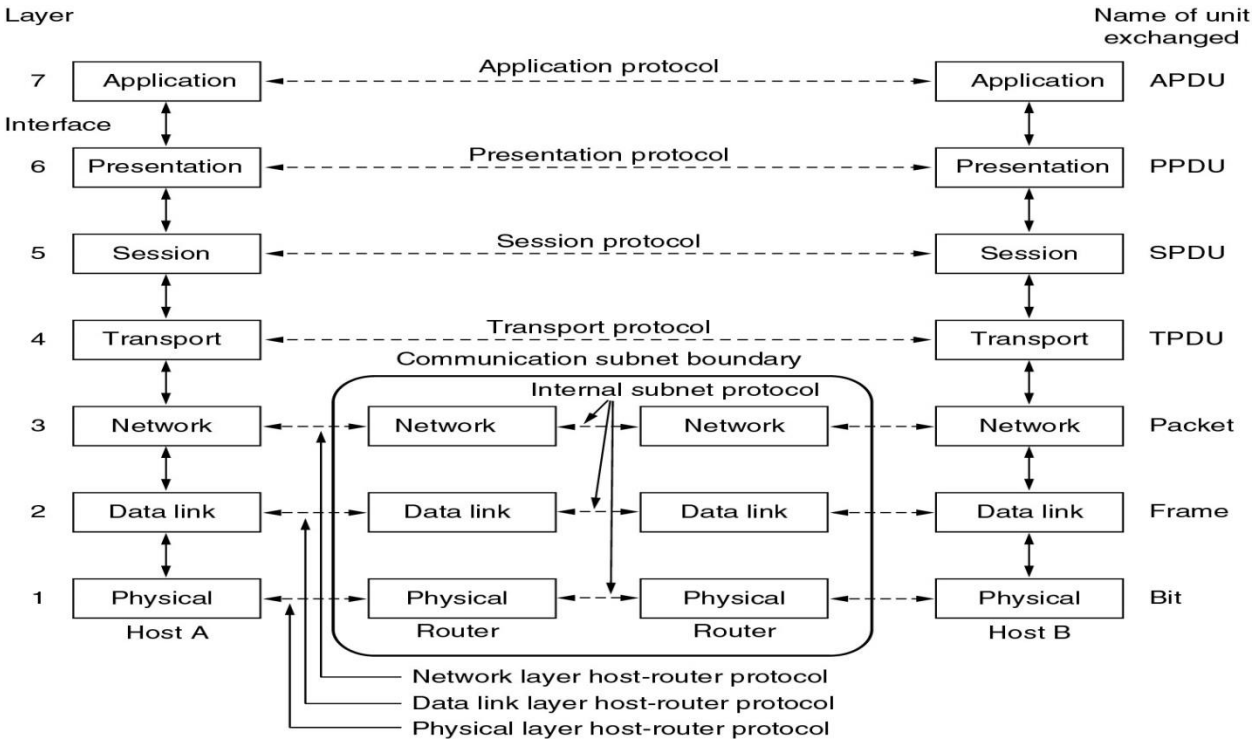


Figure 1.OSI reference model

2.2.2 TCP/IP reference model

The main objective for the development of this protocol is to connect multiple networks in a seamless way. It was first described by Cerf and Kahn (1974), and later refined and defined as a standard in the Internet community (Braden, 1989). The design philosophy behind the model is discussed by Clark (1988). Fig 2 shows [3] the TCP/IP reference model.

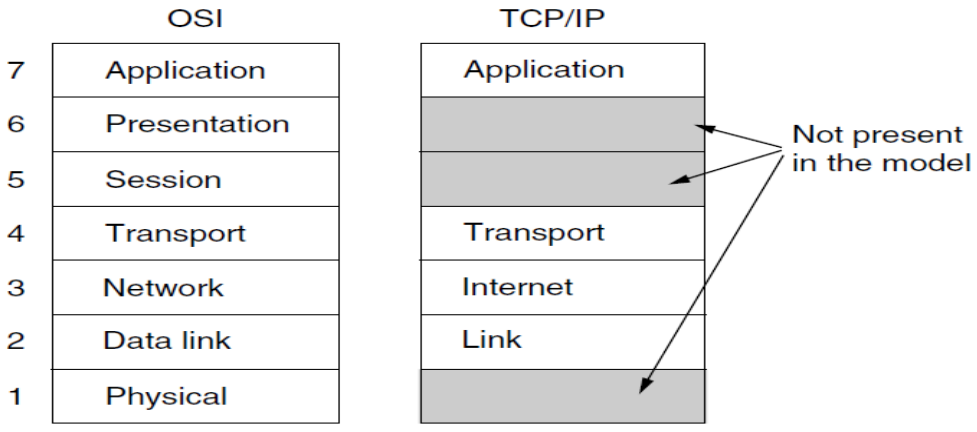


Figure 2.TCP/IP reference models

- **The Link layer: -** The lowest layer in the model, the link layer describes what links such as serial lines and classic Ethernet must do to meet the needs of this connectionless internet layer. It is not really a layer at all, in the normal sense of the term, but rather an interface between hosts and transmission links.
- **The Internet layer: -** The internet layer is the linchpin that holds the whole architecture together. Its job is to permit hosts to inject packets into any network and have them travel independently to the destination
- **The Transport Layer: -** The layer above the internet layer in the TCP/IP model is now usually called the transport layer. It is designed to allow peer entities on the source and destination hosts to carry on a conversation, just as in the OSI transport layer.
- **The Application Layer: -** The TCP/IP model does not have session or presentation layers. No need for them was perceived. Instead, applications simply include any session and presentation functions that they require.

2.3 Data link layer

Data link layer is the layer that founded on both reference models. Data link layer is has many algorithms for achieving reliable, efficient communication of whole units of information called frames. In order to accomplish this data link layer will break data into many frames with same amount of size.

The data link layer uses the services of physical layer to send and receive bits over the communication channel. [3] Data link layer has numbers of function which are

- Providing a well-defined service interface to the network layer

  As data link layer is below the network layer and each layer has services that will be provided to the adjacent layers, this function is focused on providing a well-defined service interface to the network layer.
- Dealing with transmission errors

  Since there is no perfect channel to transmit data without any chance of error, Data link layer should deal with errors. This means the layer has a responsibility to detect and correct error.
- Regulating the flow of data so that slow receivers are not swamped by fast senders

This means that the layer has to control the flow of the data transmission. In this case if there is a performance difference between the sender and the receiver, the data link layer has a responsibility to adjust the flow of the transmission.

2.4 Functions of Data Link Layer

Whenever there is a data to be sent first the network layer will notify the data link layer about the data. Then there will be many things that the data link layer will perform before it gives it to the physical layer and also during the transmission. The following are aspects that the data link layer will handle.

2.4.1 Framing

To provide service to the network layer, the data link layer must use the service provided to it by the physical layer. What the physical layer does is accept a raw bit stream and attempt to deliver it to the destination. The usual approach is for the data link layer to break up the bit stream into discrete frames, compute a short token called a checksum for each frame, and include the checksum in the frame when it is transmitted. When a frame arrives at the destination, the checksum is recomputed. If the newly computed checksum is different from the one contained in the frame, the data link layer knows that an error has occurred and takes steps to deal with it.

[16] Data-link layer takes packets from Network Layer and encapsulates them into Frames. Then, sends each frame bit-by-bit on the hardware. At receiver's end Data link layer picks up signals from hardware and assembles them into frames.

Breaking up the bit stream into frames is more difficult than it at first appears. A good design must make it easy for a receiver to find the start of new frames while using little of the channel bandwidth.

2.4.2 Flow Control

[15] In the data communication, flow control manages the transmitted data between sender and receiver. First, data will be transmit to the receiver, after it is received by receiver, an acknowledgment will be sent back to sender, and then the sender sends the next data.

Another important design issue that occurs in the data link layer is what to do with a sender that systematically wants to transmit frames faster than the receiver can accept them. This situation can occur when the sender is running on a fast, powerful computer and the receiver is running on a slow, low-end machine.

2.4.3 Error Control

The usual way to ensure reliable delivery is to provide the sender with some feedback about what is happening at the other end of the line. [14] In ARQ-based error control the packet is retransmitted if it is found to have errors. Such packets are retransmitted until it is received error free. The error detection is usually implemented through a cyclic redundancy check (CRC).Typically, the protocol calls for the receiver to send back special control frames bearing positive or negative acknowledgements about the incoming frames. If the sender receives a positive acknowledgement about a frame, it knows the frame has arrived safely. On the other hand, a negative acknowledgement means that something has gone wrong and the frame must be transmitted again.

2.5 Data link layer implementation

The data link layer provides basic addressing and controlling access to the physical layer, as well as subnet routing. A subnet is a small part of the complete network like LAN or small component of LAN. This layer provides easy mobility of hosts among the subnet, without requiring reconfiguration.

The data link layer is implemented in three phases, each of which performs different tasks. These are generation, execution and analysis. In the first phase, due to the large number of hosts under execution and heterogeneous nature of LAN topologies, it was better to generate the network topologies dynamically (Tree, Torus, Mesh and Cube) in order to send the data. The main output of generation phase is a topology file which contains the number of hosts, bridges and connections between them.

In the execution phase, once the topology has been determined, either by creating the topology manually or the topology generated from the generation phase. This is the core phase and was responsible for the major work which includes setting up the network by reading a topology file and converting it into a series of respective objects, setting up the data streams across the

network based on the input file which details which hosts to send what data to and from which hosts, enabling the tracing on each of the nodes and the traced files are connected to the correct sources.

The analysis phase was used to get the useful data. The files emitted from the above phase must be analyzed and compared. By analyzing we can get the information like how efficient the routing protocols are and how long each packet took to traverse the network and other execution results.

2.6 Data link layer protocols

In any broadcasting networks, the stations must ensure that only one station transmits at a time on the shared communication channel. The data link layer protocols are used to perform the corresponding operations in the sub layers of data link layer.

2.6.1 Elementary protocols

The protocol that determines who can transmit on a broadcast channel are called Medium Access Control protocol. The MAC protocol is implemented in the MAC sub layer which is the lower sub layer of the data link layer. The higher portion of the data link layer is often called logical link control (LLC).

- **Serial Line IP:** First protocol for sending IP data grams over a dial up links
- **Point to Point Protocol:** Used for dial up and for high speed routers. The supporting protocols are link control protocol, password authentication protocol and network control protocol, IP control protocol.
- **High level Data Link:** Default protocol for serial links in Cisco routers and it does not uses addresses. Maximum frame size is 1500.

2.6.2 Sliding window protocols

[11] This shows a different performance in terms of their efficiency, complexity and buffer requirement. Sliding window protocol assumes full duplex communication. It uses two types of frames, first data and second acknowledgment. One of important features of all the sliding windows protocol is that each outbound frame contains a sequence number, ranging from 0 to $2^n - 1$, where the value of n can be arbitrary. Sliding window refers to imaginary boxes at the

transmitter and receiver. This window provides the upper limit on the number of frames that can be transmitted before acknowledgment requirement. Window holds the number of frame to provide above mention limit. The frames which are being transmitted to send are falling in sending window similarly frames to be accepted are store in the receiving window.

[13] A sliding window protocol provide for reliable data transfer over an error channel. Basically any ARQ protocol, the receiver sends an acknowledgment for any correctly received packet. If the acknowledgment is not received by the transmitter within specified time period (because either the data packet or acknowledgment itself was lost), the transmitter's timer expires and the transmitter resends the presumably lost data message.

The three sliding window protocols are bidirectional protocols. These three differ among themselves in terms of efficiency, complexity and buffer management. The main use of sliding window protocols is that at any point of time, the sender maintains a set of sequence numbers corresponding to frames it is permitted to send. Similarly the receiver also maintains a receiving window corresponding to the set of frames it is permitted to accept. The sender window and receiver window need to have same upper and lower limits i.e. same frame size. In most of the protocols the frame size is fixed, but in some protocols the frame size is variable.

The sequence number in the senders window represents the frames that have been sent or been sent but not ACK. Whenever a new packet arrives from the network layer, it is given the next highest sequence number and upper edge of the window is incremented by one. When an ACK come in the lower edge will be incremented by one. In this way the window continuously maintain non ACK frames.

At the receiving side, any frame falling outside the window is discarded without a comment. When a frame whose sequence number is equal to the lower edge of the window is received, it is passed to the network layer, an ACK is generated and window is rotated by one. The receiver window always remains at initial size.

2.6.2.1 GO-Back-N

[12] The Go-Back-N protocol tends to increase the efficiency of Stop-and-Wait by introducing parallelism in the frame transmission. The transmitter continuously transmits information frames

without waiting for ACKs. The window size, which represents the limit on the number of frames at the transmitter. It is important to note that due to the pipeline effect in the Go-Back-N protocol, new information frames are being transmitted and received during the processing and propagation periods and the time spent on transmitting the ACK.

In this scheme, the transmitter sends the code words continuously and stores them to wait for acknowledgements. Buffer space for n packets is needed at the transmitter. The acknowledgements for code word arrive after round trip delay, during which N-1 other code words are transmitted. In this protocol, the sender is allowed to transmit multiple packets without waiting for an ACK, but is constrained to have no more than some maximum allowable number let N, of unacknowledged packets in the pipeline. The range of permissible sequence numbers for transmitted but not yet ACK packets can be viewed as Window size N. The Go-Back-N protocol is referred as Sliding Window protocol. In this protocol, an ACK for packet with sequence number n will be taken to be a cumulative ACK, which indicates that all packets with sequence number up to and including n have been correctly received at the receiver. Go-Back-N is derived from the sender's behavior.

2.6.2.2 Selective Repeat

The basic idea behind the selective repeat protocol is that the receiver accepts packets that are out of order and requests only those frames that are to be transmitted which are in error. The receiver must have enough buffer space to store the error free packets when an error is detected in previously received packets.

[10] Selective repeat protocol allows the receiver to accept frames out of order. The out-of-order frames will be stored in a buffer, sorted, and passed to higher layers in the correct order. The selective Repeat protocol is much more efficient than Go – Back –N protocol, since only negatively acknowledged code words are retransmitted. After resending the negatively acknowledged code word, the transmitter continues transmitting new code words in the transmitted buffers.

In this protocol if the channel is error free the transmissions continues from the first frame to the last frame in the frame order. This means that on the start of the transmission, the sender will send frames until it reach the window size. As soon as the receivers send acknowledgment for

the transmitted packet, the senders send new packets. This process continues until the send finish all frames. This process is clearly shown in the next figure.

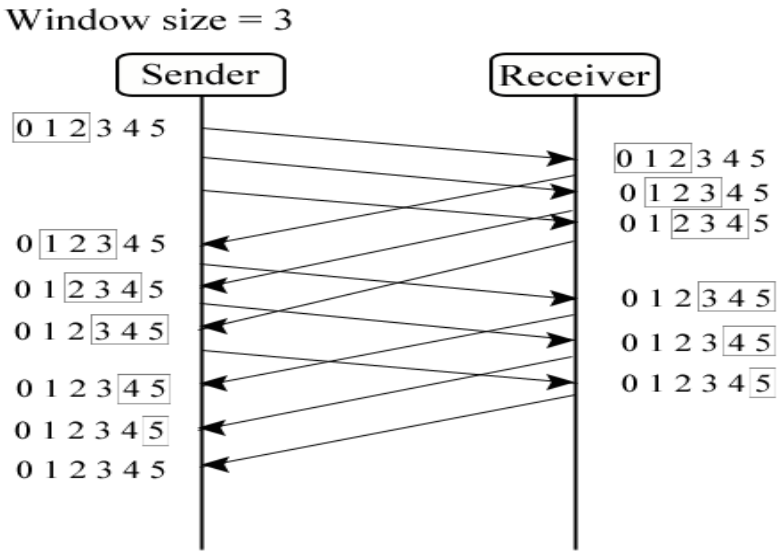


Figure 3.Selective Repeat protocol on error free channel

But if the communication channel is not stable and if it is nosy channel there will be a chance of lost packet or the received packet may contain error. In this case the receiver send negative acknowledgment for the send, this negative acknowledgment indicate that the there is a lost packet in the middle of the transmission or it can indicate that the packet contain error. The next figure shows how the Selective Repeat protocol will deal when the packet lost in the middle of the transmission.

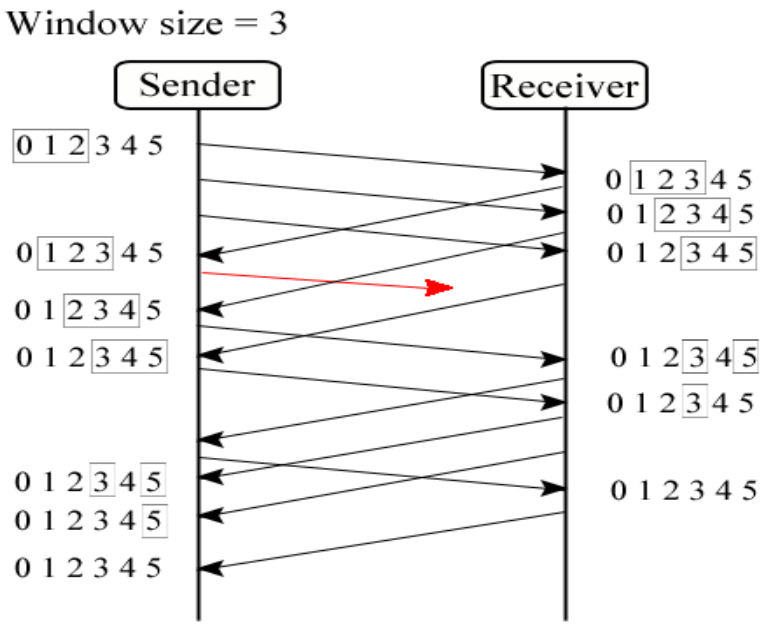


Figure 4.Selective Repeat with noisy channel

There is a two steps process in the selective repeat protocol. (i) Setup Window size and Operation. For n bit sequence number the maximum window size is 2n-1. In case of damaged or lost frame the sender retransmits those which are negatively acknowledged and those for which timer expires.

2.7 Related works

In order to make the study well organized we review related papers that are previously done that are very useful input for our study. In this section we briery review those researches in order to point out what other methods has been developed as a solution and also to show the differences with the present work.

In [5] Implementation of Data Link Control protocols in wired network they proposed a new protocol that is called Priorities Retransmission. Their proposed protocol has buffer on both side of receiver and sender. It allows the sender to have more than one outstanding frame at a particular time and receiver to accept out of order frames and store it in its window. Sender for Priorities Retransmission is modified from that of Selective Repeat. It collects all the NAK which is received from receiver side and store it into an array. If only one NAK is received from

receiver side it simply retransmits that frame for which a NAK is received. If more than one NAK is received then find the minimum sequence number of NAK and retransmitted that frame, this process is going on until all NAK frames are not retransmitted. It uses optimum bandwidth than Selective Repeat protocol.

Accordingly to their research they used a priority mechanism to allow the receiver to tell the sender what frame has to be re-transmitted first and based on the priority the sender re-transmit the NAK frames. This study proposed best solution in case of NAK frames, but still there is gap in case of ACK frames because still the receiver have to send acknowledgment for each individual frames.

The research proposed new protocol which will be best in error channel because during the transmission there might be damaged packet so there proposed protocol will send the packets with the negative acknowledgment by ordering them in there priority, this will be grate in case of quality of transmissions.

The drawback of this research is their modification of the Selective Repeat protocol is focused on only for negative acknowledgment for providing quality of transmission. In case of our study, our proposed protocol is focusing on minimize the acknowledgment which can reduce the load of the transmission. Our proposed protocol is will best solution in case of error free and noisy channel.

In [6] Good put analysis of a modified cumulative ARQ they try to increase throughput, reduce delivery delay by improvement of cumulative ARQ. In the process of the traditional cumulative ARQ mechanism transmission, when the first lost PDU appears, the successful received PDU before the first lost PDU will be fed back. This PDU and the PDU after it will wait to the next transmission opportunity to retransmit. In this way, the transmission causes large delivery delay and waste of channel.

Their modified mechanism works, in the first frame, since the PDU with sequence number 3 has been lost and identified by the receiver, the largest sequence number among all successfully received PDUs is 2. The received PDUs with continuous lost state or continuous successful state are divided into a sequence and we need contain the number of PDUs in each sequence. In this

way, they just need to know the information of sequence feedback and retransmit the lost sequence. In this study storing the received frame state is the main aspect that they use to improve the protocol.

The idea behind the research is they identify the lost frame sequence and they transmit the sequence of the lost frame to the sender to show that what are the frames that are received damaged. But still we have to acknowledge each frame.

The drawback of the research is they only focused on the delay of delivery which cause when a damaged frame is received in the middle of the transmissions. But our proposed protocol will focus on the changing the individual acknowledgment process to cumulative acknowledgment which can solve delivery delay as well. Because when the packet is damaged form the sent window, the damaged packet will not affect the rest of the packet since the proposed protocol include the damaged packet in the next window with other new packets to be transmitted.

The related works that we mention on the above indicates that there have been attempts to create more efficient protocol accordingly to provide packets without delivery delay and packets that are have priority before other packets. This research plays a great role to insure quality of transmission.

But still there is a chance to make more efficient protocol by using the acknowledgment process as a key aspect of the improvement. In fact this study focused on changing the individual acknowledgment to cumulative acknowledgment which can provide the protocol that use less bandwidth and time beyond quality of the transmissions. In fact if we change the individual acknowledgment to cumulative acknowledgment there is also a chance of minimizing the probability of the packet which can be damaged in the transmissions.

CHAPTER THREE

3. Methodology

Research Methodology is the general principle that guides the research. In order to conduct, a good research, a well-defined approach and principle has to be followed. In this section, a specific methodology that guides this research work is described.

3.1 Research Design

The research strategy provides a framework for design a systematic study that would address the study's goal, objectives, and questions. This section summarizes the overall study design. The research strategy type is Quantitative research, because the research will be contacted by comparing and evaluating based on the previous protocol with the new protocol.

3.2 Process Model

Most of the process models are considered which are characterized by the values of bit error rate and packet error rate. The main concern in data communications systems is how to control transmission errors so that the received data is error free. The communication channels may have different kinds of behavior such as additive noise and fading which was caused by multipath propagation and inter symbol interference. As a result, the received signal is badly destroyed that the received message cannot be reconstructed unless some kind of error control is used. There are two approaches to control the error are forward error correction and selective repeat. In this research we are more concentrating on selective repeat protocol. If the receiver detects errors in the received word, it generates a retransmission request, or a NACK. If no errors are detected, the receiver sends positive ACK to the transmitter.

3.3 Issues of Reliability and Validity

The reliability and validity are two important terms at Application, Transport and Data Link Layers. The reliable data transfer means that the data are received ordered and error free. The main problem of implementing reliable data transfer occurs not only at the transport layer but also at data link layer. The service provided to the upper layer entries is that of through reliable channel where no transferred data bits are corrupted or lost and all are delivered in the order in

which they are sent. It is the responsibility of the receiver to validate the received packets and sending respective ACK.

3.4 Sampling Techniques

The sampling technique that we will use for this research is based on the average of the test result that we will conduct for Stop-and-Wait, Selective Repeat and the optimized Selective Repeat protocol. This means that after we recorded test result that we will get from each protocol on different situations, we will sample the results based on different factors that we will discuss on the next chapter.

3.5 The Development of the Prototype

The research question that we intended to find answer needs prototype in order to show that what we try to accomplish and also to what level of point that the research will be evaluated to show what are the key aspects that the research provide.

As we try to state the necessary of the prototype development for this research, we try to develop three different prototypes. The prototypes will be developed using the same idea of sending packets from the sender to the receiver. But the prototype will use Stop-and-Wait, Selective Repeat, and our new protocol as one protocol for one prototype. By developing these three prototypes we intended to show how the new protocol will be better by comparing it with the previous protocols.

For Stop-and-Wait, Selective Repeat and the optimized Selective Repeat protocol we will develop the server and the client prototypes. The server on each prototype will make sure that the packet is forwarded between the clients. Firstly we will start the server JAVA application, this server application will open a port and it will wait the client to join on the server. The client on each prototype will ask to connect to the server when we opened them. If the server send ok packet they will be added on the client list and the server will automatically notify the other clients about the new client. Each client will use different port that can help the server to identify and transmit packets to specifically the required client. In the clients application the other client information will be displayed.

The prototype that we will develop will allow each client to send plain text to other client which is currently online on the server. Since each client notified about neighbors clients, simply the client that wishes to transmit text to other client will select the receiver for the listed online clients.

Whenever the client wants to transmit data to another client, firstly there must be at least one client which is active on the server. After the sender client selects the receiver client, the sender will provide the text. When the sender press the send button, automatically request packet is sent to the server and if the server is busy it will be rejected and notified to the sender. But if the server is not busy, the request packet will be sent to the receiver. When the receiver receive the request packet, if the receiver is busy it will send busy packet to the server and the server notify the sender. But if the receiver is not busy and want to accept the connection, it will select the window size and send accepted packet to the server and the server transfer the packet to the sender. When the sender receive the accept packet it will start sending the packet.

The data link layer on each client is responsible to breaking the message data in to packets. The framing operation is based on the each character on the message. Each character is treated as a single packet on the transmission. So based on the number of the packets the transmission continued until there is no packet left.

3.5.1 Reference Model Design for the Prototypes

As reference models differ from network to network, the simulation prototypes for Stop-and-Wait, Selective Repeat and Optimized Selective Repeat protocol have their reference model. We design the reference model by categorizing the functions into five layers. Each layer have its own function to provide to the above and the below layers. Fig 5 shows the reference model that will be used in the prototype simulations.

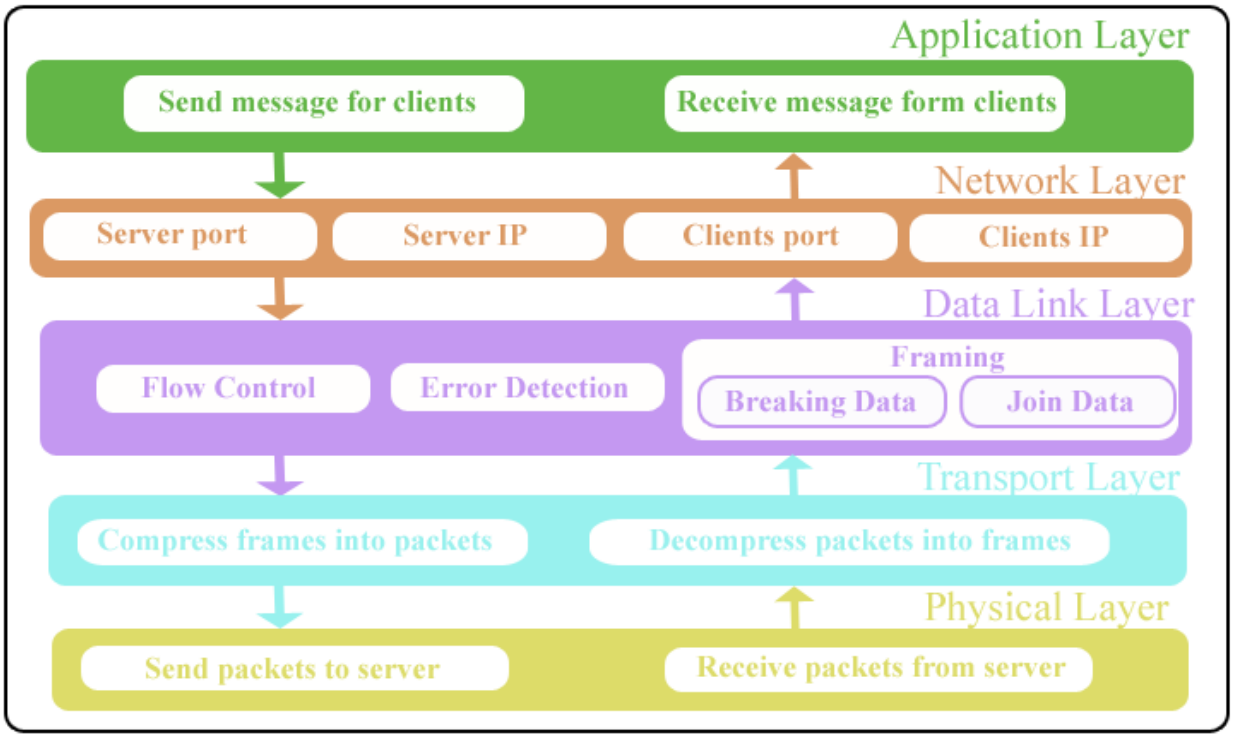


Figure 5.Reference model used on the research

As we try to describe the reference model that we will implemented for the development of the prototype. In the following section we will try to describe what are the responsibilities and functions of each layer from the reference model design.

- **Application Layer: -** This layer will be responsible to allow the sender to send and the receiver to receive message. This will be implemented by using JAVA application interface that we will develop for the prototypes.
- **Transport Layer: -** This layer is responsible to compress frames that are provided by Data Link layer into Datagram packets and to decompress the Datagram packets that are received by Physical layer into frames as well as providing the decompressed frames for Data Link layer.
- **Network Layer: -** This layer is responsible to learn the server and connected client's information such as on port they are running and what the IP address is. When Application layer provide the message this layer provides the message, the server information and the destination client information to the Data Link layer. And also when

message is provided from Data Link layer to this layer, the layer will provide the message with the sender information to the Application Layer.

- **Data Link Layer: -** This layer is responsible to flow control, error detection, breaking messages into frames and joining frames to message. And also it is responsible to provide message to the Network layer with the sender information.
- **Physical Layer: -** This layer is responsible to transmit the packet that is provided from Transport layer to the server with the information about the destination IP address and port number. And also it is responsible to provide packets received from the port to the Transport Layer.

3.5.2 The Architecture of the System

The architecture of the system will be the same for the Stop-and-Wait, Selective Repeat and Optimized Selective Repeat protocol prototypes. The difference of the protocols will be based on how the data link layer will perform on the acknowledgment process and how many packets will be sent before acknowledgment.

Based on how the network reference model is designed on the prototype that shown on Fig 5, on this section we will try to show the architecture design of the optimized Selective Repeat protocol. Fig 6 shows the architecture design, on this design it shows the general overview of the sender and receiver design. Based on the deign we will try to discuss the idea of the architected design on the sender on the receiver

Sender

- **Prepare to send module:** This module is responsible to prepare information about the receiver and the packets to be sent to the receiver. This module has three sub modules.
  - **Receiver Info:** This sub module will prepare the receiver information's including the port and IP address.
  - **Set WS:** This sub-module will set the window size that will be used on the time of the transmissions. Which means that the maximum amount of packets that can be sent to the receiver without waiting acknowledgment.
  - **Packeting:** This sub-module is responsible for breaking messages into packets and putting the packets on the buffer.

- **Send packet module:** This module is deals with sending packets to the receiver and also this module have two sub-modules.
    - **Count packet on buffer:** This sub-module is responsible to count the packets that are putted on the buffer from the above module. Based on the packets on the buffer this sub-module will identify how many packets will be left after sending the first window.
    - **Send first window:** This sub-module will responsible to send packets that are on the first window.
- **Acknowledgment module:** This module is responsible to wait for acknowledgment and handle the next step based on the received acknowledgment. This module has three sub-modules.
    - **Wait for acknowledgment:** This sub-module is responsible to wait acknowledgment for the first window which is sent on the send packet module.
    - **Positive acknowledgment:** This sub-module is responsible to send next window after waiting for when acknowledgment and if it is positive acknowledgment.
    - **Negative acknowledgment:** This sub-module is responsible to send the previous window again when acknowledgment is received and if it is negative acknowledgment.
    - **Some acknowledgment:** This sub-module is responsible to identify the damaged packets on the previous window, adding new packets from the buffer in the correctly received packets and prepare the new window and sending the window if the received acknowledgment some acknowledgment.

Receiver

- **Prepare to receive module:** This module is responsible to prepare the receiver to receive packets from the sender. This module has three sub-modules.
    - **Sender Info:** This sub-module is responsible to identify and store sender information like port number and IP address.
    - **Select WS:** This sub-module is responsible to identify what is the maximum amount of packets should be on the window. So be selecting the window size it sends to the sender the window size.

- **Create buffer:** This sub-module is responsible to create a buffer for the packets that will be received from the sender.

❖ **Receive packet module:** This module is responsible to count how many packets are received from the sender, check errors on the packets and store the correct packets on the buffer. This module has three sub-modules.

- **Store on the buffer:** This sub-module is responsible to store the correctly received packets on the buffer.
- **Error detection:** This sub-module is responsible to check the received packet for errors.
- **Count packets:** This sub-module is responsible to count the received packets from the window which can be a base to send acknowledgment for the sender.

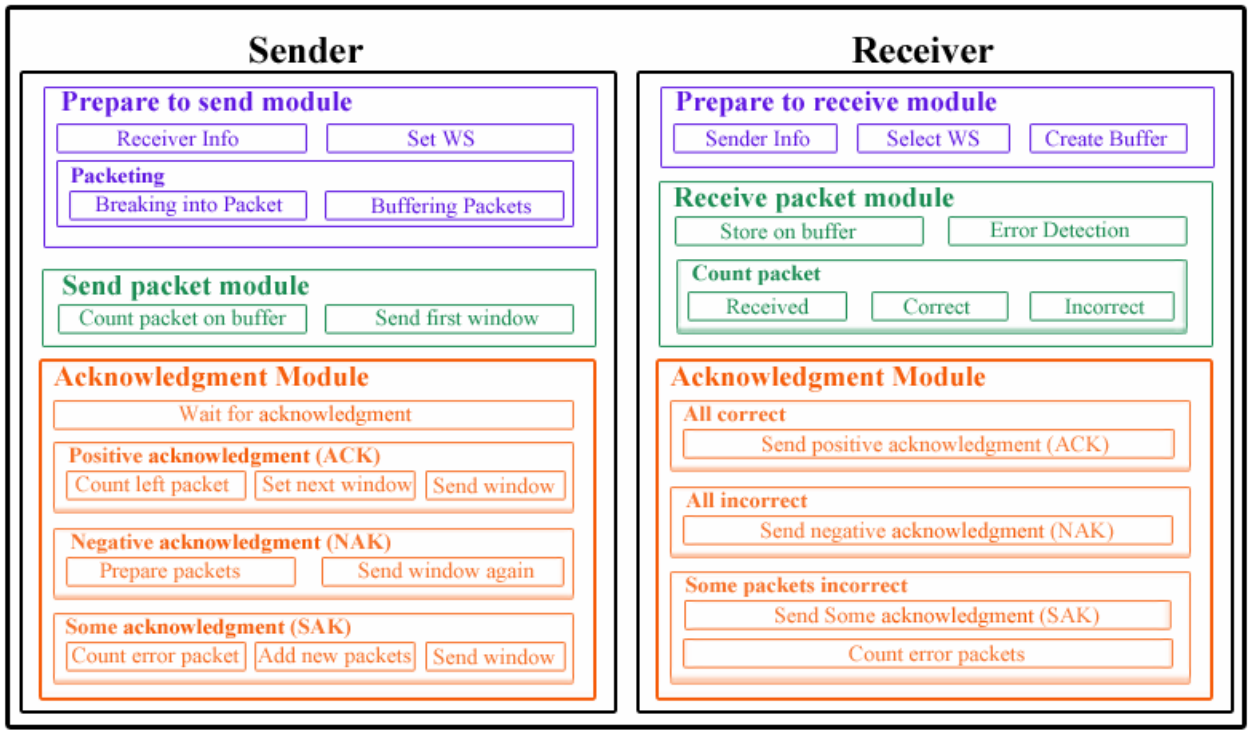


Figure 6.Architecture of the prototype

❖ **Acknowledgment module:** This module is deals with sending acknowledgment to the sender based on the count of packets received and the status of the packets. This module has three sub-modules.

- **All correct:** This sub-module is responsible to send a positive acknowledgment (ACK) when all packets on the window is received and all packets are correct.
- **All incorrect:** This sub-module is responsible to send a negative acknowledgment (NAK) when all packets on the window are received and all packets are incorrect.
- **Some incorrect:** This sub-module is responsible to responsible to send some acknowledgment (SAK) when all packets are received and some packets are incorrect. On the some acknowledgment packet this sub-module is also responsible to send the id of the packets that are received incorrectly.

As we try to show by using the design and explanations the Optimized Selective Repeat protocol uses cumulative acknowledgment. To transform the individual acknowledgment to cumulative acknowledgment in addition to the positive and negative acknowledgment it needs third acknowledgment type which indicated from the window, some packets are correctly received and some are incorrectly received or lost in the time of the transmission. So when the sender receive positive acknowledgment it sends the next window packets but if it receive negative acknowledgment it will send the packets on the window again. But if the receive some acknowledgment it will identify the id of the lost or damaged packets and it will formulate new window by adding new packets in the place of the correctly received packets and it sends the window.

3.5.3 Requirements Modeling

Selective Repeat ARQ is a specific instance of the automatic repeat request protocol used for communications. It may be used as a protocol for delivery and acknowledgment of message units, or it may be used as a protocol for the delivery of sub divided message sub units. It is reliable and offers a distinct advantage over the naïve Stop-and-Wait protocols. Since the protocol uses shifting windows at both the sender and receiver side. The requirement modeling may deals with

- ✓ The goal is to totally or partially transfer a certain non-empty original sequential file from one site to another.
- ✓ Total transfer means that the transmitted file is a copy of the original.

- ✓ Each site may end up either if believes that the protocol has transmitted successfully or the protocol has aborted.
- ✓ The receiver should maintain a sliding window and buffer the packet it receives if it lies in the range of window.
- ✓ The sender also has to maintain a sliding window. All packets with index not larger than the send base, ACK have been received and packets beyond the window have not been sent.
- ✓ All ACK packets at the sender should have been correctly received at the receiver.

To meet the requirement modeling we consider the combinations of data and object modeling to show how the research prototype will be implemented by using Data flow diagram and System flow chart diagram.

The data flow diagram describes shows the basic logic how the sender will send to the receiver and how the receiver from the sender with the mechanism that they use for sending and receiving acknowledgment on both selective repeat protocol and optimized selective repeat protocol simulation prototype.

The system flow chart diagram is designed only to the optimized selective repeat protocol simulation. It describes the producers and mechanism that the server follow in case of new client request to join, old client says good bye, clients request to send message and when the transmission is finished. On the client side also it shows the general review of how the client simulation prototype operates.

- ➢ **Data flow diagram**

The data flow diagram for this prototype is designed for the sender and the receiver separately. As the Stop-and-Wait, Selective Repeat and our new protocol have different data flow, in this section we will show only the Selective Repeat protocol and the new protocol data flow diagrams.

The data flow diagram for Selective Repeat protocol will be considered as two different aspects which are the sender and the receiver flow. As we try to show on the below diagram firstly the

sender will send setup packet to the receiver and if the receiver want to accept the data it will send setup packet with the size of the window size.

When the sender receiver the setup packet it will set the window size and start sending packet up to the window size and it will wait for the acknowledgment. As soon as the receiver send positive acknowledgment the sender will send next packet but if the receiver send negative acknowledgment the error or lost packet will be transmitted again.

On the receiver side the receiver will wait for any event and if there is a setup request event automatically it will select window size and it send setup packet to the sender. After that when packet is receiver it will check the packet send positive or negative acknowledgment for correct packet and incorrect packet respectively.

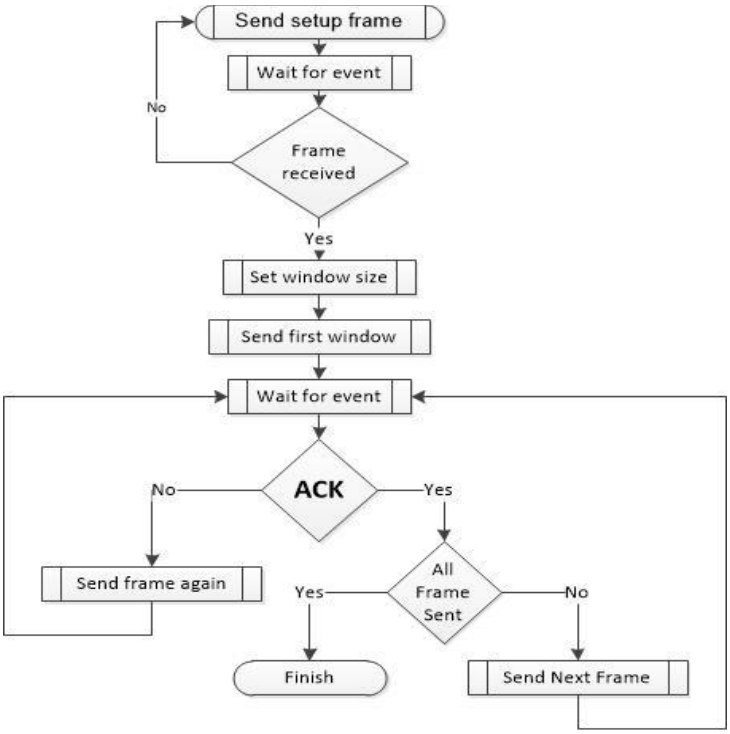


Figure 7.Selective Repeat sender data flow diagram

Fig 7 shows Selective Repeat sender data flow diagram for the prototype. The following are description of the steps that shown on the diagram

A. The sender will send setup packet to the server to check whether the server or the receiver can handle new transmission and the sender will wait for event or for replay

B. If the sender didn't receive packet or if the server say it is busy it will delay for some time and send the setup packet again
C. When packet is received the sender will set the window size based on the receiver request and it will send the first window packets to the server
D. After sending the first window the sender will wait for acknowledgment for sent packet
E. When an event is occur the sender check whether it is positive acknowledgment or not
F. If it is positive acknowledgment then it will check if all packets are transmitted if all packets are transmitted it will finish the transmission but if there are left packet it will send new packet to the sender
G. If it is negative acknowledgment it will send the packet again

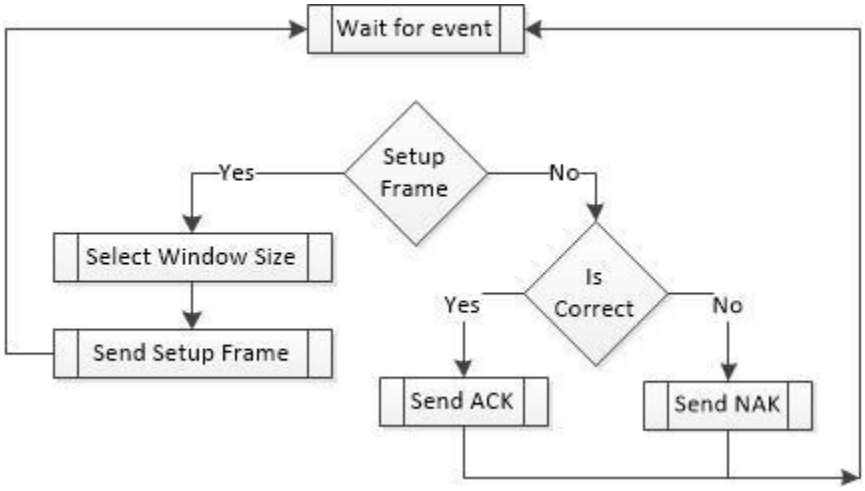


Figure 8.Selective Repeat receiver data flow diagram

Fig 8 shows Selective Repeat receiver data flow diagram for the prototype. The following are description of the steps that shown on the diagram

A. The receiver will wait for packet
B. When packet is received it will check whether it is setup packet or not
C. If it is setup packet it will set window size and send the setup packet back to the server
D. But if it is data packet it will check whether the packet is correct or not
E. If it is correct it will send positive acknowledgment
F. If it is in correct it will send negative acknowledgment

The new protocol that we aim to design to minimize the positive and negative acknowledgment also has two different aspects which are the sender and receiver data flowchart diagram.

Fig 9 shows the optimized Selective Repeat sender data flow diagram for the prototype. The difference between the Selective Repeat sender and Optimized Selective Repeat sender is on the Selective Repeat protocol one acknowledgment is only for one packet but in the Optimized Selective Repeat sender one acknowledgment is for the whole window.

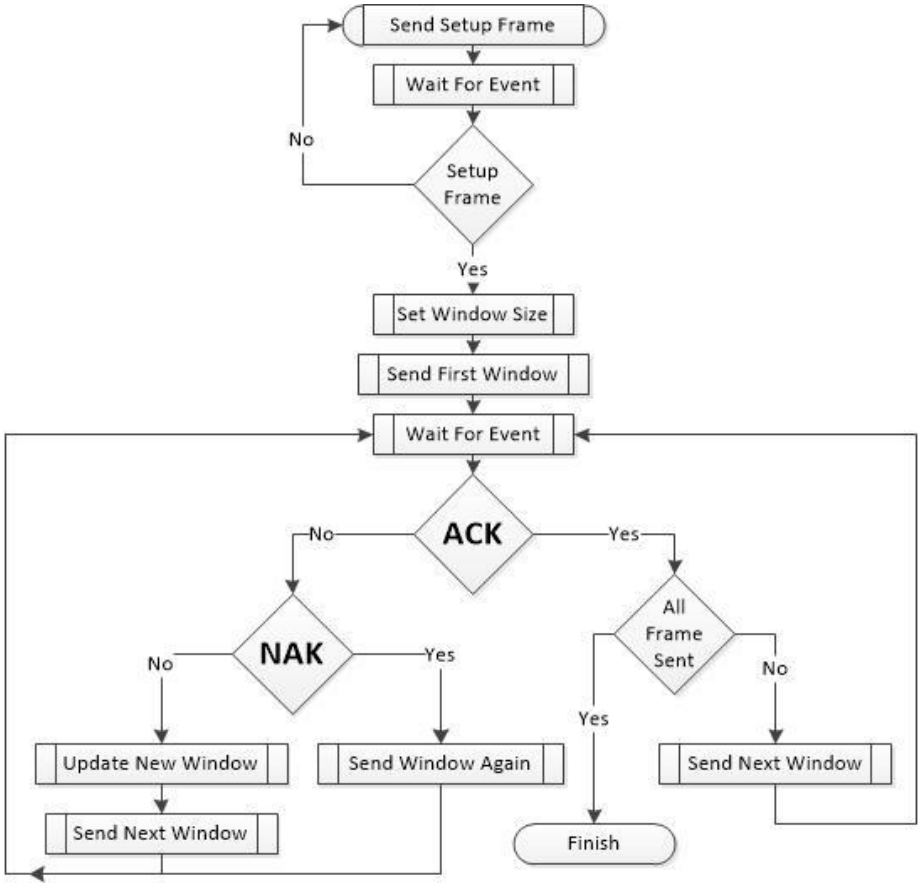


Figure 9.Sender data flow diagram

The following are description of the steps that shown on the diagram

A. The sender will send setup packet to the server to check whether the server or the receiver can handle new transmission and the sender will wait for event or for replay
B. If the sender didn't receive packet or if the server say it is busy it will delay for some time and send the setup packet again

C. When packet is received the sender will set the window size based on the receiver request and it will send the first window packets to the server
D. After sending the first window the sender will wait for acknowledgment for sent packet
E. If it is positive acknowledgment then it will check if all packets are transmitted if all packets are transmitted it will finish the transmission but if there are left packet it will send the next packets with the size of the window size to the sender
F. If it is negative acknowledgment it will send the first window packets again
G. If it is not positive and negative acknowledgment it means that only some packets are incorrect from the window size packets, so the sender create new window that contain the previously incorrect packet and new packets and send the window to the receiver

Fig 10 shows the optimized Selective Repeat receiver data flow diagram for the prototype. The difference between the Selective Repeat receiver and Optimized Selective Repeat receiver is on the Selective Repeat protocol the receiver send acknowledgment for each packet but in the Optimized Selective Repeat receiver it sends one acknowledgment for the window.

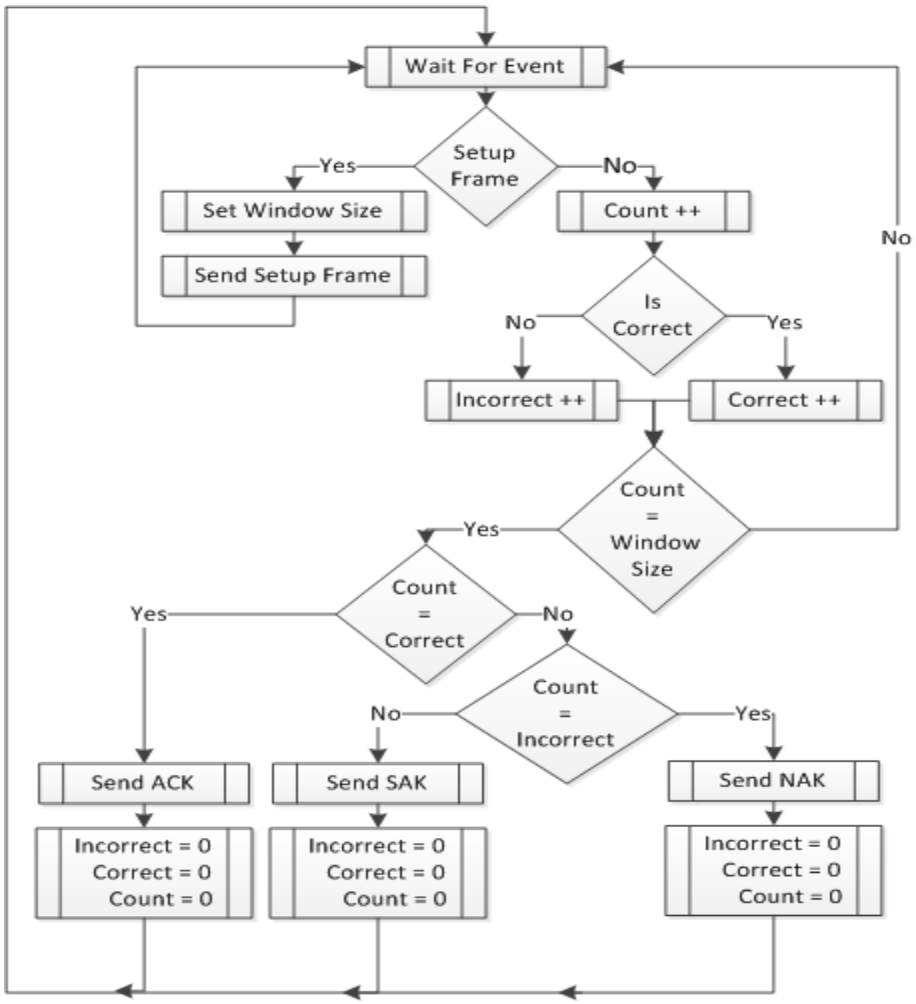


Figure 10.Receiver data flow diagram

The following are description of the steps that shown on the diagram

A. The receiver will wait for packet
B. When packet is received it will check whether it is setup packet or not
C. If it is setup packet it will set window size and send the setup packet back to the server
D. But if it is data packet it will increment the count of packet and check whether the packet is correct or not
E. If the count is equal with the window size it will check if all packets are received correctly
F. If all packets in the windows size are correct it will send positive acknowledgment
G. If all packets in the windows size are incorrect it will send negative acknowledgment

H. If only some packets in the windows size are incorrect it will send some acknowledgment
I. After sending positive, negative or some acknowledgment it will set the incorrect, correct and count to zero

- **System flowchart diagram**

The system flowchart diagram refers set of producer that shows how the designed prototype will work. This includes both the server and clients, when the client can be sender or the receiver. The following key aspects have a direct or indirect influence on the prototype system design. We are considering the following on the server side

- How to generate and open port for clients
- How to handle new connection request to join from new clients
- How to handle if old client close the connection
- How to decide to allow or deny when a sender request to send data to the receiver

And also the following are considerations on the client's side

- How to generate and use port
- How to generate another port if the port is already on use
- How to send data to another client
- How to receiver data from another client
- How to set parameters that will be used in the time of receiving data
- How to set error

As we try to list the main consideration that have influence on how the designed prototype should work, the following diagrams shows the final system flowchart design for the optimized protocol implementation.

Figure 11 shows the system flow diagram for the server. The diagram shows how the system behave when new client ask to join, when old client leave the connection, when the sender request to send message to the receiver and when the message transfer finished successfully.

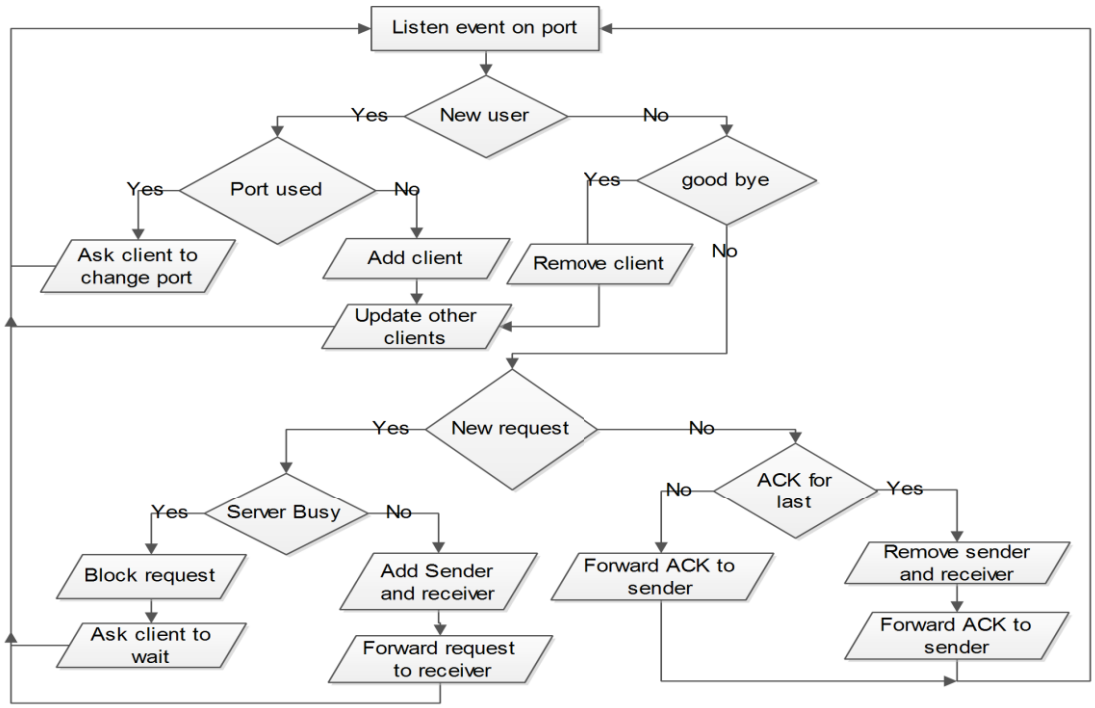


Figure 11.System flow diagram for server

As Fig 11 shows the system flow diagram for the server on the next section we will try to describe the diagram by using steps

A. The server will listen an event on the port that it creates
B. When an event occur it will check if the received packet is sent by a new user if it is new user it will check if the port that the client running is already in use if it's used it ask the client to change the port but if it free the server will add the client and send information of the new client to the old clients
C. If it is a good bye packet it will remove the client and send information for old clients
D. It is a new request to send message to other clients it will check whether the server is busy or not. If it is busy it will reject the request and sent it back to the sender but if it is free it will forward the request to the receiver
E. Or if it is acknowledgment for the last packet it will finish the transmission
F. But if it is other packet it will forward the packet to the destination

Figure 12 shows the system flow diagram for the clients. The diagram shows starting from asking the server information up to message transfer completed. This includes asking server

information, generating port, asking the server to allow the client to join and other flows of the client when it receive packet from the server.

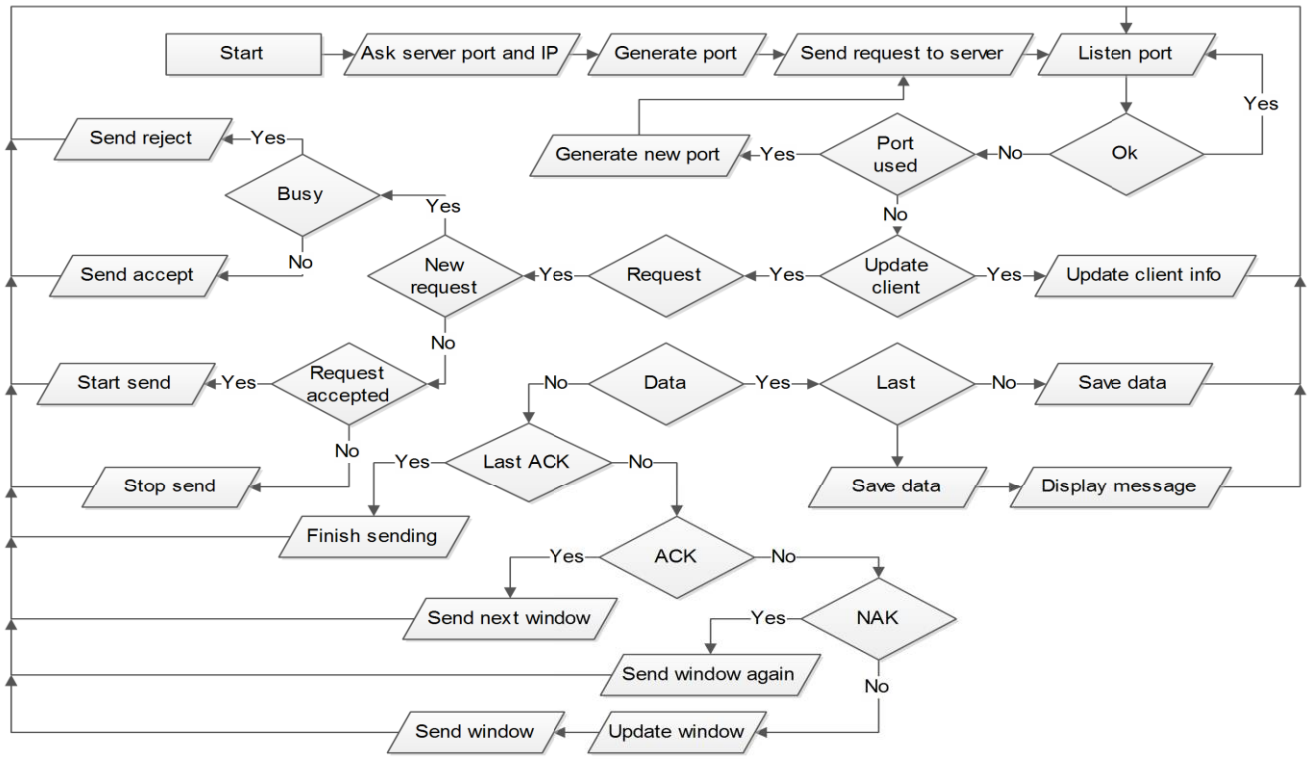


Figure 12.System flow diagram for client

As Fig 12 shows the system flow diagram for the client on the next section we will try to describe the diagram by using steps

A. The client start the application and ask the server port and IP from the user
B. Then the client generate a port, then send request to the server to join the network
C. Listen event on the port
D. When event occurred on the port it will check whether the server sends ok to the request or not. If the server send ok it will listen to the port again but if it is port used packet it will generate new port and send request again go to step C
E. If it is update client packet it will update the information's of the connected users and go to step C

F. If it is a request packet and if it new requests it will check whether it is busy or not. If it is busy it sends reject packet and go to step C or if it is request accepted it start sending the message packet but if it is request reject is stop the transmission and go to step C

G. If it is message packet and it is last message packet it saves the message, display the message to the user and go to step C but if it not the last data message it will save the data and got to step C

H. If it is acknowledgment for sent packet, it will check if it is acknowledgment for the last message packet, if it is it will finish the transmission but if it is not it will check whether it is positive acknowledgment or not

I. If it is positive acknowledgment it will send the next window to the receiver and go to step C or if it is negative acknowledgment it will send the window again and go to step C but if it is some acknowledgment it send the damaged packets with new packet by one window and go to step C

3.5.4 Data Design

This section will discuss the data design that we will use on the prototype development.

3.5.4.1 Data Dictionary

In order to allow communication between the server and clients as well as client to clients (in the case of sender and receiver) we categorize the packets into two, which are during the connection and during the transmission of data.

- **During the connection: -** this category includes when the client asks to join, when the server accepts or reject the request, when new client is joining and when client leave.
  - ✓ **ASK_TO_JOIN: -** This packet is sent from new client to the server in order to ask the server to allow them to join the connection.
  - ✓ **OK: -** This packet send from the server to the new client that send ask to join packet to the server to indicate that the server accept them in the connection.
  - ✓ **BLOCK: -** This packet send from the server to the new client that send ask to join packet to the server to indicate that the server reject the request. When the clients get this packet automatically it will wait for certain seconds and send ask to join packet again.

- ✓ **UPDATE_CLIENT: -** This packet will be sent from the server to all active clients when new client connected or old client disconnect from the server.

- **During the transmission of data: -** This category includes when sender request to send, when receiver accepts or reject the request, when the sender send packet, when receiver receive packet, when the receiver send acknowledgments and when the transmission finished.
  - ✓ **SETUP_REQUEST: -** This packet sent from the sender to the server when the sender wants to send data to the receiver and when the server receive this packet it will forward the packet to the destination (receiver).
  - ✓ **SETUP_ACCEPTED: -** This packet sent from the receiver to the server to indicate that the receiver accepts the request and then the server forward the packet to the sender.
  - ✓ **SETUP_REJECTED: -**This packet sent from the receiver to the server to indicate that the receiver rejects the request and then the server forward the packet to the sender. When the sender gets this packet automatically it will wait for certain seconds and send request to the receiver again.
  - ✓ **DATA_SEND: -** This packet is sent from the sender to the server and then the server forwards it to the receiver. This packet means that this packet is a data packet and there will be anther packet will come.
  - ✓ **DATA_FINISH: -** This packet is sent from the sender to the server and then the server forwards it to the receiver. This packet means that this packet is a data packet and it is the last data packet.
  - ✓ **DATA_ACK: -** This packet sent from the receiver to the server to indicate that the receiver receives all packets from the sliding window correctly and then the server forward the packet to the sender. When the sender gets this packet it will start sending the new packets.
  - ✓ **DATA_NAK: -** This packet sent from the receiver to the server to indicate that the receiver receives all packets from the sliding window incorrectly and then the server forward the packet to the sender. When the sender gets this packet it will re-send the packets again.

- ✓ **DATA_SAK: -** This packet sent from the receiver to the server to indicate that the receiver receives some packets from the sliding window incorrectly and then the server forward the packet to the sender. When the sender gets this packet it will re-send incorrectly received packets with new packets.
- ✓ **DATA_ACK_FINISH:** This packet sent from the receiver to the server to indicate that the receiver receives all packets from the last sliding window correctly and then the server forward the packet to the sender. When the sender gets this packet it will stop the communication.

3.5.5 Programming Language Used and Justification

The prototype implementation for this research will be developed using JAVA network programming. [7] Network programming is probably one of the features that is most used in the current world. As soon as people want to send or receive data over a network in a program, you need to use sockets. Java is a language born after the advent of the Internet and hence has very good integration of sockets in the standard API.

The reason that we propose Java network programing for the prototype development is that it supports both acknowledged and non-acknowledged communication types. Since these researches focused on minimize the positive and negative acknowledgement we will consider acknowledged service with connectionless type. We selected connectionless acknowledged type of connection because this connection type is not more unreliable connection type.

Java programing language is high level programming languages that support both connection-oriented connection (TCP) and connectionless connection (UDP). Since the connection type that is considered for this research is based on connectionless connection type we will use Datagram Packet (UPD) as the main connection between the clients and the server.

[8] Datagram Packet (UDP) doesn't guarantee that the packets will be delivered and doesn't guarantee that they will arrive in the order they were sent. It's called an "unreliable protocol".

By implementing the concept of socket programming using UDP we will make the server and the clients to send and receive datagram packet on specific port. The server and the clients will use different port number to avoid collusion and to identify the clients specifically.

3.6 Test Procedure

The procedure that we will use to test our results depends on several factors. The major factors we considered are Error rate, window size and packet size. We compare the result produced by selective repeat protocol with other sliding window protocols like Stop-and-Wait with the help of algorithm. To get the effective and accurate comparisons we repeat the same test procedure for five times with various factors like different packets and different window sizes, each sample size about three hundred and we got total of one thousand result sets. By depending on the results we will try to show that how the current acknowledgment process consumes time and produce load on the both sender and receiver.

CHAPTER FOUR

4. Presentation, Analysis, and Interpretation of Data

4.1 Presentation

This section will try to discuss how the implementation of the prototype is presented for testing the new algorithm. This includes what are the scenarios that the prototype will be used to compare and contrast each protocols.

The prototype is developed by considering many real time scenarios as part of the implementation. To simulate the error packet and packet lost scenario the simulation have error rate function. This function have user define and auto mode option of error making scenario. In case of auto mode the prototype will generate the error rate type but in case of user define mode the user will set the error rate type. The error range has the following four NORMAL, LOW, MEDIUM, and HIGH error rate type. These options will be used to simulate error free and highly error transmissions. The error rate of the options will be as the following

Table 1.Error option

No	Option	Error probability (out of 10)	Percentages (100%)
1	NORMAL	0/10	0%
2	LOW	2/10	20%
3	MEDUIM	3/10	30%
4	HIGH	5/10	50%

The NOMAL option will indicate the scenario of error free transmission and the HIGH option will show highly error occurrence transmission scenario.

The other function that the prototype will provide is adjusting window size for a transmission at the receiver side. This functionality gives an option to change the size of the receiver window size capacity so that the sender will use the selected window size in the transmission period. By using this option we can analysis results of the protocols when the window size changes. The main reason that this functionality added to the prototype is, since the new protocol use blocked acknowledgment it will give us the advantage to evaluate each protocol when the window size is varies from high to low. The following are options to select the window size for a transmission

Table 2.Window size option

No	Option	Window Size
1	3	3
2	4	4
3	5	5
4	6	6
5	7	7

As we described the scenarios that the prototype will considered, the next part will try to describe what are the analysis part will be conducted.

By using the option that is provided by the prototype, the following three factors are considered in the analysis of the protocols performances.

- **Error rate: -** Based on this factor we will try to evaluate the performance of each protocol when the transmission media is error free, highly error and medium. By using the output of this analysis we will be able to indicate how the new ideal protocol will be better on error free transmission as well as on highly error able transmission.
- **Window size: -** This factor has five different options by itself which will be from the minimum number of window size of three up to the maximum number of window size of seven. This can be used to analyze the performance of each protocol when the window size ranges from three up to seven. Based on the output of this analysis we will try to indicate how the new ideal protocol is more preferable when the window size increase.
- **Number of packet: -** This factor considers the number of packet to be transmitted from the sender to the receiver. In this factor we will consider when the packet count is equal to the window size, when the packet count is less than the window size and when the packet count is greater than the window size. We will consider greater than the window size factor as three different size of packet. This factor will be used to measure the performance of each protocol how they will perform when the packet count is differs.

Based on previously described three factors we will use six main key aspects to compare and contrast the performance each protocol when the factors various one to another. The followings are the six key aspects that we will be considered on the compare and contracts process of each protocol

- **Total packets**: - these will indicate how many packets the protocol used to finish the transmission of all packets. Based on the result of this factor since we use the same number of packets on all protocol we can evaluate how each protocol performs.
- **Time taken: -** these will show the time that the protocol takes to finish send packets. Based on the result of this factor we can measure which protocol is more time effective.
- **Number of acknowledgment: -** this factor will focus on counting both positive and negative acknowledgements have been used between the sender and the receiver up to the end of the transmissions. Based on the result of this factor we will be able to answer how the new protocol minimizes the positive and negative acknowledgments.
- **Packet dropped: -** this factor will count how many packets has been lost or dropped during the transmission. Since the implementation of error rate scenario in this protocol is based on probability, the packet dropped factors will be considered because since it is based on probability if one protocol dropped more packet that another protocol the protocol that dropped more packet will score less result that the protocol that dropped less packets.
- **Number of positive acknowledgment: -** this factor will count only the positive acknowledgments that the sender and the receiver used during the transmission period. So based on this factor we can know how many positive acknowledgments have been used on the transmission.
- **Number of negative acknowledgment: -** this factor will count only the negative acknowledgments that the sender and the receiver used during the transmission period. So based on this factor we can know how many positive acknowledgments have been used on the transmission.

Based on the analysis result of the above six key aspects we will try to describe the following things.

- ✓ How the selective repeats protocol can be optimize in order to minimize the positive and negative acknowledgment.
- ✓ We will try to find answer for each research questions that are listed on research specific question section.

- ✓ Indicate how the optimization process should be.

4.2 Simulation Scenarios and Step Performed

In this section we will try to discuss the simulation scenarios and the step performed in order to collect the test results.

Step performed

The first step we performed in order to collect the test results is we export the executable jar file for client and server for Stop-and-Wait, Selective Repeat and Optimized Selective Repeat protocol from the Net beans.

Then we create an Excel documents for the three protocols by categorizing based on NORMAL, LOW, MEDUIM and HIGH error rates to document the test results of the protocols. After creating the Excel documents we identified the major factors which the test result document must include.

Lastly by running the server and clients we send different message by using different window sizes, error rates and messages from sender to the receiver for the same three protocols one by one.

Simulation Scenarios

A. Interface of the client and server

The simulations of the Stop-and-Wait, Selective repeat and Optimized Selective Repeat protocols have interface for client and server individually. The client and server simulation interface is the same for the three protocols. The difference is how many packets they send without acknowledgments and how they handle on acknowledgment is received. Based on this factor in the following section we will show the interface of client and server for Optimized Selective Repeat protocol.

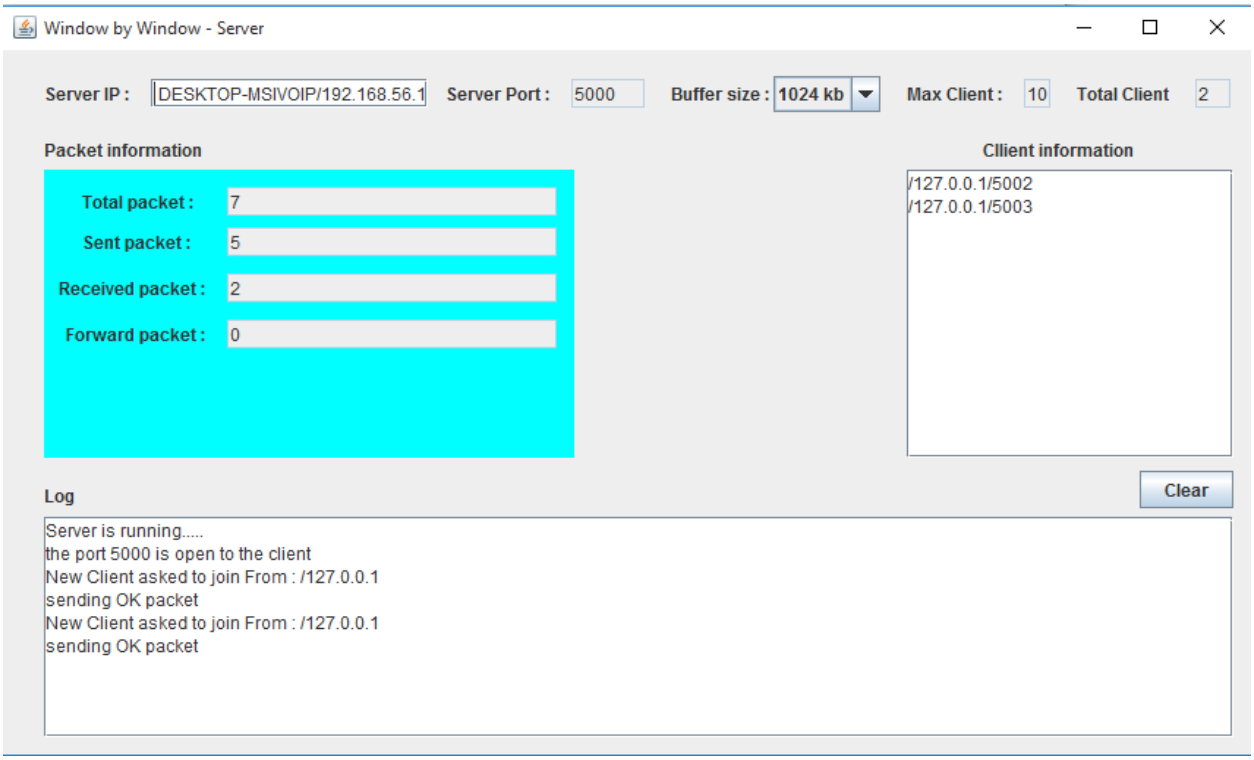


Figure 13. Server Interface

The server simulation application includes the maximum amount of clients allowed, how many clients currently connected, how many packets is received, forwarded , sent and the port number and IP of the server. The server simulation also has log messages that show what the server is doing in the background.

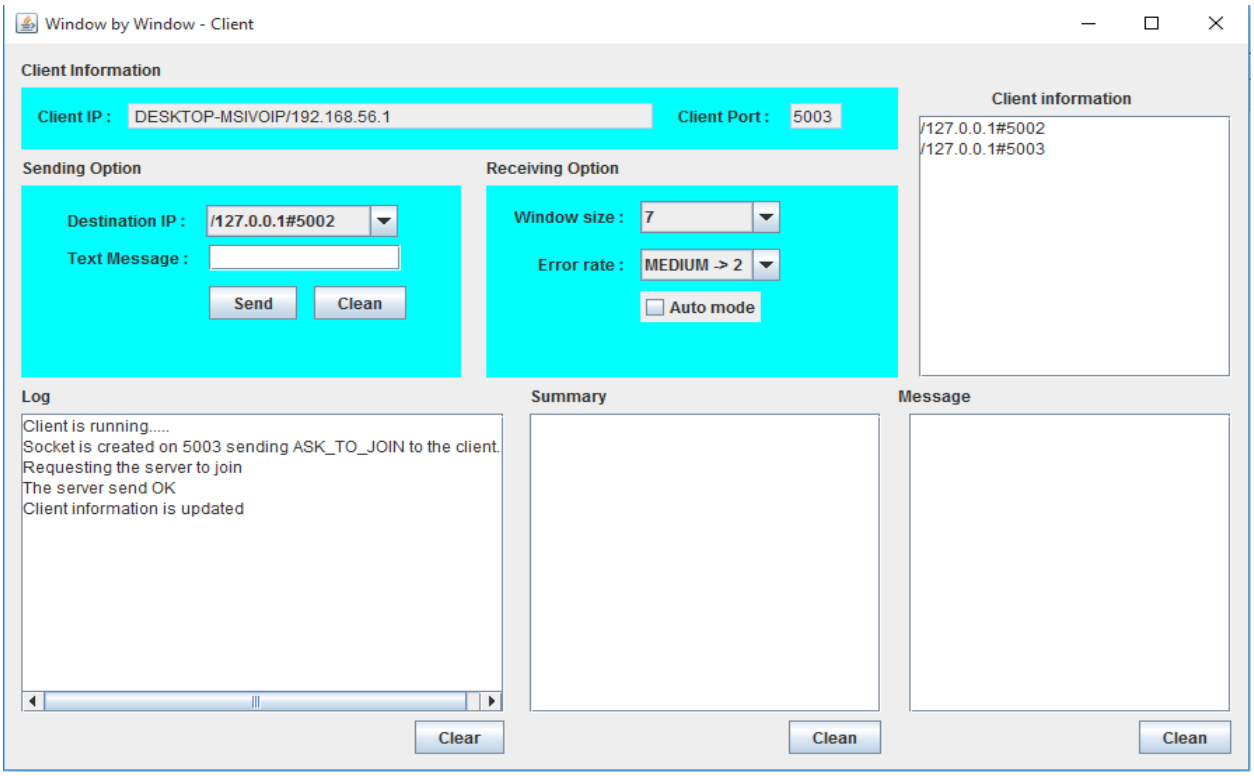


Figure 14. Client Interface

The client simulation application includes the port number and IP address information about the clients and other connected clients. The window size and error rate mode for receiving message from other clients and log messages for the background process, summary of transmission message and message that received or sent for and from clients.

B. Implementation of proposed algorithm on simulation

Since the clients act as both sender and receiver proposed algorithm implemented on the client part of application on the Optimized Selective Repeat protocol simulation. The following table shows the general idea of the algorithm by separating the clients act based on the time of sending and the time of receiving.

Table 3. Implementation of proposed algorithm

Sender	Receiver
❖ Send the first window ❖ Wait for acknowledgment ❖ When acknowledgment is received • Positive acknowledgment (ACK)	❖ Receive packets ❖ Add received packets count by one ❖ Check for error • If it is correct

○ Send the next window • Negative acknowledgment (NAK) ○ Send window again • Some acknowledgment (SAK) ○ Identify the packet id's of incorrectly received packets ○ Create new window ○ Add incorrect packets on window ○ Add other packets from the next packets from next window ○ Send the window	○ Add one on correct packets count ○ Add data on the packet on buffer • If it is incorrect ○ Add one on incorrect packet count ○ Take packet id ❖ When the received packets count is equal to the window size • If all packets are correct send positive acknowledgment (ACK) • If all packets are incorrect send negative acknowledgment (NAK) • If some packets are incorrect collect all incorrect packets id's and Send the id's by using some acknowledgment (SAK)

C. Implementation of the factors in the test results

As we try to explain the simulation option that can be used on the transmission on the previous section, now we will try to describe how many test results will be produce by selecting one factor from each option.

The options are how many message sent from the client to clients, window size used on the transmission and error rate option which used to simulate packets loss and damaged packets. For the three protocol simulation we use the following combination of options.

Table 4. Implementation of factors in the test

Error rate	Window size	Length of message	Total combination	Total number of test conducted
Normal	3,4,5,6 & 7	<WS, =WS,10,20 & 30	25	125
Low	3,4,5,6 & 7	<WS, =WS,10,20 & 30	25	125
Medium	3,4,5,6 & 7	<WS, =WS,10,20 & 30	25	125
High	3,4,5,6 & 7	<WS, =WS,10,20 & 30	25	125

As we show on the previous table totally there will be one hundred test combinations and since we test one combination five times there will be five hundred test results for one protocol simulation test results. Since we will test three protocols there will be one thousand five hundred test results.

D. Simulation steps scenarios

In the steps that we follow in the test result conducting process are the following

Table 5. Simulation step scenarios

Step number	Work
Step 1	Select the protocol that will be tested
Step 2	Select factors options of Window size, total sent message and error rate
Step 3	Run the server
Step 4	Run the clients and give server IP and port
Step 5	Set window size for the clients
Step 6	Send the message from one client to other clients
Step 7	Record test result of the summary for the transmission on excel
Step 8	Go to **Step 6** until we conducted five test result for the factors
Step 9	Close clients and server
Step 10	Go to **Step 1**

By using the steps that described on the table we successfully conducted test results for each factor that can be used to compare and contrast the three protocols.

4.3 Analysis

In this section we will try to document and discus the analysis result of the test which is conducted on the Stop-and-Wait, Selective Repeat and the optimized Selective Repeat protocols simulations.

As we try to discuss in the previous section there will be total of one thousand five hundred test results. In the time of testing we conduct the following test factors for each testing

- Total used time to transmit all packets from sender to receiver
- Total used packet to transmit the packets from the sender to the server and from the server to the sender
- In case of HIGH, LOW and MEDUIM error rates option we conducted the number of dropped packets in the time of transmission

- Total number of positive and negative acknowledgments that is used in the packet transmission

After we successfully conduct the one thousand five hundred, results since we conduct five tests for three hundred different combinations that we get from four error rate types, five window size and five total sent packet. For this reason we try to find the average values for the five tests on the three different combinations. Then we last with one test result for one combination, specifically we left with three hundred results.

Then by using the three hundred test results we try to find average results values based on the total number of packets for each window size and error rate. Since use less that window size, equal to window size, ten packets, twenty packets and thirty packets as total number of packets, we try to find the average values for the five types of numbers of sent packets. After this step we left with 60 test results. In the following tables we try to show the test results of the 60 output by categorizing by using error rate.

A. Test results for NORMAL error rate with five different window sizes for Stop-and-Wait, Selective Repeat and optimized Selective Repeat protocol simulations.

Table 6.Test result of NORMAL error rate

Protocol	WS	Acknowledgements				Dropped Packet	Total Packet	Time
		ACK	NAK	SAK	Total			
Stop And Wait	3	13	0	-	13	0	36	181.08
	4	13.4	0	-	13.4	0	36.8	172.64
	5	13.8	0	-	13.8	0	37.6	174.4
	6	14.2	0	-	14.2	0	38.4	179.64
	7	14.6	0	-	14.6	0	39.2	187.6
Selective Repeated	3	13	0	-	13	0	36	137.34
	4	13.4	0	-	13.4	0	36.8	125.56
	5	13.8	0	-	13.8	0	37.6	112.6
	6	14.2	0	-	14.2	0	38.4	119.6
	7	14.6	0	-	14.6	0	39.2	114.8
Optimized Selective Repeated	3	4.6	0	0	4.6	0	28	91.96
	4	3.6	0	0	3.6	0	27	78.16
	5	2.8	0	0	2.8	0	26.6	74.28
	6	2.6	0	0	2.6	0	26.8	78.8
	7	2.4	0	0	2.4	0	27	74.88

B. Test results for LOW error rate with five different window sizes for Stop-and-Wait, Selective Repeat and optimized Selective Repeat protocol simulations

Table 7.Test result of LOW error rate

Protocol	WS	Acknowledgements				Dropped Packet	Total Packet	Time
		ACK	NAK	SAK	Total			
Stop And Wait	3	13	2.6	-	15.6	2.52	41.2	206.36
	4	13.4	3.12	-	16.52	3.12	43.04	215.92
	5	13.8	2.84	-	16.64	2.84	43.28	197.36
	6	14.2	2.88	-	17.08	2.88	44.8	208.12
	7	14.6	2.96	-	17.56	2.96	45.12	223.6
Selective Repeated	3	13	3.2	-	16.8	3.2	42.32	153.04
	4	13.4	2.92	-	16.32	2.92	42.64	143.72
	5	13.8	2.64	-	16.44	2.64	42.88	139.84
	6	14.2	2.8	-	17	2.8	44.4	142.44
	7	14.6	3.2	-	17.8	3.2	45.6	144.6
Optimized Selective Repeated	3	4.08	0.12	3.08	7.28	4.04	38.24	133.28
	4	3.04	0.08	3.16	6.28	4.64	38.84	127.4
	5	2.56	0	3.36	5.92	5.28	42.08	133.32
	6	2.28	0	3.12	5.4	4.88	44.24	131.56
	7	1.92	0	4.2	6.12	7.16	54.48	159.32

C. Test results for MEDUIM error rate with five different window sizes for Stop-and-Wait, Selective Repeat and optimized Selective Repeat protocol simulations

Table 8.Test result of MEDUIM error rate

Protocol	WS	Acknowledgements				Dropped Packet	Total Packet	Time
		ACK	NAK	SAK	Total			
Stop And Wait	3	13	4.06	-	17.04	4.04	44.32	217.64
	4	13.4	3.96	-	17.36	3.96	44.72	217.08
	5	13.8	4.2	-	18.04	4.2	45.92	209.96
	6	14.2	5.28	-	19.48	5.28	48.4	227.92
	7	14.6	5.48	-	18.88	5.48	50.16	232.64
Selective Repeated	3	13	4.92	-	17.92	4.92	45.84	163.76
	4	13.4	4.24	-	17.64	4.24	44.72	158.28
	5	13.8	4.76	-	18.56	4.76	47.12	163.24
	6	14.2	5.12	-	19.32	5.12	48.64	166.6
	7	14.6	4.84	-	19.44	4.84	49.04	161.4
Optimized	3	4.12	0.16	5.32	9.52	7.64	47.28	168.8
	4	3.12	0.04	4.72	7.8	7.68	47	153.84

Selective Repeated	5	2.52	0.08	6.32	8.88	10.92	60.64	178.52
	6	2.16	0	8.48	10.12	15.52	78.08	218.08
	7	1.96	0	9.24	11.2	19.8	93.88	228.68

D. Test results for HIGH error rate with five different window sizes for Stop-and-Wait, Selective Repeat and optimized Selective Repeat protocol simulations

Table 9.Test result of HIGH error rate

Protocol	WS	Acknowledgements				Dropped Packet	Total Packet	Time
		ACK	NAK	SAK	Total			
Stop And Wait	3	13	11.12	-	24.12	11.12	58.24	271.2
	4	13.4	9.04	-	22.44	9.04	56.48	277.24
	5	13.8	10.6	-	24.4	10.6	54.72	263.28
	6	14.2	11.16	-	25.36	11.16	60.96	281.32
	7	14.6	9.52	-	23.72	9.52	58.24	267.08
Selective Repeated	3	13	11.28	-	24.28	11.28	58.56	214.36
	4	13.4	10.72	-	24.12	10.72	58.24	206.16
	5	13.8	10.24	-	24.04	10.24	58.08	189.96
	6	14.2	11.16	-	25.36	11.6	60.72	211.2
	7	14.6	13.6	-	28.76	13.76	64.96	228.04
Optimized Selective Repeated	3	3.76	1.76	12.36	17.92	11.04	80.64	260.52
	4	2.64	1	18.04	21.68	38.08	113.88	297.56
	5	2.28	0.6	25.4	28.28	62.76	173.12	240.72
	6	2.08	0.4	41.8	44.32	118.84	313.12	496.8
	7	1.88	0.2	54.16	56.28	168.12	446.28	617.68

4.4 Interpretation of Data

In this section we will try to represent the analysis of data by using graphical representation as well as the description of the data values that for the optimized Selective Repeat, Selective Repeat protocol and Stop-and-Wait protocol based on the representations.

Firstly let us represent the original analysis of data by using four graphical representations. Each representation the analysis data of the three protocols with one error rate option. This charts contain the total acknowledgment used, total time taken and total used packets to finish the transmissions.

The following four graphical representations shows the analysis data results for NORMAL, LOW, MEDUIM and HIGH error rate mode. In case of NORMAL error rate since we assumed

that it is error free transmissions there is no need to show how many packets are dropped during the transmissions so the graphical representation shows only how many acknowledgment packet are sent, how many total packets are used to finish the message transmissions and how many time they used to finish the message transmission when we send the same amount of messages in different window size on the three protocol simulations. But in case of LOW, MEDUIM and HIGH error rate modes the graphical representation will also show how many packets are dropped during the transmissions.

- When the error rate mode is NORMAL

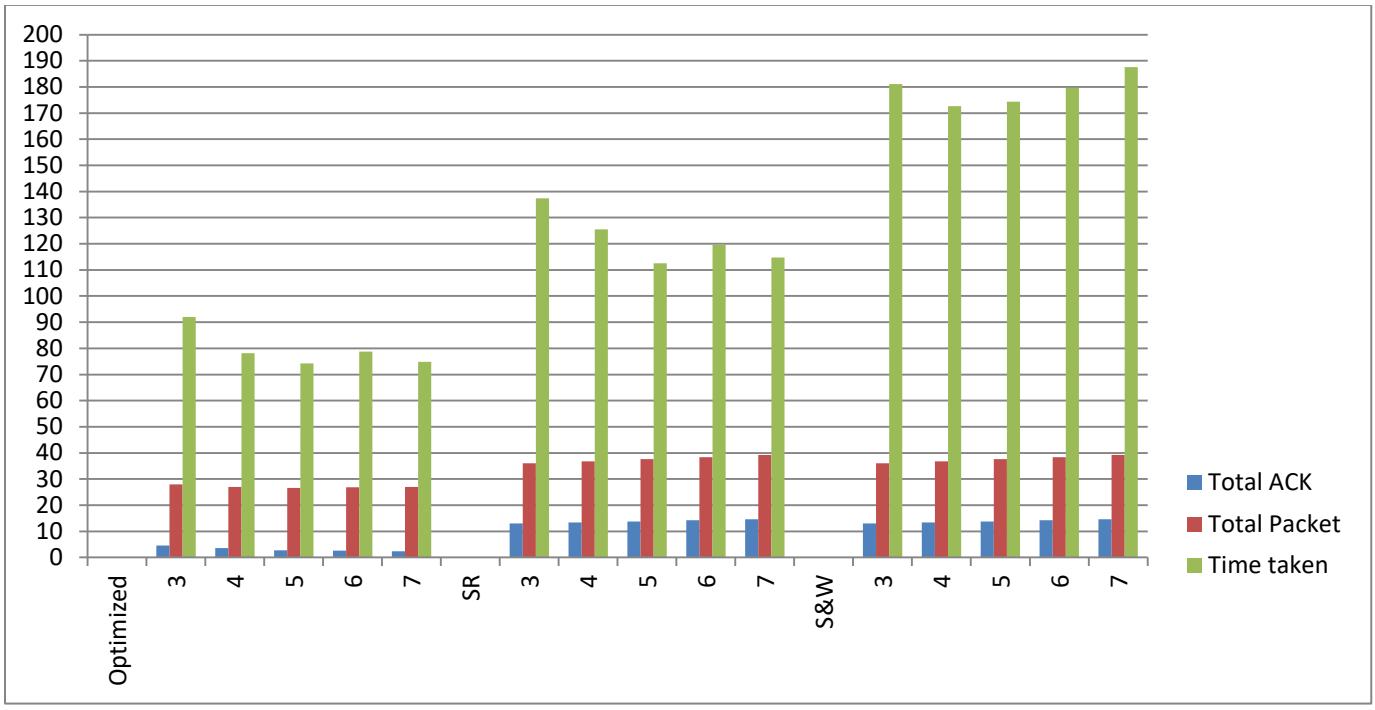


Figure 15.Error rate NORMAL

This representation show that the total amount of used acknowledgment between the sender and receiver in the transmissions, total amount of time taken and how many total packets that the server received or forwarded in the transmissions. As the representation show the optimized Selective Repeat protocol used fewer resources than Selective Repeat protocol and the Selective Repeat protocol used fewer resources than Stop-and-Wait protocol.

➢ When the error rate mode is LOW

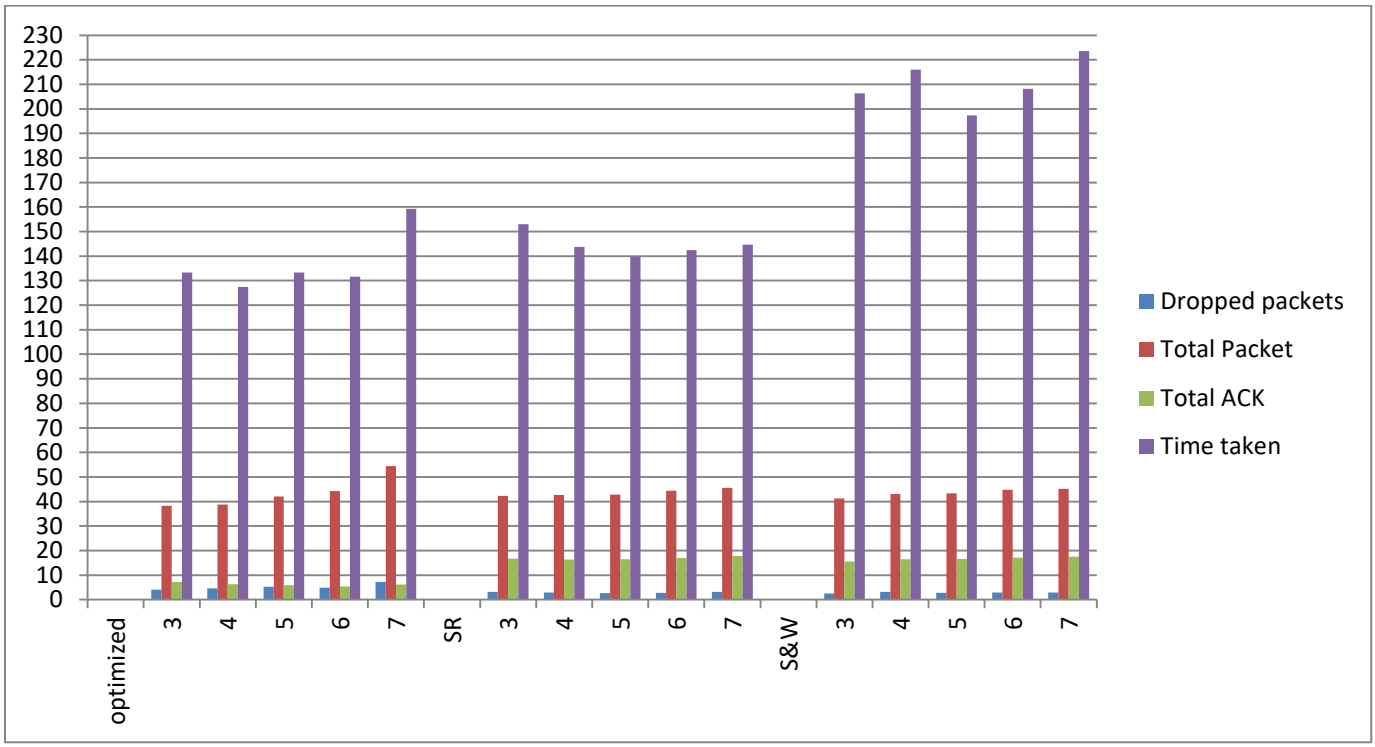


Figure 16.Error rate HIGH

This representation show that the total amount of used acknowledgment between the sender and receiver in the transmissions, total amount of time taken, how many total packets that the server received or forwarded in the transmissions and how many packets are dropped during the transmissions when the transmission is on LOW error rate.

As the representation shows on Stop-and-Wait protocol there are less dropped packets but the protocol used more time that the others but the optimized Selective Repeat protocol dropped more packets but still the protocol used less acknowledgment than the others and less time on the four sliding window sizes out of the five window sizes. On the Selective Repeat protocol it used the similar acknowledgment and total packets with the Stop-and-Wait protocol but it used less time than the Stop-and-Wait protocol and more time than the optimized Selective Repeat protocol.

➢ When the error rate mode is MEDUIM

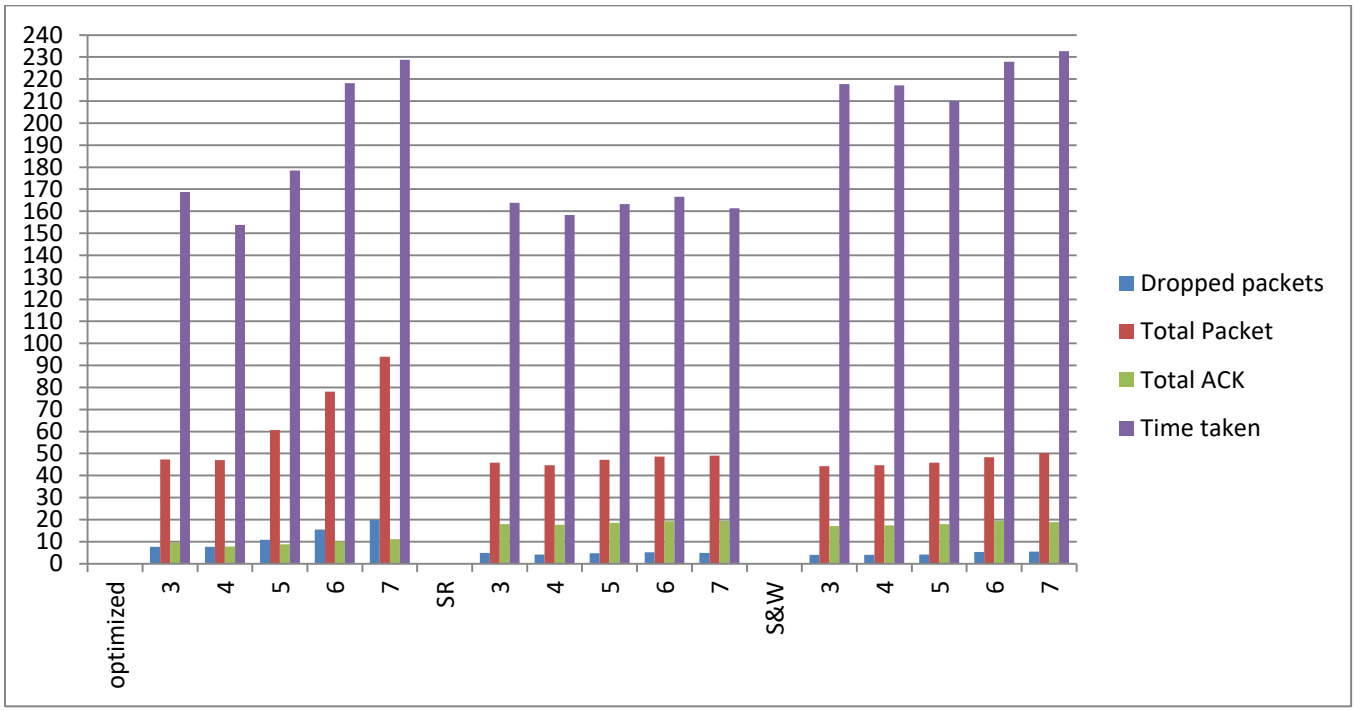


Figure 17.Error rate MEDUIM

This representation show that the total amount of used acknowledgment between the sender and receiver in the transmissions, total amount of time taken, how many total packets that the server received or forwarded in the transmissions and how many packets are dropped during the transmissions when the transmission is on MEDUIM error rate.

As the representation shows Selective Repeat and Stop-and-Wait protocol used almost the same acknowledgment and total packets but Selective Repeat protocol used less time to finish the transmissions than Stop-and-Wait protocol.

On the optimized Selective Repeat protocol the transmission used more total packets than the other protocol because of the optimized protocol dropped more packets when we compare it with the other protocols but still these protocols used less acknowledgment that Selective Repeat protocol and Stop-and-Wait protocol.

When we compare the optimized Selective Repeat protocol with the other protocols based on time used the optimized Selective Repeat protocol used less time than Stop-and-Wait protocol but it used more time than Selective Repeat protocol.

➢ When the error rate mode is HIGH

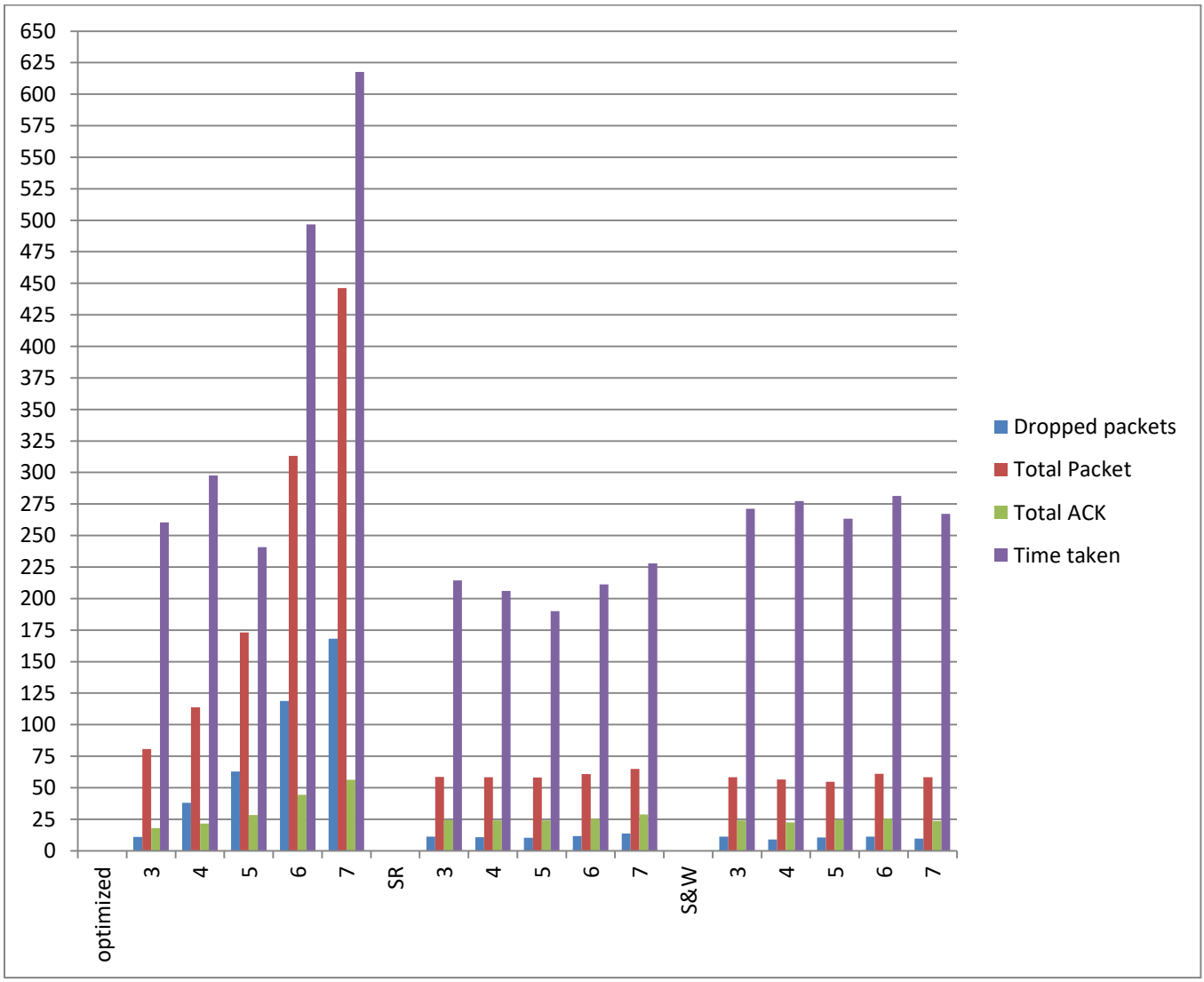


Figure 18.Error rate HIGH

This representation show that the total amount of used acknowledgment between the sender and receiver in the transmissions, total amount of time taken, how many total packets that the server received or forwarded in the transmissions and how many packets are dropped during the transmissions when the transmission is on HIGH error rate.

As the representation shows Selective Repeat and Stop-and-Wait protocol used almost the same acknowledgment and total packets but Selective Repeat protocol used less time to finish the transmissions than Stop-and-Wait protocol.
When we compare the optimized Selective Repeat protocol with the other protocols it used more total packet than the other protocols on all window size, more time, more acknowledgment and it dropped more packets than the other protocols on four window sizes. This is because the optimized Selective Repeat dropped more packets than the other two protocols.

When we see the values on the window size three the optimized Selective Repeat protocol dropped the same amount of packets, in fact the protocol used more time and more packets than the other protocols. But still the optimized Selective Repeat protocol uses less acknowledgment than the Selective Repeat protocol as well as Stop-and-Wait protocol.

4.5 Comparisons of Data

In this section we will try to compare and contrast the optimized Selective Repeat protocol with Selective Repeat and Stop-and-Wait protocol based on different points of view. Since the protocols responds differently when the error rate is increased as we had seen in the previous section. We will focus on error free transmissions or NORMAL error rate mode as the basement of the comparison to show how the Selective Repeat protocol is better than Stop-and-Wait protocol and how the optimized Selective Repeat protocol will be more better that the current Selective Repeat protocol when the current individual acknowledgment process is optimized to per window acknowledgment.

The comparison will have three factors, the three factors will be based on acknowledgment they used, based on total packets they used and based on the time they used for the same amount of packets, window size and error rate.

A. Based on how many total acknowledgment used

On this section we will try to compare and contrast the two protocol simulations by using how many packets are sent from the sender to the receiver, how many packets the transamination used to finish all sent packet and how many total acknowledgments the transmission used during the transmissions. The following figure is shows the graphical representations of three selected

comparison factors for the new optimized Selective Repeat protocol and the current Selective Repeat protocol. The reason we will not use the Stop-and-Wait data is Stop-and-Wait protocol has the same idea with Selective Repeat protocol on acknowledgment.

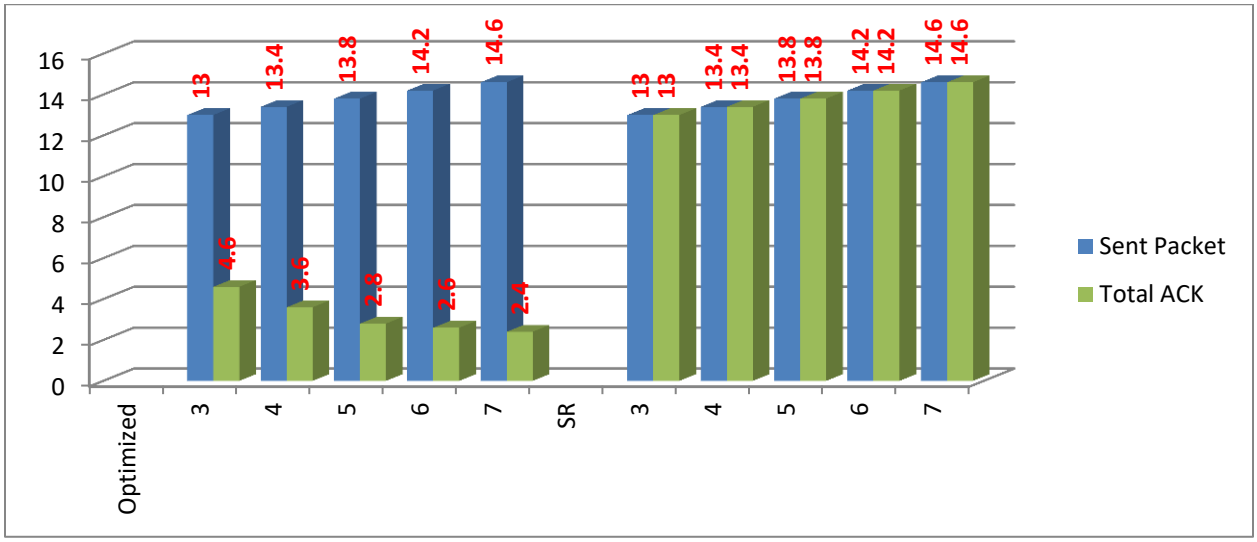


Figure 19.Based on acknowledgment used

Based on the data representation shown on the above Figure the following aspects can be arise

- In case on Selective Repeat protocol it shows that it used exactly the same amount of acknowledgment for every sent packet for the receiver.
- In case of the optimized Selective Repeat protocol is shows that the prototype simulation used less acknowledgment that the sent packet to the receivers. In fact it shows that it used more less acknowledgment when the window size increased.

B. Based on total packets used

On this section we will try to compare and contrast the two protocol simulations by using how many packets are sent from the sender to the receiver and how many packets the transamination used to finish all sent packet used during the transmissions. The sent packet values shows only the main message that sent from the sender to the receiver and used packet values show including the packets that the clients used to ask to join to the server and requesting the receiver to send with the packet used to finish the sent packet. The used packet value is taken from the

server, which means that the server records packets since the clients are requesting to join and how many packets are receiver or forwarded to the clients on the data transmissions.

The following next figure is shows the graphical representations of two selected comparison factors for the new optimized Selective Repeat protocol and the current Selective Repeat protocol.

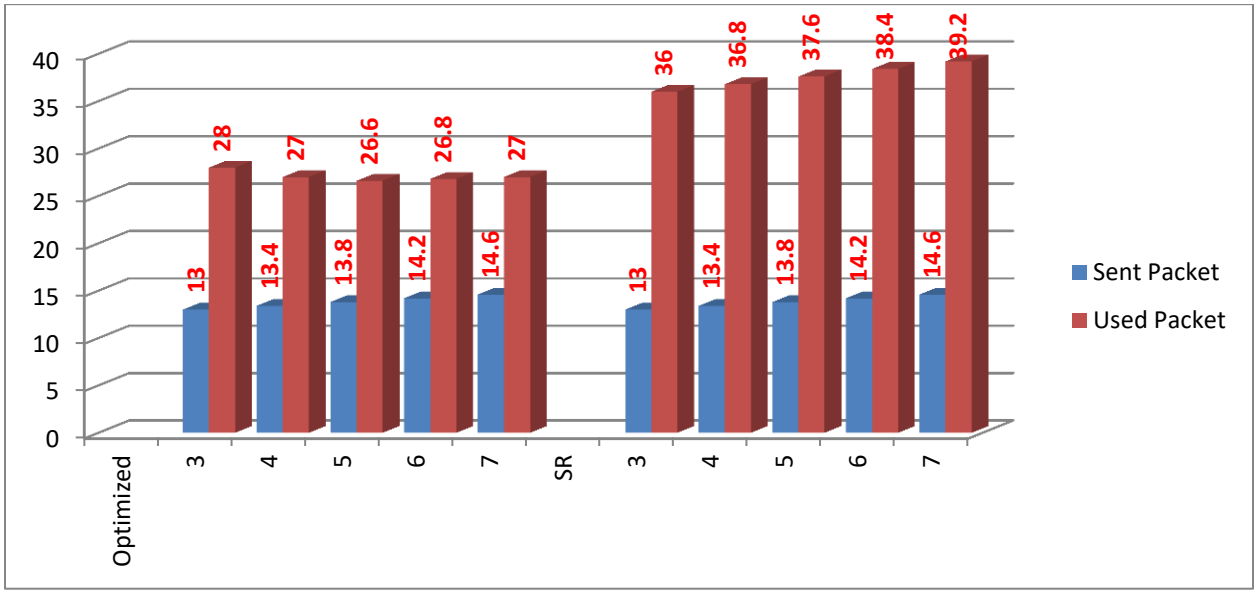


Figure 20.Based on total packet used

As the figure shows when we send the same amount of message on both protocols the optimized protocol used fewer amounts of packets. The reason why the optimized protocol used less packets that the Selective Repeat protocol is the optimized protocol used window acknowledgment rather than individual acknowledgment.

C. Based on how many time taken used

On this section we will try to compare and contrast the three protocol simulations by using how much take the simulation protocols used during the transmissions. The following Fig is shows the graphical representations of the selected comparison factor for the new optimized Selective Repeat protocol, the current Selective Repeat protocol and Stop-and-Wait protocol.

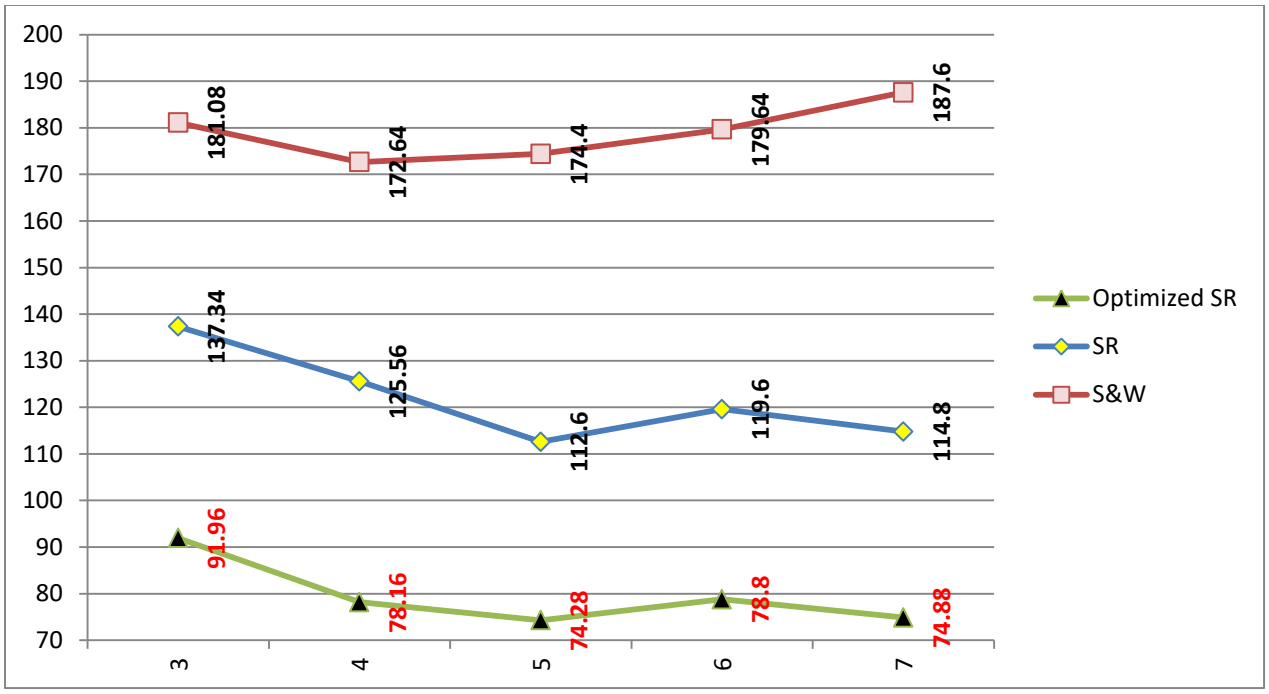


Figure 21.Based on time they use

Based on the analysis data value representation the optimized Selective Repeat protocol used less time that the Selective Repeat protocol. And the Selective Repeat protocol used less time that Stop-and-Wait protocol. This means that

- ❖ When we see that Selective Repeat protocol takes less time to finish than Stop-and-Wait protocol
  - The prototype reflects that it is correctly developed
  - The reason that the Stop-and-Wait protocol was changed to Selective Repeat protocol is more appropriate.
- ❖ When we see that optimized Selective Repeat protocol takes less time to finish than Selective Repeat protocol
  - As we seen that rather than sending one packet and waiting for the acknowledgment it will be more best if we send packets with the window size, rather than sending the packet after the window size when we receive the acknowledgment it will be more best if we sent another window when we received acknowledgment for the previous window
  - Since the most valuable resource is time if it is changed the individual acknowledgment to window acknowledgment we can save this resource.

CHAPTER FIVE

5. Summary, Conclusions, and Recommendations

5.1 Summary of Findings

In this research, how the current Selective Repeat protocol individual acknowledgment process can be optimized to minimize the positive and negative acknowledgments has been studied. During the study by creating an mechanism to minimize the acknowledgment as the first step the research try's to show how the optimized Selective Repeat protocol gives advantage to provide better protocol which will use less acknowledgment, less time, less bandwidth and less load on the communication channel.

After the optimization process is partially created the study continued by designing simulation for the Stop-and-Wait, Selective Repeat and the new optimized Selective Repeat protocols. This design includes the client and receiver and server for each protocol. The client simulation designed with the ability to act as the sender or the receiver by using different port. The server is designed to create a specific port and accepts the clients to join the network before any packet transmission and the server will transmit the packets from one client to anther clients.

After the designing process is finished the study continues by developing the designed simulations by using java networking concept. During the development process the study firstly focused on Stop-and-Wait protocol simulation development because the Selective Repeat protocol is an encasement of Stop-and-Wait protocol and the new optimized Selective Repeat protocol is an enhancement of the Selective Repeat Protocol. So by using this step the study firstly successfully develop the Stop-and-Wait protocol simulation and after that by using the Stop-and-Wait protocol simulation by implementing the sliding window concept on the Stop-and-Wait protocol simulation the study develop the protocol simulation for Selective Repeat protocol. Finally by using the Selective Repeat protocol simulation by implementing one acknowledgment for one window rather than individual acknowledgment the study develop protocol simulation for the new optimized Selective Repeat protocol.

After the development process is completed the study tries to conduct the analysis data by using the three protocol simulations. The analysis of the protocol is conducted by focusing how the

three protocols respond on different window sizes and error rate. The analysis data contain how many packets are used, how may time used, how many packets are dropped on error rate is high and how may acknowledgment is used to successfully transmit the packets in different window size and error rate on the three protocol simulations. The analysis data totally contain one thousand five hundred test results, which are the minimized to twenty per protocol by taking the average of twenty five to one.

After the development and test analysis data is conducted the study focused on comparing and contrasting the test analysis data by selecting error free transmissions data. As the study try's to identify from the comparison the optimized Selective Repeat protocol used less acknowledgment, time and total packets than the Selective Repeat and Stop-and-Wait protocols. This means that the study has successfully optimized the positive and negative acknowledgments of the Selective Repeat protocol by changing the individual acknowledgment to per window acknowledgment.

5.2 Conclusions

In this research we presented an optimized Selective repeat protocol that will minimize the positive and negative acknowledgment of the current Selective Repeat protocol. We found out how the proposed optimized protocol use valuable resources like time and bandwidth with better efficient manner than the current Selective Repeat protocol. We used Java based example prototype simulation for showing how the optimized protocol has better performance than the current Selective Repeat protocol. The prototype developed for the optimized protocol, the current Selective Repeat protocol and Stop-and-Wait protocol. By using the simulations for each protocol we try to transmit the same amount of data and we recorded the acknowledgment, time and total packet used on each simulation. Based on the recorded values we found out how Selective Repeat protocol have better performance than Stop-and-Wait protocol and how the optimized protocol have even better performance than the current Selective Repeat protocol.

5.3 Recommendations

The research highly recommends that by conducting another similar research on the idea of optimizing the individual acknowledgment to per window acknowledgment in order to minimize the positive and negative acknowledgment, we can have a bright feature of providing new

protocol that can transmit data with less acknowledgment, time and bandwidth. Here is a list of some further word to be done:

1. The proposed optimized protocol developed and tested without any consideration of how it performs when the sent packets are timeout. So before we try to implement on the real time it is better to make it combatable on timeout scenario.
2. Based on the research that conducted by considering how timeout affects the optimized, if the performance is still good. The research recommends that to conduct a research to study how the optimized protocol can be implemented on the real time scenario.

References

[1]. B. Epple, H. Hennigera, C. S. Solsonaa, (2008). “An Error Protection Protocol for user-transparent bridging of Fast Ethernet data transmission over the optical fading channel in an Aeronautical environment”, Institute of Communications and Navigation, 82234 Wessling, Germany.

[2]. O. Hasan, S. Tahar, “Performance Analysis of ARQ Protocols using a Theorem Prover”, Concordia University, Canada.

[3]. S. Tanenbaum, (2011). “Computer Networks”, Fifth Edition.

[4]. M. Eltayeb, (2008). “A study on the performance of some ARQ”.

[5]. S. Maurya, V. K. Nayak, Dr. A Nagaraju, (2014). “Implementation of Data Link Control Protocols in Wired Network”, International Journal of Engineering Trends and Technology (IJETT), Volume 18 Number2.

[6]. LI Suoping, LIU Mei, BAI Zhongyu, (2015). “Goodput Analysis of a Modified Cumulative ARQ”, 3rd International Conference on Mechatronics, Robotics and Automation.

[7]. Manuel Oriol,(2007). “Java Programming: Sockets in Java”.

[8]. Prentice-Hall,(2000). “Thinking in Java”, Second Edition.

[9]. KNX Association. ” Serial Data Transmission and KNX Protocol”.

[10]. Kamtorn Ausavapattanakun and Aria Nosratinia (2007). “Analysis of Selective-Repeat ARQ via Matrix Signal-Flow Graphs”. IEEE TRANSACTIONS ON COMMUNICATIONS (IEEE),Volume 55 Number 1.

[11]. B. Kaur, M. Kaur, P. Mudgil, H. Singh (2013). “IMPORTANCE OF SLIDING WINDOW PROTOCOL”, International Journal of Research in Engineering and Technology(IJRET), Volume 02 Issue 10.

[12]. Osman Hasan and Sofi`ene Tahar (2007). “Performance Analysis of ARQ Protocols using a Theorem Prover”, Concordia University.

[13]. Drago Hercog (2010). “The Importance of Sliding Window Protocol Generalisation in a Communication Protocols Course”, International Conference on Engineering Education(ICEE).

[14]. Oskar Eriksson (2011).” Error Control in Wireless Sensor Networks”, UPPSALA Universit.

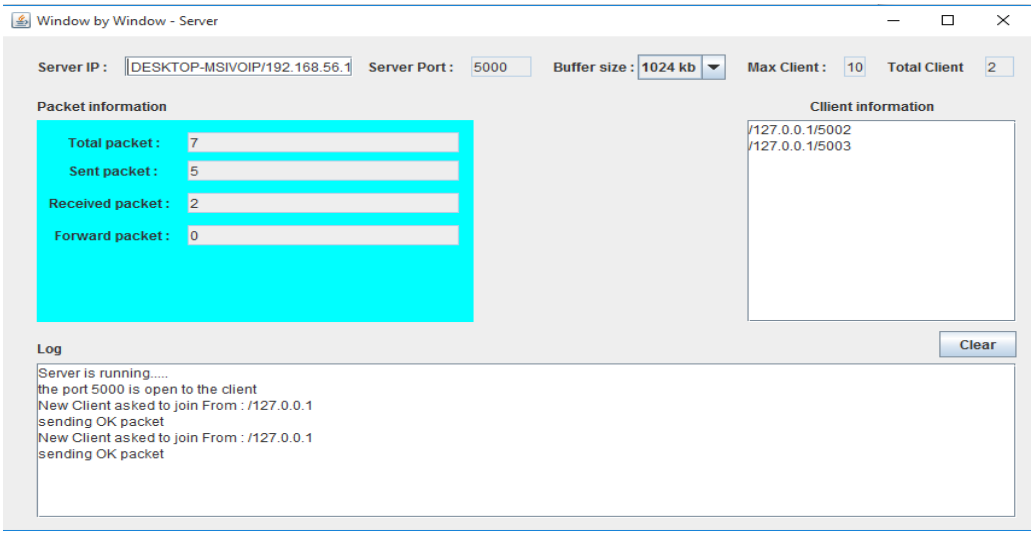
[15]. Hu Hai(2013). “Automatic Repeat-Request Courseware”, UNIVERSITY OF APPLIED SCIENCES.

[16]. Monika Singh & Ruhi Saxena(2014). “Data Link Layer Designing Issues: Error Control—A Roadmap”, Global Journals Inc. (USA), Volume 14 Issue 8 Version 1.0.

Appendix

A. Interface of the prototype for optimized protocol

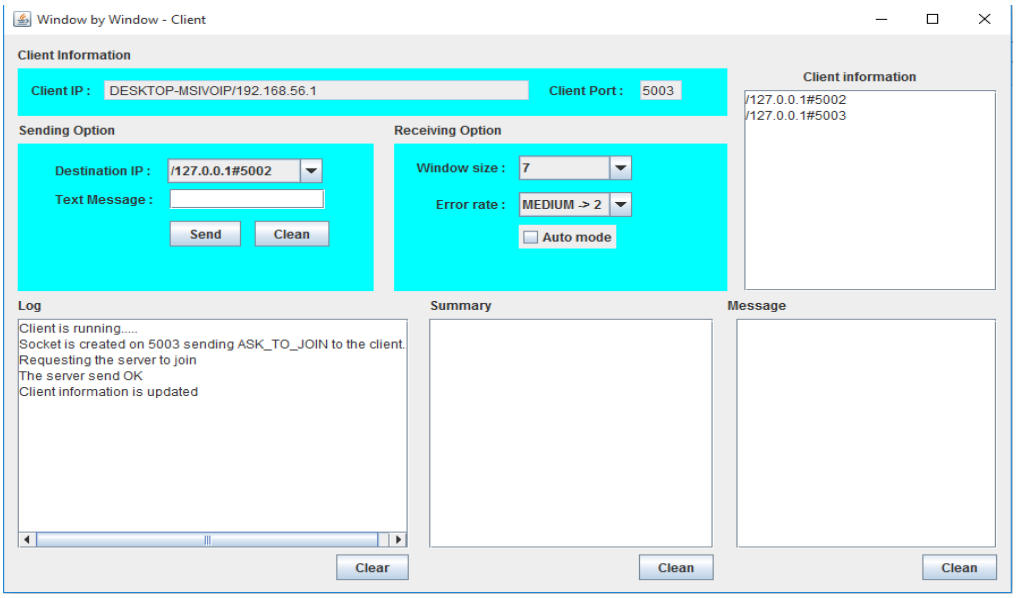
- Server



The following are description of the elements of the server interface

- **Server IP :** shows the IP address of the server
- **Server Port :** shows the port on which the server is running on
- **Max Client :** shows the maximum amount of allowed clients
- **Total Client :** shows how many clients are connected
- **Client Information :** shows the IP and port information of the connected clients
- **Total Packet :** shows how many packet are received, sent, and forwarded
- **Received Packet:** shows how many packets received from the clients. On the image it shows that it received two packet, which the two clients sent to the server to ask to join
- **Forward Packet :** shows how many packets are received from the clients to server which the server will forwarded it to the target client
- **Log :** shows log message to the user
- **Clear Button :** this clear the Log box in case of more message

➢ Client



The following are description of the elements of the client interface

- **Client IP :** shows the IP address of the client
- **Client Port :** shows the port on which the client is running on
- **Destination IP :** shows the connected clients which can the client can send message
- **Text Message :** give the client write a message that will be sent to the receiver
- **Client Information :** shows the IP and port information of the connected clients
- **Window size :** give an option to the user to select window size that will be used when the client receive message from another client
- **Error Rate :** give an option to the user to select error rate that will be applied when the client receive message from another client
- **Auto Mode :** this option will generate an error mode
- **Log :** shows log message to the client
- **Summary :** this shows the summary of the sending or receiving operation used to finish
- **Message :** this shows the message that is received from sender or sent to the received
- **Clear Buttons :** this buttons used to clear the above box messages in case of more message

B. Type of packet used and their description

No	Name of packet	From To	Description
1	ASK_TO_JOIN	• Client - Server	When client want to join
2	OK	• Server - Client	When the server accept the request
3	PORT_USED	• Server – Client	When the client ask to join with already used port the server send this packet to the client that ask to join
4	BLOCK	• Server – Client	When the server is full to accept new client the server send this packet for the client that ask to join
5	UPDATE_CLIENT	• Server – Client	When new client is connected or old client is disconnected the server send this packet to all connected used
6	BUSY_TRY_AGIAN	• Server - Sender	When the sender request to send message but if the server is busy the server send this packet to the sender
7	SETUP_REQUEST	• Sender - Server • Server - Receiver	When the sender want to send message to the receiver the client send this packet and if the server is not busy the server send it to the receiver
8	SETUP_REJECTED	• Receiver - Server • Server - Sender	When the receiver is busy to receive message it send this packet to the server and the server send it to the sender
9	SETUP_ACCEPTED	• Receiver - Server • Server - Sender	When the receiver is free to receive message it send this packet to the server and the server send it to the sender
10	DATA_SEND	• Sender - Server • Server - Receiver	When the sender send message packet it send this packet with the message to the server and the server send it to the sender
11	DATA_FINISH	• Sender - Server • Server - Receiver	When the sender send the last message packet it send this packet with the message to the server and the server send it to the sender
12	DATA_ACK	• Receiver - Server • Server - Sender	When the receiver received data packet without error it sends this packet to the server and the server send it to the sender
13	DATA_NAK	• Receiver - Server • Server - Sender	When the receiver received data packet with error it sends this packet to the server and the server send it to the sender
14	DATA_SAK	• Receiver - Server • Server - Sender	When the receiver received data packets with some packets error and some error free it sends this packet with the error packet to the server and the server send it to the sender
15	DATA_ACK_FINISH	• Receiver - Server • Server - Sender	When the receiver received the last data message packet without error it sends this to the server and the server send it to the sender
16	DATA_NAK_FINISH	• Receiver - Server	When the receiver received the last data

		• Server - Sender	message window with some error and some without error it sends this packet with the error packet to the server and the server send it to the sender

C. Case that are considered on prototype design

No	Case	Case considerations
1	When the server is open	• Create a specific port for the clients • Wait for client connections
2	When the clients are opened	• Ask the user to enter server port and IP • Generate a specific port • Send ASK_TO_JOIN packet to the server • If the server send PORT_USED packet generate another port and send ASK_TO_JOIN packet to the server until the server send OK packet • If the server send BLOCK packet wait and send ASK_TO_JOIN packet again
3	When new client ask to connect	• Check the maximum amount of client if it is full send BLOCK packet • Check whether the port is used or free, if it is used send PORT_USED packet • If it id free send OK packet • Update the clients buffer • Send UPDATE_CLIENT packet to all connected clients
4	When old client is disconnected	• Update the clients buffer • Send UPDATE_CLIENT packet to all connected clients
5	When the client want to send message	• Send SETUP_REQUEST packet to the server with the destination IP and port • If the server send BUSY_TRY_AGIAN wait for a specific time and send SETUP_REQUEST packet to server again • If the receiver send SETUP_REJECTED packet abort • If the receiver send SETUP_ACCEPTED packet send DATA_SEND packet with the message data until the last packet • If it is the last packet send the message data with DATA_FINISH packet • If the receiver send DATA_ACK send the next packet or window • If the receiver send DATA_NAK send the packet or window again • If the receiver send DATA_SAK send the error packet with the new packet • When it send DATA_FINISH and the receiver send DATA_ACK_FINISH finish the sending

		message and display transmission summary
6	When the client are receiving packet	• When is received check whether it is free to receive or not • If it is not free send SETUP_REJECTED packet to the server • If it is free send SETUP_ACCEPTED to the server and wait for the data packet and wait for DATA_SEND packet • When it receiver DATA_SEND packet and it is received without error add message on the buffer and send DATA_ACK • If it receive DATA_SEND with error send DATA_NAK • When it receive DATA_FINISH packet send DATA_ACK_FINISH, display message and summary for the user and delete message from buffer
7	When the server receive permission to send message from clients	• Check whether the server can handle another transmission if it cannot handle send BUSY_TRY_AGIAN to the sender • If it can handle send SETUP_REQUEST packet to the destination • If the receiver send SETUP_REJECTED packet forward it to the sender • If the receiver send SETUP_ACCEPTED packet forward it to the sender and update server load • Until the receiver send DATA_ACK_FINISH packet to the sender forwarded the data packet and acknowledgment packet between them • When the transmission is over update server load

D. Receiver implementation description

Variables
int Packet_Count = 0 , Error_Count = 0 , receiver _index = -1; int RECEIVER_WINDOW_SIZE = 0; String Error_Packet = "";

Class Name	**waitForPackets()**
Parameters	-
Descriptions	This class used to wait packets on the created port and collect the packet if any packet is received. Then it will chech the packet type and perform the next step based on the packet type.
	if SETUP_REQUEST check if it free if free

	sendPacket(SETUP_ACCEPTED); select RECEIVER_WINDOW_SIZE else *sendPacket(SETUP_REJECTED);* if DATA_SEND Packet_Count = Packet_Count + 1 if error packet Error_Count = Error_Count + 1 Error_Packet = Error_Packet + message_id else correct packet *message_buffer_receiver.add(messgage);* if Packet_Count == Window_Size if Error_Count == 0 *sendPacket(DATA_ACK);* else if Error_Count == Window_Size sendPacket(DATA_NAK); else *sendPacket(DATA_SAK);* Packet_Count = 0 Error_Count = 0 Error_Packet = "" if DATA_FINISH Packet_Count = Packet_Count + 1 if error packet Error_Count = Error_Count + 1 Error_Packet = Error_Packet + message_id else correct packet *message_buffer_receiver.add(messgage);* if Error_Count == 0 *sendPacket(DATA_ACK_FINISH);* *displayMessage(received_message);* else if Error_Count == Window_Size *sendPacket(DATA_NAK_FINISH);* else *sendPacket(DATA_SAK);* Packet_Count = 0 Error_Count = 0 Error_Packet = ""

Class Name	**displayMessage(***final String messageToDisplay***)**
Parameters	• *messageToDisplay* :- The message that will be displayed on the application
Descriptions	This class used to display informations on the application.
	append *messageToDisplay* on the display

Class Name	**sendPacket**(*byte[] data, int length, InetAddress address, int port*)
Parameters	• *byte[] data* :- The data that will be sent using the packet. • *int length* :- The length of the the data. • *InetAddress address* :- The IP address of which the packet will be sent. • *int port* :- The port number of which the packet will be sent.
Descriptions	This class used to send data to another client using the client IP and port.
	Send data to client on *address* IP and *port* port number

E. Sender implementation algorithm description

Variables
int start = -1 , last = -1; String message; int send_index = -1 , SENDER_WINDOW_SIZE = 0; List<Message_Buffer_sender> message_buffer_sender = new ArrayList<Message_Buffer_sender>();

Class Name	**displayMessage(**final String messageToDisplay**)**
Parameters	• *messageToDisplay* :- The message that will be displayed on the application
Descriptions	This class used to display informations on the application.
	append *messageToDisplay* on the display

Class Name	**sendPacket**(byte[] data, int length, InetAddress address, int port)
Parameters	• *byte[] data* :- The data that will be sent using the packet. • *int length* :- The length of the the data. • *InetAddress address* :- The IP address of which the packet will be sent. • *int port* :- The port number of which the packet will be sent.
Descriptions	This class used to send data to another client using the client IP and port.
	Send data to client on *address* IP and *port* port number

Class Name	**waitForPackets()**
Parameters	-
Descriptions	This class used to wait packets on the created port and collect the packet if any packet is received. Then it will chech the packet type and perform the next step based on the packet type.
	if SETUP_REJECTED abort sending message if SETUP_ACCEPTED set SENDER_WINDOW_SIZE *send_data_first_window(index);* if DATA_ACK *send_data_ACK(index,id);* if DATA_NAK *send_data_NAK(index, start, last);* if DATA_SAK

	put error packet id on *id_s* *send_data_SAK(index,id_s, last);* if DATA_ACK_FINISH *displayMessage(sent_message);* if DATA_NAK_FINISH *send_data_NAK(index,start,last);*

Class Name	**send_data_first_window**(*int index*)
Parameters	• *int index* :- This used to identify the receiver which the first window will be sent.
Descriptions	This class is used to send the first window to receiver.
	int total_packet = -1 count left packet on total_packet if total_packet == 0 *sendPacket(data with DATA_FINISH);* else if total_packet == SENDER_WINDOW_SIZE for(int p = 0 ;p <= SENDER_WINDOW_SIZE; p++) { if(p == SENDER_WINDOW_SIZE) *sendPacket(data with DATA_FINISH);* else *sendPacket(data with DATA_SEND);* } else if total_packet > SENDER_WINDOW_SIZE for(int p = 0 ;p <= SENDER_WINDOW_SIZE;p++) { *sendPacket(data with DATA_SEND);* } else if total_packet < SENDER_WINDOW_SIZE for(int p = 0 ;p <= total_packet;p++) { if(p == total_packet) *sendPacket(data with DATA_FINISH);* else *sendPacket(data with DATA_SEND);* }

Class Name	**send_data_ACK**(*int index,int id*)
Parameters	• *int index* :- The receiver index which the next window will be sent. • *int id* :- The packet id of the last packet on the previous window.
Descriptions	This class used to send the next window to thereceiver if the receiver send posetive acknowledgment for the window..
	message_buffer_sender.remove(id); int total_packet = -1 count left packet after message_id on total_packet if total_packet == 0 *sendPacket(data with DATA_FINISH);* else if total_packet == SENDER_WINDOW_SIZE for(int p = 0 ;p <= SENDER_WINDOW_SIZE;p++)

	{ if(p == Window_size) *sendPacket(data with DATA_FINISH);* else *sendPacket(data with DATA_SEND);* } else if total_packet > SENDER_WINDOW_SIZE for(int p = 0 ;p <= SENDER_WINDOW_SIZE;p++) { *sendPacket(data with DATA_SEND);* } else if total_packet < SENDER_WINDOW_SIZE for(int p = 0 ;p <= total_packet;p++) { if(p == total_packet) *sendPacket(data with DATA_FINISH);* else *sendPacket(data with DATA_SEND);* }

Class Name	**send_data_NAK**(int index,int start,int last)
Parameters	• *int index* :- The receiver index which the window will be sent again. • *int start* :- The packet id of the first packet on the window. • *int last* :- The packet id of the last packet on the window.
Descriptions	This class used to send the window again to the receiver if the receiver send negative acknowledgment for the window.
	int after_id = -1 int before_id = -1 count message from the start on before_id count message from the last on after_id if before_id == SENDER_WINDOW_SIZE if after_id < 0 for(int p = packet_id- WINDOW_SIZE;p <= WINDOW_SIZE;p++) { if(p == packet_id) *sendPacket(data with DATA_FINISH);* else *sendPacket(data with DATA_SEND);* } else for(int p = packet_id - WINDOW_SIZE ;p <= WINDOW_SIZE;p++) { *sendPacket(data with DATA_SEND);* } else for(int p = packet_id - before_id ;p <= message_id;p++) { if(p == packet_id) *sendPacket(data with DATA_FINISH);*

	else *sendPacket(data with DATA_SEND);* }

Class Name	**send_data_SAK**(int index,String id_s,int last)
Parameters	• *int index* :- The receiver index which the window will be sent again. • String id_s :- The packet id's which incorrectly received from the previous window. • int last :- The packet id of the last packet on the prevois window.
Descriptions	This class used to create new window which contain incorrect packets on the prvoius window and new packets and send the window to the receiver.
	int total_packet = -1 int error_count = -1 count the error packet on error_count count left packet on total_packet if total_packet < 0 if error_count == 0 sendPacket(data with DATA_FINISH); else if error_count > SENDER_WINDOW_SIZE for(int p = 0 ;p <= SENDER_WINDOW_SIZE;p++) { sendPacket(data with DATA_SEND); } else if error_count < SENDER_WINDOW_SIZE for(int p = 0 ;p <= error_count;p++) { if(p == error_count) sendPacket(data with DATA_FINISH); else sendPacket(data with DATA_SEND); } else if error_count == SENDER_WINDOW_SIZE for(int p = 0 ;p <= SENDER_WINDOW_SIZE;p++) { if(p == SENDER_WINDOW_SIZE) sendPacket(data with DATA_FINISH); else sendPacket(data with DATA_SEND); } else if error_count + total_packet == SENDER_WINDOW_SIZE for(int p = 0 ;p <= SENDER_WINDOW_SIZE;p++) { if(p == Window_size) sendPacket(data with DATA_FINISH); else sendPacket(data with DATA_SEND); } else if error_count + total_packet > SENDER_WINDOW_SIZE for(int p = 0 ;p <= SENDER_WINDOW_SIZE;p++) {

	sendPacket(data with DATA_SEND); } else if error_count + total_packet < SENDER_WINDOW_SIZE for(int p = 0 ;p <= total_packet;p++) { if(p == total_packet) sendPacket(data with DATA_FINISH); else sendPacket(data with DATA_SEND); }

F. Buffer packet class for Sender and Receiver

Sender	Receiver
class **Message_Buffer_sender** { String receiver_ip; int receiver_port; int id; int message_id; String message; Message_Buffer_sender (int id,String ip, int port, String message,int message_id) { this.receiver_ip = ip; this.receiver_port = port; this.message = message; this.id = id; this.message_id = message_id; } public String get_receiver_ip() { return receiver_ip; } public int get_receiver_port() { return receiver_port; } public int get_id() { return id; } public String get_message() { return message; } public int get_message_id() { return message_id; } }	class **Message_Buffer_sender** { String receiver_ip; int receiver_port; int id; int message_id; String message; Message_Buffer_sender(int id,String ip, int port, String message,int message_id) { this.receiver_ip = ip; this.receiver_port = port; this.message = message; this.id = id; this.message_id = message_id; } public String get_receiver_ip() { return receiver_ip; } public int get_receiver_port() { return receiver_port; } public int get_id() { return id; } public String get_message() { return message; } public int get_message_id() { return message_id; } }