

# **Education Tool For OSI Model**

Project Report submitted in partial fulfillment of the requirement for the  
degree of

Bachelor of Technology

in

**Computer Science & Engineering**

under the Supervision of

*Dr. Nitin Chanderwal*

By

**Aastha Sharma 111211**

to



Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

## Certificate

This is to certify that project report entitled “Education Tool For OSI Model”, submitted by Aastha Sharma(111211)in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

**Date:**

**Supervisor’s Name: Dr.Nitin Chanderwal**

**Designation: Associate Professor**

## **Acknowledgement**

It is indeed a matter of great privilege and honor for me to express my sincere and deepest acknowledgements towards all those who have contributed either directly or indirectly towards the completion of my project.

At the very outset, I wish to place on record my sincere thanks to my Project Guide **Dr. Nitin Chanderwal**, for his most valuable suggestions and support. He has motivated me throughout the entire duration of my work and this could not have been possible without his blessed guidance and unmatched wealth of knowledge.

I am also thankful to Jaypee University of Information Technology for the guidance and constant supervision.

Lastly, I would like to thank the almighty and my parents for their moral support and my friends with whom I shared my day to day experience and received lots of suggestions that improved my quality of work

Date:

Name of the student: Aastha Sharma

## Table of Content

<b>S. No.</b>	<b>Topic</b>	<b>Page No.</b>
1.	Abstract	1
2.	Motivation	2
3.	Introduction	3
3.1	Importance of OSI Model	4
3.2	OSI Layering	6
3.3	Principles of ISO for the the Layers	7
4.	Layers of OSI Model	9
4.1	Physical Layer	9
4.2	Data Link Layer	11
4.3	Network Layer	14
4.4	Transport Layer	16
4.5	Session Layer	18
4.6	Presentation Layer	20
4.7	Application Layer	22
4.8	FEC Encoder and Decoder	23
4.9	Activity Diagram	29
4.10	Use-Case Diagram	30
5.	Screenshots	31
6.	Conclusion	42
7.	Tools and Technologies	43
8 .	References	44

9. Appendices	46
Appendix A	45
Appendix B	55

## List of Figures

<b>S.No.</b>	<b>Title</b>	<b>Page No.</b>
1.	OSI Layers	6
2.	Physical Layer	9
3.	Data Link Layer	12
4.	Network Layer	15
5.	Transport Layer	17
6.	Session Layer	19
7.	Presentation Layer	21
8.	Application Layer	23
9.	FEC Encoder and Decoder	24
10.	FEC Encoder	26
11.	FEC Interleaver	27
12.	FEC De-interleaver	28
13.	FEC Decoder	28

# 1. Abstract

The project is based on making an education tool for the simulation of the layers of the OSI model which will help to understand the function of these layers.

A reference model is a conceptual blueprint of how communication should take place. It addresses all the process required for effective communication and divides these processes into logical grouping called layers. When a communication system is designed in this manner, it is known as layered architecture. The OSI isn't a physical model, though. Rather, it's a set of guidelines that application developers used to create and implement application that run on network. It also provides a framework for creating and implementing networking standards, devices, and internetworking schemes which comprises of seven different layers. Each layer is having its own responsibilities.

It was ratified in 1984, is like an INTERFACE between two parties i.e., one sender and other receiver. The OSI model was basically developed to simplify network complexity, facilitate network training and introduce easy network troubleshooting.

OSI model is called reference model because it provides reference to implement network communication. It gives set of rules, how different networks communicate with each other. The OSI defines a networking framework to implement protocols in seven layers. Control is passed from one layer to the next, starting at the application layer in one station, and proceeding to the bottom layer, over the channel to the next station and back up the hierarchy. A layer serves the layer above it and is served by the layer below it. For example, a layer that provides error-free communications across a network provides the path needed by applications above it, while it calls the next lower layer to send and receive packets that make up the contents of that path. Two instances at one layer are connected by a horizontal connection on that layer.

## 2. MOTIVATION

Over the past couple of decades many of the networks that were built used different hardware and software implementations, as a result they were incompatible and it became difficult for networks using different specifications to communicate with each other.

To address the problem of networks being incompatible and unable to communicate with each other, the International Organisation for Standardisation (ISO) researched various network schemes. The ISO recognised there was a need to create a NETWORK MODEL that would help vendors create interoperable network implementations.

The purpose of the OSI reference model is to guide vendors and developers so the digital communication products and software programs they create will interoperate, and to facilitate clear comparisons among communications tools. Most vendors involved in telecommunications make an attempt to describe their products and services in relation to the OSI model. And although useful for guiding discussion and evaluation, OSI is rarely actually implemented, as few network products or standard tools keep all related functions together in well-defined layers as related to the model.

The main concept of OSI is that the process of communication between two endpoints in a telecommunication network can be divided into seven distinct groups of related functions, or layers. Each communicating user or program is at a computer that can provide those seven layers of function. So in a given message between users, there will be a flow of data down through the layers in the source computer, across the network and then up through the layers in the receiving computer. The seven layers of function are provided by a combination of applications, operating systems, network card device drivers and networking hardware that enable a system to put a signal on a network cable or out over Wi-Fi or other wireless protocol).



### **3. INTRODUCTION**

The OSI, or Open System Interconnection, model defines a networking framework to implement protocols in seven layers. This model defines a networking framework to implement protocols in seven layers. Control is passed from one layer to the next, starting at the application layer in one station, and proceeding to the bottom layer, over the channel to the next station and back up the hierarchy.

The International Standard Organization (ISO) is a multinational body dedicated to world wide agreement on international standard which was established in 1947. The ISO proposed a model named as OSI (Open System Interconnection) in 1983, which covers all aspects of network communication. The purpose of OSI model is for open communication between different systems without requiring changes to logic of the underlying hardware and software. The OSI model is not a protocol, it is a model for understanding and designing a network architecture that flexible, robust and interoperable. In 1977, the International organization for Standardization (ISO) recognized the special and urgent need for standards for heterogeneous informatics networks and decided to create a new subcommittee (SC16) for “Open Systems Interconnection”. The universal need for interconnecting systems from different manufacturers rapidly became apparent, leading ISO to decide for the creation of SC16 with the objective to come up with standards required for “Open Systems Interconnection”. The term “open” was chosen to emphasize the fact that by conforming to those systems obeying the same standards throughout the world.

The purpose of the OSI reference model is to guide vendors and developers so the digital communication products and software programs they create will interoperate, and to facilitate clear comparisons among communications tools. Most vendors involved in telecommunications make an attempt to describe their products and services in relation to the OSI model. And although useful for guiding discussion and evaluation, OSI is rarely actually implemented, as few network products or standard tools keep all related functions together in well-defined layers as related to the model.

## **3.1 IMPORTANCE OF OSI MODEL**

The OSI model has many benefits which include:

### **3.1.1 Reduces Complexity:**

It breaks the network communication process into smaller and simpler components thus aiding component development, design and troubleshooting.

### **3.1.2 Compatibility:**

The OSI model can fit to any compatible software/hardware from different users in other parts of the world. As software/hardware differs among various users so OSI is a model that is compatible to all.

### **3.1.3 Simplifies teaching and learning**

OSI model is very interactive and even guides us to know what a model is, how it operates, and common methodologies, how new technologies are developed in existing networks.

### **3.1.4 Security:**

OSI model have functionality for Encryption and Decryption which has a major contribution for security purpose. This makes it Reliable.

### **3.1.5 Interoperability between Vendors:**

It allows multiple vendor development through standardization of network components. Defines the process for connecting two layers together, promoting interoperability between vendors. It allows vendors ti compartmentalize their design efforts to fit a modular design, which eases the implementations and simplifies troubleshooting.

### **3.1.6 Accelerates evolution:**

It provides for effective updates and improvements to individual components without affecting other components or having to rewrite the entire protocol.

### **3.1.7 Ensures interoperable technology:**

It prevents changes in one layer from affecting the other layers, allowing for quicker development.

### **3.1.8 Easy Troubleshooting:**

Since each layer in an OSI is independent of each other so it makes it easier to detect and solve all errors prevailing in it.

**3.1.9 Add multiple network models:** The OSI model is designed in such a way that user can further extend.

## **3.2 OSI LAYERING**

Layering is a structuring technique which permits the network of Open Systems to be viewed as logically composed of a succession of layers, each wrapping the lower layers and isolating them from the higher layers. The subcommittee (SC16) which is created by ISO has given an illustration of layering shown in figure where successive layers are represented in a vertical sequence, with the physical media for Open Systems Interconnection at the bottom.

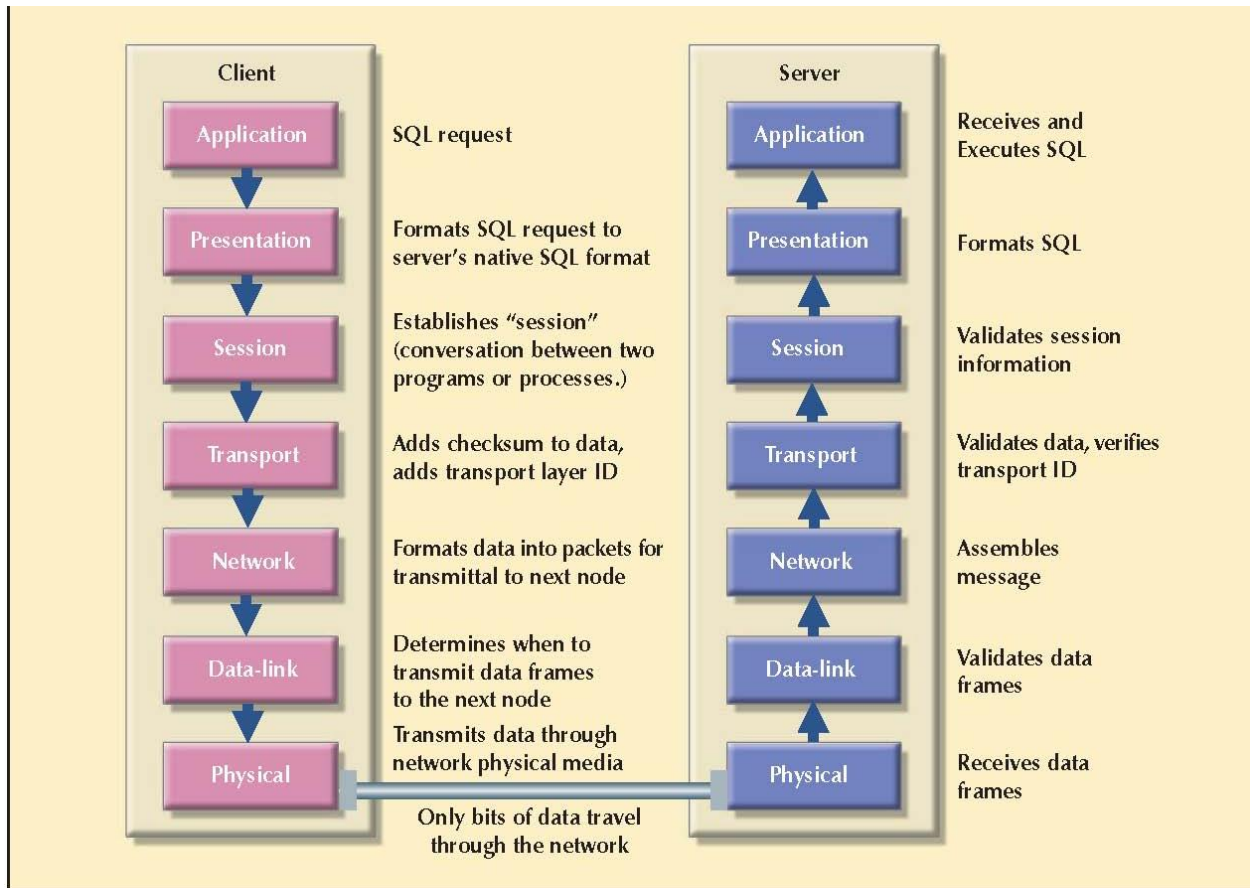


Fig 1 (OSI Layers)

Each individual system itself is viewed as being logically composed of a succession of subsystems, each corresponding to the intersection of the system with a layer. In other words, a layer is viewed as being logically composed of subsystems of the same rank of all interconnected systems. Each subsystem is, in turn, viewed as being made of one or several entities. In other words, each layer is made of entities, each of which belongs to one system. Entities in the same layer are termed peer entities.

The basic idea of layering is that each layer adds value to services provided by the set of lower layers in such a way that the highest layer is offered the set of services needed to run distributed applications. Layering thus divides the total problem into smaller pieces. Another basic principle

of layering is to ensure independence of each layer by defining services provided by layer to the next higher layer, independent of how these services are performed. This permits changes to be made in the way a layer or a set of layers operate, provided they still offer the same service to the next higher layer.

### **3.3 PRINCIPLES OF ISO FOR THE SEVEN LAYERS OF THE OSI ARCHITECTURE**

ISO determined a number of principles to be considered for defining the specific set of layers in the OSI architecture, and applied those principles to come up with the seven layers of the OSI Architecture. Principles to be considered are as follows-

- i. Do not create so many layers to make difficult the system engineering task describing and integrating these layers.
- ii. Create a boundary at a point where the services description can be small and the number of interactions across the boundary is minimized.
- iii. Create separate layers to handle functions which are manifestly different in the process performed or the technology involved.
- iv. Collect similar functions into the same layer.
- v. Select boundaries at a point which past experience has demonstrated to be successful.
- vi. Create a layer of easily localized functions so that the layer could be totally redesigned and its protocols changed in a major way to take advantages of new advances in architectural, hardware, or software technology without changing the services and interfaces with the adjacent layers. Create a boundary where it may be useful at some point in time to have the corresponding interface standardized.
- vii. Create a layer when there is a need for a different level of abstraction in the handling of data, e.g., morphology, syntax, semantics.
- viii. Enable changes of functions or protocols within a layer without affecting the other layers.

- ix. Create for each layer interfaces with its upper and lower layer only.
- x. Create further subgrouping and organization of functions to form sublayers within a layer in cases where distinct communication services need it.
- xi. Create, where needed, two or more sublayers with a common, and therefore minimum, functionality to allow interface operation with adjacent layers.

## 4. LAYERS OF OSI MODEL

### 4.1 PHYSICAL LAYER

The physical layer is responsible for individual *bits* from one node to another. It coordinates the rule for transmitting bits. It launches the raw bits in the channel or link. This layer defines,

- Physical network structures.
- Mechanical and electrical specifications of the transmission medium.
- Bit transmission encoding and timing rules.

The following network connectivity hardware are normally associated with physical layer are,

- Hubs or Switches
- Transmission media connectors
- Modems

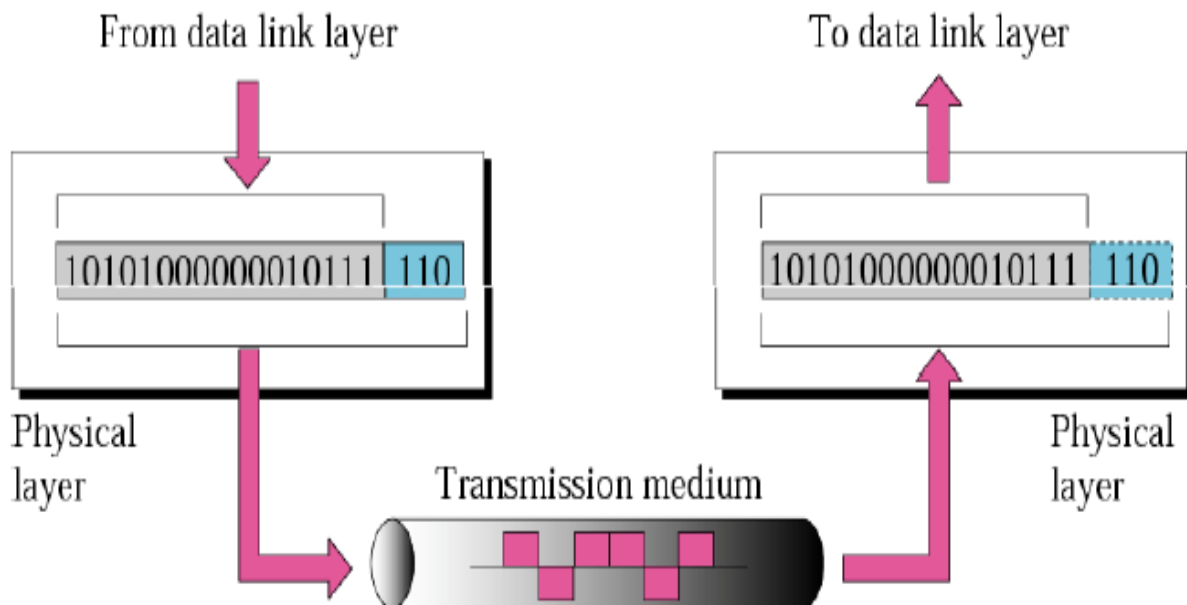


Fig 2  
(PHSICAL LAYER)

Major duties of this layer are:

**a) Physical Characteristics of Interface and Media:**

It defines the characteristics of the interface between the devices and transmission medium. It also defines the type of transmission medium used.

**b) Representation of Bits:**

The physical layer data consists of stream of bits(Sequence of 0's and 1's) without any interpretation. For transmission, bits are encoded into electrical or optical signals. The physical layer defines the type of representation (optical or electrical).

**c) Data Rate (or) Transmission Rate:**

The Sender and receiver must be synchronized at bit level.

**d) Synchronization of bits:**

The sender and receiver not only must use the same bit rate but also must be synchronized at the bit level. In other words, the sender and the receiver clocks must be synchronized.

**e) Line Configuration:**

This layer is concerned with the connection of devices to the media. In a point-to-point configuration, two devices are connected through a dedicated link. In a multipoint configuration, a link is shared among several devices.

**f) Physical Topology:**

The physical topology defines how devices are connected to make a network. Devices can be connected by using a mesh topology (every device is connected to every other device), a star topology (devices are connected through a central device), a ring topology (each device is connected to the next, forming a ring), a bus topology (every device is on a common link), or hybrid topology (this is a combination of two or more topologies).

**g) Transmission Mode:**

The physical layer also defines the direction of transmission between two devices: simplex, half-duplex, or full-duplex. In simplex mode, only one device can send; the other can only receive.

The simplex mode is a one-way communication. In the half-duplex mode, two devices can send



and receive, but not at the same time. In a full-duplex (or simply duplex) mode, two devices can send and receive at the same time.

## 4.2 DATA LINK LAYER

This layer is responsible for the two party communications by exchanging *frames* between the two nodes. It describes methods for moving information between multiple devices within the same logical network based on physical device addressing. It makes the physical layer, a raw transmission facility, to a reliable link. It makes the physical layer error free to upper layer (network). The basic purposes for Data Link Layer protocol implementations are:

- a) Organize Physical Layers bits into logical groups of information called *frames*.
- b) Detect errors.
- c) Control data flow.
- d) Identify computers on the network.

Data link layer protocols:

- a) HDLC (High-Level Data Link Control)
- b) Frame relay

The protocol packages the data into frames that contain source and destination addresses. These frames refer to the physical hardware address of each network card attached to the network cable. Ethernet, Token Ring, and ARC net (Attached Resource Computer network) are examples of LAN(Local Area Network) data link protocols. If communication extends beyond the LAN onto the Internet, the network might use other data link protocols, such as Point-to-Point Protocol (PPP) or Serial Line Internet Protocol (SLIP). The data link layer sends block of data with necessary synchronization, bit error detection/correction error control, and flow control. This control of data flow controls approximately 70 percent of all error handling. Since the physical layer merely accepts and transmits a stream of bits without any regard to the meaning of the structure, it is up to the data link layer to create and recognize frame boundaries. This can be

accomplished by attaching special bit patterns to the beginning and end of the frame as shown in figure T2 (trailer of layer 2) and H2 (header of layer 2) are attached at the beginning and ending of the frame.

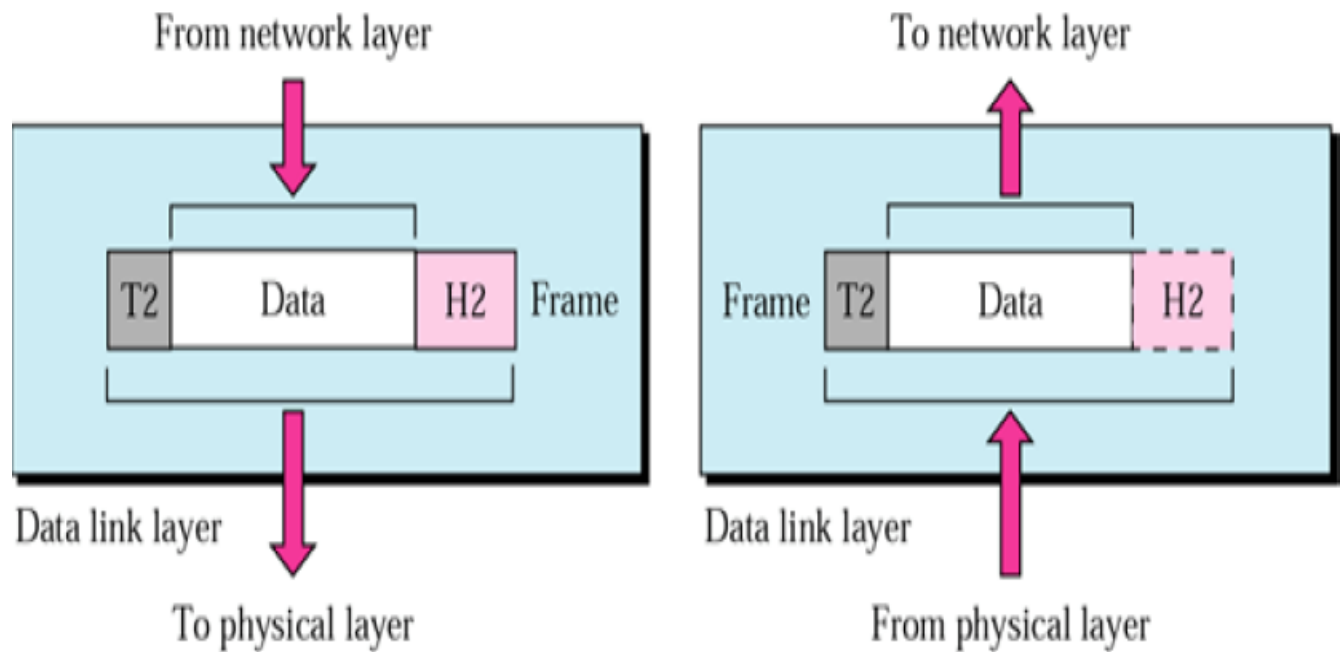


Fig 3  
(DATA LINK LAYER)

This layer is subdivided into two sub-layers:

**a) Logical Link Control (LLC):**

This sub layer functions include – Managing frames to upper and lower layers, error control and flow control.

**b) Media Access Control (MAC):**

The MAC sublayer carries the physical address of each device on the network. This address is more commonly called a device's MAC address. MAC address is a 48 bit address which is burned into the NIC card (Network Interface Card) on the device by its manufacturer. NICs have a MAC address. A switch uses this address to filter and forward traffic, helping relieve congestion and collision on a network segment.

The major duties of the Data link layer are,

**a) Framing:**

The data link layer divides the stream of bits received from network layer into manageable data units called frames.

**b) Physical Addressing:**

If frames are to be distributed to different systems on the network, the data link layer adds a header to the frame to define the sender and/or receiver of the frame. If the frame is intended for a system outside the sender's network, the receiver address is the address of the device that connects the network to the next one.

**c) Flow Control:**

The data link layer imposes flow control mechanism to prevent overwhelming the receiver, i.e., if the rate at which the data is absorbed by receiver is less than the rate of which it is produced at sender.

**d) Error Control:**

The data link layer adds reliability to the physical layer by adding mechanisms to detect and retransmit damaged or lost frames. It also uses a mechanism to recognize duplicate frames. Error control is normally achieved through a trailer added to the end of the frame.

**e) Access Control:**

When two or more devices are connected to the same link, then medium access mechanism should be evolved to determine which device has control over the link at any given time.

## **4.3 NETWORK LAYER**

The network layer is responsible for the delivery of *packets* from original source to final destination. Before receiving the packets to the respective destination, *buffering* of packets takes place in the network. This layer takes care of addressing and routing issues. It describes methods

for moving information between multiple independent networks based on network layer addressing. The data link layer oversees the delivery of the packet delivery between two systems on the same network, whereas the network layer that each packet gets to the final destination. If systems are connected on same link, there is no need for network layer. If systems are attached to different networks then it is inevitable.

While the data link layer deals with the method in which the physical layer is used to transfer data, the network layer deals with organizing that data for transfer and reassembly. The network layer provides the functional and procedural means of transferring variable length data sequences (called datagrams) from one node to another connected to the same *network*. A network is a medium to which many nodes can be connected, on which every node has an *address* and which permits nodes connected to it to transfer messages to other nodes connected to it by merely providing the content of a message and the address of the destination node and letting the network find the way to deliver ("route") the message to the destination node. In addition to message routing, the network may (or may not) implement message delivery by splitting the message into several fragments, delivering each fragment by a separate route and reassembling the fragments, report delivery errors.

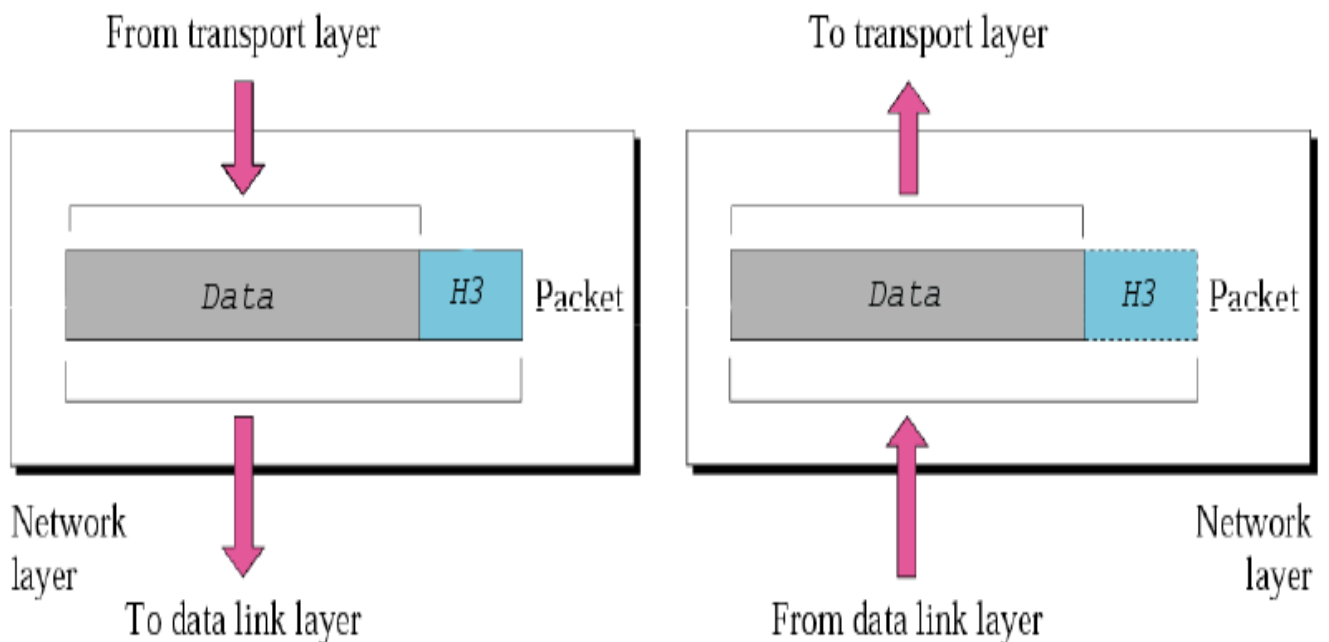


Fig 4  
(NETWORK LAYER)

This layer defines,

- a) Logical Network structures and addressing.
- b) Route discovery and selection.
- c) Network layer flow control and error control.
- d) The network connectivity hardware associated with network layer.
- e) Routers.
- f) Network layer protocols.
- g) IP (Internet Protocol) etc.

The major duties of network layer are:

**a) Logical Addressing:**

The physical address implemented at data link layer handles the addressing problem locally. If packet passes the network boundary, then we need logical address. The network layer adds a header to the packet coming from upper layer with source and destination logical address.

**b) Routing:**

In a large network, the connecting devices (router or switches) route or switch the packets to their final destination.

## **4.4 TRANSPORT LAYER**

The transport layer is responsible for delivery of a message from one process to another. It is responsible for process to process delivery of the entire message. The network layer ensures host to destination delivery of individual packets. It does not recognize the relationship between the packets. It does not recognize the relationship between the packets. So the transport layer ensures that the whole message arrives intact and in order, overseeing both error control and flow control at process-to-process level. It is responsible for end-to-end connection between the source and

the destination. The name of the data unit in transport layer is *TPDU* (Transport Protocol Data Unit).

This layer is mainly concerned with Reliability (security like encryption/decryption, firewall etc.), maintenance of flow of packets by reducing CONGESTION (When a Network receives more packets than its capacity) and performs error checking ensuring quality of service by resending data when data has been corrupted.

This layer provides transparent transfer of data between end systems, or hosts, and is responsible for end-to-end error recovery and flow control. It ensures complete data transfer.

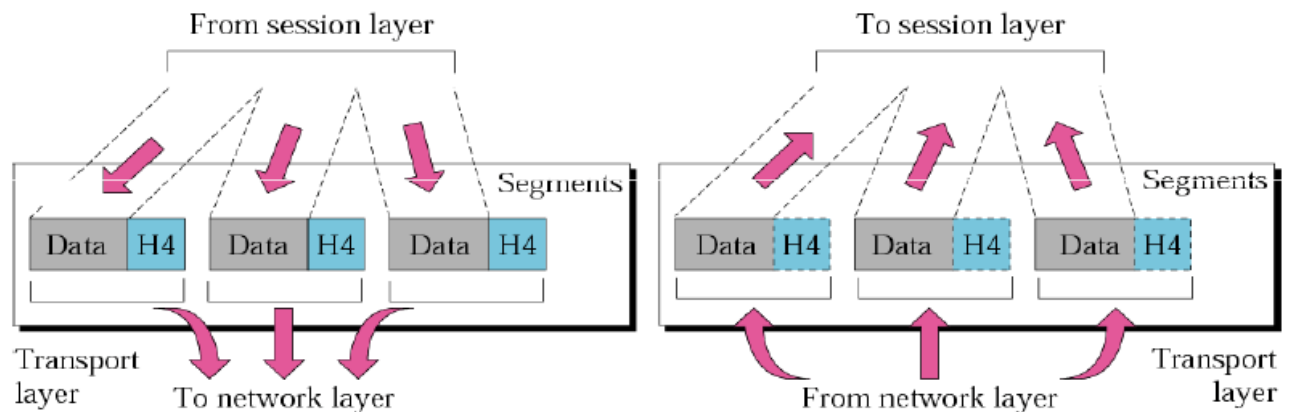


Fig 5  
(TRANSPORT LAYER)

This layer defines,

- a) Connection and transaction identifiers.
- b) Segment development.
- c) Connection services.
- d) TCP
- e) UDP etc.

The duties of transport layer are:

**(a) Service-point addressing:**

Computers often run several programs at the same time. For this reason, source-to-destination delivery means delivery not only from one computer to the next but also from a specific process (running program) on one computer to a specific process (running program) on the other. The transport layer header must therefore include a type of address called a *service-point address* (or port address). The network layer gets each packet to the correct computer; the transport layer gets the entire message to the correct process on that computer.

**b) Segmentation and reassembly:**

If the message is big, it is divided into transmittable segments. Each segment contains a sequence number, which enables the transport layer to reassemble the segments upon arrival at the destination. It is also used to identify the packet lost during transmission.

**c) Connection control:**

The transport layer can be either connection-oriented or connectionless. A connectionless transport layer treats each segment as an independent packet and delivers to the transport layer of the destination machine. Whereas, a connection-oriented transport layer establishes a connection with the transport layer of the destination machine first before delivering the packets. After data is transferred, the connection is terminated.

**d) Flow Control:**

Like the data link layer, the transport layer is responsible for flow control. However, flow control at this layer is performed end to end rather than across a single link.

**e) Error Control:**

Like the data link layer, the transport layer is responsible for error control. However, error control at this layer is performed process-to-process rather than across a single link. The sending transport layer makes sure that the entire message arrives at the receiving transport layer without error (damage, loss, or duplication). Error correction is usually achieved through retransmission.

## 4.5 SESSION LAYER

The session layer permits two parties to hold ongoing communications called a session across a network. The applications on either end of the session can exchange data or send packets to another for as long as the session lasts. The session layer handles session setup, data or message exchanges, and tear down when the session ends. It also monitors session identification so only designated parties can participate and security services to control access to session information. A session can be used to allow a user to log into a remote time-sharing system or transfer a file between two machines.

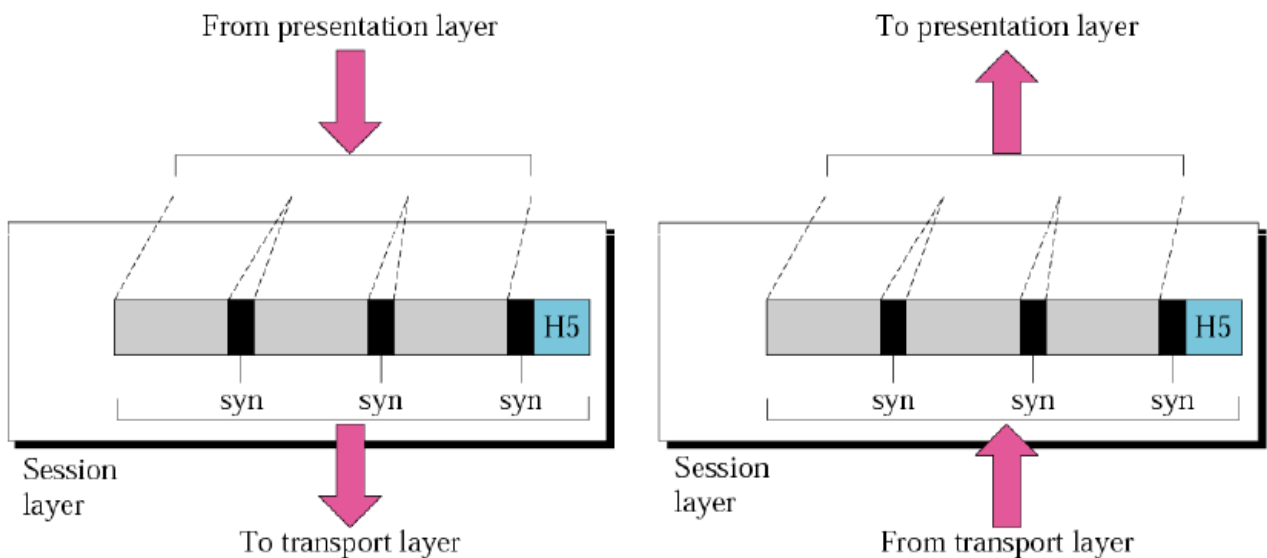


Fig 6  
(SESSION LAYER)

The session layer has the option of providing one-or-two-way communication called dialogue control. Sessions can allow traffic to go in both directions at the same time, or in only one direction at a time. Token management may be used to prevent both sides from attempting the same operation at the same time. To manage these activities, the session layer provides tokens that can be exchanged. Only the side holding the token is permitted to perform the critical operation. Another session service is synchronization. Consider the problems that occur when transferring a file between two machines and the system crashes not being able to complete the



transfer. This process must be restarted from the beginning. To avoid this problem, the session layer provides a way to insert checkpoints into the data stream, so that after a crash, only the data after the last checkpoint has to be repeated. It accepts the data from presentation layer and provides services to it and accepts the services of the transport layer. The name of data unit in the session layer is *SPDU* (Session Protocol Data Unit) or sessions.

Therefore session layer functionality includes:

- a) Virtual connection between application entities
- b) Synchronization of data flow
- c) Creation of dialog units
- d) Connection parameter negotiations
- e) Partitioning of services into functional groups.
- f) Acknowledgments of data received during a session
- g) Retransmission of data if it is not received by a device

Specific responsibilities of the session layer include the following:

**(a)Dialog control:**

The session layer allows two systems to enter into a dialog. It allows the communication between two processes to take place in either half duplex (one way at a time) or full-duplex (two ways at a time) mode.

**(b)Synchronization:**

The session layer allows a process to add checkpoints, or synchronization points, to a stream of data. For example, if a system is sending a file of 2000 pages, it is advisable to insert checkpoints after every 100 pages to ensure that each 100-page unit is received and acknowledged independently. In this case, if a crash happens during the transmission of page 523, the only pages that need to be resent after system recovery are pages 501 to 523. Pages previous to 501 need not be resent.

## 4.6 PRESENTATION LAYER

The presentation layer is responsible for the format of the data transferred during network communications. This layer is concerned with the syntax and semantics of the information transmitted. For outgoing messages, it converts data into a generic format for the transmission. For the incoming messages, it converts the data from the generic form to a format understandable to the receiving application. Different computers have different codes for representing data. The presentation layer makes it possible for computers with different representation to communicate. The presentation layer provides common communication services such as encryption, text compression, and reformatting. The presentation layer is also concerned with other aspects of information representation. Data compression can be used to reduce the number of bits that have to be transmitted. Cryptography is frequently required for privacy and authentication. The name of data unit in the presentation layer is *PPDU* (Presentation Protocol Data Unit).

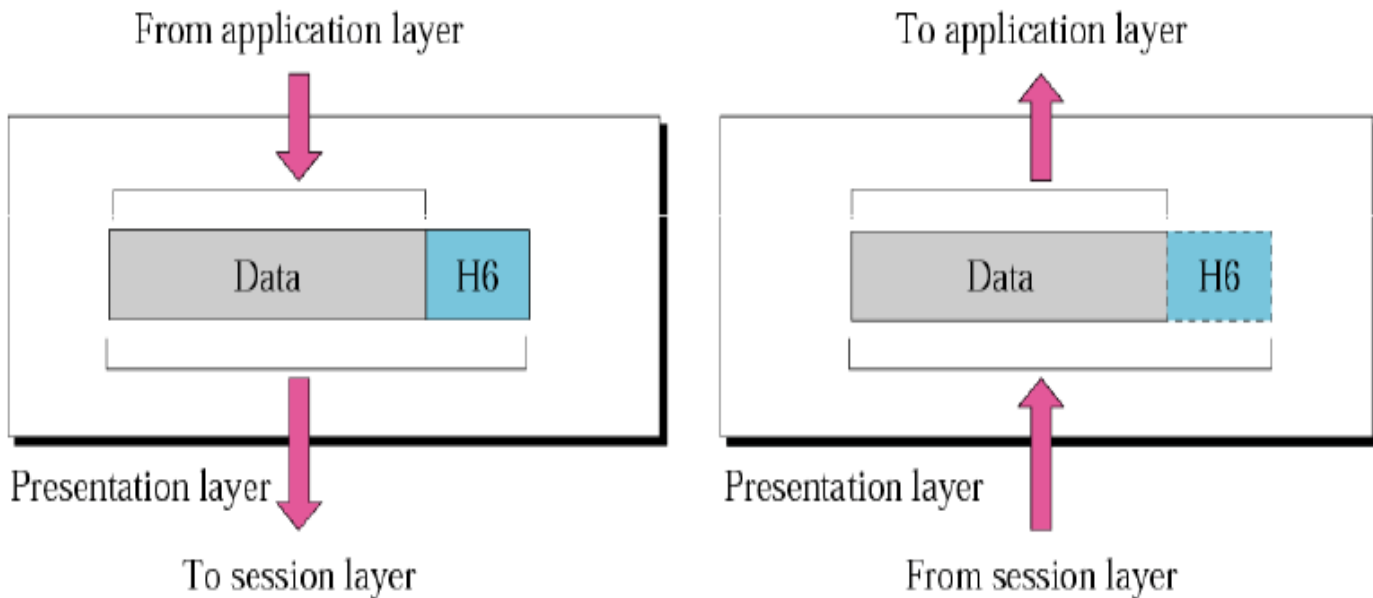


Fig 7  
(PRESENTATION LAYER)

Specific responsibilities of the presentation layer include the following:

**(a) Translation:**

The processes (running programs) in two systems are usually exchanging information in the form of character strings, numbers, and so on. The information must be changed to bit streams before being transmitted. Because different computers use different encoding systems, the presentation layer is responsible for interoperability between these different encoding methods. The presentation layer at the sender changes the information from its sender-dependent format into a common format. The presentation layer at the receiving machine changes the common format into its receiver-dependent format.

**(b) Encryption**

To carry sensitive information, a system must be able to ensure privacy. Encryption means that the sender transforms the original information to another form and sends the resulting message out over the network. Decryption reverses the original process to transform the message back to its original form.

**(c) Compression**

Data compression reduces the number of bits contained in the information. Data compression becomes particularly important in the transmission of multimedia such as text, audio, and video.

This layer works by taking care of the directions given by the user at the application layer. It is where the human readable programming languages are translated into machine code instructions used by the lower layers. In general the main work of Presentation is Data representation, security encryption, and converts computer code to network formatted code. This layer (i.e. presentation layer) provides independence from data representation (e.g., encryption) by translating between application and network formats. The presentation layer transforms data into the form that the application accepts. This layer formats and encrypts data to be sent across a network. It is sometimes called the syntax layer.

## **4.7 Application Layer**

The application layer enables the user, whether human or software to access the network. It provides user interface and support for services such as e-mail, remote file access and transfer, shared database management etc.

**a)** It accepts the services from presentation layer and data unit in this layer is called APDU (Application Protocol Data Unit).

### **b) File Transfer, Access and Management:**

Provides handling services in the network. This includes the movement of files between different systems, reading, writing and deletion of remote files, and management of remote file storage.

### **c) Network virtual Terminal:**

Provides services to access applications in different remote computer systems through stimulating a real terminal.

### **d) Electronic Mail and Messaging Handling:**

Facilitates the electronic exchange of documents.

### **e) Directory Services (DS):**

Provides services with the ability to match names with addressing information.

### **f) Common management Information Protocol:**

Provides services for network management

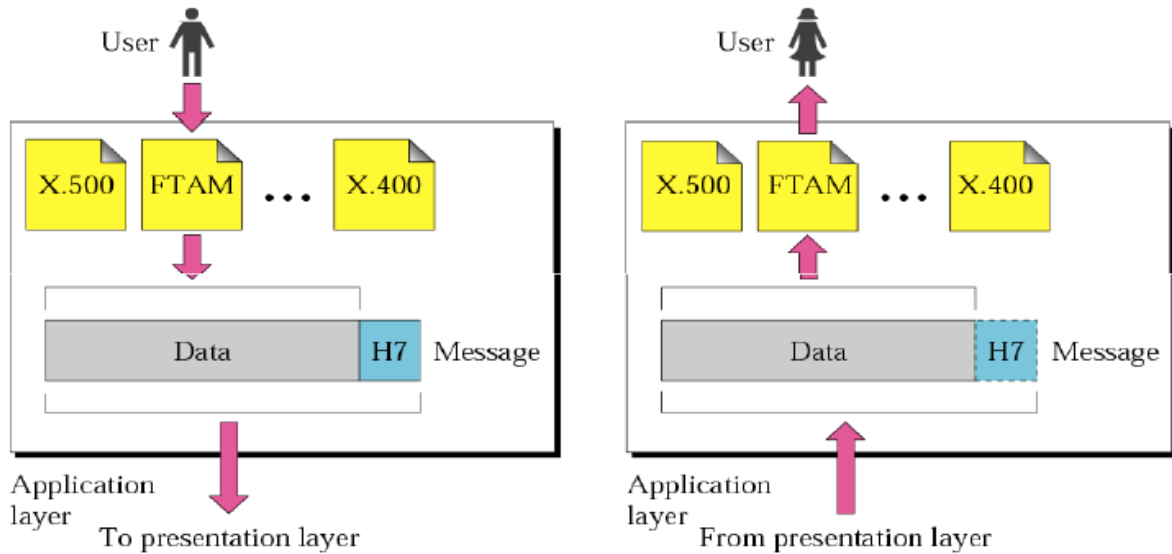


Fig 8  
(APPLICATION LAYER)

This is the level that the user often interacts with. This is where data turns into websites, chat programs and so on

This layer supports application and end-user processes. Communication partners are identified, quality of service is identified, user authentication and privacy are considered, and any constraints on data syntax are identified. Everything at this layer is application-specific.

This layer provides application services for file transfers, e-mail, and other network software services.

## 4.8 FEC ENCODER AND DECODER

In telecommunication, information theory, and coding theory, **forward error correction (FEC)** or channel coding is a technique used for controlling errors in data transmission over unreliable or noisy communication channels. The central idea is the sender encodes his message in a redundant way by using an error-correcting code (ECC). The American mathematician Richard

Hamming pioneered this field in the 1940s and invented the first error-correcting code in 1950: the Hamming code.

The redundancy allows the receiver to detect a limited number of errors that may occur anywhere in the message, and often to correct these errors without retransmission. FEC gives the receiver the ability to correct errors without needing a reverse channel to request retransmission of data, but at the cost of a fixed, higher forward channel bandwidth. FEC is therefore applied in situations where retransmissions are costly or impossible, such as one-way communication links and when transmitting to multiple receivers in multicast. FEC information is usually added to mass storage devices to enable recovery of corrupted data, and is widely used in modems.

FEC processing in a receiver may be applied to a digital bit stream or in the demodulation of a digitally modulated carrier. For the latter, FEC is an integral part of the initial analog-to-digital conversion in the receiver. The Viterbi decoder implements a soft-decision algorithm to demodulate digital data from an analog signal corrupted by noise. Many FEC coders can also generate a bit-error rate (BER) signal which can be used as feedback to fine-tune the analog receiving electronics.

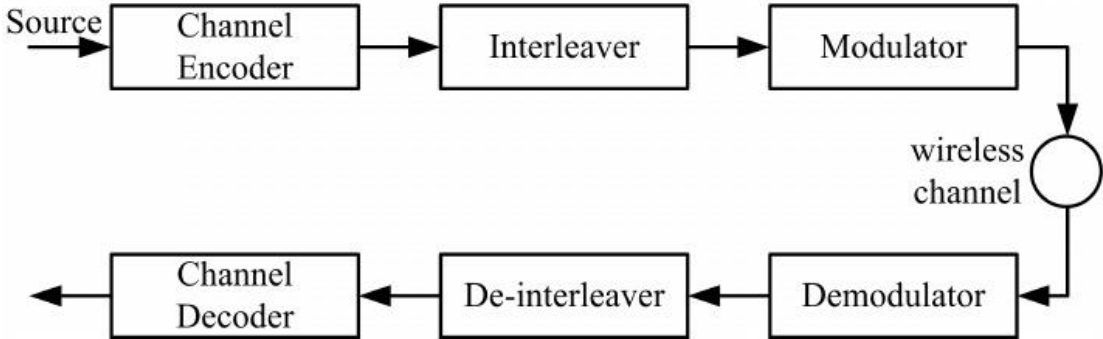


Fig. 9 (FEC Encoder and Decoder)

FEC adds redundancy to transmitted information using a predetermined algorithm. The redundant bits are complex functions of the original information bits. Bits are sent multiple

times, because an error may appear in any of the samples transmitted. FEC codes generally detect the last set of bits to determine the decoding of a small handful of bits.

With FEC, each character is sent two or three times, and the receiver checks instances of each character. It is accepted only if conformity occurs in both instances. If conformity is satisfied for an instance, the character conforming to the protocol is accepted. If no characters conform to the protocol, the character is rejected and an underscore or blank is displayed in its place.

The steps involved are:

### **1. FEC Encoder:**

FEC is a system of error control for data transmission, where the sender adds redundant data to its messages. This allows the receiver to detect and correct errors (within some bounds) without the need to ask the sender for additional data. In this module we add redundant data to the given input data, known as FEC Encoding.

The text available in the input text file is converted into binary. The binary conversion is done for each and every character in the input file. Then we add the redundant data for each bit of the binary. After adding we have a block of packets for each character.

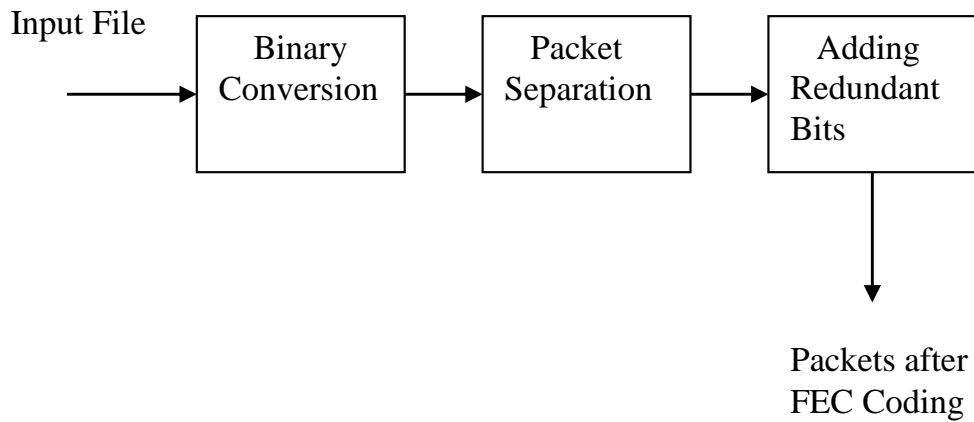


Fig 10  
(FEC Encoder)

## 2. Interleaver:

Interleaving is a way of arranging data in a non-contiguous way in order to increase performance. It is used in data transmission to protect against burst errors. In this module we arrange the data (shuffling) to avoid burst errors which is useful to increase the performance of FEC Encoding.

This module gets the input as blocks of bits from the FEC Encoder. In this module we shuffle the bits inside a single block in order to convert burst errors into random errors. This shuffling process is done for each and every block comes from the FEC Encoder. Then we create a Socket connection to transfer the blocks from Source to the Queue. This connection is created by using the Server Socket and Socket class Available in Java.



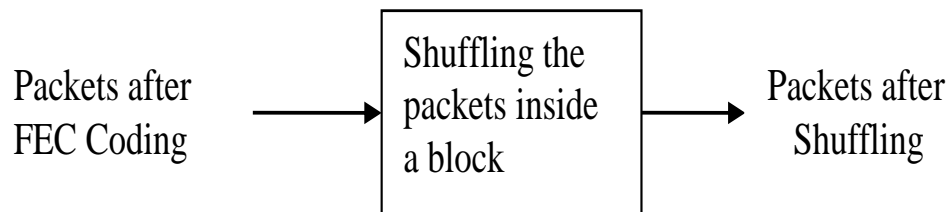


Fig 11

(FEC Interleaver)

The next two steps are followed at the receiver side. When the packets are received at the receiver side then they further undergo de-interleaving and decoding process to get the original file.

### 3. De-Interleaver:

This module receives the blocks of data from the Queue through the socket connection. These blocks are the remaining packets after the loss in the Queue. In this module we arrange the data packets inside a block in the order in which it is before Interleaving. This process of Interleaving and De-Interleaving is done to convert burst errors into random errors. After De-Interleaving the blocks are arranged in the original order. Then the data blocks are sent to the FEC Decoder.

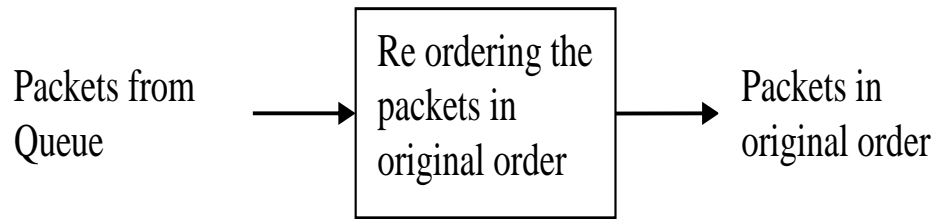


Fig 12

(FEC De-interleaver)

#### 4. FEC Decoder:

This module gets the input from the De-Interleaver. The received packets are processed to remove the original bits from it. Thus we recover the original bits of a character in this module. After retrieving the original bits, we convert it to characters and write it inside a text file.

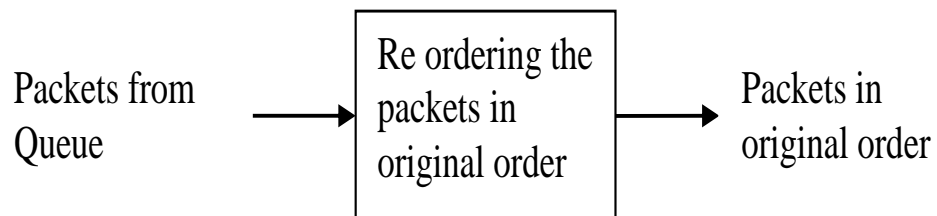
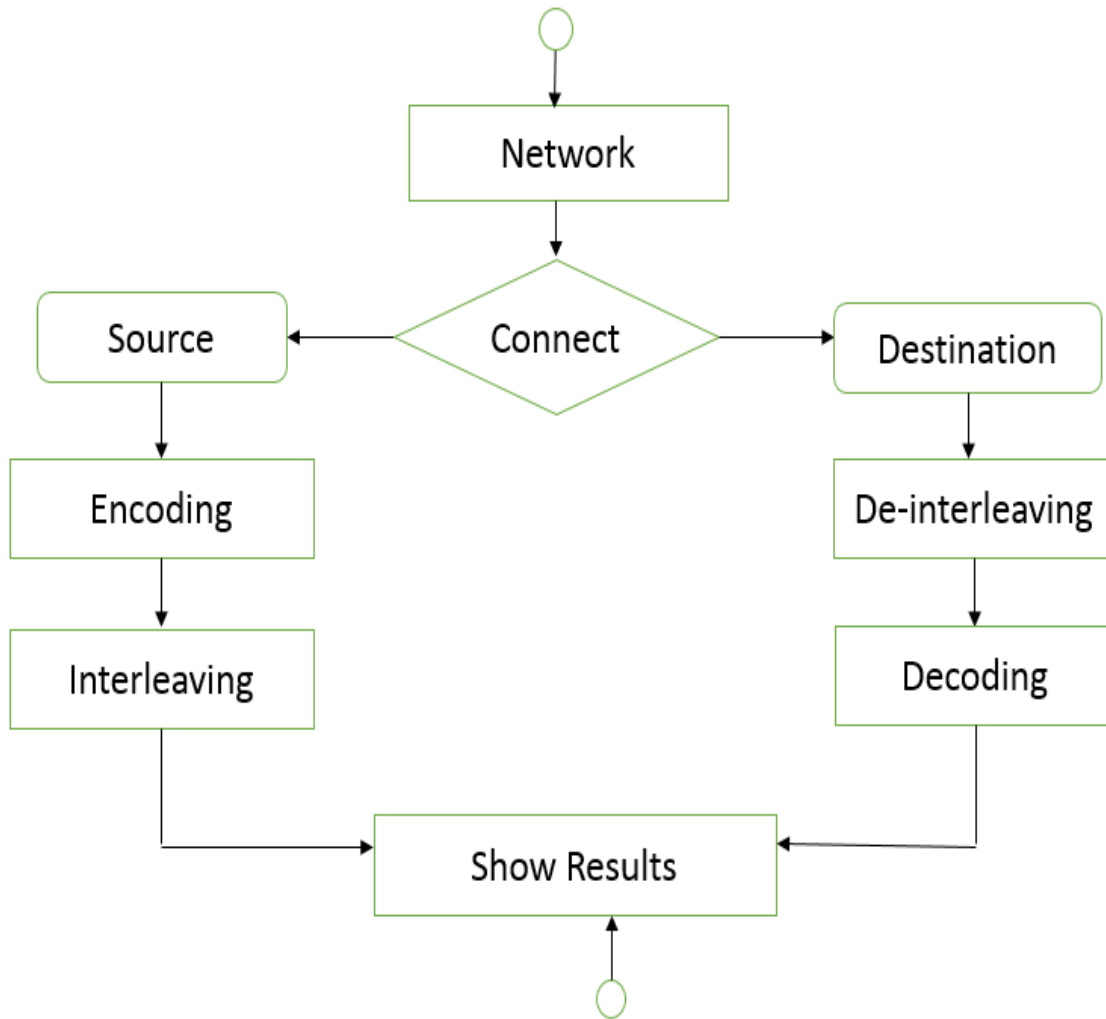


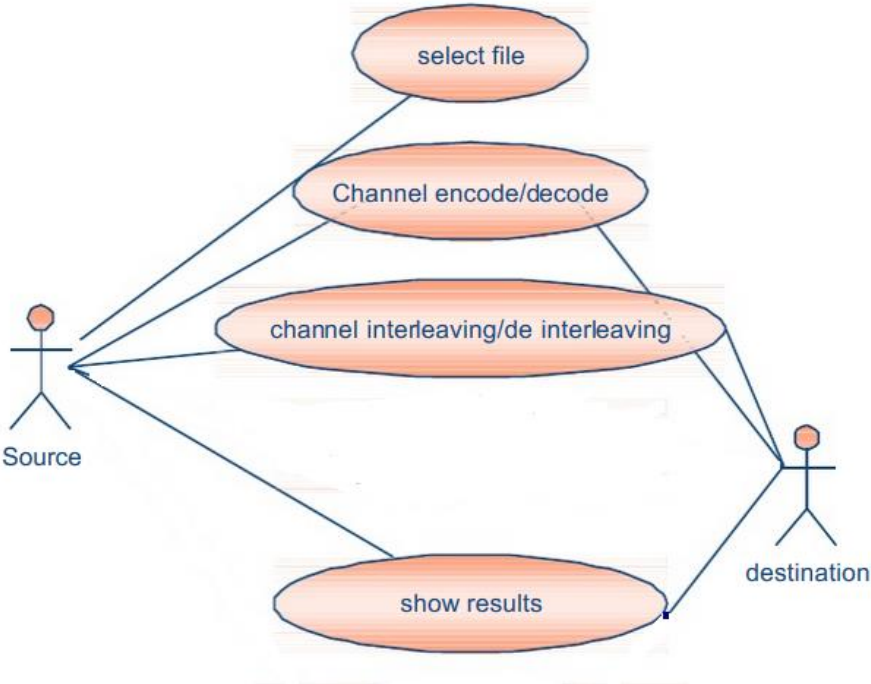
Fig 13

(FEC Decoder)

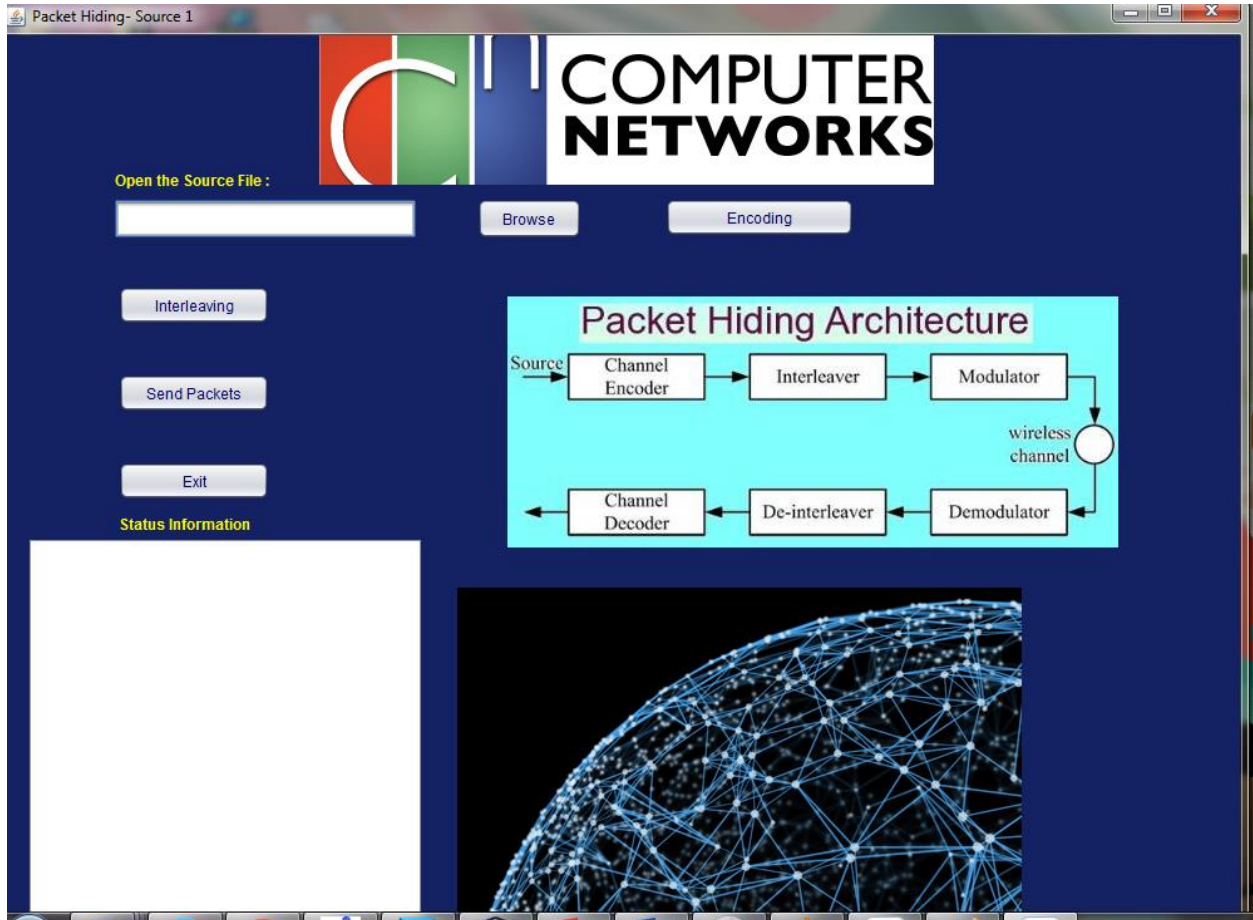
## 4.9 Activity Diagram



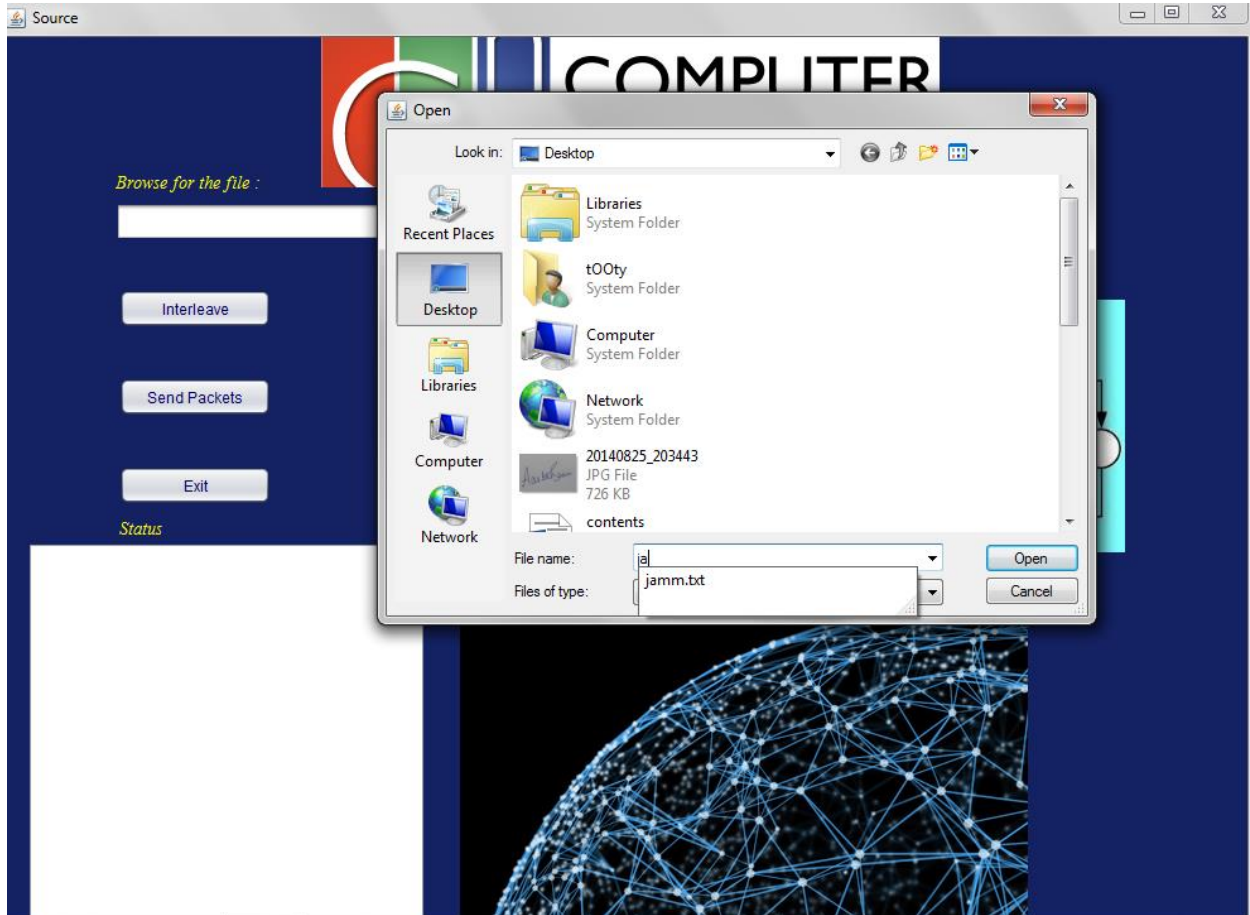
4.10 Use Case Diagram



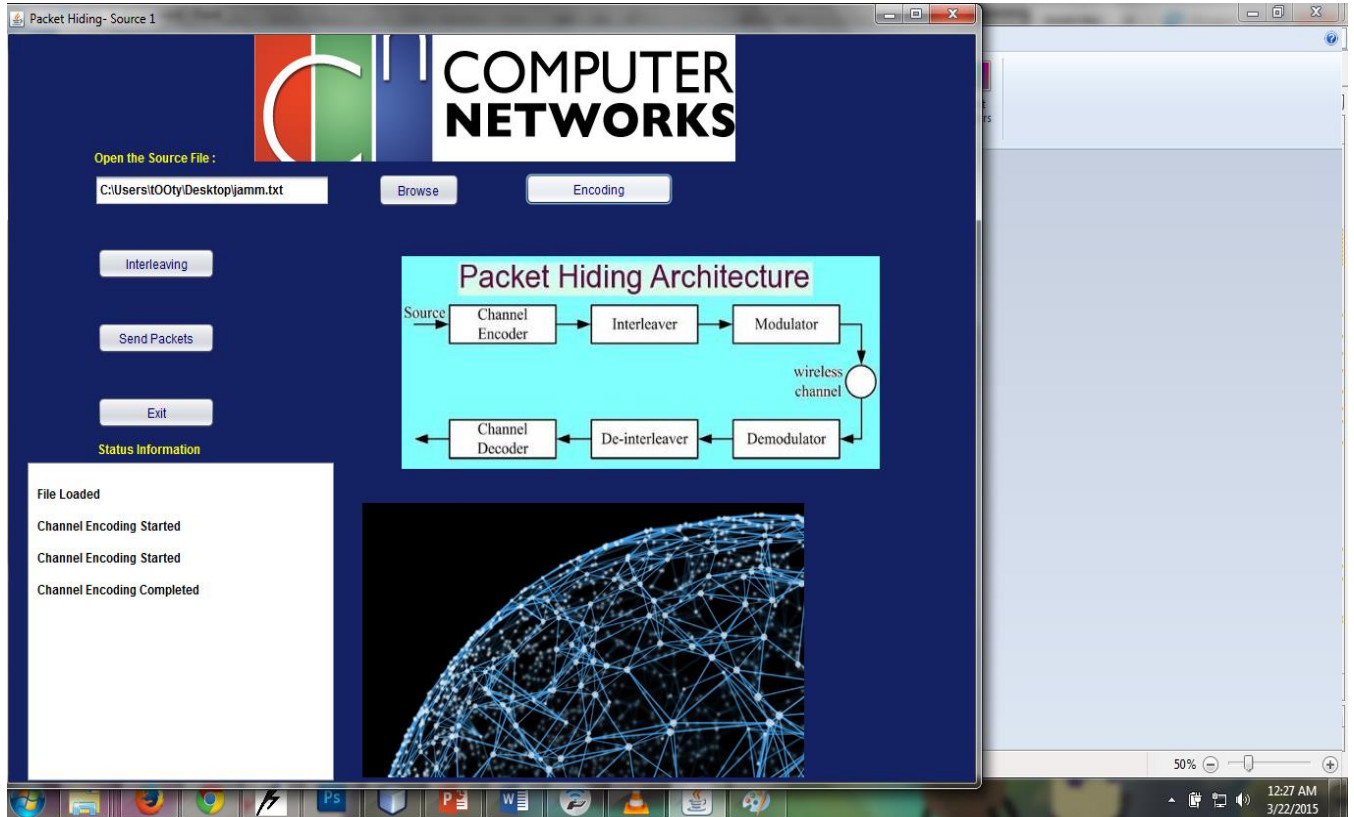
## 5. Screenshots



User interface for sender



Selecting file using browse button



File loaded using browse button and encoding process is completed

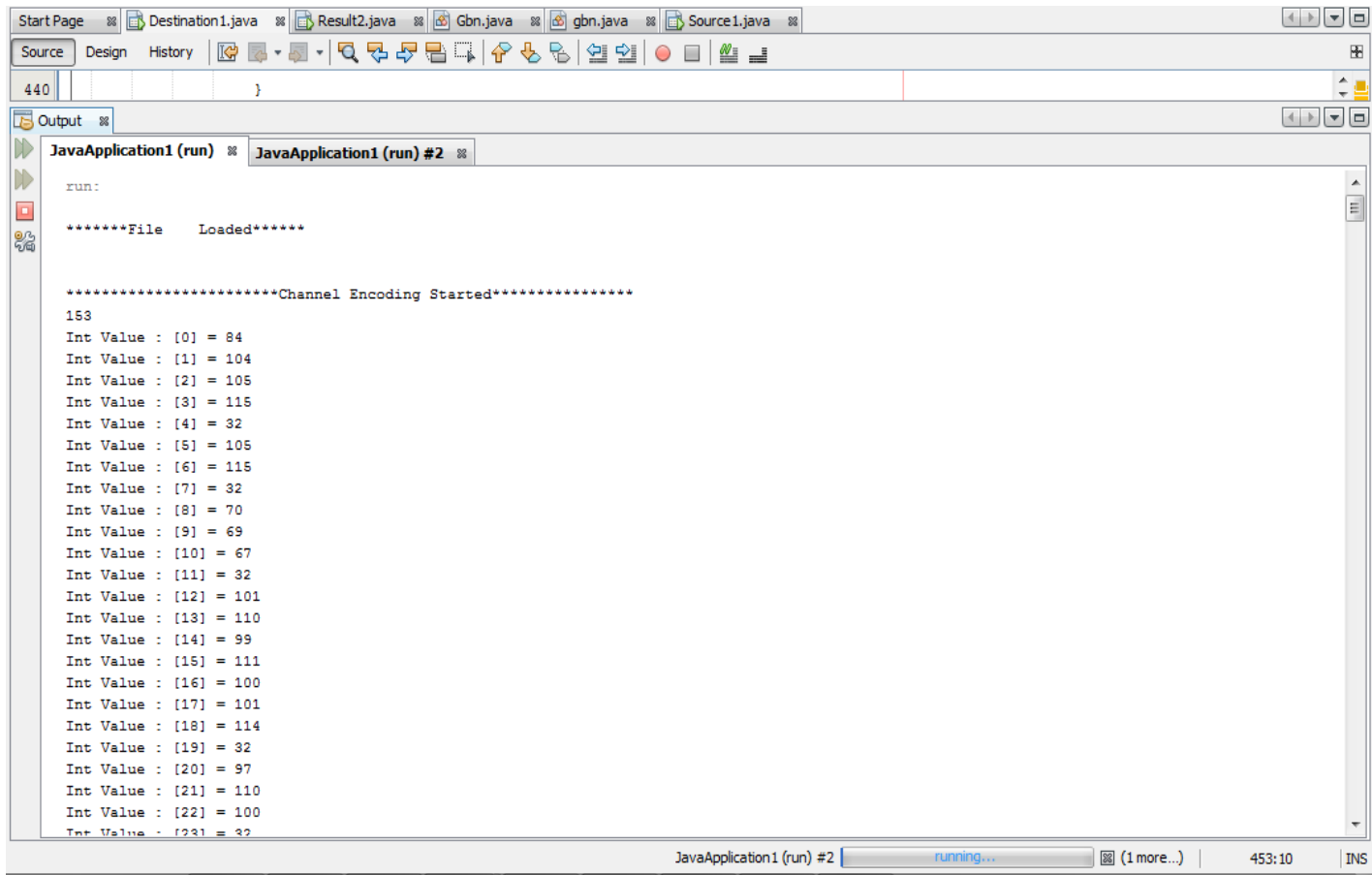
The screenshot displays a software application titled "COMPUTER NETWORKS". The interface includes a file path input field containing "C:\Users\it00ty\Desktop\jamm.txt", with "Browse" and "Encoding" buttons. Below these are buttons for "Interleaving", "Send Packets", and "Exit". A "Status Information" section lists the following steps: "File Loaded", "Channel Encoding Started", "Channel Encoding Completed", "Interleaving Process Started", "Interleaving Process Completed", "Sending Packets to Destination", and "Packets Sent to Destination".

In the center, a diagram titled "Packet Hiding Architecture" illustrates the data flow. It starts with a "Source" pointing to a "Channel Encoder", which connects to an "Interleaver", then to a "Modulator". The "Modulator" is connected to a "wireless channel" (represented by a circle). From the "wireless channel", the signal goes to a "Demodulator", then to a "De-interleaver", and finally to a "Channel Decoder".

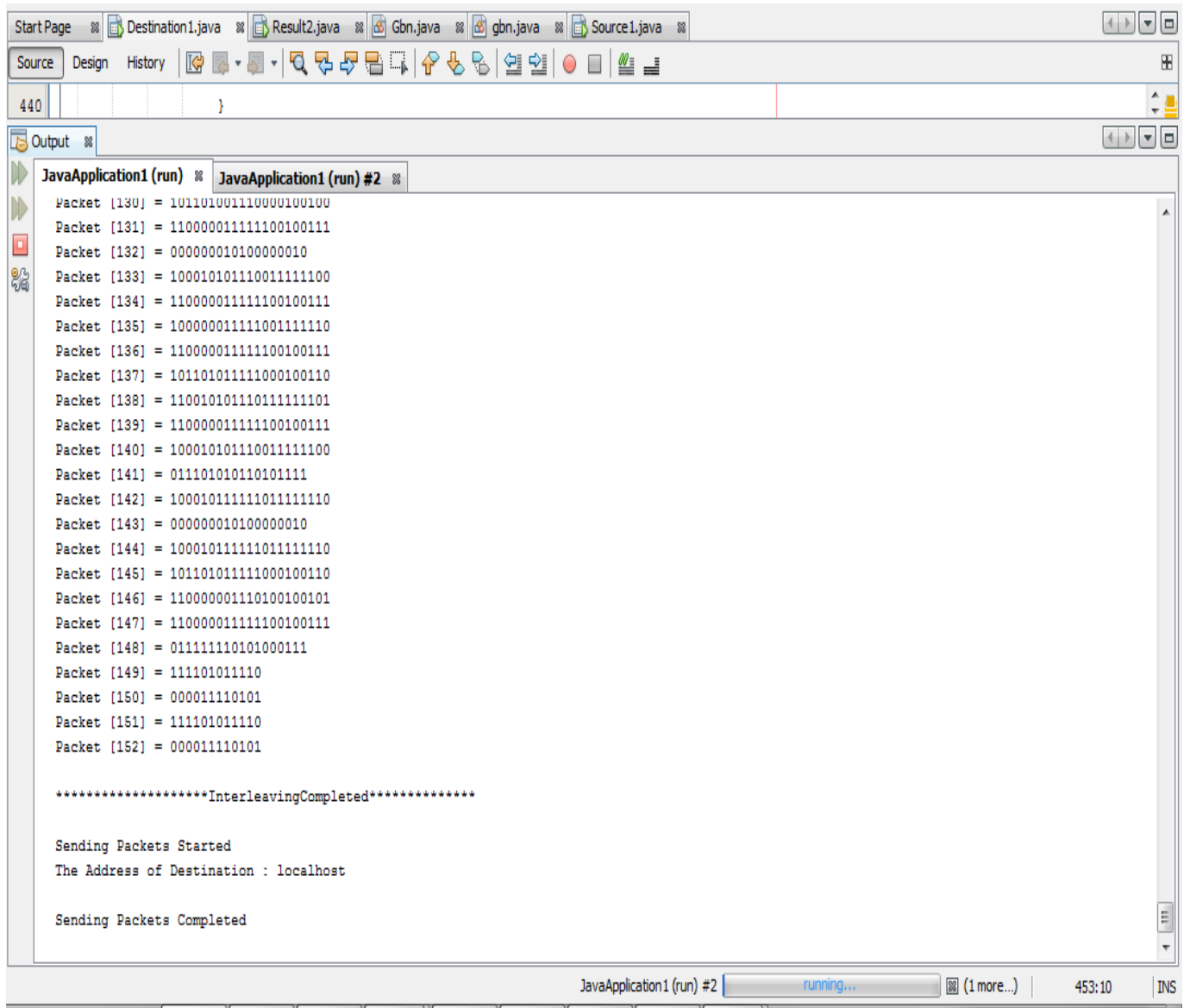
At the bottom right of the interface, there is a graphic of a globe with a network of blue lines and nodes overlaid on it.

Interleaving process is completed and packets are sent to destination






Corresponding output in IDE



Packets are formed from the input file

Destination



# COMPUTER NETWORKS

*Received File Location :*

De-Interleaving

Decoding

Result

Exit

*Status Information*


Packets Recieving Started

Packets Received

### Packet Hiding Architecture


```

graph LR
    Source --> CE[Channel Encoder]
    CE --> I[Interleaver]
    I --> M[Modulator]
    M --> WC((wireless channel))
    WC --> D[Demodulator]
    D --> DI[De-interleaver]
    DI --> CD[Channel Decoder]
  
```



Packets are received at the receiver side

Destination



# COMPUTER NETWORKS

Received File Location :

De-Interleaving

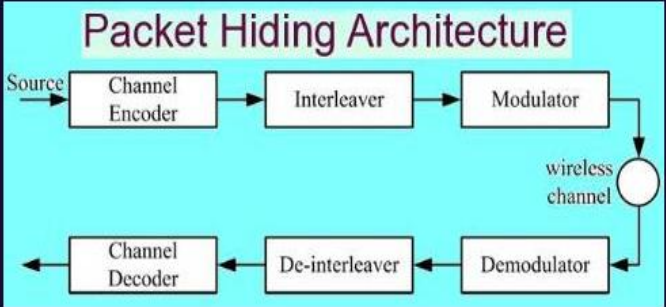

Decoding

Result

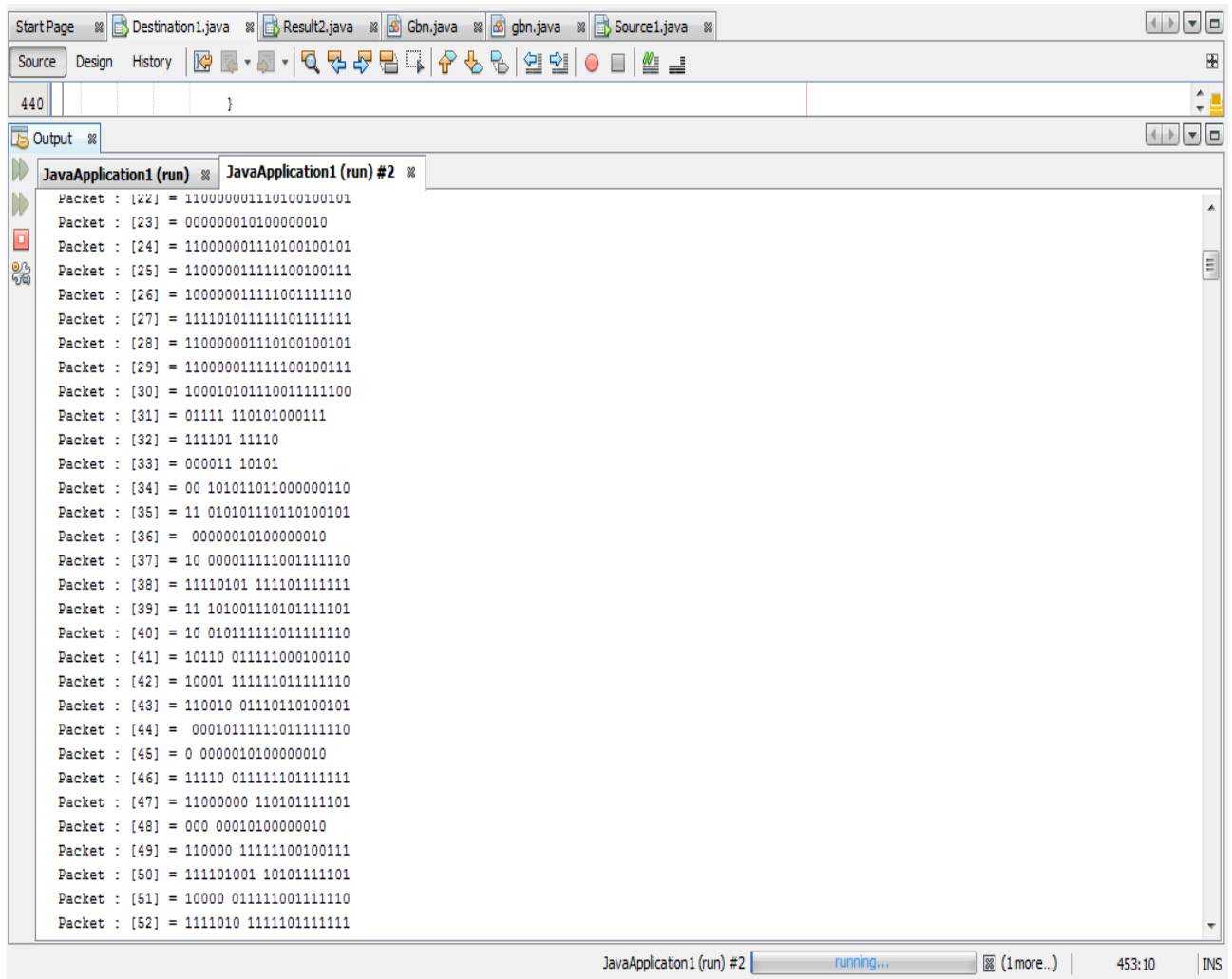
Exit

Status Information

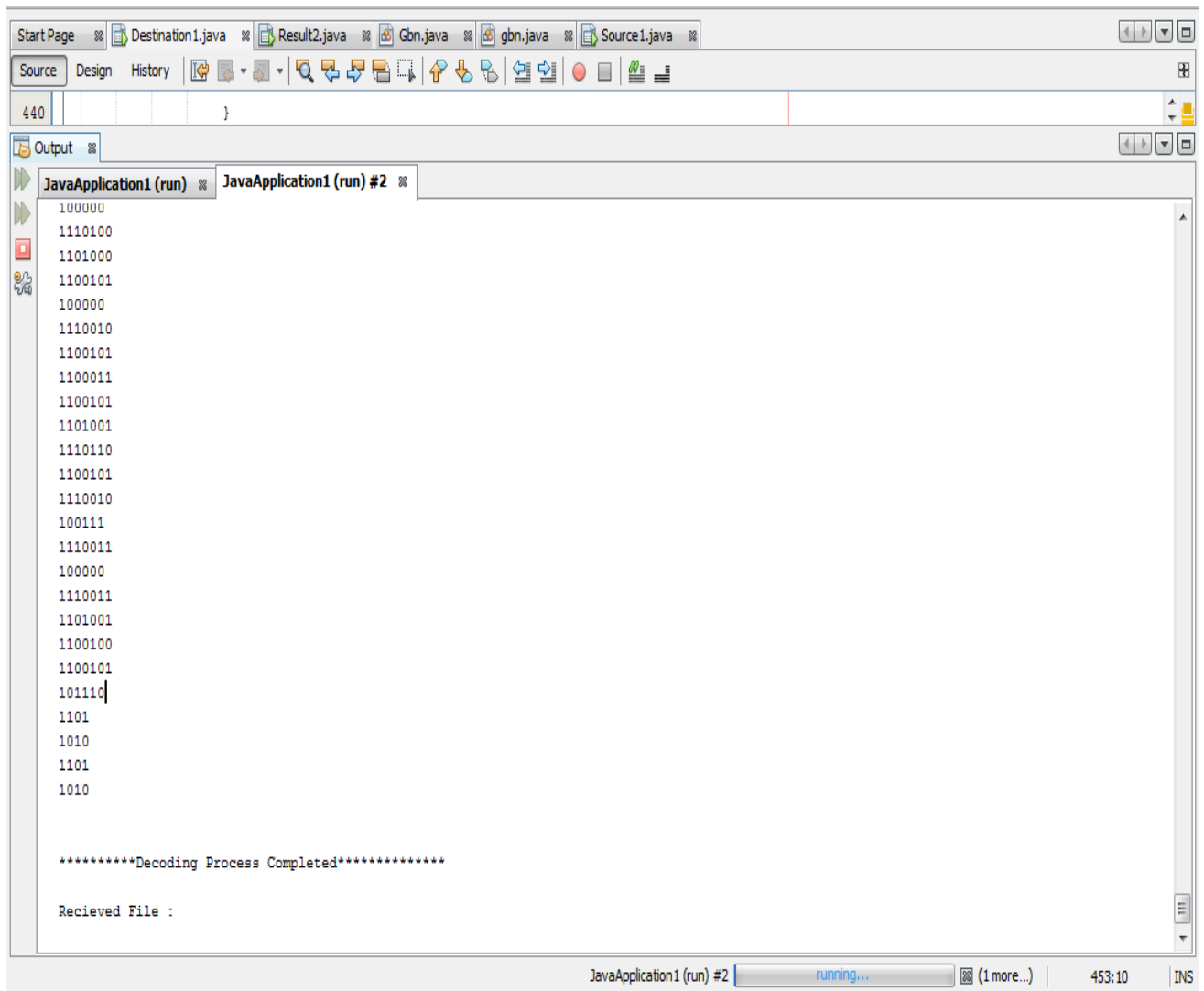
- Packets Recieving Started
- Packets Received
- De-Interleaving Process Started
- De-Interleaving Process Completed
- Channel Decoding Process Started
- Converting Binary to String
- Decoding Process is Completed

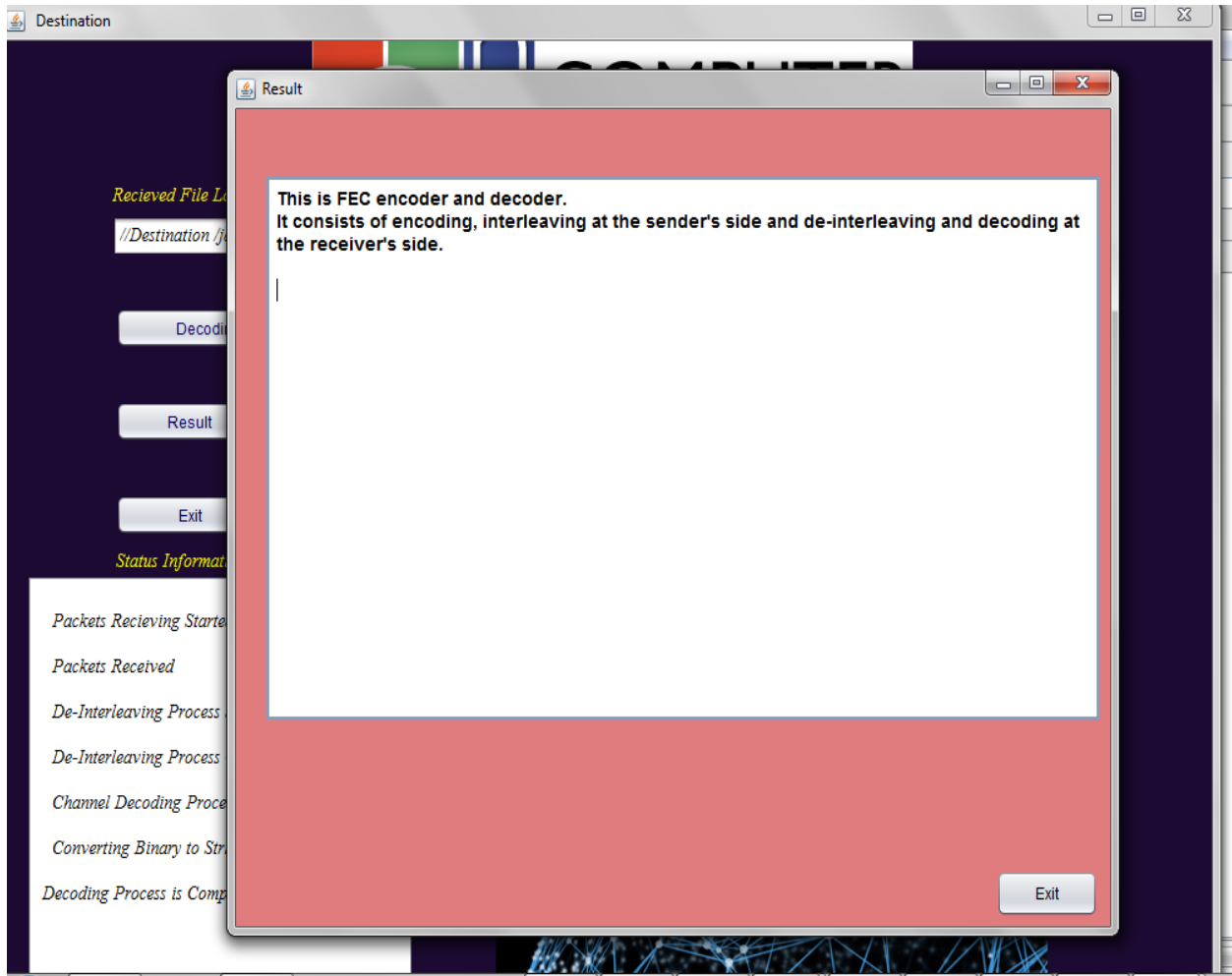
De-interleaving and decoding process completed



Packets received at the receiver's side



Packets are de-interleaved and decoded



The file is displayed when result button is clicked

## 6. Conclusion

OSI is basically an architecture which only gives us an idea how packets transfer over the network during any communication. OSI enhancements are done time to time for developing new technologies. Future implementation in OSI will lead to enhancement in security and many other fields. The development of OSI Standards is a very big challenge, the result of which will impact all future computer communication developments. If standards come too late or are inadequate, interconnection of heterogeneous systems will not be possible or will be very costly. The work collectively achieved so far is very promising, and additional efforts should be expanded to capitalize on these initial results and come up rapidly with the most urgently needed set of standards which will support initial usage of OSI (mainly terminals accessing services and file transfers).

FEC code is a system of adding redundant data, or parity data, to a message, such that it can be recovered by a receiver even when a number of errors (up to the capability of the code being used) were introduced, either during the process of transmission, or on storage. Since the receiver does not have to ask the sender for retransmission of the data, a back-channel is not required in forward error correction, and it is therefore suitable for simplex communication such as broadcasting. Error-correcting codes are frequently used in lower-layer communication, as well as for reliable storage in media.



## 7. Tools and Technologies

### 7.1 JAVA SWINGS

**Swing** is a GUI widget toolkit for Java. It is part of Oracle's Java Foundation Classes (JFC) — an API for providing a graphical user interface (GUI) for Java programs.

Swing was developed to provide a more sophisticated set of GUI components than the earlier Abstract Window Toolkit (AWT). Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. It has more powerful and flexible components than AWT. In addition to familiar components such as buttons, check boxes and labels, Swing provides several advanced components such as tabbed panel, scroll panes, trees, tables, and lists.

### 7.2 NET BEANS IDE 8.0.2

**NetBeans** is an integrated development environment (IDE) for developing primarily with Java, but also with other languages, in particular PHP, C/C++, and HTML5. It is also an application platform framework for Java desktop applications and others. The NetBeans IDE is written in Java and can run on Windows, OS X, Linux, Solaris and other platforms supporting a compatible JVM. The NetBeans Platform allows applications to be developed from a set of modular software components called *modules*. Applications based on the NetBeans Platform (including the NetBeans IDE itself) can be extended by third party developers

## 8. REFERENCES

1. Computer Networks by Andrew S. Tanenbaum. Published by Pearson Education.3<sup>rd</sup> Edition [671 pages ]
2. DATACOMMUNICATIONS AND NETWORKING by Behrouz A. Forouzan.Published by Tata McGraw Hill.4<sup>th</sup> Edition [1171 pages]
3. Gaurav Bora, Saurabh Bora, Shivendra Singh, Sheikh Mohamad Arsalan, “OSI Reference Model: “at International Journal of Computer Trends and Technology (IJCTT) – volume 7 number 4– Jan 2014 [5 pages]
4. Margaret Rouse, “OSI (Open Systems Interconnection)”, in <http://searchnetworking.techtarget.com/definition/OSI>.
5. Depavath Harinath, “OSI Reference Model – A Seven Layered Architecture of OSI Model” at International Journal of Advanced Research in Computer Science and Software Engineering. Volume 3, Issue 8, August 2013. [ 8 pages]
6. Hubert Zimmermann, “OSI Reference Model The ISO Model of Architecture for Open System Interconnection” IEEE transaction on communications, vol.28, issue 4, April 1980.
7. Webopedia ([http://www.webopedia.com/quick\\_ref/OSI\\_Layers.asp](http://www.webopedia.com/quick_ref/OSI_Layers.asp))
8. <http://computer.howstuffworks.com/osi.htm>
9. <http://whatis.techtarget.com/reference/OSI-Reference-Model-illustrated>
10. <http://media.techtarget.com/digitalguide/images/Misc/osi.g>
11. <http://www.escotal.com/osilayer.html>
12. <http://encyclopedia2.thefreedictionary.com/Seven-layer+OSI+model>
13. Wikipedia([http://en.wikipedia.org/wiki/OSI\\_model](http://en.wikipedia.org/wiki/OSI_model))

14. <http://searchnetworking.techtarget.com/definition/OSI>
15. [http://www.mikrozona.hu/Elmelet/uj\\_technologiak/Forward%20Error%20Correction.pdf](http://www.mikrozona.hu/Elmelet/uj_technologiak/Forward%20Error%20Correction.pdf)
16. <http://www.tutorialspoint.com>
17. <http://www.techopedia.com/definition/824/forward-error-correction-fec>
18. [ijcttjournal.org/Volume4/issue-3/IJCTT-V4I3P131.pdf](http://ijcttjournal.org/Volume4/issue-3/IJCTT-V4I3P131.pdf)
19. [Wikipidea-en.wikipedia.org/wiki/OSI\\_model](http://en.wikipedia.org/wiki/OSI_model)
20. [Wikipdia-en.wikipedia.org/wiki/Forward\\_error\\_correction](http://en.wikipedia.org/wiki/Forward_error_correction)
21. <http://whatis.techtarget.com/reference/OSIReferenceModelillustrated>

## 9. Appendices

### Appendix A

#### Code for the source:

##### Encoding

```
for(i=0;i<length;i++) {
fint[i]=(int)f1[i];
System.out.println("Int Value : ["+i+"] = "+fint[i]);
fstring[i] = Integer.toBinaryString(fint[i]); }
for(i=0;i<length;i++) {
System.out.println(fstring[i]);
try{ Thread.sleep(2); }
catch (Exception er)
{
System.out.println("Sleep Disturbed : "+er);
} }
for(i=0;i<length;i++)
{
for(j=0;j<fstring[i].length();j++)
{
sepr[i][j]=Character.toString(fstring[i].charAt(j));
} }
for(i=0;i<length;i++)
{
for(j=0;j<fstring[i].length();j++)
{
System.out.print(sepr[i][j]+" ");
}
System.out.print("\n");
```

```

try{
Thread.sleep(2);
}
catch (Exception er)
{
System.out.println("Sleep Disturbed : "+er);
}
}
for(i=0;i<length;i++)
{
for(j=0;j<fstring[i].length();j++)
{
sepr[i][j]=sepr[i][j]+sepr[i][j]+sepr[i][j];
}
}
for(i=0;i<length;i++)
{
for(j=0;j<fstring[i].length();j++)
{
System.out.print(sepr[i][j]+" ");
}
System.out.print("\n");
try{
Thread.sleep(2);
}
catch (Exception er)
{
System.out.println("Sleep Disturbed : "+er);
}
}
}

```

```
for(i=0;i<length;i++)
{
merge[i]="";
for(j=0;j<fstring[i].length();j++)
{
merge[i]+=sepr[i][j];
}
}
for(i=0;i<length;i++)
{
System.out.println(merge[i]+" ");
try
{
Thread.sleep(2);
}
catch (Exception er)
{
System.out.println("Sleep Disturbed : "+er);
}
}
```

## Interleaving

```
for(i=0;i<length;i++)
{
for(j=0;j<merge[i].length();j++)
{
pack[i][j]=merge[i].charAt(j);
}
}
for(i=0;i<length;i++)
{
for(j=0;j<merge[i].length();j++)
{
System.out.print(pack[i][j]+" ");
}
System.out.print("\n");
try{
Thread.sleep(2);
}
catch (Exception er)
{
System.out.println("Sleep Disturbed : "+er);
}
}
for(i=0;i<length;i++)
{
for(j=0;j<1;j++)
{
if((merge[i].length())==21)
{
```

```

shuff[i][0]=pack[i][5];
shuff[i][1]=pack[i][12];
shuff[i][2]=pack[i][11];
shuff[i][3]=pack[i][9];
shuff[i][4]=pack[i][6];
shuff[i][5]=pack[i][10];
shuff[i][6]=pack[i][8];
shuff[i][7]=pack[i][20];
shuff[i][8]=pack[i][0];
shuff[i][9]=pack[i][4];
shuff[i][10]=pack[i][1];
shuff[i][11]=pack[i][19];
shuff[i][12]=pack[i][13];
shuff[i][13]=pack[i][7];
shuff[i][14]=pack[i][16];
shuff[i][15]=pack[i][3];
shuff[i][16]=pack[i][17];
shuff[i][17]=pack[i][15];
shuff[i][18]=pack[i][2];
shuff[i][19]=pack[i][18];
shuff[i][20]=pack[i][14];
}
else if((merge[i].length()==18)
{
shuff[i][0]=pack[i][5];
shuff[i][1]=pack[i][12];
shuff[i][2]=pack[i][11];
shuff[i][3]=pack[i][9];
shuff[i][4]=pack[i][6];
shuff[i][5]=pack[i][10];

```



```
shuff[i][6]=pack[i][8];
shuff[i][7]=pack[i][0];
shuff[i][8]=pack[i][4];
shuff[i][9]=pack[i][1];
shuff[i][10]=pack[i][15];
shuff[i][11]=pack[i][7];
shuff[i][12]=pack[i][16];
shuff[i][13]=pack[i][3];
shuff[i][14]=pack[i][17];
shuff[i][15]=pack[i][13];
shuff[i][16]=pack[i][2];
shuff[i][17]=pack[i][14];
}
else
{
shuff[i][0]=pack[i][5];
shuff[i][1]=pack[i][11];
shuff[i][2]=pack[i][10];
shuff[i][3]=pack[i][9];
shuff[i][4]=pack[i][6];
shuff[i][5]=pack[i][2];
shuff[i][6]=pack[i][8];
shuff[i][7]=pack[i][0];
shuff[i][8]=pack[i][4];
shuff[i][9]=pack[i][1];
shuff[i][10]=pack[i][3];
shuff[i][11]=pack[i][7];
}
}
}
```

```

if(length<=50){
    l=(int)(Math.random()*3);
for(int a=0;a<=3;a+=1)
{
    j=(int)(Math.random()*10);
shuff[a][j]='\0';
}
}
else if(length>=51&&length<=210)
{
l=(int)(Math.random()*4);
for(int a=31;a<=100;a+=1)
{
    j=(int)(Math.random()*10);
shuff[a][j]='\0';
}
}
else if(length>=251&&length<=500)
{
l=(int)(Math.random()*4);
for(int a=110;a<=192;a+=1)
{
    j=(int)(Math.random()*10);
shuff[a][j]='\0';
}
}
else if(length>=501&&length<=750)
{
l=(int)(Math.random()*4);

```

```

for(int a=440;a<=501;a+=1)
{
j=(int)(Math.random()*10);
shuff[a][j]='\0';
}
}
else if(length>=751&&length<=1000)
{
l=(int)(Math.random()*4);
for(int a=652;a<=751;a+=1)
{
j=(int)(Math.random()*10);
shuff[a][j]='\0';
}
}
else
{
l=(int)(Math.random()*4);
for(int a=500;a<=610;a+=1)
{
j=(int)(Math.random()*10);
shuff[a][j]='\0';
}
}
System.out.println("\n\n");
for(i=0;i<length;i++)
{
for(j=0;j<merge[i].length();j++)
{
System.out.print(shuff[i][j]+" ");
}
}
}

```

```

}
System.out.print("\n");
try
{
Thread.sleep(2);
}
catch (Exception er)
{
System.out.println("Sleep Disturbed : "+er);
}
}
for(i=0;i<length;i++)
{
fr[i]="";
for(j=0;j<merge[i].length();j++)
{
fr[i]+=shuff[i][j];
}
}
for(i=0;i<length;i++)
{
System.out.println("Packet ["+i+"] = "+fr[i]);
try
{
Thread.sleep(2);
}
catch (Exception er)
{
System.out.println("Sleep Disturbed : "+er); } }

```

## Appendix B

### Code for Destination

#### De-interleaving

```
for(i=0;i<length;i++)
    {
        for(j=0;j<fr[i].length();j++)
            {
                pack[i][j]=fr[i].charAt(j);
            }
    }
for(i=0;i<length;i++)
    {
        for(j=0;j<fr[i].length();j++)
            {
                System.out.print(pack[i][j]+" ");
            }
        System.out.print("\n");
        try
            {
                Thread.sleep(2);
            }
        catch (Exception er)
            {
                System.out.println("Sleep Disturbed : "+er);
            }
    }
for(i=0;i<length;i++)
    {
```

```

for(j=0;j<1;j++)
{
    if((fr[i].length())==21)
    {
        bklen=fr[i].length();
        reshuff[i][0]=pack[i][8];
        reshuff[i][1]=pack[i][10];
        reshuff[i][2]=pack[i][18];
        reshuff[i][3]=pack[i][15];
        reshuff[i][4]=pack[i][9];
        reshuff[i][5]=pack[i][0];
        reshuff[i][6]=pack[i][4];
        reshuff[i][7]=pack[i][13];
        reshuff[i][8]=pack[i][6];
        reshuff[i][9]=pack[i][3];
        reshuff[i][10]=pack[i][5];
        reshuff[i][11]=pack[i][2];
        reshuff[i][12]=pack[i][1];
        reshuff[i][13]=pack[i][12];
        reshuff[i][14]=pack[i][20];
        reshuff[i][15]=pack[i][17];
        reshuff[i][16]=pack[i][14];
        reshuff[i][17]=pack[i][16];
        reshuff[i][18]=pack[i][19];
        reshuff[i][19]=pack[i][11];
        reshuff[i][20]=pack[i][7];
    }
    else if((fr[i].length())==18)
    {
        reshuff[i][0]=pack[i][7];
    }
}

```

```

reshuff[i][1]=pack[i][9];
reshuff[i][2]=pack[i][16];
reshuff[i][3]=pack[i][13];
reshuff[i][4]=pack[i][8];
reshuff[i][5]=pack[i][0];
reshuff[i][6]=pack[i][4];
reshuff[i][7]=pack[i][11];
reshuff[i][8]=pack[i][6];
reshuff[i][9]=pack[i][3];
reshuff[i][10]=pack[i][5];
reshuff[i][11]=pack[i][2];
reshuff[i][12]=pack[i][1];
reshuff[i][13]=pack[i][15];
reshuff[i][14]=pack[i][17];
reshuff[i][15]=pack[i][10];
reshuff[i][16]=pack[i][12];
reshuff[i][17]=pack[i][14];
}
else
{
reshuff[i][0]=pack[i][7];
reshuff[i][1]=pack[i][9];
reshuff[i][2]=pack[i][5];
reshuff[i][3]=pack[i][10];
reshuff[i][4]=pack[i][8];
reshuff[i][5]=pack[i][0];
reshuff[i][6]=pack[i][4];
reshuff[i][7]=pack[i][11];
reshuff[i][8]=pack[i][6];
reshuff[i][9]=pack[i][3];

```

```

        reshuff[i][10]=pack[i][2];
        reshuff[i][11]=pack[i][1];
    }
}
for(i=0;i<length;i++)
{
    for(j=0;j<fr[i].length();j++)
    {
        System.out.print(reshuff[i][j]+" ");
    }
    System.out.print("\n");
    try
    {
        Thread.sleep(2);
    }
    catch (Exception er)
    {
        System.out.println("Sleep Disturbed : "+er);
    }
}

```

## Decoder

```

for(i=0;i<length;i++)
{
    rec[i]="";
    for(j=0;j<fr[i].length();j++)
    {
        if(reshuff[i][j+2]!='0')
        {

```



```

        rec[i]+="" +reshuff[i][j+2];
        j+=2;
    }
    else if(reshuff[i][j+1]!='\0')
    {
        rec[i]+="" +reshuff[i][j+1];
        j+=2;
    }
    else if(reshuff[i][j]!='\0')
    {
        rec[i]+="" +reshuff[i][j];
        j+=2;
    }
    }
}
for(i=0;i<length;i++)
{
    System.out.println(rec[i]);
    try
    {
        Thread.sleep(2);
    }
    catch (Exception e)
    {
        System.out.println("Sleep Disturbed : "+e);
    }
}

jTextArea1.append("\n\n Binary to String Conversion");
for(i=0;i<length;i++)

```

```
{
    l2[i]= Integer.parseInt(rec[i],2);
    f1[i]=(char)l2[i];
}
try
{
    Fien="";
    for(i=0;i<length;i++)
    {
Fien+=Character.toString(f1[i]);
    }
}
catch (Exception e)
{
    e.printStackTrace();
}
}
```