ABSTRACT

ATM ETHERNET BRIDGE is a software which connects WAN'S & translate the data format so that LAN'S can communicate over the WAN. Basically LAN'S can sent / Receive data only in Ethernet frame formats, and LAN'S work with Date link layer, where as WAN'S work in Network layer of the OSI reference model and WAN'S can sent / receive data in cell / packet format. WAN device which is used to connect is ATM – Switch.

As per the basic rule in networking, LAN'S can only sent / receive data in frame formats. Assume that you have 2 separate LAN'S which are connected over the WAN. You can establish communications through ATM switch.

But, over the WAN, data can be sent / received only packet / cell format. This project is going to be software switch which is used to connect to both LAN"S and it main purpose is to convert frames to cell & cell which be sent to the destination again cell to frame conversion happens at to the other end. At last is sent to the destination. This project demonstrates the integration of two heterogeneous networks and develops a software bridge between the ATM and the Ethernet. This acts as a translator for converting the information cells to Ethernet packets and vice-versa. ATM is a versatile protocol used for a converged data transfer through a single channel, that provides much high reliable and faster transfer. Since the data is transferred in digital form.

The actual information to be sent is stuffed into a cell which has got a standard size of 53 bytes out of this, 5 bytes are alloted for the overload and the remaining is allocated for payload(actual data 48 bytes).

This project is going to implement a type of interface two dissimilar networks. This is going to act like a software bridge that interconnects and transfers data packets between two heterogeneous networks, one type of network is an ATM and the other is Ethernet. The packet structure of both differs in their architecture and length. This software bridge is going to modify the packets as they go through the bridge, either from an ATM network to Ethernet or vice-versa.

The data from the client are captured in the form of packets which are then converted into fixed size ATM cells (53 Bytes) payload and header in the server1. The cells are then converted into frames in server2 and then to client2 as per the ATM routing concept. ‘C’ – socket programming in Linux based servers are used to build this software.

TABLE OF CONTENTS

INTRODUCTION

NETWORKING:

A network is an interconnection of two or more devices in order to enable transfer of data or information from one place to another.

The concept of networking envisages the interconnection of two or more computers by means of cables and information can be sent and received between these computers.

The main purpose of setting up a network is to share information. Exchange of information over a network of computers is easier and faster. In addition, resources like disk space, printers, modem, scanner, etc. can be shared among the computers in a network.

ASYCHRONOUS TRANSFER MODE:

ATM technology involves the process of sending /receiving data via asychronous multiplex transmission, in which data is transmitted sporadically (only when necessary) rather than in a synchronous manner which requires communications between the transmitting and receiving sides.

ETHERNET:

Ethernet is the name given to a popular packet-switched LAN technology invented at Xerox PARC in the early 1970's. Xerox corporation Intel Corporation, and Digital Equipment Corporation standardized Ethernet in 1978; IEEE released a compatible version of the standard using the number 802.3. Ethernet has become a

popular LAN technology, most medium or laarge corporations use Ethernets. Because Ethernet is so popular,many variants exist.

Each Ethernet cable is about 1/2 inch in diameter and upto 500 meters long. A resistor is between the center wire and shield at each end to prevent reflection of electrical signals.

PROBLEM DEFINITION:

The interface between two different networks, ATM and ETHERNET is economical. Here the packet structure of both the networks differs in their architecture. Hence the packets are modified accordingly as they go through the bridge.

ATM ETHERNET bridge will enhance industry specifications that help deliver the combined benefits of ATM and Ethernat technology for products that deliver the quality, scalability, price and performance demanded by enterprise customers. This project has the ability to incorporate both technologies and the ability to evolve with a company's business needs.

This project is divided into three modules known as packet capturing which performs the operation of servicing the requests in the data link layer. The packets from the network layer are forwarded to the DATA LINK LAYER with respect to the IP address specified in the packets as the gateway IP address. If the IP address of the packets matches with the gateway IP address of the DLL then the packets are forwarded to the DLL.

The DLL obtains the frames and stores them in the memory where the packets are routed to their corresponding destinations. When the service is requested in the network layer, the packets obtained in this layer are checked for their IP address.

If this IP address belongs to the smae network layer then it is forwarded to its destination, since this process is very complicated the process of servicing the request is being done in the data link layer.

The second module implemented in this project is cell formation where the captured packets are converted into fixed size ATM cells (53 Bytes) payload and header. Here Segmentation & Reassembly concept is implemented in order to segment the captured frames into ATM cells for transmission and to reassemble the frames from ATM cells on reception. The third module is known as ATM routing which performs the operation of routing the packets from the server to the desnied client according to the destination IP address. Here in the project PVC connection is established using the socket where the two sockets operate as server and client.

NEED FOR THE PROJECT:

- Ethernet is the most widely used Local area network technology to interconnect multiple PCs in various applications, such as: sharing files, sharing peripheral and internet accesses.
- QoS is not as important for conventional system, because they are used for primary data transfer.
- Asynchronous transfer mode (ATM) on the other hand is widely adopted for the wide area network (WAN) interconnections. ATM can handle multiple types of traffic that have quite different performance requirements on the communication channels.
- Quality of service (QoS) is an important performance measure fir WAN based system, because of the scarcity of bandwidths.
- With increasingly tight integration of remote systems, a uniform QoS framework for both LAN and WAN transmission is important.

GOALS OF THE PROJECT:

- Cost effective
- Efficient to support cross network transport
- Framework for effective synthesis of bridging and routing with ATM, in an environment of diverse network technologies.
- Match the bandwidth differences between the two transmission technologies .Mapping of QoS between ATM & Ethernet packets.
- To increase speed
- Less hardware

LITERATURE SURVEY

SOFTWARE REQUIREMENTS SPECIFICATION:

Software requirement specification records the outcome of the software requirements definition activity. The software requirement specification is a technical specification of requirements for the software product.

The software requirement specification is based on the system definition. High level requirements specified during initial planning elaborated and make more specific in order to characterize the features that the software product will incorporate.

DESIRABLE PROPERTIES:

There are number of properties that a software requirements specification should possess. In particular, a requirement document should be:

- Correct
- Complete
- Consistent
- Unambiguous
- Functional
- Verification
- Traceable
- Easily changed

An incorrect or incomplete set of requirements can result in a software product that satifies its requirements but does not satisfy customer needs. An inconsistent specification states contradictory requirements in different parts of the document ,

while an ambiguous requiredment is subject to different interpretations by different people.

Software requirement shouldbe funtional in nature ; i.e.,that should describe what is required without implying how the system will meet its requireds.

Requirements must be verifiable from two points of view;it must be possible to verify that the requirements satisfy the customer needs , and it must be possible to verify that the subsequent works satisfy the requirements..

Finally the requirements should be indexed , segmented and cross referenced to permit easy use and easy modification. Ideally, every software requirement should be traceable. to specify customer statements and to specify statements in the system definition.

FORMAT OF A SOFTWARE REUIREMENT SPECIFICATION:

- Product overviewand summary
- Development, Operating and Maintenance Environments
- External interfaces and Data flow
- Exception handling
- Early subsets and implementation and Enhancements

PRODUCT OVERVIEW AND SUMMARY:

This project is going to implement a type of interface between two dissimilar networks. This is going to act as a software bridge that interconnets and transfers data packets between two heterogeneous networks. One type of network is ATM and the other is ETHERNET. The packet structure of both differs in their

architecture and length. So this software bridge is going to modify the packets as they go through the bridge, either from an ATM network to ETHERNET or vice versa.

ATM is a versatile protocol used for a converged data transfer through a single channel that provides much high reliable and faster data transfer. The actual information to be sent is stuffed into a cell which has got a standard size of 53 bytes (bytes-header and 48 bytes-payload).

DEVELOPMENT, OPERATING AND MAINTENANCE ENVIRONMENT:

LINUX:

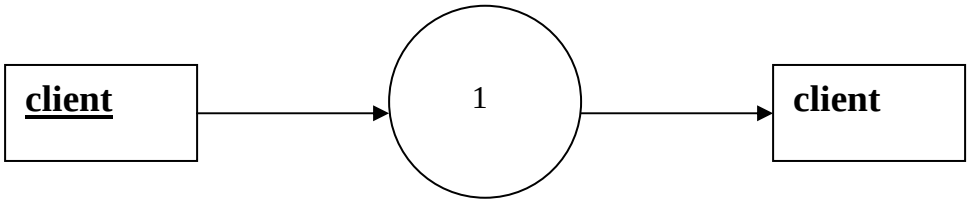
Originally Linux was written to run in Intel 80386 and 80486 processors, but nowadays it has been ported to several other architectures. The development of the Linux system was started by Linus Torvarlds, and currently software for the operating system kernal is being developed around the world by several individuals. Linux is mostly compatible with a number of UNIX standards on the source code level, including IEEE POSIX.1, System V, and BSD feature. The operating system kernal is nowadays written mostly in C, some parts are written using assembly language.

MAIN FEATURES:

- Multitasking
- Multi-user accessibility
- Executable loaded on demand
- Memeory pagination
- Dynamic disk chche

- Shared libraries
- Several executable file formats
- Several file system formats

EXTERNAL INTERFACE AND DATA FLOW:



.ATM ETHERNET BRIDGE

EXCEPTION HANDLING:

(Actions to be taken)

- Packet capturing
- Cell formation
- Sending the cells

- Receiving the cells
- Frame formation
- Frame forwarding

EARLY SUBSETS AND IMPLEMENTATION PRIORITIES:

The interface between two different networks ATM & ETHERNET existing in VLSI & EMBEDDED Technologies involves high cost. The concept of the system already existing in the hardware section is being implemented in the present system.

FORESEEABLE MODIFICATIONS AND ENHANCEMENTS:

The software bridge which is been developed is economical. Here the packets which differ in their architecture are modified as they go through the bridge. This project can be implemented in layers AALI-AAL4. This project can be implemented to support protocols IPX/SPX, APPLE TALK, etc.,

LANGUAGES AND PROCESSORS FOR REQUIREMENTS

'C' seems a strange name for programming language. But this strange sounding language is one of the most popular computer languages today. C was an offspring of the 'Basic Combined Programming Language'(BCPL) called B, developed in the 1960's at Cambridge University. B language was modified by Dennis Ritchie and was implemented at Bell Laboratories in 1972. The new language was named C. Since it was developed along with the UNIX operating system, it is strongly associated with UNIX. This operating system, which was also developed at Bell Laboratories, was coded almost entirely in C.

For many years, C was used mainly in academic enviroments,but eventually with the release of C compilers for commercial use and the increasing popularity of UNIX, it began to gain widespread support among computer professional. Today, C DOS is a dominant operating systems for microcomputers, it is natural that C has begun to influence the is running under a number of operating systems including MS-microcomputer community at large.

IMPORTANCE OF C:

The increasing popularity of C is probably due to its many desirable qualities. It is robust language whose rich set of built-in functions and operators can be used to write any complex program. The C compiler combines the capabilities of an assembly language with the features of a high-level language and therefore it is well suited for writing both system software and business packages. In fact, many of the C compilers available in the market are written in C.

Programs written in C are efficient and fast. This is due to its variety of daata types and powerful operators. It is many times faster than BASIC. There are only 32 keywords and its strength lies in its built-in functions. Several standard functions are available which can be used for developing programs.

C is highly portable. This means that C programs written for one computer can be run on another with little or no modification. Portability is important if we plan to use a new computer with a different operating system.

C language is well suited for structured programming, thus requiring the user to think of a problem in terms of function modules or blocks. A proper collection of these modules would make a complete program. This modular structure makes program debugging, testing and maintenance easier.

Another important feature of C is its ability to extend itself. A 'C' program is basically a collection of function that are supported by the C library. We can continuously add our own funtions to the C library. With the availability of a large number of funcitons, the programming task becomes simple.

SYSTEM SPECIFICATION

HARDWARE REQUIREMENTS:

- NIC CARDS
- 64 MB RAM
- ANY HIGH POWER PROCESSOR

SOFTWARE REQUIREMENTS:

- LINUX
- NETWORK PROGRAMMING IN 'C'

SYSTEM ALALYSIS:

EXISTING SYSTEM:

The integration of two dissimilar networks ATM & ETHERNET has been implemented using VLSI & EMBEDDED Technologies. The cost involved in implementing this is high and it involves more hardware.

PROPOSED SYSTEM:

This project develops a software bridge between ATM & ETHERNET taking into consideration the cost factor involved in hardware implementation. Framework for effective synthesis of bridging and routing with ATM, in an environment of diverse network technologies is possible with this project.

SCOPE :

- ATM ETHERNET bridge will enhance industry specifications that help deliver the combined benefits of ATM and Ethernet technology for products that deliver the quality, scalability, price and performance demanded by enterprise customers.

- This project has the ability to incorporate both technologies and the ability to evolve with a company's business needs.

- Utilizing this technique, data is transmitted sporadically (only when necessary) rather than in a synchronous manner.

- The application of ATM enables to improve overhead, reducing line use to only the time of actual data transfer and allows to transfer large quantities of data efficiency.

END USERS:

End users are those who makes use of the software which is been developed by the Project developers. E.g. the clients who need communication between two different networks.

ARCHITECTURE AND SYSTEM STUDY

INTRODUCTION:

According to Webster, the process of design involves "conceiving and planning out in the mind" and "making a drawing, pattern, or sketch of". In software design, there are three distinct types of activities: external design, architectural design, and detailed design.

External design of software involves conceiving , planning out, and specifying the externally observable characteristics of a software product. These characteristics include user displays and report formats, external data sources and data sinks, and the functional characteristics, performance requirements, and high level process structure for the product.

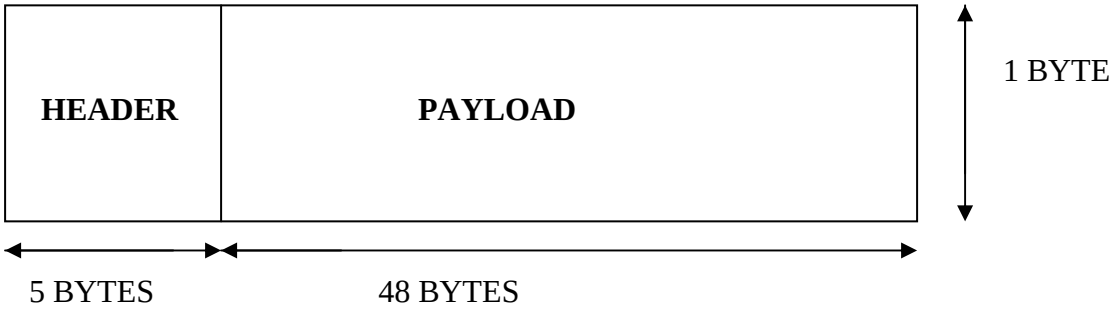
External design begins during the analysis phase and continues into the design phase. In practice, it is not possible to perform requirements definition without doing some preliminary design. Requirements definition is concerned with specifying the external, functional, and performacne requirements for a system.

Internal design involves conceiving , planning out, and specifying the internal structure and processing details of the software product. The goals of internal design are to specify internal structure and processing details, to record design decisions and indicate why certain alternatives and trade-offs were chosen, to elaborate the test plan, and to provide a blueprint for implementation, testing, and maintenance activities. The work products of internal design include a specification of architectural structure, the details of algorithms and data structures, and the test plan.

Architectural design is concerned with refining the conceptual view of the system, identifying internal processing functions, decomposing high-level functions into sub funcitons, defining internal data streams and data stores, and establishing

relationships and interconnections among functions, data streams, and data stores. Issues of concern during detailed design include spefication of algorithms that implement the functions, concrete data structures that implement the data stores, the actual interconnctions among functions and data structures, and the packaging scheme for the system. The test plan describes the objectives of testing, the test completion criteria, the integration plan, particular tools and techniques to be used, and the actual test cases and expected results. Functional tests and performance tests are developed during requirements analysis and are refined during the design phase. Tests that examine the interal structure of the software product and tests that attempt to break the system are developed during detailed design and implementation.

ATM CELL BASIC FORMAT:



ATM INFRASTRUCTURE

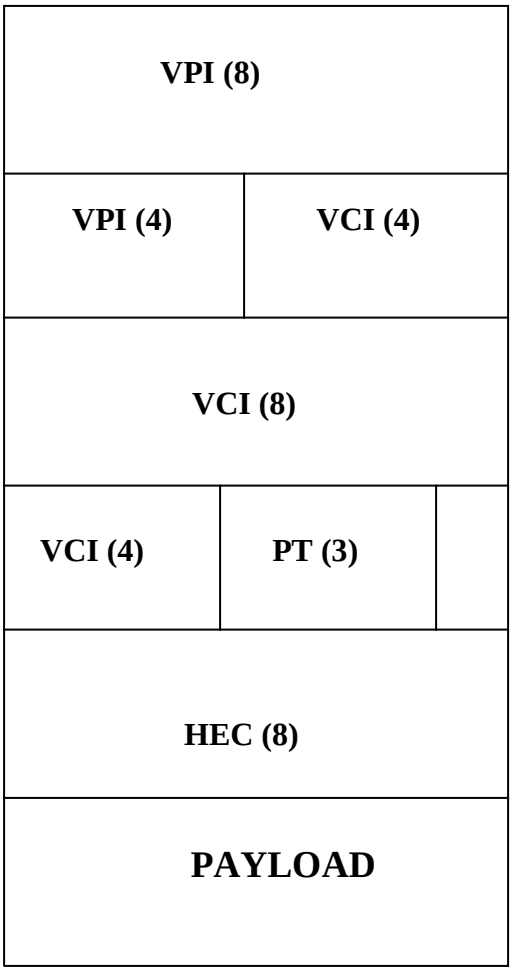
A telecommunication network is designed in a series of layers. A typical configuration may have utilized a mix of time division multiplexing, Frame Relay, ATM and/ or IP. Within a network, carriers often extend the characteristic strengths of ATM by blending it other technologies, such as ATM over SONET/SDH or DSL over ATM. By doing so, they extend the management features of ATM to other platforms in a very cost effective manner.

ATM itself consists of layers. The first layer-known as the adaptation layer-holds the bulk of the transmission. This 48-byte payload divides the data into different types. The ATM layer contains five bytes of additional information, referred to as overhead.

This section directs the transmission. Lastly, the physical layer attaches the electrical elements and network interfaces.

The vast majority (roughly 80 percent) of the world's caarriers use ATM inthe core of their networks. ATM has been widely adopted because of its unmatched flexibility in supporting the broadest arraay of technologies, including DSL,IP Ethernet, Frame Relay, SONET/SDH and wireless platforms. It also acts a unique bridge between legacy equipment and the new generation of operating systems and platforms. ATM freely and easily communicates with both, allowing carriers to maximize their infrastructure investment.

ATM CELL HEADER FORMAT



VPI—Vitual Private Identifier

CLP—Cell Loss Priority

PT-----Payload Type

HEC---Header Error Control

- VPI-8 bits of virtual path identifier that is used, in conjunction with the VCI, to identify the next destination of a cell as it passes through a series of switch routers on its way to its destination.

- VCI—16 bits of virtual channel identifier that is used, in conjunction with the VPI, to identify the next destination of a cell as it passes through a series of switch routers on its way to its destination.

- PT—3 bits of payload type. The first bit indicates whether the cell contains user data or control data. If the cell contains user data, the second bit indicates congestion, and the third bit indicates whether the cell is the last in a series of cells that represent a single AAL5 frame.

- CLP—1 bit of cell loss priority that indicates whether the cell should be discarded if it encounters extreme congestion as it moves through the network.

- HEC—8 bits of header error control that are a checksum calculated only on the header itself.

ATM is a proven technology that is now in its fourth generation of switches . Its maturity alone is not its greatest asset. Its strength is in its ability to anticipate the market and quickly respond, doing so with the full confidence of the industry behind it.

DESIGN NOTATION:

In software design, as in mathematics, the represenation schemes used are of fundamental importance. Good notations can clarify the interrelationshops and interactions of interest, while poor notaion can complicater and interfere with good design practice. At least three levels of design specifications exist: external design specifications, which describes the external characteristics of a software system; architectural design specifications, which describe control flow, data representation, and other algorithmic details within the modules.

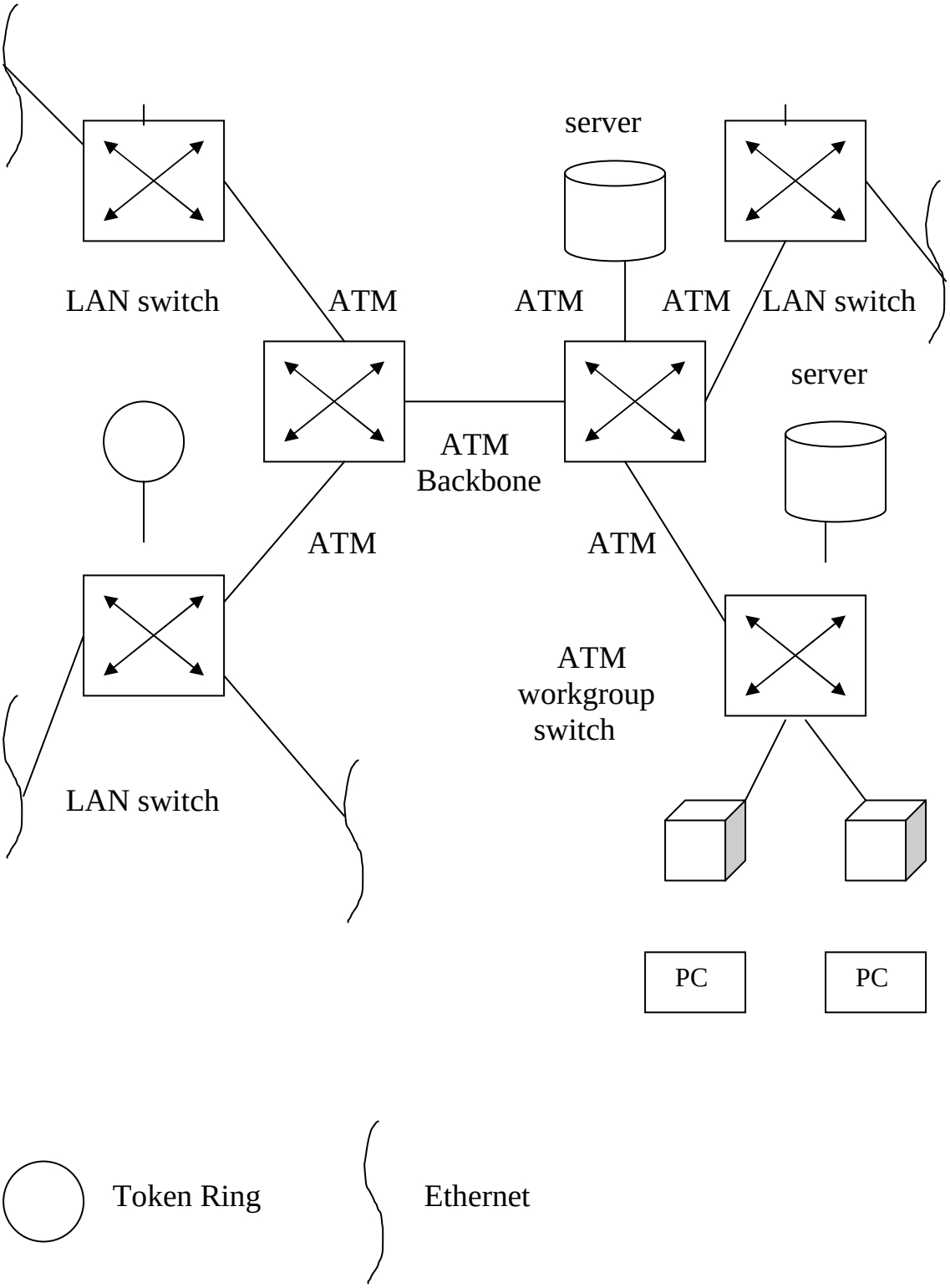
Some design representations are appropriate for use on more than one level of design, while others are appropriate for only one level. Because the boundary between requirements analysis and external design is not well defined, some of the notation are also useful for external design.

Notation used to specify the external characteristics, architectural structure, and processing details of a software system include data flow diagram ,structure charts, HIPO diagrams, procedure specifications, pseudo code, structured English, and structured flowcharts. The LAN enviroment of a campus or building appears sheltered from the headaches associated with high-volumes of traffic that deluge larger networks. But the changes of LAN interconnection and performance are no less critical.

The ATM /LAN relationship recently took a giant step forward when a prominent U.S. vendor announced a patent for its approach to extending ATM's quality of service to the LAN. The filing signals another birth in a long lineage age of applications that prove the staying power and adaptability of ATM.

ATM IN LAN:

ATM BACKBONE WITH LAN WORKGROUP



ATM SERVICES

ATM provides a connection-oriented service with best-effort delivery and can provide quality of service (QoS) guarantees with regard to throughput. An ATM connection between two network endpoints is called a Virtual Channel Connection (VCC) Two distinct connection types are supported.

ETHERNET

Ethernet is the name given to a popular packet-switched LAN technology invented at Xerox PARC in the early 1970's. Xerox
corporation Intel Corporation, and Digital Equipment Corporation
standardized Ethernet in 1978; IEEE released a compatible version of the standard using the number 802.3. Ethernet has become a popular LAN technology, most medium or large corporations use Ethernets.Because Ethernet is so popular, many varients exits.

Each ethernet cable is about 1/2 inch in diameter and upto 500 metres long. A resistor is added between the center wire and shield at each end to prevent reflection of electrical signals.

The original Ethernet design used a coaxial cable called the ether; the cable itself is completely passive; all the active electronic components that make the network function are associated with computers that are attached to the network.

The connection between a computer and a coaxial ethernet5 cable requires a hardware device called a transceiver. Physically, the connection between a transceiver and the Ethernet requires a small hole in the outer layers of the cable Technicians often use the term tape to describe the connection between an Ethernet transceiver and the cable.

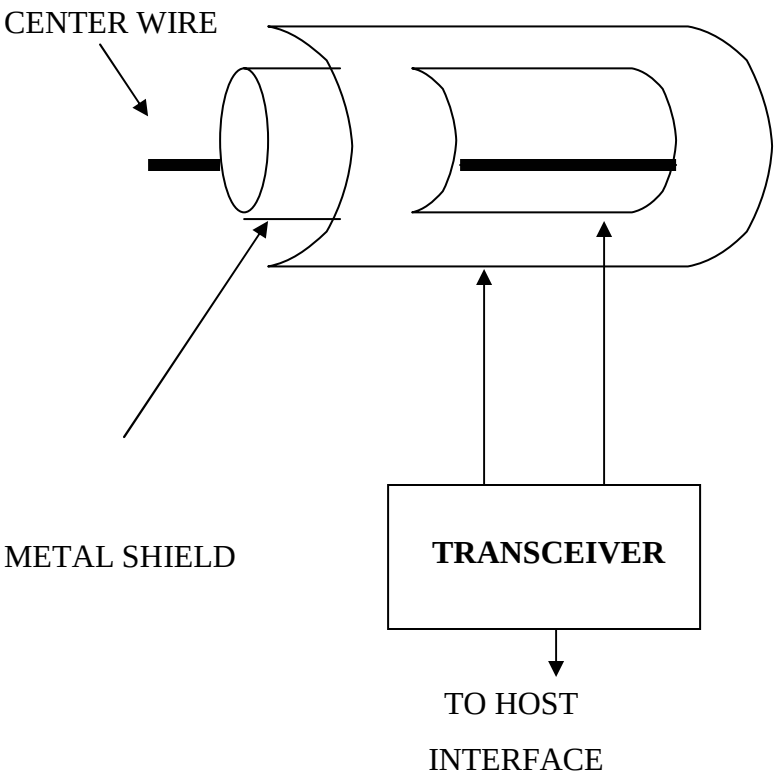


FIG: A cutway of Ethernet cable showing the details of electrical connection between a transceiver and cable.

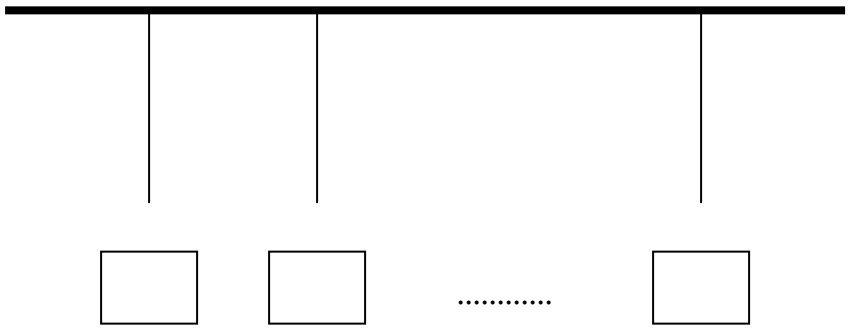


FIG: The schematic diagram of an Ethernet with many computers connect.

Each connection to an Ethernet has two major electronic components. A transceiver connects to the center wire and braided shield on the cable, sensing and

sending signals on the ether. A host interface or host adapter plugs into the computer's bus and connects to the transceiver.

A transceiver is a small piece of hardware usually found physically adjacent to the ether. In addition to the analog hardware that senses and controls electrical signals on the ether, a transceiver contains digital circuitry that allows it to communicate with a digital computer.

The transceiver can sense when the Ethernet is in use and can translate analog electrical signals on the ether to digital form. A cable called the attachment unit interface (AUI) connects the transceiver to an adapter board in a host computer, informally called a transceiver cable. The AUI cable contains many wires. The wires carry the electrical power needed to operate the transceiver, the signals that control the transceiver operation, and the contents of the packets being sent or received.

Each host interface controls the operation of one transceiver according to instructions it receives from the computer software. To the operating system software, the interface appears to be an I/O device that accepts basic data transfer instructions from the computer, controls the transceiver to carry them out, interrupts when the task has been completed, and reports status information.

Although the transceiver is a simple hardware device, thr host interface can be complex. In practice, organizations that use the original Ethernet in a conventional office environment run the Ethernet cable along the seiling in each hall, and arrange for a connection from each office to attach to the cable.

PROPERTIES OF ETHERNET

The Ethernet is a 10 mbps broadcast bus technology with best-effort delivery sematics and distributed access control. It is a bus because all stations share a single communication channel; it broadcasts because all transceivers receive every

transmission. The method used to direct packets from one station to just one other or a subset of all stations.

A transceiver passes all packets from the cable to the host interface, which chooses packets the computer should receive and filters out all others. Ethernet is called a best-effort delivery mechanism because the hardware provides no informations to the sender about whether the packet was delivered. For example, if the destination machine happens to be powered down, packets sent to it will be lost, and the sender will not be notified.

The Ethernet access scheme is called Carrier Sense Multiple Access with Collision Detect (CSMA/CD). It is CSMA because multiple machines can access the Ethernet simultaneously and each machine determines whether the ether is idle by sensing whether a carrier wave is present. When a host interface has a packet to transmit, it listens to the ether to see if a message is being transmitted. When no transmission is sensed, the host interface starts transmitting. Each transmission is limited in duration because there is a maximum packet size.

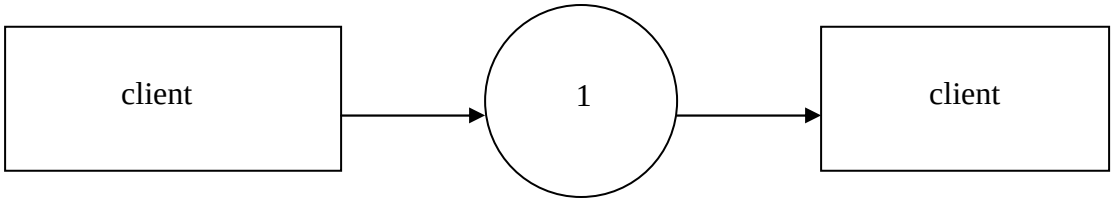
DATA FLOW DIAGRAM:

A data flow diagram is a graphical representation that depicts informatio flow and the transforms that are applied as data move from input to output. The basic form of a data flow diagram, also known as a data flow graph or a bubble chart.

The data flow diagram may be used to represent a system or software at any level of abstraction. In fact, DFDs may be partitioned into levels that represent increasing information flow and functional detail. Therefore, the DFD provides a mechanism for functional modeling as well as information flow modeling. In so doing,it satisfies the second operational analysis principle.

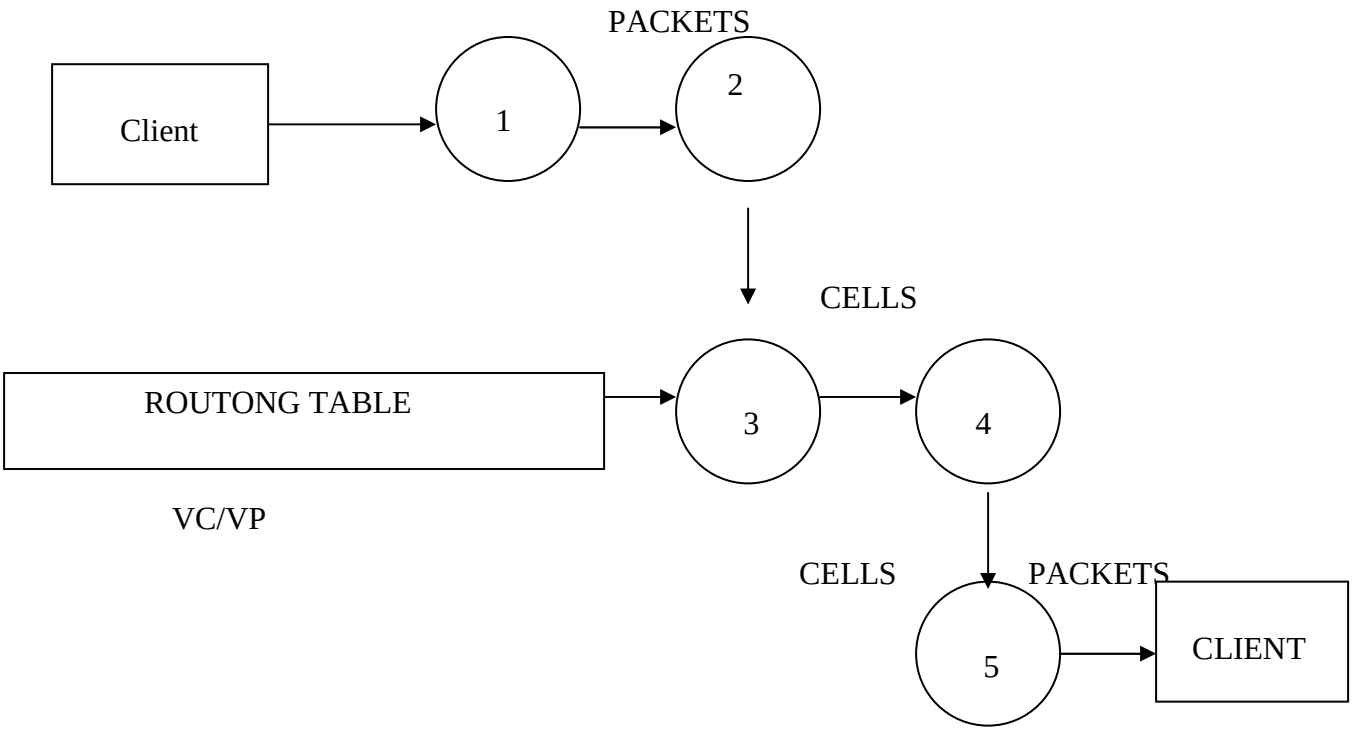
A rectangle is used to represent an external entity; that is, a system element or another system that produces information for transformation by the software or receives information produced by the software. A circle sometimes called a bubble represents a process of transform that is applied to data or control and changes it in some way. An arrow represents one or more data items. All arrows on a data flow diagram should be labeled. The double line represents data store-stored information that is used by the software. The simplicity of DFD notation is one reason why structure analysis techniques are widely used.

DFD AT LEVEL 0



In software design, as in mathematics, the represenation schemes used are of fundamental importance. Good notations can clarify the interrelationshops and interactions of interest, while poor notaion can complicater and interfere with good design practice. At least three levels of design specifications exist: external design specifications, which describes the external characteristics of a software system; architectural design specifications, which describe control flow, data representation, and other algorithmic details within the modules.

DFD AT LEVEL 1



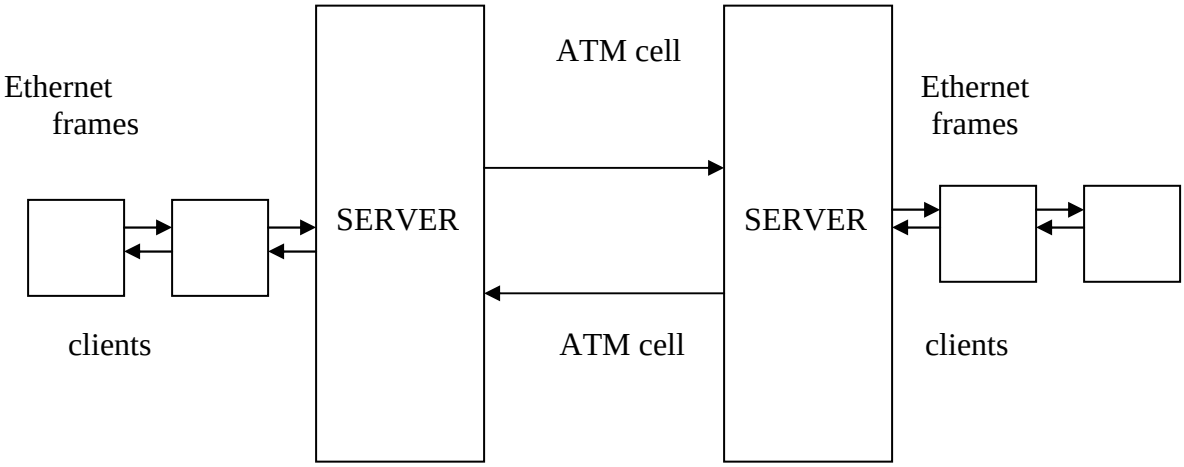
0. PACKET CAPTURING
1. CONVERT PACKET TO CELLS
2. ROUTING
3. SERVER 2-PACKET CONVERSION
4. WRITE PACKET

STRUCTURE CHART:

Structure charts are used during architectural design to document hierarchial structure, paremeters, and interconnections in a system. A structure charts differs from a flowchart in two ways; a structure chart has no decision boxes, and the sequential ordering of tasks inhetent in a flow chart can be suppressed in a structure chart.

In software design, as in mathematics, the represenation schemes used are of fundamental importance. Good notations can clarify the interrelationshops and interactions of interest, while poor notaion can complicater and interfere with good design practice.

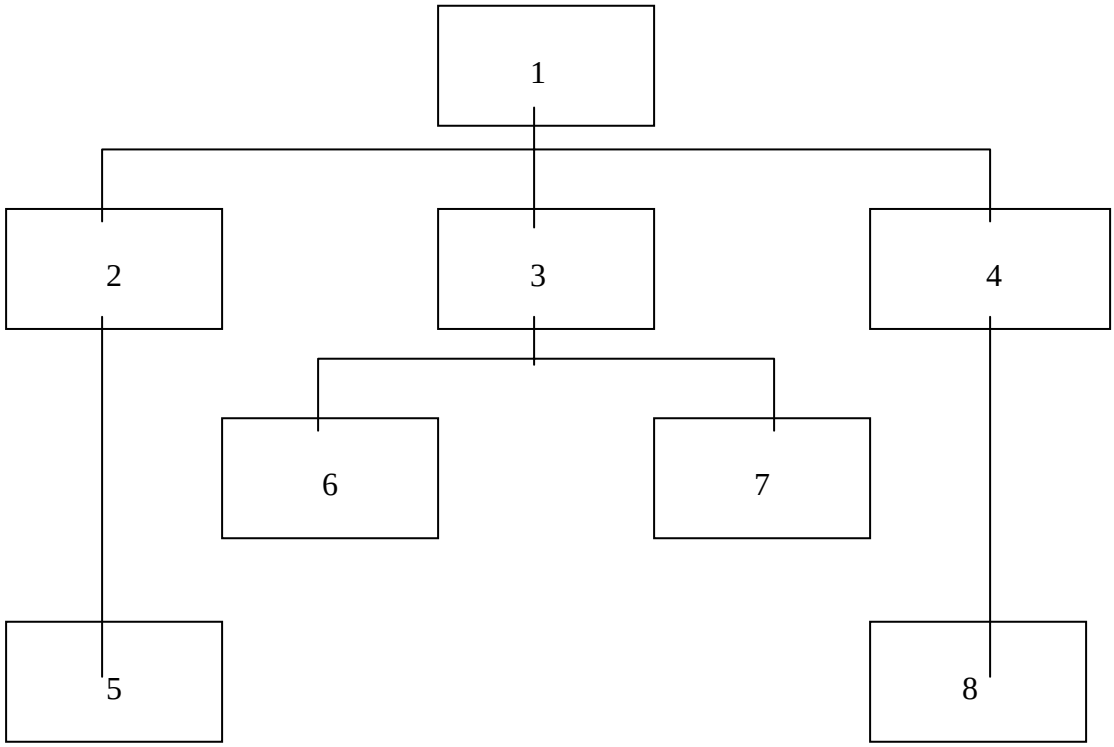
The structure of a hierarchial system can be specified using a structure chart as illustrated in fig. The chart can be augmented with module-by-module specification of the input and output parameters, as well as the input output parametet attributes. During architectural design the parameter attributes are abstract; they are refined into concrete representations during detailed design.



HIPO DIAGRAM:

HIPO diagrams (Hierarchy-Process-Input-Output) contains a visual table of contents, a set of detail diagrams. The visual table of contents is a directory to the set of diagrams in the package; it consists of a tree-structureed) directory, a summary of the contents of each overview diagram, and a legend of symbol definitions. The visual table of contents is a stylized structure chart. Overview diagrams specify the functional processes in a system. Each overview diagram describes the inputs, processing steps, and outputs for the function being specified.

HIPO DIAGRAM



1.ATM ETHERNET BRIDGE

2.PACKET CAPTURING

3.CELL FORMATION

4. ATM ROUTING

5.CAPTURE THE PACKETS & SEND IT TO SERVER1

6.CONVERT THE FRAMES TO CELLS IN SERVER1

7.CONVERT THE CELLS TO FRAMES IN SERVER2

8.ROUTE THE PACKETS TO THE DESTINATION

STRUCTURED FLOWCHARTS:

Flowcharts are the traditional means for specifying and documenting algorithmic details in software system. Flowcharts incorporate rectangular boxes for actions, diamond shaped boxes for decisions, directed arcs for specifying interconnections between boxes and variety of specially shaped symbols to denote input, output, data stores, etc.

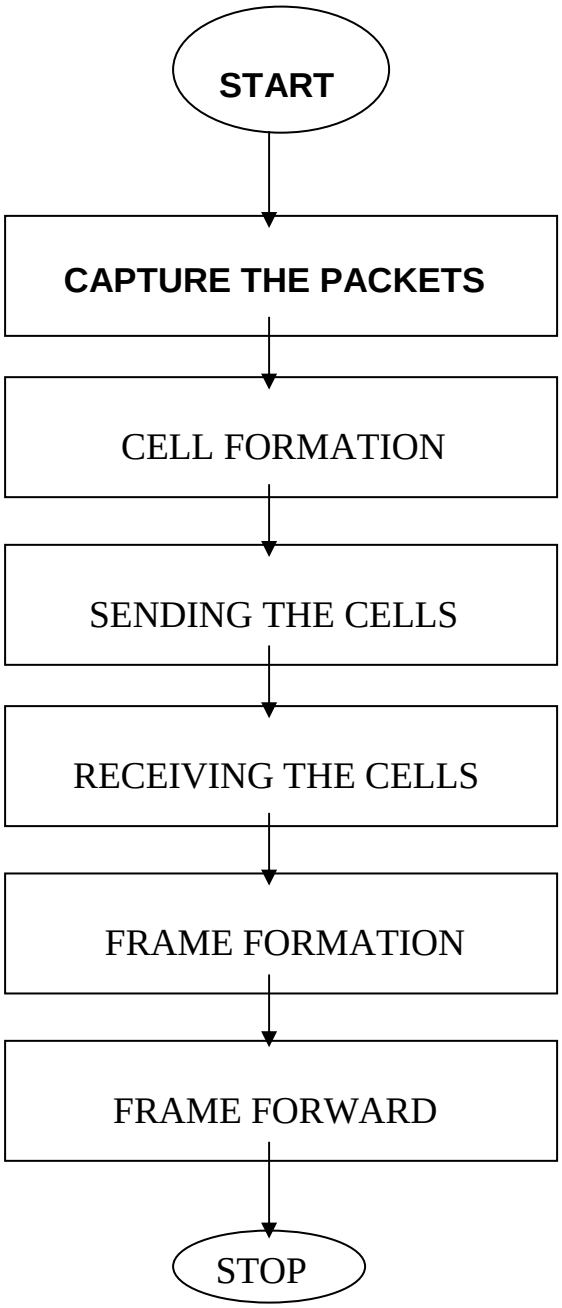
Structued flowcharts differ from traditional flowcharts in that structure flowcharts are restricted to compositions of certain basic forms. this makes the resulting flowcharts the graphical equivalent of a structured pseudo code description.

The basic forms are characterized by single entry into and single exit from the form. Thus, forms can be nested within forms to any arbitrary
depth, and in any arbitrary fashion, so long as the single entry property is reserved.

Structure flowcharts are preferred in situations where clarity of control flow is to emphasize.

The single entry, single exit property allows hierarchial nesting of structure flowcharts to document a design in top-down fasion, starting with top level structure and proceeding through detailed design.

FLOW CHART


START
CAPTURE THE PACKETS
CELL FORMATION
SENDING THE CELLS
RECEIVING THE CELLS
FRAME FORMATION
FRAME FORWARD
STOP

IMPLEMENTATION METHEDOLOGY

INTRODUCTION

The implementation phase of software development is concerned with translating design specifications into source code. The primary goal of implementation is to write source code and internal documentation so that conformance of the code to its specification can be easily verifiled, and so that debugging, testing and modification are eased. This goal can be achieved by making the source code as clear as possible.

NETWORKS:

A network is an interconnection of two or more devices in order to enable transfer of data or information from one place to another.

EVOLUTION OF COMPUTER NETWORKS:

Computers were solely used for processing and storing data. Interaction through it was restricted to the particular user sitting in front of the computer. There was no way for people to communicate with each other through computers. All this changed with the evolution of the concept of NETWORKING.

The concept of neworking envisages the interconnection of two or more computers by means ofcables and information can be sent and received between these computers.

The main purpose of setting up a network is to share information. Exchange of information over a network of computers is easier and faster. In addition, resources like hard disk space, printers, scanner, etc. can be shared among the computers in a network.

ADVANTAGES OF COMPUTER NETWORK:

- Sharing of hardware resources
- Sharing of software resources
- Central storage and data security
- Easier and faster sharing of information

TYPES OF COMPUTER NETWORKS:

There are three broad categories of networks; they are LAN, MAN, WAN.

LOCAL AREA NETWORK:

A network consisting of two or more computers that are interconnected by means of cable in a single location is called as Local Area Network. In a LAN,the computers can take any one of the three functions as detailed below:

TYPES OF LAN

- Peer to peer

  This is an examble of a simple network where two or more computers are directly connected to each other and share resources. There is no central control over the network. Peer networks are organized into workgroups.

- Server based networks

In this type of networks, there is a main computer called as the SERVER that controls the network and provides central storage space for information. The other computers that are connected to the Server are called as CLIENTS, or WORKSTATIONS or DUMB TERMINALS.

- Hybrid networks

  These types of network are a combination of both peer-to-peer network and server network.

METROPOLITAN AREA NETWORK

A network of computers that are interconnected within a specific geographical area like a university campus or a city is called as METROPOLITAN AREA NETWORK or Campus Area Network(CAN).

WIDE AREA NETWORK:

A network of computers that are interconnected over a large area is called as a WIDE AREA NETWORK. The computers are connected by means of cable, telephone lines, satellites or radio transceivers.

COMPONENTS OF COMPUTER NETWORK

SERVER:

This is the main computer that controls the network and therefore should have a fast processor with high capacity RAM & Hard Disk.

CLIENT:

This is the computer that is connected to the server. The configuration depends on the requirements of the user.

NETWORK INTERFACE CARD:

This is an add-on card that is plugged on any one of the slots available on the motherboard of the server computer as well as the client computer. It is responsible for transmitting & receiving data to and from the server.

NETWORK MEDIA:

This comprises of cables used to physically connect each computer in the network.

CONNECTOR:

A connector is used to connect the cable with the NIC.

CONNECTING DEVICES:

These are special devices used to amplify the electrical signals. Some of them are Repeaters, Switches, Bridges and Gateways.

ASYNCHRONOUS TRANSFER MODE:

ATM technology involves the process of sending/ receiving data via asynchronous multiplex transmission, in which data is transmitted in fixed packets of 53 bytes with an address. Utilizing this technique, data is transmitted sporadically (only when necessary) rather than in a synchronous manner which requires communications between the transmitting and receiving sides.

WAN TECHNOLOGIES:

- Switching systems
- Frame relay systems
- Broadcast systems
- X.25 networks
- Frame relay
- Switched Multi-megabit data service
- Point-to-point protocol
- Digital Data service
- Fibre Distribution Data Interface
- Asynchronous tranfer mode
- Synchronous Optical Network

ADVANTAGE OF ATM OVER OTHER WAN TECHNOLOGIES

ATM is a connection oriented a packet-oriented transfer mode. ATM has the advantage of the digital line's higher reliability and fidility to provide faster packet switching. It allows multiple logical connections to be switched over a single physical interface. The information flow on each logical connection is organised into fixed-size packets, called cells. ATM does not provide any link-by-link error control or flow control.

The cell is how ATM packages information to be sent over the network.The size of the cell is always 53 bytes, which is a relatively small package for transmitting data. Packaging the information into a cell is typically referred to as cellificaation. This method of cellificaation allows network to handle a combination of voice, data, and video information and still provide an optimized transmission enviorement for each information type. ATM has been optimized to accommodate a both real-time and non real-time data efficiently and cost effectively.

The switching of cells enables two important benefits of ATM:

- The ability to intermingle voice, data,and video without compromising the unique needs of each application.

- The ability to switch the information very quickly

PERMANENT VIRTUAL CONNECTIONS (PVC):

A PVC is a connection set up by some external mechanism, typically network management, in which a set of swiches between an ATM source and destination ATM system are programmed with the appropriate VPI/VCI values. ATM signaling can

facilitate the set up of PVCs, but, by definition, PVCs always require some manual configuration. As such, their use can often be cumbersome.

SWITCHED VIRTUAL CONNECTIONS (SVC)

An SVC is a connection that is set up automatically through a signaling protocal. SVCs do not require the manual interaction needed to set up PVCs and, as such, are likely to be much more widely used. All higher layer protocals operating over ATM primaarily use SVCs.

ATM ADAPTATION LAYERS

AAL1

AALI is designed to support real-time constant bit rate (CBR) traffic such as pulse code modulated (PCM) voice via ATM networks. The function of the AALI layer include detecting lost cells, and providing time stamp to support a common clock between end stations. The cells on the AALI frame are numbered to allow the reassembly process. Like all AAL layers, this aadaptation layer checks the frame integrity. Some of these mechanism involve lost or misinserted cells, faulty headers, etc. An example for traffic handled by AALI is 64kbps digital voice.

1.CBR (constant bit rate)

The CBR service class is intented for real-time applications, i.e. those requiring tightly constrained delay and delay variation, as would be appropriate for voice and video applications. The consistent availability of a fixed quantity of bandwidth is considered appropriate for CBR service. Cells which are delayed beyond the value specified by CTD (cell transfer delay) are assumed to be significantly less value to the application.

AAL2 was initally intended for real-time variable bit rate (VBR) traffic. It allows the support of bursty traffic to be exploited. The AAL2 was initally standard has not yet been finally defined, but will definitely cover error checking as well as delay monitoring and handling. An example for traffic handled by AAL2 is VBR encoded video.

2.REAL TIME VBR

The real time VBR service class is intended for real-time applications, i.e., those requiring tightly constrained delay and delay variation, as would be appropriate for voice and video application. Source are expected to transmit at a rate which varies with time. Equivalently the source can be described "bursty". Cells which are delayed beyond the value specified by CTD are assumed to be of significantly less value to the application. Real-time VBR service may support statistical multiplexing of real-time sources, or may provide a consistently guaranteed QoS.

AAL3/4

Type 3/4 adaptation layer specifies the connection-oriented and non connection-oriented transfer of data packets via ATM networks. Like the other adaptation layers, it also deals with error checking and correction functions before and after the original data.

The information is then chopped into 44-byte chunks. The payloads include 2bytes of header and 2bytes of trailer. A CRC is performed an each cell to check for bit errors. There is also message identification (MID) which allows multiplexing and interleaving of large packets an a single virtual connection.

Two examples of traffic handled by AAL3/4 is frame relay over ATM andCCITT 1.364 (SMDS) over ATM.AAL3 andAAL4 were both designed for non real-time VBR traffic (i.e., traditional data networks), and because the specifications were so similar in format and functionality, they were eventually merged creating AAL3/4.

NON-REAL TIME VBR

The non-real time VBR service class is intended for non-real time applications which have 'bursty' traffic characteristics and which can be characteristics in terms of a GCRA. For those cells which are transferred, it expects a bound on the cell transfer delay. Non real time VBR service supports statistical multiplexing of connections.

AALs 1-4 where designed by the international telecommunications Union (ITU) and their design refelected the ITU's historical focus an telephone (as opposed to computer) networks. The computer industry found AAL3/4 excessively complex and inefficient and proposed the Simple and Efficient Adaptation Layer (SEAL) as a replacement. SEAL was subsequently standaraized as AAL5. AAL5 is now the de-facto-standard for all VBR traffic in ATM based computer networks2.

SEGMENTATION AND REASSEMBLY (AAL5)

ATM adaptation layers are often viewed as consisting of two sub layers: the convergence sub layer (CS) and the segmentation and reassembly (SAR) sub layer. In AAL5 the function of the CS is to perform framing an error detection while SAR's job is to segment AAL5 frames into ATM cells for transmission and to reassemble AAL5 frames from ATM cells on reception. For examble, Suppose a 450 byte packet is passed by an upper layer protocol to the CS.

The CS pads the packet (if necessary) so that its length is a multiple of 48, minus 8 (length -8 mod 48). In this case 22 bytes of padding will be added yielding 472

bytes. The Cs then adds an 8-byte trailer containing a 4byte error detecting code and a 2byte length field set in this case to 450. The padded packet with CS trailer appended is thus always an even multiple of 48 bytes in length and is sometime called an AAL5 protocol data unit (PDU) or simply an AAL5 frame.

The CS passes AAL5 frames to the SAR sub layer for segmentation into 53 byte ATM cells. For AAL5 each ATM cell consists of a standard 5-byte cell header followed by a 48 byte payload. In this example, the SAR sub layer segments the 480 bytes of the AAL5 frame into 10 ATM cells, and then schedules them for transmission.

ATM IN TELECOMUNICATIONS INFRASTRUCTURE

A telecommunication network is designed in a series of layers. A typical configuration may have utilized a mix of time division multiplexing, Frame Relay, ATM and/ or IP. Within a network, carriers often extend the characteristic strengths of ATM by blending it other technologies, such as ATM over SONET/SDH or DSL over ATM. By doing so, they extend the management features of ATM to other platforms in a very cost effective manner.

ATM itself consists of layers. The first layer-known as the adaptation layer-holds the bulk of the transmission. This 48-byte payload divides the data into different types. The ATM layer contains five bytes of additional information, referred to as overhead. This section directs the transmission. Lastly, the physical layer attaches the electrical elements and network interfaces.

ATM AS THE BACKBONE FOR OTHER NETWORKS

The vast majority (roughly 80 percent) of the world's caarriers use ATM inthe core of their networks. ATM has been widely adopted because of its unmatched

flexibility in supporting the broadest arraay of technologies, including DSL,IP Ethernet, Frame Relay, SONET/SDH and wireless platforms. It also acts a unique bridge between legacy equipment and the new generation of operating systems and platforms. ATM freely and easily communicates with both, allowing carriers to maximize their infrastructure investment.

LAN EMULATION OVER ATM

LANE emulates the services of existing Local Area Network (LAN) technologies and provides upper protocol layers an inaterface similaar to the MAC layer.It was developed to give the possibility for connecting end station to ATM networks,while network software can interact as if they were connected to traditional LANs, e.g., to IEEE 802.3 Ethernet [IEEE802.3][Ethernet] or IEEE802.5 Token Ring [IEEE 802.5] network.

The services provided by the ATM network differ from those in today's LANs. The LAN end stations can send data without first establishing a connection, while ATM is connection-oriented technology. In traditional LANs the end stations share the same physical media, so there is a possibility to effectively send packets to all or sum of the end stations (broadcast and multicast operation).

COLLISION DETECTION AND RECOVERY

When a tranceiver begins transmission, the signal does not reach all parts of the network simultaneously. Instead it travels along the cable at approximately 80% of the speed of light. Thus it is possible for two transceivers to both sense that the network is idle and begin transmission simultaneously. When the two electrical signals cross they become scrambled, such that neither is meaningful. Such incidents are called collisions. The Ethernet handles collisions in an ingenious fashion. Each transceiver monitors the cable while it is transmitting to see if a foreign signal interferes with its transmission.

Tehnically, the monitoring is called collision detect, making the Ethernet a CSMA/CD network.

When a collision is detected, the host interface aborts transmission, waits for activity to subside, and tries again. Care must be taken. If collision, twice as long if a second attempt to transmit also produces a collision, four times as long if a third attempt results in a collisions, and so on.

The motivation for exponential back off is that in the unlikely event many stations attempt to transmit simultaneously, a severe traffic jam could occur. In sucn a jam, there is a high probability two stations will choose random back offs together.

Thus, the probability of another collision is high. By doubling the random delay, the exponential back off strategy quickly spreads the station's attempts to retransmit over a reasonably long peroid of time, making the probability of further collisions extremely small.

ETHERNET CAPACITY

The standard Ethernet is rated at 10 mbps, which means that the data can be transmitted onto the cable at 10 million bits per second. Although a computer can generate data at Ethernet speed, raw network speed should not be thought of as the rate at which two computers can exchange data. Instead; network speed should be thought of as a measure if network total traffic capacity. High bandwidth makes it possible to carry heavy traffic loads while low bandwidth means the highway cannot carry as much traffic.

Ethernet follows the IEEE 802.3 standard, using the carrier sense multiple access with collision detect (CSMA/CD) method of network access or protocol. It costitutes the lower two layers of the ISO model (Data link, Physical layer).

It function at a transfer speed of 10 Bits/sec. With this transfer speed and the CSMA/CD access method, Ethernet is an excellent choice for networks that occasional bursts of heavy traffic . Because of its low cost (existing cabling like the shielded and unsheilded twisted pair), wide availability, and technical maturity, Ehernet is by far the most popular medium for LANs.

Ethernet is a multi-access, packet-switched communications system for carrying digital data amoung locally distributed computer systems. Ethernet is a bus system with a random access mechanism for the transmission protocol. An adaptive delay interval is used in the case of collisions. The adaptive method is a binary exponential back-off. Although the CSMA/CD procedure can be used for almost all bidirectional media, the basis Ethernet uses a coaxial cable.

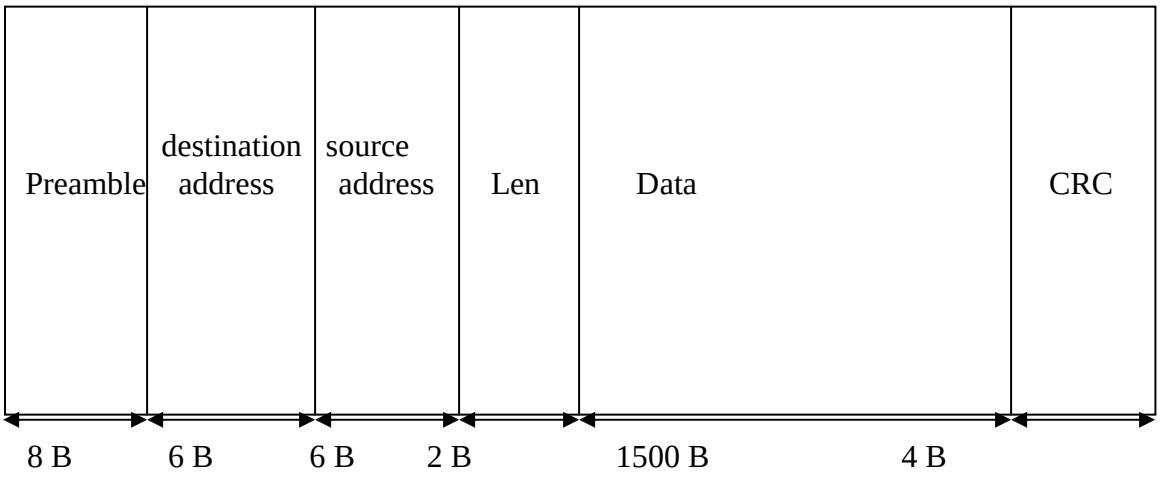
The Ethernet system has several basic components: the station, the controller, the transmission system, and the interface. The station is usually a microprocessor that makes use of the communications capability of the network. Each station has an interface between the station's operating system and the Ethernet controller. The controller is usually connected to the system bus and can thus be addressed identification, error detection, intermediate storage, and CSMD/CD and packet formation.

The transmission system contains all the components necessary to establish a communication route between controllers. It contains transmission medium appropriate send and receives units, and perhaps repeaters to enlarge the network.

The interface represents the connection between the controllers and the transmission system. Since the controller performs a majority of the communication function, the interface is a relatively simple device. It also contains the data routes to and from the transmission system, it must also be able to transmit status reports and control information.

Stations must be able to send and receive packets over the common coaxial cable system using the agreed format and inter packet gap. Each packet should be seen as a series of 8-bit bytes. The least significant bit of each byte is transferred first.The addressing uses a 48-bit address which is fixed in the hardware of every Ethernet controller, which has unique address.

STRUCTURE OF THE ETHERNET PACKET



A packet is a unit of information which not only contains user data but also the header information. This header information is the control information that is applied to the data in the formation of a packet. Each respective layer (network, transport, and session) that is implemented on the network station will apply its header information into the packet. This information is used by the receiving station to take action.

Maximum packet size is 1526 bytes, of which 1500 data. Minimum packet size is 72 bytes, of which 46 are data. The preamble(p) to an Ethernet packet is a special bit pattern which servers to synchronize the receiving station. This is followed by Destination address (DA) of the station to which the packet is sent (IP address in case of the TCP/IP protocol) is an 6-byte field.

Source addresses (SA), which contains a 6-byte address of the sending station. Length field, which contains a 2-byte type identifier of the for interpretation by higher-level protocols, for aiding in interpretation of the data. Data field(data), which contains a whole number of data bytes in the region 8N < 1500. The minimum ensures that correct packets and collision fragments can be distinguished. Each bit sequence shorter that the maximum packet length is identified as a collision fragment. FCS, Frame Check Sequence, which is a 4-byte field containing a cyclic redundancy checksum produced by a polynomial.

Each station reads the packet on the bus cable and compares the destination address with its own.If the two are the same, the station is the destination and it receives the packet completely. Because of this property, all stations in the network read packet.

THEORY OF OPERATION

Ethernet access method basically performs three functions: transmitting and receiving data packets, decoding the data packets for transmission or reception, and detecting errors within the data packet or on the networks. A station wishing to transmit checks whether the cable is busy(using carrier sense) and defers transmission of the packet untill the cable is quiet. when the cable is quiet, the deferring station immediately begins to transmit.

When a station is ready to send a frame, the transceiver checks to see if any other host is transmitting. if not, the transceiver sends the frame, which monitoring the signal on the cable if any other host started to send at the same time. If another station is also transmitting, a collision results. A collision is not a disastrous event. Rather, it garbles the signal on the cable, just like two people on a singleplex telephone line simultaneously, when a collision is detected, the transceiver notifies the network interface, which invokes the back off algorthm .

Ethernet is one of the LAN technologies most popularly used. But there are many other technologies available for the LAN internet work. Some of the other LAN tchnologies are listed below

LANtastic	2	Mbps
ARCnet	2.5	Mbps
Ethernet	10	Mbps
Token ring	4/16	Mbps
Token bus(IEEE 802.4)		
Apple Talk/Local Talk		
Banyan VINES		
FDDI(Fibre Distributed Data Interface)	100	Mbps

SEGMENTATION AND REASSEMBLY (AAL5)

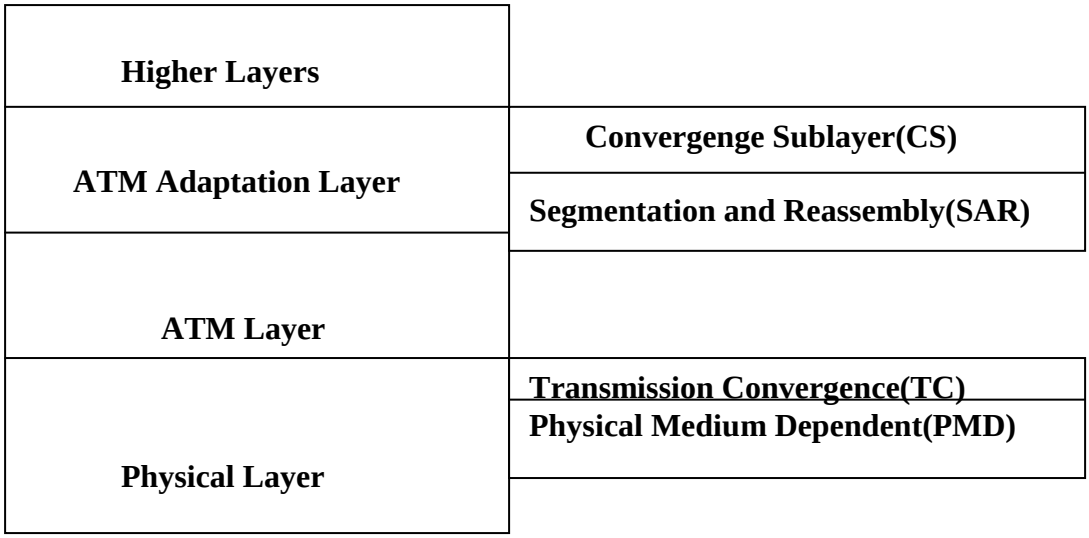
ATM adaptation layers are often viewed as consisting of two sub layers: the convergence sub layer (CS) and the segmentation and reassembly (SAR) sub layer. In AAL5 the function of the CS is to perform framing an error detection while SAR's job is to segment AAL5 frames into ATM cells for transmission and to reassemble AAL5 frames from ATM cells on reception. For example, suppose a 450 byte packet is passed by an upper layer protocol to the CS.

The CS pads the packet (if necessary) so that its length is a multiple of 48, minus 8 (length -8 mod 48). In this case 22 bytes of padding will be added yielding 472 bytes. The Cs then adds an 8-byte trailer containing a 4byte error detecting code and a 2byte length field set in this case to 450. The padded packet with CS trailer appended is thus always an even multiple of 48 bytes in length and is sometime called an AAL5 protocol data unit (PDU) or simply an AAL5 frame.

The CS passes AAL5 frames to the SAR sub layer for segmentation into 53 byte ATM cells. For AAL5 each ATM cell consists of a standard 5-byte cell header

followed by a 48 byte payload. In this example, the SAR sub layer segments the 480 bytes of the AAL5 frame into 10 ATM cells, and then schedules them for transmission.

As cells arrive at the other end of the connection, the receiving SAR sub layer strips the 5 byte cell headers and concatenates the 48-byte payloads in a reassembly buffer. When end of frame is indicated (by a bit in the last cell header), the reassembled frame is passed to the CS. The CS checks the frame for validity and passes the frame and a status indicator to the layer above it.



TCP/IP

TCP/IP stands for Transmission Control Protocol. It was developed for the US department of Defense to allow communication between different types of computers and networks. Later, TCP/IP was included with the Berkeley software distribution of BSD UNIX. TCP/IP is a widely used networking protocol. TCP provides reliable sequence delivery of packets between clients. The internet Protocol provides packet delivery between hosts.

The Defence Advanced Research Project Agencg(DARPA) developed the TCP/IP suite of protocols to allow communication between networks of different computers.

TCP/IP has a number of features that make it a very robust protocol: First, the network works even if part of the network fails. A TCP/IP network can handle high error rates by retransmitting and rerouting lost or corrupted data.

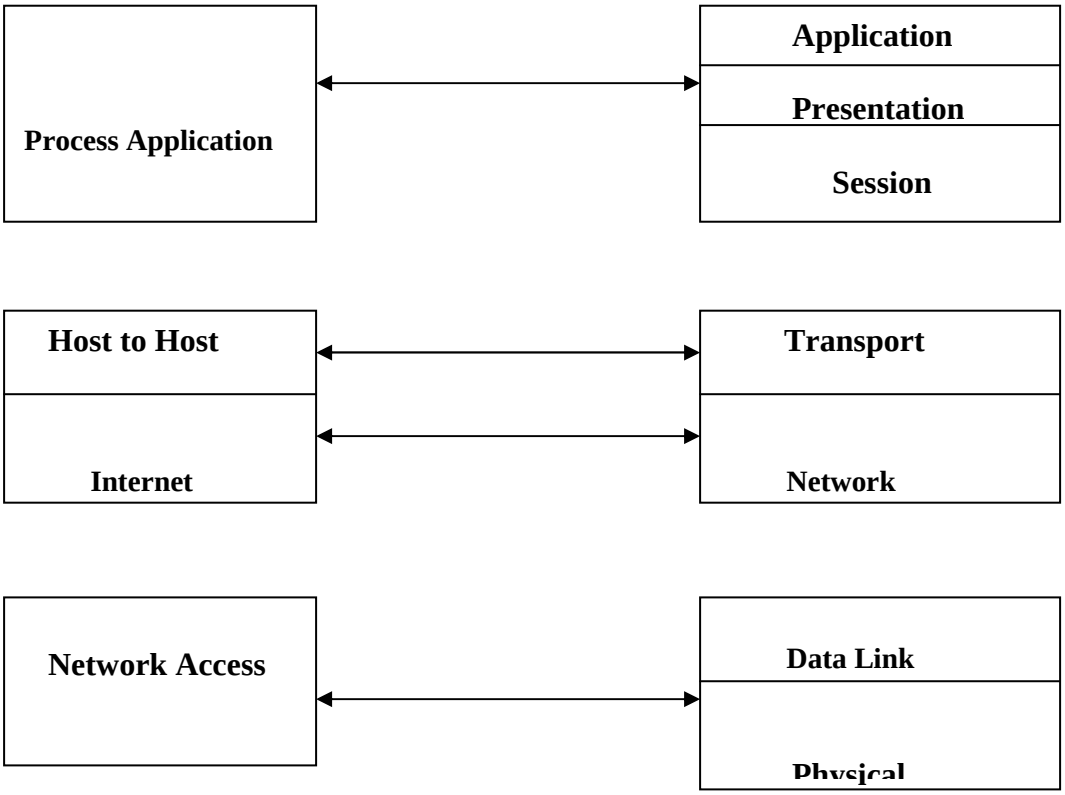
It is a scalable, which means we can add new subnetworks, and we can do this without disrupting the network service.

The TCP/IP protocols define the format of packets. The paclet format specifies the

- Origin and destination of the packet
- The type of packet
- The way packets are acted upon.

COMPARISION OF THE TCP/IP WITH OSI MODEL

TCP/IP		Protocol
Process/ Application	⟷	TELNET, FTP, SMTP
Host-to-Host	⟷	TCP
Internet	⟷	IP
Network Access	⟷	Ethernet,FDD



The topmost three layers are represented as protocol/application layer.The application protocols like TELNEY,FTP, and SMTP belongs to this layer.

The transport layer is represented as host-host layer. TCP belongs this layer. The network layer is represented asInternet layer. IP belongs to this layer. The bottom two layers are represented as network accesses.

The OSI model describes an ideal computer network. In this network system communication take place between processes at each layer of the model. Applications developed for TCP/IP generally use several of the protocols in the suite The layers in the protocol suite are collectively known as protocol stack.

CODING AND TESTING

PROCESS DESCRIPTION

This project involves three important modules. Each module performs different operations which is used for sending the data from one client to another.

1 PACKET CAPTURING
2 CELL FORMATION
3 ATM ROUTING

PACKET CAPTURING:

The first thing to understand is the general layout of a pcap sniffer.

1. We begin by determining which interface we want to sniff on. In Linux this may be something like eth0,in BSD it may be xl1,etc. We can either define this device in a string, or we can as pcap to provide us with the name of an interface that will do the job.

2. Initialize pcap. This is where we actually tell pcap what device we are sniffing on.

3. In the event that we only want to sniff specific traffic (e.g.: only TCP/IP packets,only packets going to port 23,etc) we must create a rule set,"compile" it,and apply it.

This is a three phase process, all of which is closely related. The rule set is kept in a string, and is converted into a format that pcap can read(hence compiling it). The compilation is actually just done by calling a function within our program: it does not involve the use of an external application. Then we tell pcap to apply it to whichever session we wish for it to filter.

4. Finally, we tell pcap to enter its primary execution loop. In this state, pcap waits until it has received however many packets we want it to. Every time it gets a new packet in, it calls another function that we have already defined. The function that it calls can do anything we want: it can dissect the packet and print it to the user, it can save it in a file, or it can do nothing at all.

The user specifies the device by passing the name of it as the first argument to the program. Now the string "dev" holds the name of the interface that we will sniff on in a format that pcap can understand. In this case, pcap just sets the device on its own. Most of the pcap commands allow us to pass them a string as an argument. In the event that the command fails, it will populate the string with a description of the error. In this case, if pcap_lookupdev() fails, it will store an error message in errbuf. The task of creating a sniffing session is really quite simple. For this, we use pcap_open_live(). The prototype of this function (from the pcap man page) is as follows:

Pcap_t*pcap_open_live(char *device,int snaplen, int promise, int to_ms, char *ebuf)

The first argument is the device. Snaplen is an integer which defines the maximum number of bytes to be captured by pcap. Promise, when set to true, brings the interface into promiscuous mode(however, even if it is set to false, it is possible under specific cases for the interface to be in promiscuous mode, anyway).to_ms is the read time out in milliseconds(a value of 0 sniffs until an error occurs: -1 sniffs indefinitely). Lastly, ebuf is a string we can store any error messages within (as we did above with erbut). The function returns our session handler.

To demonstrate, consider this code snippet:

```
#include<peap.h>

pcap_ t *handle:

handle = pcap_open_live (somedev, BUFSIZ, 1,0, errbuf);
```

This code fragment opens the device stored in the strong “somedev”, tells it to read however many bytes are specified in BUFSIZ (which is defined in pcap.h). We are telling it to put the device into promiscuous mode, to sniff until an error occurs, and if there is an error, store it in the string errbut.

PROMISCUOUS VS. NON-PROMISCUOUS SNIFFING

The two techniques are very different in style. In standard, non-promiscuous sniffing, a host is sniffing only traffic that is directly related to it. Only traffic to, from, or routed through the host will be picked up by the sniffer. Promiscuous mode, on the other hand, sniffs all traffic on the wire. In a non-switched environment, this could be all network traffic. The obvious advantage to this is that it provides more packets for sniffing, which may or may not be helpful depending on the reason you are sniffing the network. However, there are regressions. Promiscuous mode sniffing is detectable; a host can test with strong reliability to determine if another host is doing promiscuous sniffing. Second, it only works in a non-switched environment (such as a hub,or a switch that is being ARP flooded). Third, on high traffic networks, the host can become quite taxed for system resources.

FILTERING TRAFFIC:

Often times our sniffer may only be interested in specific traffic. For instance, there may be times when all we want is to sniff on port 23(telnet) in search of passwords. Or perhaps we want to highjack a file being sent over port 21(FTP). Enter pcap_compile() and pcap _ setfilter().

The process is quite simple. After we have already called pcap_open_live() and have a working sniffing sessions, we can apply our filter. Two reasons. First , pcap's filter is far more efficient, because it does it directly with the BPF filter; we eliminate numerous steps by having the BPF diver do it directly. Second, this is a lot easier. Before applying our filter, we must"compile" it. The filter expression is kept in a regular string (char array).

To compile the program we call pcap_compile O. The prototype defines it as:

Int pcap_compile (pcap_t *p, struct bpf_program *fp, char *str, int optimize, bpf_u_int32 netmask)

The first argument is our session handle (pcap_t *handle in our previous example). Following that is a reference to the place we will store the compiled version of our filter. Then comes the expression itself, in regular string format. Next is an integer that decides if the expression should be "optimized" or not (O is false, 1 is true. Standard stuff.) finally, we must specify the net mask of the network the filter applies to. The function returns-1 on failure; all other values imply success.

After the expression has been compiled, it is time to apply it. Enter pcap_setfilter O. Following our format of explaining pcap, we shall look at the pcap_setfilter O prototype:

int pcap_setfilter (pcap_t *p, struct bpf_program *fp)

This is very strait forward. The first argument is our session handler, the second is a reference to the compiled version of the expression (presumably the same variable as the second argument to pcap_compile O).

There are two main techniques for capturing packets. We can either capture a single packet at a time, or we can enter a loop that waits for n number of packets to be sniffed before being done. For this we use pcap_next O.
The prototype for pcap_next O fairly simple:

U_char*pcap_next (pcap_t*p, struct pcap_pkthdr*h)

The first argument is our session handler. The second argument is a pointer to a structure that holds general information about the packet, specifically the time in which it was sniffed, the length of this packet, and the length of his specific portion (incase it is fragmented, for example.) pcap_nextO returns a u_char pointer to the packet that is described by this structure.

int pcap_loop (pcap_t*p,int cnt, pcap_handler callback, u_char*user)

The first argument is our session handle. Following that is an integer that tells pcap_loop O how many packets it should sniff for before returning (a negative value means it should sniff until an error occurs). The third argument is the name of the callback function (just its identifier, no parenthesizes). The last argument is useful in some applications, but many times is simply set as NULL.

The format as the prototype for our callback function:

void got_packet (u_char *args, const struct pcap_pkthdu *header, const u_char *packet);

The function has a void return type. This is logical, because pcap_loop O wouldn't know how to handle a return value anyway. The first argument corresponds to the last argument of pcap_loop O. Whatever value is passed as the last argument to pcap_loop O is passed to the first argument of our callback function every time the function is called. The second argument is the pcap header, which contains information about when the packet was sniffed, how large it is, etc. A packet contains many attributes, so as you can imagine, it is not really a string, but actually a collection of structures (for instance, a TCP/IP packet would have an Ethernet header, an IP header, a TCP header, and lastly, the packet's payload). This u_char is the serialized version of these structures.

Typecasting

Ethernet = (struct sniff_Ethernet*) (packet);

ip = (struct sniff_ip*) (packet + size_ethernet);

tcp = (struct sniff tcp*) (packet + size_Ethernet + size_ip);

Payload + (u_char*) (packet + size_Ethernet + size ip + size_tcp);

For instance, a sniff_Ethernet structure has a size of 14 bytes. A sniff_ip structure is 20 Bytes, and likewise a sniff_tcp structure is 20 bytes. The u_char pointer is really just a variable containing an address in memory. That's what a pointer is; it points to a location in memory. For the sake of simplicity, we'll say that the address this pointer is set to is the value X. Well, if our three structures are just sitting in line, the first of them (sniff_Ethernet) being located in memory at the address X, then we can easily find the address of the other structures.

Variable	Location (in Bytes)
sniff_ethernet	X
sniff_ip	X+14
sniff_tcp	X+14+20
Payload	X+14+20+20

The sniff_Ethernet structure, being the first in line is simply at location X. siniff_ip,who follows directly after sniff_Ethernet, is at the location X, plus however much space sniff_Ethernet consumes(14 in this examples). Sniff_tep is after both sniff_ip and sniff_Ethernet, so it is location at X plus the sizes of sniff_ethernet and sniff_ip (14 and 20 bytes, respectively). Lastly, the payload(which isn't really a structure, just a character string) is located after all of them.

CELL FORMATION

In this module the captured packets are converted into fixed size ATM cell(53 Bytes) payload and header. Here Segmentation & Reassembly concept is implemented in order to segment the segment the captured frames into ATM cells for transmission and to reassemble the frames from ATM cells on reception.

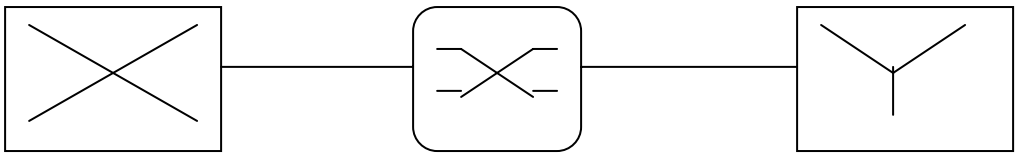
For example, Suppose a 450 byte packet is passed by an upper layer protocol to the CS. The CS pads the packet (if necessary) so that its length is a multiple of 48,minus 8(length -8 mod 48). In this case 22 bytes of padding will be added yielding 472 bytes.

The CS then adds an 8 bytes trailer containing a 4 byte error detecting and a 2 byte length field set in this case to 450. The padded packet with CS trailer appended is thus always an even multiple of 48 bytes in length and is sometimes called an AAL5 protocol data unit(PDU) or simply an AAL5 frame. AAL5 frames are passed by the CS to the SAR sub layer for segmentation into 53 bytes ATM cells. For AAL5 each ATM cell consists of a standard 5 byte cell header followed by a 48 bytes payload. In this example, the SAR sub layer segments the 480 bytes of the AAL5 frame into 10 ATM cells, and then schedules them for transmission.

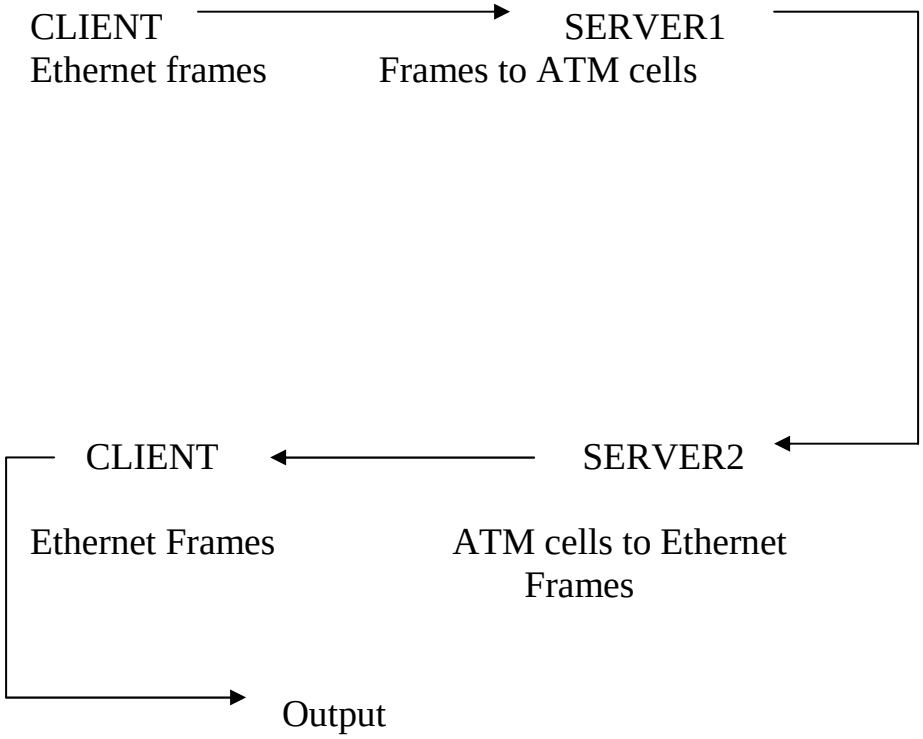
As cells arrive at the other end of the connection, the receiving SAR sub layer strips the 5 bytes cell headers and catenates the 48 bytes payloads in a reassembly buffer. When end of frame is indicated (by a bit in the last cell header), the reassembled frame is passed to the CS. The CS checks the frame for validity and passes the frame and a status indicator to the layer above it.

ATM ROUTING:

This module performs the operation of routing the packets from the server to the destined client as per the destination IP address. Here in the project PVC connection is established using the socket where the two sockets operate as server and client.



FLOW OF IMPLEMENTATION



SYSTEM TESTING

INTORDUCTION:

System testing involves two kinds of activities integration testing acceptance testing. Careful planning and scheduling are required to ensure that modules will be available for integration into the evolving software product when needed.

The integration strategy dictates the order in which modules must be available, and thus exerts a strong influence on the order in which modules are written, debugged and unit tested.

Acceptance testing involves planning and execution of functional tests, performance tests and stress tests to verify that the implemented system satisfies its requirements. Acceptance tests are typically performed by the quality assurance and customer organizations.

QUALITY ASSURANCE:

It is planned and systematic pattern of all actions necessary to provide adequate confidence that the item or product conforms to established technical requirements. The purpose of a software quality assurance group is to provide assurance that the procedures, tools and techniques used during product development and modification are adequate to provide the desired level of confidence in the work products.

INSPECTION:

Inspections can be used through out the software life cycle to asses and improve the quality of the various product.

PACKET CAPTURING:

The data captured in the form of packets and the captured packets are written in data link layer.

CELL FORMATION:

The packets are converted into 53 bytes cell of which 48 bytes are payload and 5 bytes are overload in the first server.

CELL TRANSFER:

The cells are transferred to the second server which converts the cells into frames and sends it to the client.

DATA VERIFICATION:

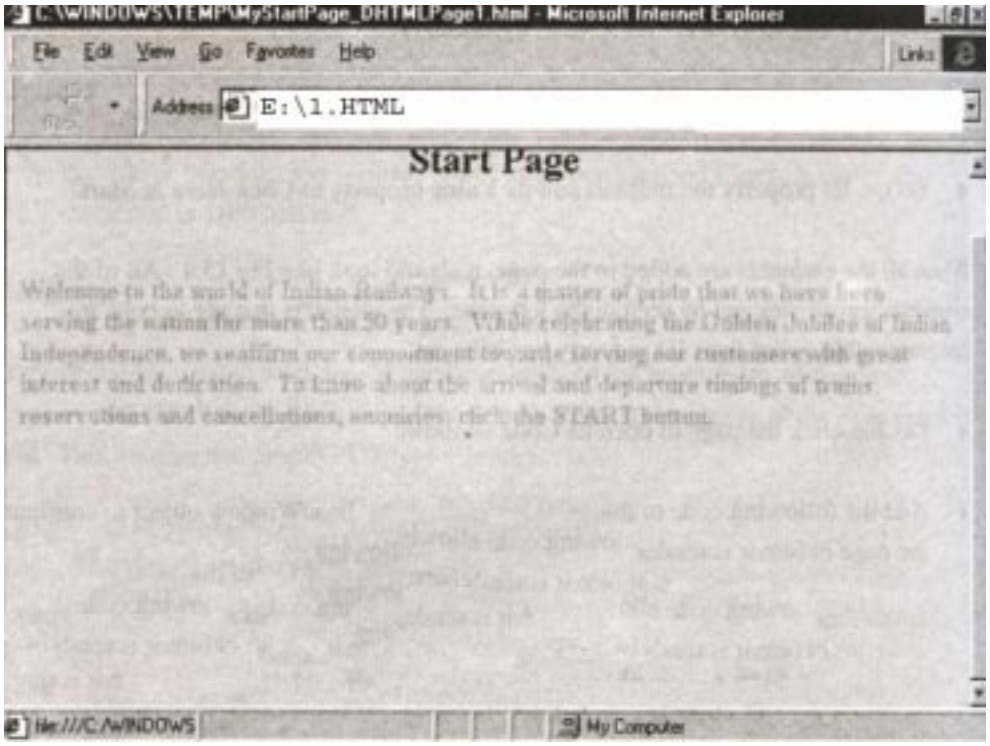
The data received is verified with the source.

USERS MANUAL

CLIENT 1:

In client 1 any file for example 1.html file is created

Client2:



The file 1.html is accessed in client 2 by crossing over two servers.

CONCLUSION

The operation of DLL obtaining the frames and storing then in the memory where the packets are routed to the servers has been performed in the packet capturing module. The packets are converted into fixed size ATM cells as payload and header in the cell formation module. The operation of routing the packets from the server to the destined client according to the destination IP address has been performed successfully.

Thus the objective of the project has benn performed successfully.

FUTURE ENHANCEMENT

- This project can be implemented to layers AAL1-AAL4 also.

- This project can be implemented to support protocols IPX/SPX,

APPLE TALK AND PLATFORM INDEPENDENT languages like JAVA.

REFERENCES

LET US ‘C’ BY YASHWANT KANITNAR

ATM NETWORKS BY RAINE HANDEL

INTERNET WORKING WITH TCP/IP BY DOUGLAS E.COMER

APPENDIX

SOURCE CODE:

PACKET CAPTURING

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <signal.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/socket.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <curses.h>
#include "pcap.h"
#include "sniff.h"
#define LOGFILE "logfile"
#define IPHDR   sizeof(struct iphdr)
#define TCPHDR  sizeof(struct tcphdr)
extern char *optarg;
extern int   optind;
FILE        *logfile;
char        *logfilename;
char        *logbuf;
char        *dev;
int         verbose=0;
int         change_log=0;
```

```
u_char      *buf;
u_short     ports[] = { 21, 23, 25,80, 110, 513, 514 };/* Default Logging Ports*/
u_short     eth_size;
struct pcap *capt_desc;
struct pcap_pkthdr pkhdr;

int main (int argc, char **argv)
{
        int   cmd_parse ;
        char *port_buf;
        char  ebuf[255];
        int i=0;
        signal (SIGINT, close_log_file);
        signal (SIGQUIT, close_log_file);
        signal (SIGKILL, close_log_file);

        if (argc < 2)
          {
                print_usage (argv[0]);
                exit (0);
          }

        while ((cmd_parse = getopt (argc, argv, "vl:i:")) != EOF)
          {
                switch (cmd_parse)
                  {
                  case 'i':
                        dev = optarg;
                        break;
                  case 'l':
                        change_log++;
```

```
                    logfilename = optarg;
                    break;
            case 'v':
                    verbose++;
                     break;
            case '?':
            default:
                    print_usage (argv[0]);
                    exit (0);
            }
  }

fflush(stdout);
capt_desc = pcap_open_live (dev, 8024, 1, 1000, ebuf);
if (capt_desc == NULL)
  {
          fprintf (stderr, "Sorry . Can't open device %s\n\n", dev);
          exit (0);
  }

switch (pcap_datalink (capt_desc))
  {
  case DLT_NULL:
          eth_size = 4;
          break;
  case DLT_EN10MB:
  case DLT_EN3MB:
          eth_size = 14;
          break;
  case DLT_PPP:
          eth_size = 4;
```

```
                break;
        case DLT_SLIP:
                eth_size = 16;
                break;
        case DLT_FDDI:
                eth_size = 21;
                break;
        case DLT_RAW:
                eth_size = 0;
                break;
        default:
                fprintf (stderr, "\n Unknown device .... Aborting ..\n\n");
                exit (0);
        }

    if (change_log)
    {
        if ((open_log_file () == ERR))
            {
                fprintf (stderr,
                        "\nCannot Open Log File ... Aborting ..\n\n");
            }
    }

     while (1)
       {
                buf = (u_char *) pcap_next (capt_desc, &pkhdr);
/*              printf("FRAME IS MOVING");

     while(buf[i] != NULL)
{
```

```
        printf(" %x ",buf[i]);
        i++;
}
i=0; */
                fflush(stdout);
                if ((buf != NULL) && ((pkhdr.len - eth_size) >= TCPHDR))
                  {
                        do_logging ();
                  }
           }

        return (SUCCESS);
}


int open_log_file (void)
  {
        if ((logfile = (fopen ((change_log) ? logfilename : LOGFILE , "w"))))
        return (SUCCESS);
        return (ERR);
}

void close_log_file (int sig)
{
        if (fclose (logfile) == EOF)
        fprintf (stderr, "\nSorry...Can't close log file ... Aborting !\n\n");
        exit (0);
}

int
do_logging (void)
```

```
{
        int istcp = 0, logging = 0, i;
        int sp;

        ip  = (struct iphdr *)  (buf + eth_size);
        tcp = (struct tcphdr *) (buf + sizeof (struct iphdr) + eth_size);

        if ((ip->protocol == IPPROTO_TCP))
          {
                /* if ((pkhdr.len - (eth_size + IPHDR)) < sizeof(struct tcphdr)) */
                istcp++;
          }

        if (istcp)
          {
                for (i = 0; ports[i] != (u_short) NULL; i++)
                  {

                        if (ports[i] == ntohs (tcp->th_sport)
                            || ports[i] == ntohs (tcp->th_dport))
                          {
                                logging++;
                          }
                  }

                if (logging)
                  {
                        struct packet *pkt;
pkt =(struct packet *)  malloc ((sizeof (pkt) + IPHDR + TCPHDR));
logbuf = (char *) (buf + sizeof (struct iphdr) + sizeof (struct tcphdr) + eth_size);
```

```
                        pkt->sourceip = ip->saddr;
                        pkt->dstip = ip->daddr;
                        pkt->sourceport = tcp->th_sport;
                        pkt->dstport = tcp->th_dport;
                         //   puts(pkt->sourceip);


                       //  printf("\n address %l\n",pkt->sourceip);
    // if(pkt->sourceip!=ntohs(192.168.0.6))
                           print_data (pkt, logbuf, tcp);
                free (pkt);
                        memset (buf, 0x0, sizeof (buf));
                    }
            }
}

void print_data (struct packet *pkt, char *logbuf, struct tcphdr *tcp)
{
        char dst[4012];
        char flag ;
        int i;
        struct in_addr in;
        in.s_addr = pkt->sourceip;

        fprintf ((change_log) ? logfile : stdout , "\n.ooOO %s:%i  ----> ", inet_ntoa (in),ntohs (pkt->sourceport));

        in.s_addr = pkt->dstip;

        fprintf ((change_log) ? logfile : stdout , "%s:%i ", inet_ntoa (in), ntohs (pkt->dstport));
```

```
/* More Interesting Flags */

if      (tcp->th_flags & TH_SYN)
         flag = 'S';
else if (tcp->th_flags & TH_FIN)
         flag = 'F';
else if (tcp->th_flags & TH_RST)
         flag = 'R';
else if (tcp->th_flags & TH_ACK)
         flag = 'A';
else
         flag = '.';

    fprintf ((change_log) ? logfile : stdout , "%c seq %u win %u ack %u OOoo.\n ",
        flag, tcp->th_seq,
        tcp->th_win, tcp->th_ack);

if (verbose)
  {
        memset (dst, 0, sizeof (dst));

        for (i = 0; i <(pkhdr.len - (sizeof (struct iphdr) +
                        sizeof (struct tcphdr) + eth_size)); i++)
            {
                    if      (isprint (logbuf[i]))
                          dst[i] = logbuf[i];
                    else if (dst[i] == '\n' || dst[i] == '\r')
                          dst[i] = '\n';
                    else
                             dst[i] = '.';
```

```
                }
        i++;
        dst[i] == '\0';

        for (i = 0; i < strlen (dst); i++)
                fputc (dst[i], (change_log) ? logfile : stdout);

                aal5(dst);
                 } /* if (verbose) */

        fflush ((change_log) ? logfile : stdout);
}

void  print_usage (char *executable_name)
{
        printf ("\n %s [-i interface] [-l logfile] [-v]\n\n", executable_name);
}


int aal5( char *data)
{
int k,padd,i,q=0,j,s=0,div,end,r=0;
int len;
len=strlen(data);
padd=(48-(len%48));
for(i=len;i<=(len+padd);i++)
data[i]='0';
end=i;
data[++i]='u';
data[++i]='c';
data[++i]=len;
```

```
data[++i]='r';
atmfor(data,end);
}
```

THE ATM CELL FORMATION

```
struct atm
{
char cdata[48];
struct atm *next;
};
 atmfor(char *data,int end)
{
struct atm *cell,*head,*pre;
struct sockaddr_in client;
int i,j,s=0,sid,sd,len,ns;
client.sin_family=AF_INET;
client.sin_port=12345;
client.sin_addr.s_addr=inet_addr("127.0.0.1");

head=(struct atm *)malloc(sizeof(struct atm));
pre=head;

for(i=1;i<=end/48;i++)
{
cell=(struct atm *) malloc(sizeof(struct atm ));
for(j=0;j<48;j++)
{
cell->cdata[j]=data[s];
s++;
}
```

```
sid=socket(AF_INET,SOCK_STREAM,0);
connect(sid,(struct sockaddr *)&client,sizeof(client));
write(sid,cell->cdata,sizeof(cell->cdata));
printf(" no %d -> %s\n",i,cell->cdata);

pre->next=cell;
pre=cell;
}
printf("\n  no of cell %d\n",i);
cell->next=NULL;
cell=head;

}
```

Writing the packet which is captured.c

```
#include<stdio.h>
#include<net/if_arp.h>
#include<sys/types.h>
#include<string.h>
#include<unistd.h>
#include<signal.h>
#include"sniff.h"
#include<net/ethernet.h>
#include<sys/socket.h>
#include<sys/errno.h>
extern int datalink;
extern char *device;
extern pcap_t *pd;
extern int rawfd;
```

```
extern int snaplen;
extern int verbose;
void cleanup(int);
void open_pcap(void);
void writepcap(void);
#include<sys/socket.h>
#include<sys/types.h>
#include<arpa/inet.h>
#include<stdio.h>
#include "write.h"
int main()
{
 FILE *fp;
 int sd,ns,i,j=0,l;
 char buf[48],buff[1500];
 struct sockaddr_in sock;
 int len;
 fp=fopen("nini.txt","w+");
 sd=socket(AF_INET,SOCK_STREAM,0);
 sock.sin_family=AF_INET;
 sock.sin_port=12345;
 sock.sin_addr.s_addr=htonl(INADDR_ANY);
 bind(sd,(struct sockaddr * )&sock,sizeof(sock));
 l=listen(sd,1);
printf(" lis %d",l);
buff1=buff;
for(;;)
 {
len=sizeof(sock);
 ns=accept(sd,(struct sockaddr *)&sock,&len);
 read(ns,buf,sizeof(buf));
```

```
fputs(buf,fp);

for(i=0;i<48;i++)
printf("%c",buf[i]);
printf("\n");
printf("%d",strlen(buf));
close(ns);
}
close(sd);
fclose(fp);
system("./Makefilew");
system("./write.o -i eth0 -l sri.txt -v");
}
```

WORKING ENVIRONMENT

WINDOWS 2000 OR WINDOWS XP IN CLIENTS.

LINUX IN SERVERS.