The Abdus Salam
International Centre for Theoretical Physics

United Nations
Educational, Scientific
and Cultural Organization

International Atomic
Energy Agency

310/1779-6

Fourth Workshop on Distributed Laboratory Instrumentation Systems
(30 October - 24 November 2006)

Internet Protocol Architectures and Network Programming

Abhaya S. INDURUWA
Department of Computing
Canterbury Christ Church University
North Holmes Road
Canterbury CT1 QU
U.K.

Internet Protocol Architectures and Network Programming

Abhaya Induruwa*

Department of Computing
Canterbury Christ Church University
Canterbury
United Kingdom

Supporting material for the
lectures given at:
The Fourth Trieste Workshop on
Distributed Laboratory Instrumentation Systems
Trieste, Italy 30 October – 24 November 2006

LNS

*asi1@canterbury.ac.uk

Abstract

This Chapter is intended to give a broad overview of the Internet and its architecture comprising network protocols, standards, hardware and supporting technologies. Various protocol architectures are presented in order to be able to understand the most desirable features that are required in a protocol to support the characteristics of modern day communication systems. The Internet Protocol (IP) architecture and its components used in time critical data communication are discussed.

Also discussed are High Speed LANs based on ATM technology and the new variants of Ethernets running at Gigabit speeds. More recent attempts based on WAP to deliver Internet content and services to hand-held mobile devices and the deployment of ADSL to deliver Internet to the home user are introduced.

Bluetooth, a wireless communication technology capable of location/ device aware connection and switching, and is aiming to take the place of wires for connecting peripherals, telephones, computers, is also briefly discussed.

The lectures also briefly cover the features in Java™ that enable todays network programmers to implement sophisticated networking applications using Java APIs.

Contents

1 Introduction

Computer networks have become an integral and indispensable part of scientific as well as public life today. Over the last couple of decades data networks have changed their character from a few slow speed point to point links to a high speed data communication backbone supporting full multimedia information transfer. The last few years have seen a phenomenal rise in the interest in the collection of computers and networks known as the **Internet**. The **Web** (World Wide Web) has truly transformed the access to information content, and their delivery over the Internet.

More recently the emergence of hand-held **PDAs** (Personal Data Assistants) including wireless phones based on **WAP** (Wireless Application Protocol), and **ADSL** (Asynchronous Digital Subscriber Line) technology that supports high speed data transfer rates over twisted pair subscriber lines, have revolutionised the provision of public access to the Internet and its information services.

In parallel with these advances in technologies associated with the content delivery over the Internet, **Java** and other *object oriented* programming language variants have given the programmer the opportunity to bring to life the hitherto static multimedia Web presentations by adding 'dynamic content' to the Web pages. As a results the Web pages of today have become *live*; they have become *interactive* and *animated*. The static 2D images on Web pages have now become 3D animated images.

The underlying network architecture supporting the above capability is based on the **TCP/IP** protocol suite which lets hundreds of millions of computers and hundreds of thousands of computer networks spread across hundreds of countries to be internetworked to form the global Wide Area Network (WAN), the Internet, which is now used by hundreds of millions of users. The familiar application protocols like **FTP**, **HTTP**, **SMTP**, etc., make use of TCP/IP to achieve a reliable stream transport across inherently unreliable network links. TCP/IP is also the basis for **client/server** computing. Other more specific protocols such as the **RTP** (Real-time Transport Protocol), together with **RTCP** (Real-time Transport Control Protocol), have been devised to facilitate the communication of Real Time data over computer networks which have been designed and built to guarantee the delivery of time insensitive bursty data. **IPv6** is emerging as the next generation Internet Protocol with all the features necessary to support the delivery of multimedia and other real time data services at high speed.

At the Local Area Network (LAN) level, the popular **Ethernet** running at 10 Mbps has already been enhanced to operate at 100 Mbps (**Fast Ethernet**) and 1000 Mbps (**Gigabit Ethernet**) thus making it suitable as a delivery mechanism of Real Time traffic to the desktop. Work has already begun to formulate 10 Gbps Ethernet specifications which will initially be deployed to enhance the high speed local backbones.

Wireless bearer technologies have matured in the last couple of years to an extent that now they take the place of wired connections. Technologies such as Bluetooth and WiFi, which are capable of location/devices aware switching have started to revolutionise the way public access to the Internet is provided.

2 The Internet

The one and only global network of interest to the whole scientific community of the world today is the **Internet**, which is based on the **Internet Protocol (IP)** at its **network layer** and **Transmission Control Protocol (TCP)** at its **transport layer**.

The Internet is a network of computer networks and computers. It interconnects computers of all types and specifications without regard to their operating system or hardware specifics. It interconnects all types of computer networks irrespective of their topology and other physical characteristics. The Internet spans across almost all countries in the world disregarding their socio-economic status and political boundaries. The one thing that binds them onto one single network is the Internet Protocol suite, **TCP/IP**.

TCP/IP is designed to provide a reliable stream transport service across unreliable network links to the application level protocols that use it. TCP/IP is also the basis for client server networking and distributed computing.

3 Network Classification

A computer network is a collection of co-operating computers interconnected by one or more transmission paths for the purpose of transfer and exchange of data between them. Today these networks span the entire globe and belong to many different nations and network operators. Such networks can be classified in many ways depending on the switching mechanism, transmission speed, etc, [3, 40, 42]. They can also be classified on the basis of their geographical coverage or network topology.

3.1 Geographical Coverage

Networks can be classified into many categories depending on their geographical coverage. The following lists them in the order from the smallest to the largest.

1. Desktop Area Networks (DANs)[1]
2. Controller Area Networks (CANs)[2]
3. Home Area Networks (HANs)[3]
4. Storage Area Networks (SANs)[4]
5. Local Area Networks (LANs)
6. Metropolitan Area Networks (MANs)
7. Wide Area Networks (WANs)
8. Global Area Networks (GANs)[5].

[1] a classification arising out of delivering ATM or IP to the desktop at 25 Mbps or above. DANs are used to interconnect devices such as a camera, a telephone and workstations.

[2] ISO 11898 specifies CAN bus up to 1 Mbps and ISO 11519 specifies CAN bus up to 125 Kbps.

[3] mainly due to the proliferation of Bluetooth technology.

[4] networked ultra high capacity disk storage devices.

[5] the result of worldwide GRID initiatives.

There is a significant level of deployment of all or some of the above even in developing countries (most notably LANs and WANs with DANs just appearing to support desk top video conferencing) and hence should be of interest to research scientists.

3.2 Network Topology

Networks can be classified according to their topology in the following manner [39, 40]:

- Bus Topology (eg. CSMA/CD; Ethernet)
- Ring Topology (eg. Token Ring, FDDI)
- Star Topology (eg. ArcNet, Switched networks such as Fast Ethernet)
- Mesh Topology (eg. Telephone network).

The bus topology has been used initially in the Ethernet (broadcast) and later in Token Bus and the DQDB (Distributed Queue Dual Bus).

The token passing mechanism shown in Figure 1 has originally been used in the Token Ring and later in FDDI.

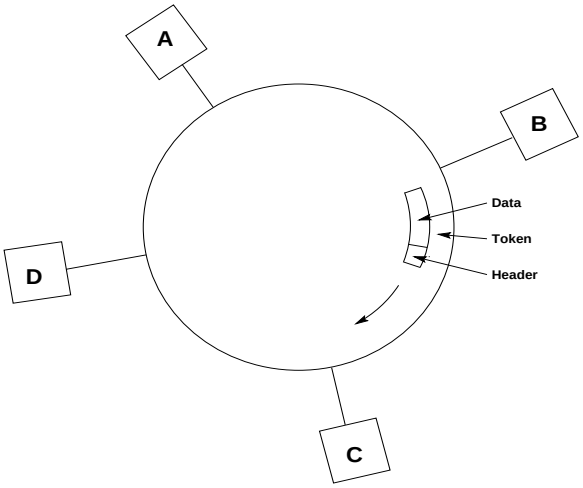


Figure 1: Token Passing Ring

If however networks are categorised according to their transfer mechanism, then the following classification results.

1. Broadcast Networks

   Figure 2 shows a broadcast network which is used to broadcast from 1 to many. CSMA/CD (Carrier Sense Multiple Access/Collision Detect), commonly known as Ethernet, is a Medium Access Control (MAC) protocol which is used to broadcast on a bus topology.

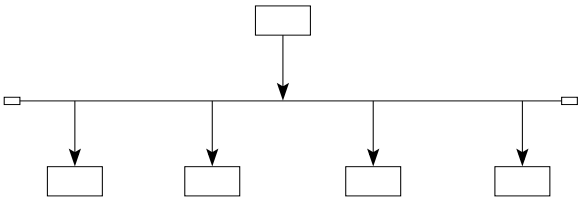


Figure 2: Broadcast Network

2. Switched Networks

   In Figure 3 is shown a switched network which is used to switch data from 1 to 1, 1 to many, or many to many. The telephone network is an example of a circuit switched network. It can be used to support applications such as tele conferencing and video conferencing (using ISDN) which involves the switching of many to many.

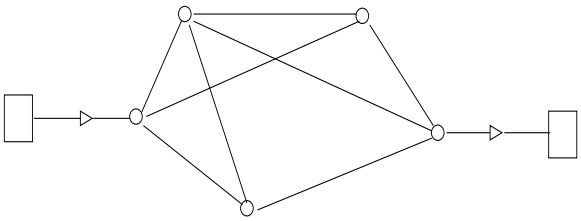


Figure 3: Switched Network

   Although initially switched networks were mainly used in telecommunication networks, today because of its superior performance, switched technologies are increasingly used in LANs (for example switched Ethernet) and ATM networks.

3. Hybrid Networks

   Figure 4 shows a hybrid network which consists of a switched part and a broadcast part.

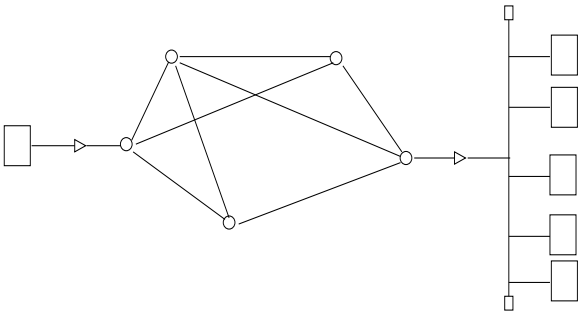


Figure 4: Hybrid Network

4 Network Architecture

The topology, transmission mechanism, and a protocol which manages the transmission mechanism together define a network architecture.

4.1 What is a Network Protocol?

A network protocol is used to facilitate the transmission of data between a sender (transmitter) and a recipient (receiver) across a data communication network in an agreed manner. Over the years several different network protocol architectures have evolved, the most notable being:

- ISO - OSI (Open Systems Interconnect) Reference Model (OSI-RM)
- IP (Internet Protocol suite comprising TCP/IP)
- ISO 8802.X (for LANs and MANs - same as IEEE 802.X)

In addition to the above there are hundreds of vendor specific protocols such as:

- IBM's SNA (Systems Network Architecture)
- DEC's DNA (DEC Network Architecture),

which are proprietary and hence are not truly interoperable.

Because todays networks span the globe, it is important to utilise standard protocols to ensure interoperability and to facilitate seamless data communication over networks belonging to many different network operators. This has been the major objective of the many national, international and professional bodies such as the American National Standards Institute (ANSI), International Standards Organisation (ISO), International Telecommunication Union (ITU), Institute of Electrical and Electronic Engineers (IEEE)[1], Internet Engineering Task Force (IETF) and the ATM Forum which are involved in preparing standards.

Figure 5 shows the layered architectures of the protocols developed by some of the standards bodies above.

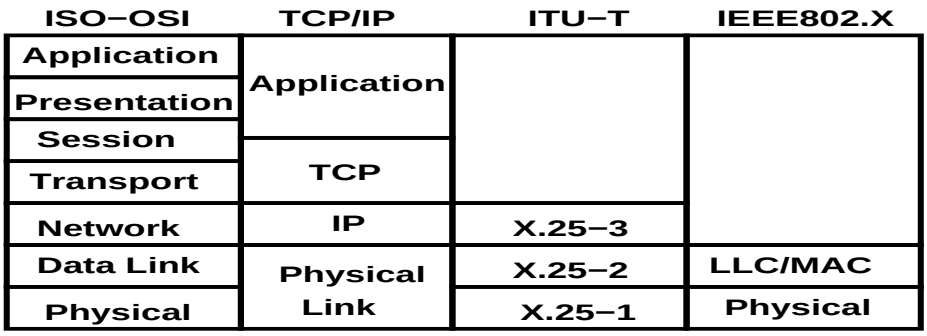


Figure 5: Layered Architectures of different Protocols

[1] http://grouper.ieee.org/groups/802/index.html

In Figure 6 is shown the architecture of the IEEE 802 family of standards for LANs and MANs. These standards cover only LLC (Logical Link Control) and MAC (Medium Access Control) protocols. It is seen that over the years the copper COAX (coaxial cable) has been replaced by UTP (Unshielded Twisted Pair), STP (Shielded Twisted Pair) and Optical Fibre (OF) for wired connections. The recent standards are mainly wireless in order to support mobile connectivity.

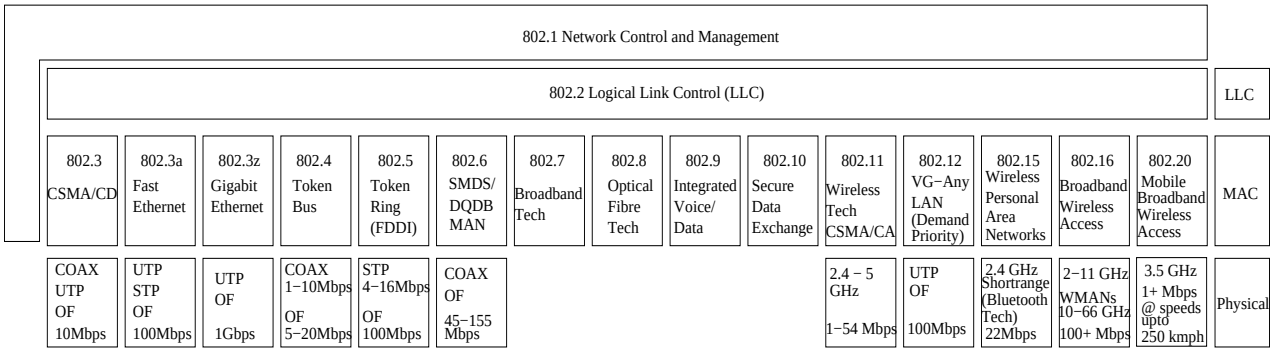


Figure 6: The Architecture of IEEE 802 LAN/MAN Standards

FDDI is a standard developed by ANSI-ASC X3T9 (Accredited Standards Committee) and provides services specified by the ISO Data link and Physical layers (ISO 9314). The key features of FDDI are 100 Mbps data rate, use of optical fibre (*multi mode fibre, single mode fibre, low cost fibre*), the token ring style MAC protocol and the reconfiguration concept (automatic healing property in case of faults). The FDDI standard however allows the FDDI traffic to be carried on other physical media such as twisted pair copper (CDDI). FDDI-II supports protocols for transmission of real time data [22].

IEEE 802.6 MAN standard specifies a DQDB (Distributed Queue Dual Bus) protocol which can support data, voice and video traffic. It can also serve as a LAN. DQDB MAN operates on a shared medium with two unidirectional buses that supports data flow in opposite directions. Two methods of gaining access to the medium depending on the type of traffic have been specified. In the first method a node on the DQDB subnetwork can queue to gain access to the medium by using a distributed queue or by requesting a fixed bandwidth through a pre-arbitrated access method. Data is transmitted in fixed size units called slots of length 53 bytes (52 bytes data + 1 byte access control field).

DQDB private networks are connected to the public network by point to point links. A DQDB MAN can typically range up to more than 50 km in diameter and can operate at a variety of speeds ranging from 34 Mbps to 150 Mbps.

DQDB has been conceived to integrate data and voice over a common set of equipment, thereby reducing maintenance and administrative costs. B-ISDN (Broadband Integrated Services Digital Network) is an attempt to provide universal and seamless connectivity for multimedia services. IEEE 802.6 MAN has been designed to provide an interim solution and to act as a migration path to B-ISDN.

Nodes in a DQDB subnetwork are connected to a pair of buses flowing in opposite directions and can operate in one of two topologies, namely; *open bus* (Figure 7) or *looped bus* (Figure 8) (open bus topology is similar to Ethernet and looped bus topology is similar to token ring).

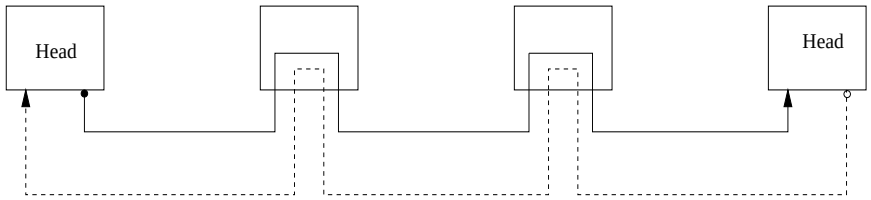


Figure 7: Open Bus DQDB Network

Although the DQDB access layer is independent of the physical medium, the speeds at which DQDB MANs operate demand the use of fibre or coaxial cables. For example, ANSI-DS3 operates at 44.736 Mbps over 75 Ω coax or fibre and ANSI SONET STS3 (Synchronous Optical NETwork) operates at 155.52 Mbps over single mode fibre. The ITU-T G.703 operates at 34.368 Mbps and 139.264 Mbps over a metallic medium.

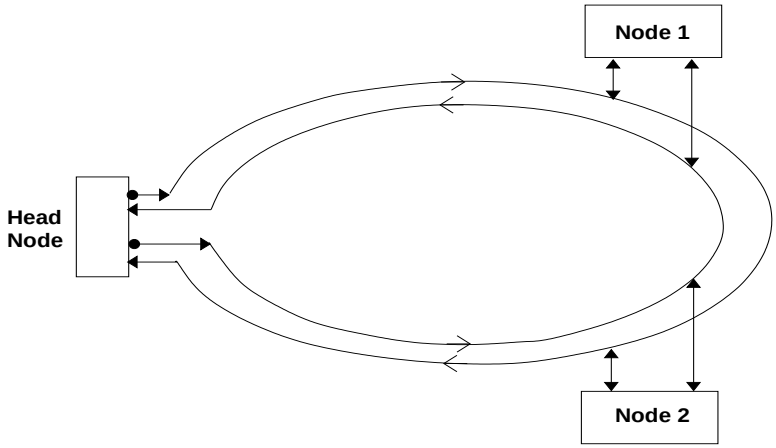


Figure 8: Looped Bus DQDB Network– DQDB Ring

SMDS (Switched Multi-megabit Data Service) which provides a connectionless service has been designed specifically to carry data instead of voice traffic on high speed public wide area networks.

A complete treatment of FDDI, DQDB and SMDS along with LANs and MANs is found in references [39, 40].

4.2 Transmission Mechanism

From the beginning, the voice data in a telephone network had been transmitted in real time using circuit switching techniques. Since circuit switching is not an efficient transmission mechanism for the communication of non-voice data, packet switching techniques have been devised.

4.2.1 Packet Switching

Transmission mechanisms based on packet switching allow the multiplexing of data packets from different sources on the same transmission path thereby making use of the channel bandwidth more efficiently. It also allows the transmission of packets from one source along different paths thus taking care of line congestion and availability problems. When the packets reach the intended destination in whatever path they may have taken, they are reassembled and presented to the user application.

The most widely used protocols which manage the packet transfer across a data network belong to the family of standards conceived by the ITU-T[1] (X.25) and Internet Architecture Board (IP). However **the variable packet lengths and the multiplexing technique introduce jitter making their Quality of Service (QOS) unacceptable for real time applications**. The inherent limitation of X.25 in high speed data transfer has been removed to some extent in the Frame Relay [37] transfer mechanism.

4.2.2 Frame Relay

Frame Relay offers a high speed version of packet switching and has the potential of operating effectively at much higher speeds compared to X.25, reaching speeds of 45 Mbps. Frame relay is well suited to high speed data applications, but not suited for delay sensitive applications such as voice and video because of the variable length of frames [37, 38].

4.2.3 Cell Switching

Cell Relay is a transmission mechanism that combines the benefits of time division multiplexing with packet switching. It operates on the packet switching principle of statistically interleaving cells on a link, on an 'as required basis', rather than on the basis of a permanently allocated time slot. The fixed cell size used enables a reasonably deterministic delay to be achieved across a network. This deterministic nature of cell relay guarantees Quality Of Service (QOS) and hence makes it suitable for all traffic types including real time traffic, within a single network.

ATM (Asynchronous Transfer Mode) has become the dominant form of cell relay. It uses a cell of 53 bytes long (5 bytes header and 48 bytes data) and typically operates at speeds of 155 and 622 Mbps. ATM is delivered to the desktop at 25 Mbps and Gbps platforms are being tested [25]. ITU-T has selected ATM as the transport technique for B-ISDN (Broadband Integrated Services Digital Network) [10, 17, 38].

[1] formerly CCITT – Consultative Committee for International Telephony and Telegraphy

4.3 Physical Media

The most commonly used physical media for wired connections are:

- Optical fibre cables
- Coaxial cables
- Unshielded Twisted Pair (UTP) cables
- Shielded Twisted Pair (STP) cables

Today all of the above media are used to carry data in excess of 100 Mbps speeds. Only the distances they cover are different (for example, optical fibre can operate at gigabit speeds for a few km whereas UTP can operate at 100 Mbps over a distance of 100 m without repeaters).

With the spread of handheld mobile communication devices such as PDAs and 3G mobile phones there is a rapidly growing demand for the delivery of high bandwidth services to these devices. This is reflected in the recent work on wireless standards carried out by various standard bodies and industry consortia [13].

5 Internetworking

By internetworking is meant the process of interconnection of computers and their networks to form a single internet. In other words to make two or more physical networks to logically look and work like one. The Internet (note the uppercase 'I') is such an internet and uses TCP/IP (Transmission Control Protocol /Internet Protocol) protocol suite [45].

TCP is connection oriented and provides a reliable stream transport service. Although TCP is commonly associated with IP (as its underlying network protocol), it is an independent, general purpose protocol that can be adapted to use with other delivery systems. The popularity of TCP has resulted in the development of ISO – TP4, which is a TCP derivative. TCP, together with IP, provides a reliable stream delivery for data traffic.

TCP/IP protocol suite has become the *de facto* standard for open system interconnection in the computer industry. It is used world wide in academic, government, private and public institutions. Some of the reasons for its wide acceptance can be attributed to the following:

- It provides the highest degree of interoperability.
- It encompasses the widest set of vendors' systems.
- It runs over more network technologies than any other protocol suite.

Over the years Unix (and hence Linux) and TCP/IP have become almost synonymous. It has now become part of the operating system kernel. The OS runs a separate process for IP, TCP input/output and UDP input/output.

The proliferation of TCP/IP is such that today there is hardly a single hardware/software platform that does not support TCP/IP.

5.1 Building Internets and Intranets

In building internets and intranets, following hardware devices used to interconnect networks are of major interest [3, 6].

1. Repeaters
2. Bridges and Switches
3. Routers and Brouters
4. Gateways.

Figure 9 shows the use of these components in relation to the ISO-OSI Reference Model.

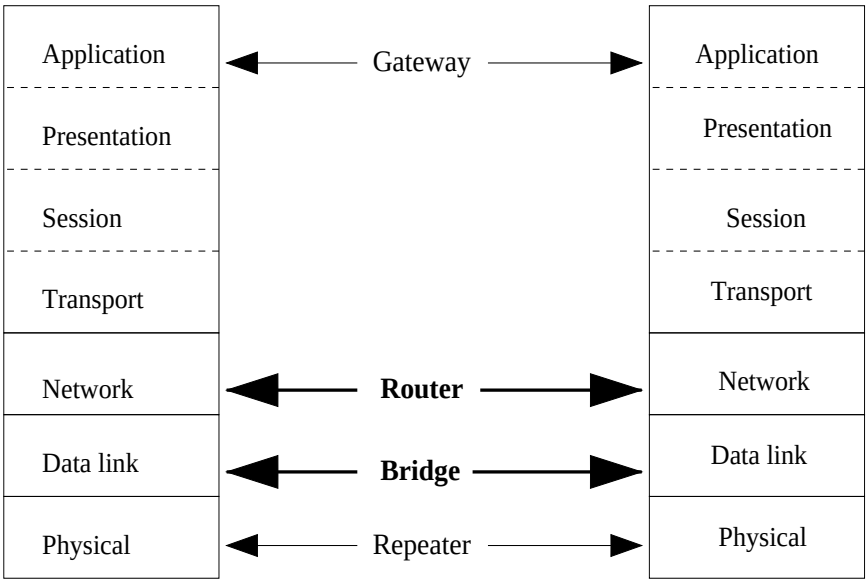


Figure 9: Interconnection Components

5.2 Repeaters

When two networks are to be connected at the lowest level ie. the physical level, an interconnecting device known as a *repeater* is used. A repeater simply takes bits arriving from one network and just forwards (repeats) them on to the other. In some cases a repeater might have to translate between two different physical layer formats, for example from optical fibre to UTP (Unshielded Twisted Pair) cable. This may involve some processing such as signal regeneration of the received signal for noise elimination. However, repeaters pass on the data received without paying attention to the address information.

5.3 Bridges and Switches

Bridges and switches are used to interconnect networks at the medium access control (MAC) layer. Typically this requires the interconnected networks to have identical MAC layers although networks with different but related MAC layers can be interconnected. Since Bridges operate at the MAC layer, they can be used to effectively segment the traffic by filtering the traffic entering one segment from another thereby reducing the unwanted traffic flow on network segments.

Unlike a repeater which replicates electrical signals, bridges replicate packets. They are superior in their function, because they do not replicate noise and errors or malformed frames. Moreover, bridges implement CSMA/CD rules and hence collisions and propagation delays remain isolated without affecting the other segments. In other words, every port on an Ethernet bridge or a switch is on a separate *Collision Domain* or physical network. As a result an almost arbitrary number of Ethernet segments can be connected together with bridges whereas with repeaters the maximum number of 10base5 segments is five giving a total length of 2.5 km. Since bridges hide details of interconnection, a set of bridged segments acts like a single Ethernet.

A bridge can be used to make a decision on which frames are to be forwarded from one segment to another. Such bridges are called *adaptive* or *learning* bridges. They learn over time, about which hosts are connected to which segment. Thus an adaptive bridge builds up the address table automatically without human intervention.

Bridges are often used to improve the performance of an overloaded network by effectively partitioning the network into segments.

The type of bridges used in CSMA/CD LANs are known as *transparent bridges* (IEEE 802.1D) since their presence is not visible to the stations. The type of bridges used to interconnect token rings are called *source routing bridges* (IEEE 802.5) and the routing information is provided by the source station.

5.4 Routers

Routers interconnect networks at the network layer (level 3 in the OSI-RM and the IP layer of the TCP/IP suite) and perform routing functions. This is the main building block in internetworking, that is building internets or intranets, using IP. Most routers now support at least one of the multicast routing protocols, which is an essential functionality to support the delivery of Real Time data over IP.

For performance reasons the functionality of a bridge and a router is combined to form a *brouter*. Whenever possible a brouter bridges data for better efficiency rather than routing.

5.5 Gateways

Gateways are used when networks based on completely different network architectures have to be interconnected at layers higher than the network layer and

when protocol translation is necessary. A typical example is interconnecting two networks based on TCP/IP suite and OSI suite of protocols. An application level gateway is required to support FTP on TCP/IP and FTAM on OSI. From this it should be clear that a different application level gateway is required for every application supported across the interconnected networks.

5.6 Multiport-Multiprotocol Devices

The multiplicity of transmission media and protocols used in today's networks require the use of multiport repeaters and bridges, as well as multiprotocol routers which support more than one protocol stack.

6 A word about the Internet

Internet Protocol is a truly scalable protocol that is used to connect computers to small LANs and to WANs forming a global internetwork. IP is available for almost all computing platforms ranging from the smallest palmtop to the largest ultrasuper computer. Nowadays IP is increasingly becoming available in handheld mobile phones and personal data assistants (PDAs) and also on stand alone micro–controller boards and embedded systems.

The Internet is an ever expanding network of networks. As of this writing (November 2004), it interconnects an estimated 200 million host computers and 2 million computer networks in more than 200 countries. It experiences a staggering growth rate of 100% per year. Figure 10 shows the Internet connectivity in the world in 1997. Today, as a result of the emergence of various transmission technologies including optical fibre, satellite and enhanced communication technologies based on robust high speed modems and DSL (Digital Subscriber Line), almost all the countries in the world have some Internet connectivity. Although it is difficult to keep track of the growth of the Internet in real time any more, Figures 11 and 12 serve to show the scale of world wide Internet deployment.

The DSL, specially ADSL (Asynchronous Digital Subscriber Line) is an innovative modem based communication technology that allows the use of the ordinary telephone line to enjoy the full multimedia capability of the Internet at home delivered at a reasonably high speed (a download speed of $\sim$1 Mbps compared to 56 Kbps using an ordinary modem).

The world wide popularity of the Internet is largely due to the World Wide Web. This is the most pervasive application on the Internet today. Figure 11 shows the staggering growth in the number of web hosts in the world. Over a period of five years this number has risen from near zero to over 35 million (in March 2002). Also this number has more than doubled in a year in the initial period.

Figure 12 shows the explosive growth of the Internet, measured in terms of the number of Internet hosts connected to the Internet.

Both these show evidence of exponential growth in the usage of the Internet. This has required some rethinking of the Internet strategy, specially in relation

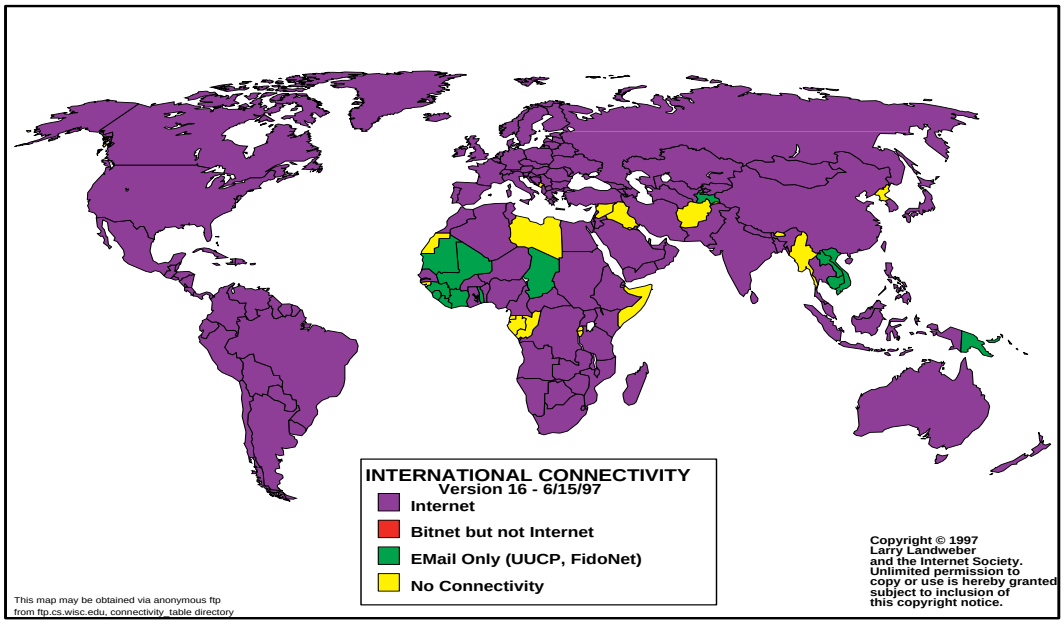


Figure 10: Worldwide Internet Connectivity

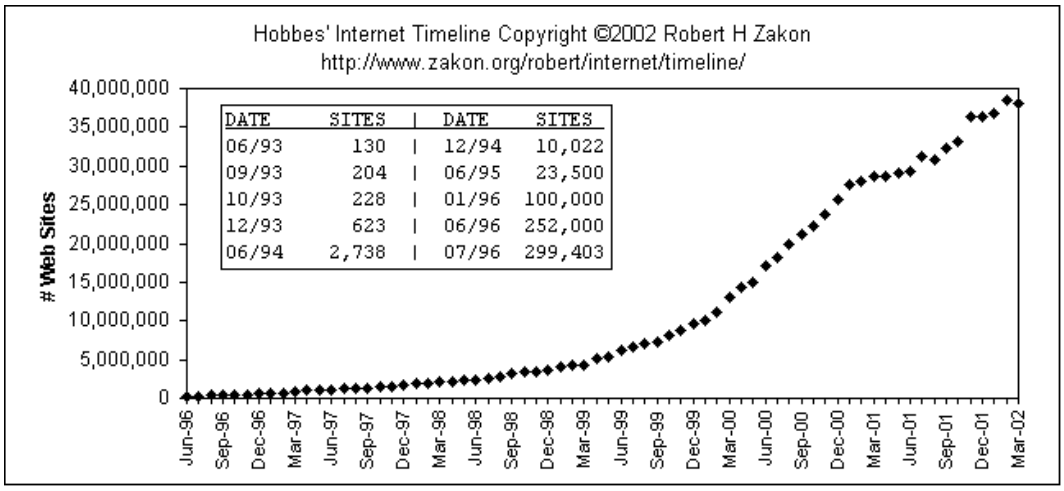


Figure 11: Growth of the number of Web hosts on the Internet

to security, address space, quality of service, etc, in order to make it sustainable in the face of this rapid growth.

Due to this unprecedented popularity and the growth of usage of IP, today IP suite is included as a standard component of all Unix and Unix like (and hence Linux) distributions, making the networking of any computer running Linux much easier. IP is also incorporated into the OSI-RM thus ensuring compliance with ISO standards.

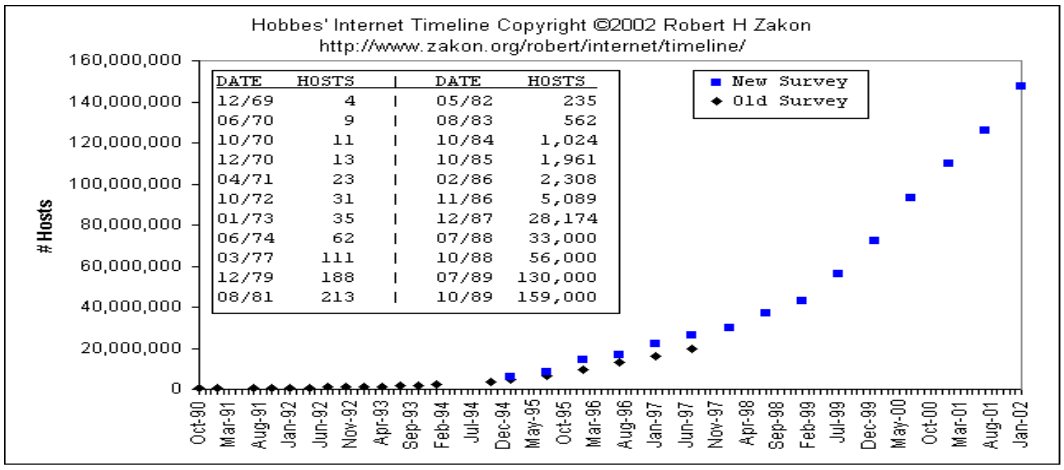


Figure 12: Growth of the number of Internet hosts

7 Internet Protocol Architecture

By far the most successful and the most widely used protocol architecture for LANs and WANs alike is the Internet Protocol (IP) [3, 6, 43].

IP is a layered architecture (see Figure 5 on page 5) consisting of only 4 layers (ISO-OSI RM has 7!). TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are the two transport protocols supported over IP. Both TCP and UDP make the assumption that the link is of acceptable reliability (measured in terms of its BER - Bit Error Rate) and is capable of delivering a packet to the intended destination without the intervention of the upper layers. Some of the services supported in TCP/IP, including RTP are shown in Figure 13.

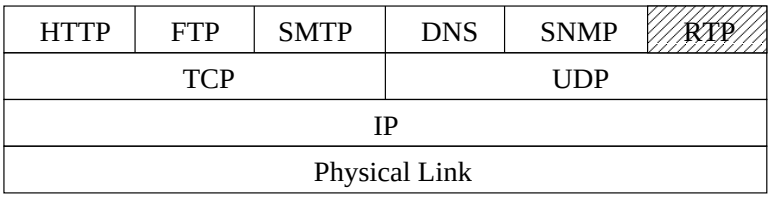


Figure 13: Some Services Supported over IP

In IP there are two transport layer protocols available to programmers (see section 11 on Networking Programming – page 60):

TCP - connection oriented, stream oriented protocol providing a reliable data flow between computers

UDP - connectionless protocol which sends independent packages (called Datagrams) with no guarantees of arrival.

The current version of IP is version 4 (IPv4) and work is underway to define the next generation IP (IPng). Already IPv6 testbeds are operational in many countries providing useful experience with this new protocol which is expected to replace IPv4.

7.1 IP Addressing

IPv4 uses 5 classes of addresses as shown in Figure 14. These map on to IP address ranges shown in Figure 15.

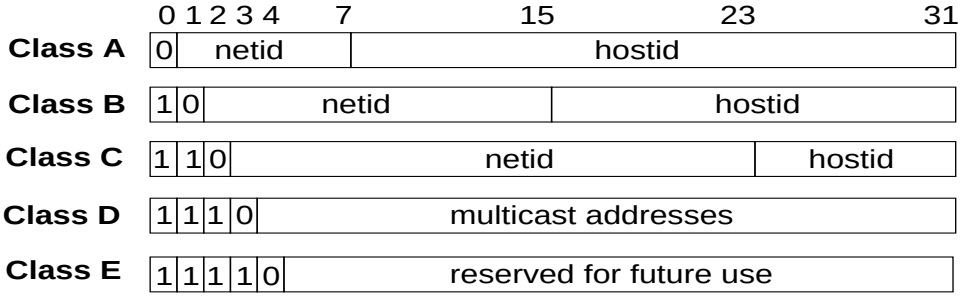


Figure 14: IP Addresses

Address Class	Address Range	Network ID	No of Networks	No of Hosts
Class A	1.0.0.0 – 126.255.255.255	1 – 126	126	16,777,214
Class B	128.0.0.0 – 191.255.255.255	128 – 191	16,384	65,534
Class C	192.0.0.0 – 223.255.255.255	192 – 223	2,097,151	254

Figure 15: IP Address Ranges

An example of IP address allocation in a typical network is shown in Figure 16.

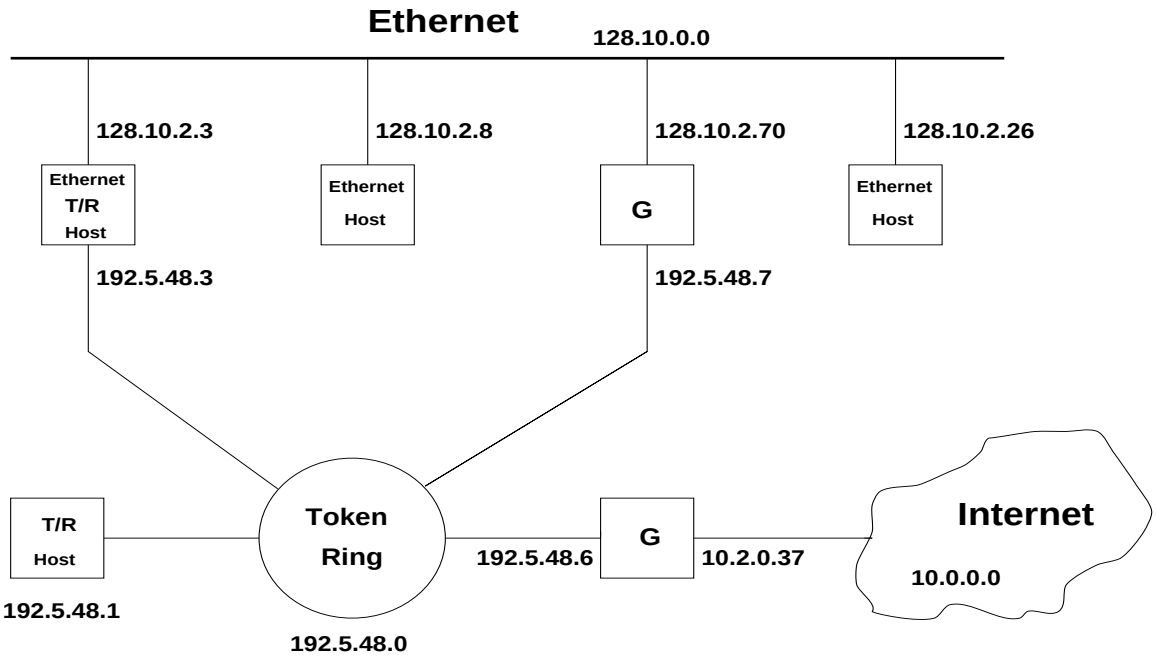
Many hosts have a host name associated with the IP address which can be used to address them. However it must be understood that an IP address does not identify a host. It identifies a network connection to a host because an IP address encodes both a network and a host on that network. Hence if the host is moved to another network, its IP address has to be changed.

No two devices on the global Internet should be allocated the same IP address, although on a private network with no connection to the outside world, arbitrary addresses can be allocated. The address allocation has been initially handled by the Network Information Centre (NIC). Now it is handled by the NIC as well as RARE in Europe and APNIC in the Asia- Pacific region. The addresses must be officially obtained from one of the above before using them on your network.

Nowadays there are Internet Service Providers (ISPs) in many countries (most of the time more than one!) who are allocated blocks of IP addresses by the relevant NIC. These ISPs in turn allocate the IP addresses to their customers.

The explosive growth of the Internet has resulted in the near exhaustion of the address space (a total of 4 giga addresses) provided in IPv4 which uses a 32 bit (4 byte) addressing scheme. IPng (IP new generation), also known as IPv6, is designed to provide among other things an enhanced address space (256 x 10^{36} addresses) using 16 byte (128 bit) long addresses [30][1], [19, 14, 24].

[1]RFCs are available from `ftp://rs.internic.net/rfc`



One address for each network / segment

Internet	**10.0.0.0**	**(Class A)**
Ethernet	**128.10.0.0**	**(Class B)**
Token Ring	**192.5.48.0**	**(Class C)**

Network Hosts

128.10.2.8

192.5.48.1

[**128.10.2.3**
192.5.48.3]

Two addresses for each gateway.

Figure 16: Allocation of IP Addresses on an internet: an Example

7.2 Resolving IP Addresses

An IP address uniquely identifies a host (machine) on a network anywhere on the Internet. Most of the time these IP addresses (in *dotted decimal quad* format) are associated with a host name (or IP name) which is more convenient to human beings but not directly useful to the machines themselves.

The Internet provides **DNS** (Domain Name System – a hierarchical data base of host names and IP addresses) service which maps a given host name to its IP address (address resolution) and an IP address to its host name (reverse lookup).

The Unix utility *nslookup* converts host names to IP addresses and vice versa.

7.3 The Internet Protocol

The Internet Protocol is a connectionless network layer protocol. TCP is a higher layer protocol which sits on top of IP layer. IP provides a connectionless (or datagram type) service with *Best Effort Delivery* to its user. In other words data given to IP (by the layer above) is not guaranteed to be delivered.

The IP datagrams are the encapsulation of data packets passed from the higher layer with an IP header as shown in Figure 17.

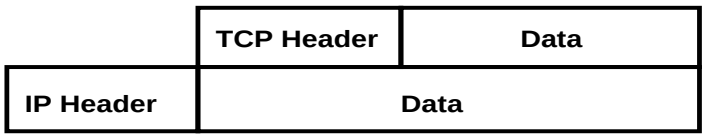


Figure 17: IP Datagram

7.4 Internetworking with IP

Figure 18 shows how IP is used to internetwork two networks running different medium access control mechanisms namely, CSMA/CD and Token Ring.

7.5 IP Datagram Format

The IP datagram format which has a preamble of ten 16 bit words and an option field of variable length is shown in Figure 19.

- Version: 4 bits; current version is IPv4
- IHL: 4 bits; Internet Header Length in 32 bit words. The minimum IHL for a valid header is 5.
- Type of Service: 8 bits; indicates the type and quality of service desired. The first 3 bits are used to define 8 different types including Network Control, Internetwork Control, etc, and the next 3 bits are used to set the Delay (Normal/Low), Throughput (Normal/High) and Reliability (Normal/High). The last 2 bits are reserved for future use.

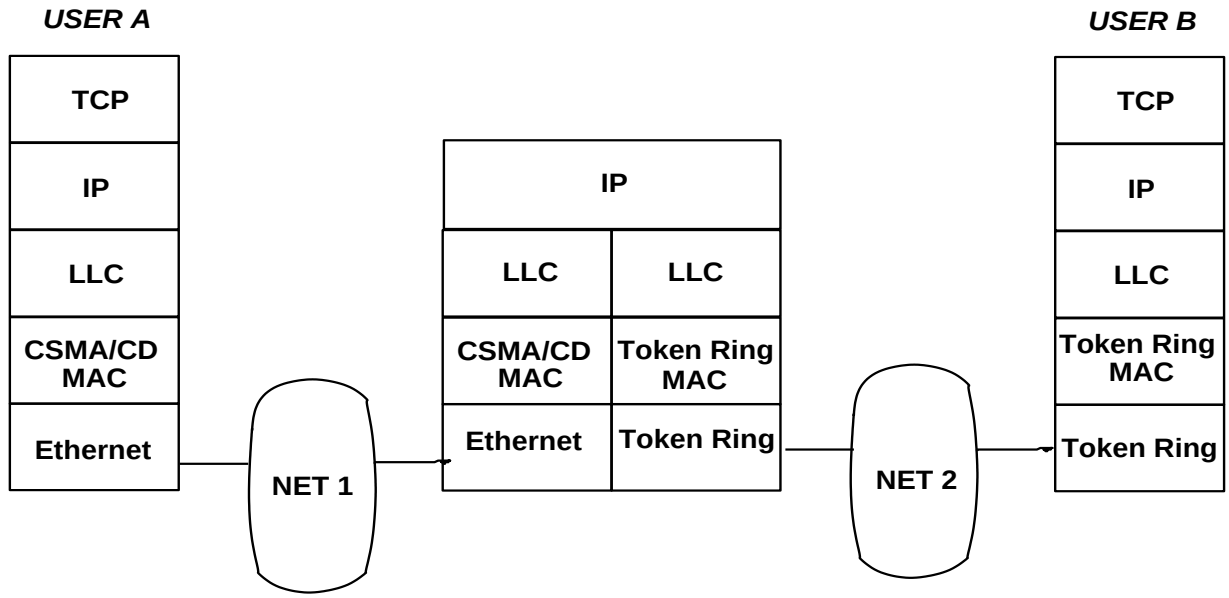


Figure 18: Internetworking Using IP

- Total Length: 16 bits; is the length of the datagram including header and data.
- Identification: 16 bits; a value assigned by the sender to aid in assembling the fragments of a datagram.
- Flags: 3 bits; Bit 0 is reserved and must be 0. Bit 1 is used to set fragmenting policy (May/Don't) and Bit 2 to indicate either last fragment or more fragments to follow.
- Fragment Offset: 13 bits; indicates where in the datagram this fragment belongs. The first fragment has zero offset.
- Time To Live: 8 bits; sets the maximum time the datagram is allowed to remain in the internet. Prevents continuous looping of datagrams.
- Protocol: 8 bits; indicates the Transport Layer Protocol that the data portion of this datagram is passed to. The values for various protocols were originally specified in the Assigned Numbers RFCs until October 1994 [29], but now they are available in an on-line database[1] (RFC 3232 [36]).
- Header Checksum: 16 bits; for header only. Since some header fields change (eg. TTL), this is recomputed and verified at each point that the header is processed.
- Source Address: 32 bits
- Destination Address: 32 bits
- Options: variable; may or may not appear in a datagram.

[1] `http://www.iana.org`

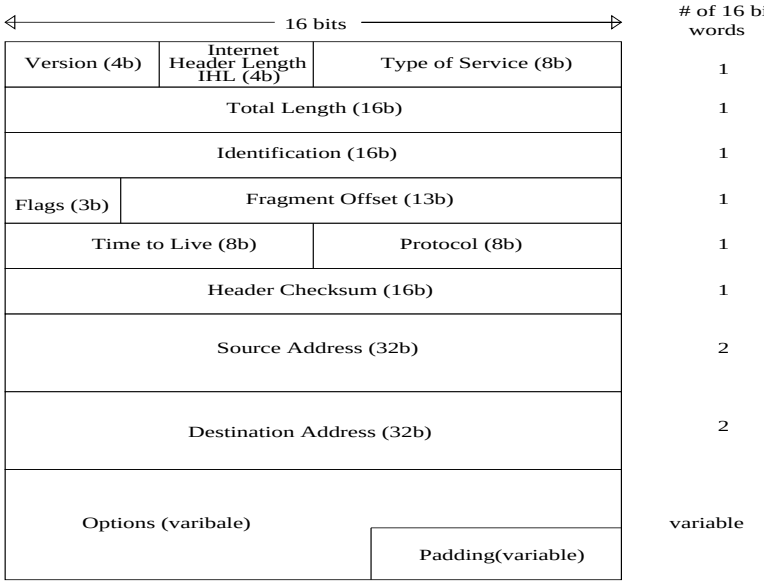


Figure 19: IP Datagram Format

7.6 Brief Description of TCP and UDP

TCP and UDP are the two transport protocols used in the IP architecture [6, 43].

TCP is a connection oriented transport protocol designed to work in conjunction with IP. TCP provides its user (application layer) with the ability to transmit reliably a byte stream to a destination and allows for multiplexing multiple TCP connections within a transmitting or receiving host computer.

Being connection oriented, TCP requires a connection establishment phase (like dialling a number to make a phone call) which is followed by the data transmission phase. A connection is terminated when it is no longer in use. TCP/IP is ideal for the transmission of bursty data. It works on the principle of retransmission of dropped packets which is one of the major contributors to delays in transmission. However, since voice and video data are time sensitive, packet technologies such as TCP/IP cannot guarantee the proper delivery of such data. Figure 20 shows the TCP header format.

- Source Port: 16 bits
- Destination Port: 16 bits
- Sequence Number: 32 bits; is the sequence number of the first data octet in this segment (except when SYN is present). If SYN is present then the sequence number is the Initial Sequence Number (ISN) and the first data octet is ISN+1.
- Acknowledgement Number: 32 bits; If the ACK control bit is set, this field contains the value of the next sequence number the sender of the segment is expecting to receive.
- Data Offset: 4 bits; is the number of 32 bit words in the TCP header. This indicates where the data begins. The TCP header is always an integral number of 32 bits long.

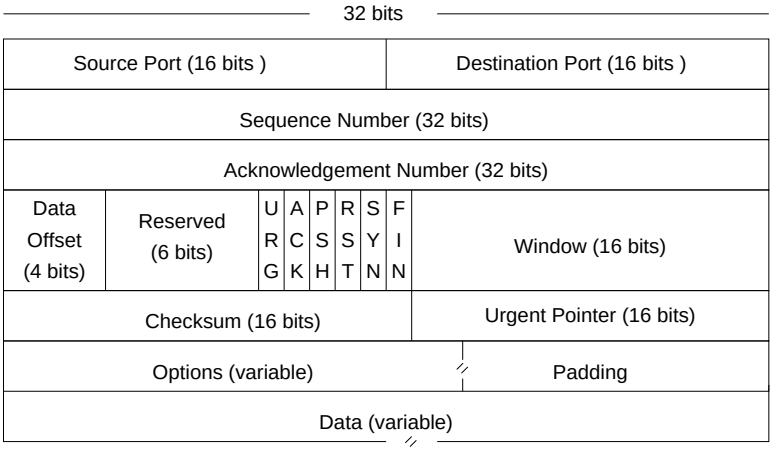


Figure 20: TCP Segment Format

- Reserved: 6 bits; for future use, must be zero.
- Control bits: 6 bits (single value bits).
  - URG: Urgent Pointer
  - ACK: Acknowledgement
  - PSH: Push function
  - RST: Reset the connection
  - SYN: Synchronise sequence numbers
  - FIN: No more data from sender
- Window: 16 bits; the number of data octets the sender of this segment willing to accept.
- Checksum: 16 bits; is the checksum for header and data.
- Urgent Pointer: 16 bits; contains the current value of the urgent pointer as a positive offset from the sequence number in this segment. It points to the sequence number of the octet following the urgent data. This field is only interpreted in segments with the URG control bit set.
- Options: variable

UDP, on the other hand, is a connectionless transport protocol designed to operate over IP. Its primary functions are error detection and multiplexing. UDP does not guarantee the delivery of packets (compare with the ordinary postal service) but guarantees that if a packet is ever delivered in error, such error will be detected (use of checksum). It also allows for communicating with multiple processes residing on the same host computer.

UDP packet format is simple (see Figure 21). It is also fast compared to the use of TCP, since there is no connection establishment phase. Moreover, UDP is important since RTP (Real time Transport Protocol) is supported over UDP.

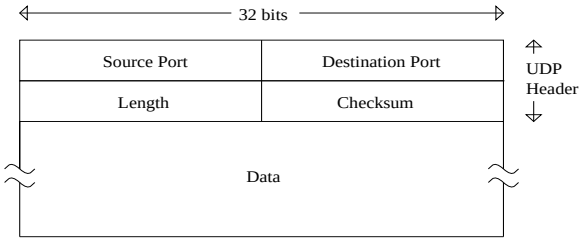


Figure 21: UDP Packet Format

7.7 Client/Server Computing

A computer on a network usually has a single physical network connection but can have more than one application or network service using the network simultaneously to send and receive data from other applications/services resident in computers connected onto the network. In effect the network protocol architecture provides a form of multiplexing/demultiplexing of data streams related to different applications or network services.

Network drivers which form part of the network operating system must ensure that each packet of data or message sent to the destination computer arrives at the intended application. This is achieved by assigning a **Port** which is a unique identifier for an application that uses the network.

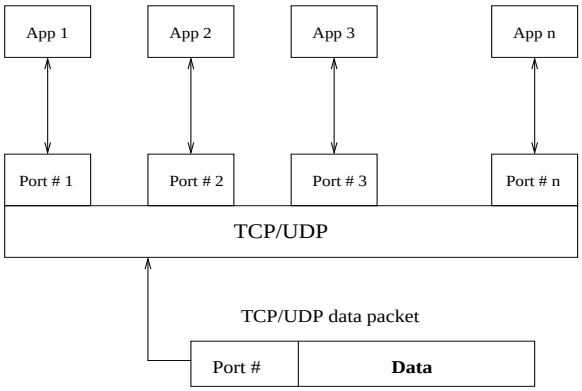


Figure 22: Use of Port Numbers

The IP datagram carries the machine identification (its IP address) and the TCP segment or UDP packet as data. The TCP segment or the UDP packet carries the **Port Numbers** (16 bits long) and the data for the application (Figure 22). On receipt of the TCP segment, the machine (already identified by its IP address) directs the data to the correct application/service associated with the port number attached to that application/service.

7.7.1 Port Numbers

When IP layer passes incoming data to the transport protocol layer (TCP), the TCP passes the data to the correct application process. The application processes,

also known as *network services*, are identified by 16 bit long port addresses.

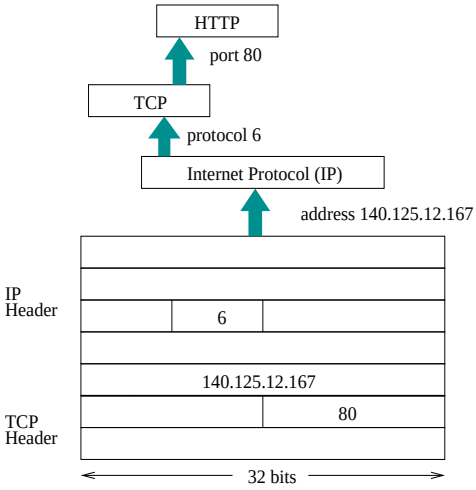


Figure 23: Mapping of Protocol and Port Numbers

Once data are routed through the network and delivered to a specific host, a mechanism must exist to deliver the data to the correct user application or process. This mechanism needs to be able to deliver data to the correct protocols in each layer as the data moves up or down the layers of the TCP/IP stack. This mechanism must also be able to combine data from many applications into a few transport protocols and from the transport protocol to the Internet Protocol. In other words the many data streams must be *multiplexed* into a single data stream before transporting on the network and the data arriving from the network must be *demultiplexed*.

IP uses protocol numbers (field 9 of the IP datagram header – see figure 19 on page 19) to identify transport protocols and the transport protocols use port numbers (fields 1 and 2 of the TCP segment header – see figure 20 on page 20) to identify applications.

Figure 23 shows how the IP address, protocol number and the port number are extracted from the IP datagram header and the TCP segment header.

Some protocol and port numbers are reserved to identify *well known services* such as HTTP and FTP. These are documented in the Assigned Numbers RFC. On a Unix (Linux) system the protocol numbers are defined in */etc/protocols* file. The first few lines of this file from a Linux machine is shown below:

```
# /etc/protocols:
# $Id: net.tex,v 1.1.1.1 2003/11/14 15:06:22 rtime Exp $
#
# Internet (IP) protocols
#
#       from: @(#)protocols     5.1 (Berkeley) 4/17/89
#
```

```
# Updated for NetBSD
# based on RFC 1340, Assigned Numbers (July 1992).

ip      0   IP       # internet protocol, pseudo protocol number
icmp    1   ICMP     # internet control message protocol
igmp    2   IGMP     # Internet Group Management
ggp     3   GGP      # gateway-gateway protocol
ipencap 4   IP-ENCAP # IP encapsulated in IP (officially ‘‘IP’’)
st      5   ST       # ST datagram mode
tcp     6   TCP      # transmission control protocol
egp     8   EGP      # exterior gateway protocol
pup     12  PUP      # PARC universal packet protocol
udp     17  UDP      # user datagram protocol
hmp     20  HMP      # host monitoring protocol
```

On Unix systems the port numbers are defined in */etc/services* file. The 16 bits gives rise to 64 K port addresses. Port numbers below 256 are reserved for well known services such as FTP and HTTP. Port numbers between 256 and 1024 are used for Unix-specific services (such as **rlogin**, which is no longer Unix specific).

The following is a few lines from */etc/services* file:

```
# /etc/services:
# $Id: net.tex,v 1.1.1.1 2003/11/14 15:06:22 rtime Exp $
#
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single
# well-known port number for both TCP and UDP; hence, most entries
# here have two entries even if the protocol doesn’t support UDP
# operations. Updated from RFC 1700, ‘‘Assigned Numbers’’
# (October 1994).  Not all ports are included, only the
# more common ones.
tcpmux          1/tcp           # TCP port service multiplexer
echo            7/tcp
echo            7/udp
daytime         13/tcp
daytime         13/udp
ftp             21/tcp
fsp             21/udp  fspd
ssh             22/tcp          # SSH Remote Login Protocol
ssh             22/udp          # SSH Remote Login Protocol
telnet          23/tcp
# 24 - private
smtp            25/tcp  mail
```

```
# 26 - unassigned
time            37/tcp  timserver
time            37/udp  timserver
whois           43/tcp  nicname
mtp             57/tcp              # deprecated
finger          79/tcp
www             80/tcp  http        # WorldWideWeb HTTP
www             80/udp              # HyperText Transfer Protocol
link            87/tcp  ttylink
kerberos        88/tcp  kerberos5 krb5  # Kerberos v5
kerberos        88/udp  kerberos5 krb5  # Kerberos v5
```

Port numbers are unique only within a specific transport protocol. In other words both TCP and UDP can, and do, assign the same port numbers. It is combination of protocol and port number that uniquely identifies the specific application and provides all of the information necessary to deliver the data to the correct application.

7.7.2 Sockets

In addition to the use of well known port numbers, the operating system can use *dynamically allocated ports*. As the name implies, these are not pre-assigned and are assigned to processes when needed. The operating system ensures that the same port number is not simultaneously assigned to another application and that the numbers assigned are above the range of standard port numbers reserved for well known services.

The dynamically assigned ports provide the flexibility needed to support concurrent multiple users. Once a service on a well known port is assigned to a user the second user requesting the same service cannot be assigned the same port number. In order to overcome this problem, the operating system dynamically allocates a port number for the source port but uses the well known port number for the destination port.

The dynamically allocated port number in conjunction with the IP address is used to uniquely identify a single network process within the entire Internet. This combination of an IP address and a port number is called a *socket*.

A **socket** is one endpoint of a two way logical communication link between two programs associated with a client and a server. A socket is bound to a port number so that the TCP layer can identify the application that data are destined to be sent. The client and the server can reside on two different machines or on the same physical machine.

The server which listens on a socket bound to a well known port (pre assigned) responds to a client requesting a connection through that port by creating another socket bound to a different port number. The server then informs the client of this port number and the client communicates with the application running on the server through this port (Figure 24). The server continues to listen on

the first socket/port. At the end of communication session the sockets/ports are closed down.

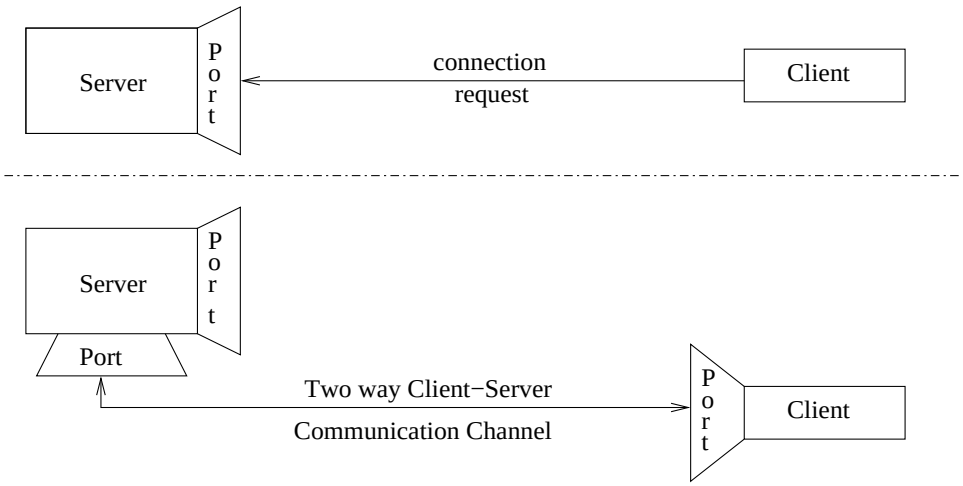


Figure 24: Socket Creation

7.8 IP Multicasting

Multicasting is important in allowing a stream of data to be sent efficiently to many receivers. In multicasting, rather than sending a separate stream of data packets to each intended user (unicasting) or transmitting all packets to everyone (broadcasting), a stream is transmitted simultaneously to a designated subset of network users. The concept of multicasting is shown in Figure 25.

IP Multicasting is defined in RFC 1112 [27] of 1989. A key to IP multicasting is the Internet Group Management Protocol (IGMP) specified in RFC 2236 [34] which updates RFC 1112. IGMP enables users to sign up for multicast sessions and allows these multicast groups to be managed dynamically, in a distributed fashion. Enhancements have been made to existing protocols to direct the traffic to the members of the group. These include Distance Vector Multicasting Routing Protocol (DVMRP – RFC 1075) and Multicast Open Shortest Path First (MOSPF – RFC 1585) protocol. An entirely new protocol developed specifically for multicasting is the Protocol Independent Multicast (PIM – RFC 2362).

At the start of a multicast session group addresses are allocated which are relinquished at the end of that session and reused later.

Internet MBONE is Internet's multicast backbone which is a collection of multicast routers that can distribute multicast traffic. MBONE participants use class D Internet addresses which identify a group of hosts rather than individual hosts.

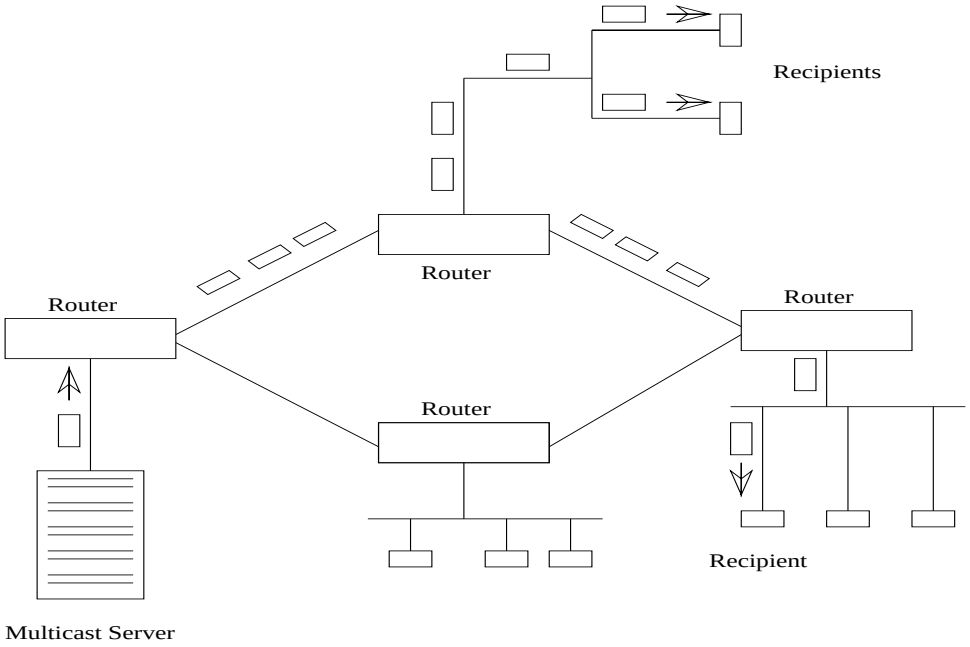


Figure 25: Multicasting

7.9 Resource Reservation Protocol – RSVP

A key factor in achieving real time quality of service is a reservation set up protocol, a mechanism for creating and maintaining flow specific state information in the end point hosts and in routers along the data flow path. The IETF has developed its Resource Reservation Protocol (RSVP) specifically for the packet switched multicast environment (RFC 2205 [33]).

RSVP has been designed to meet a number of requirements:

1. support for heterogeneous service needs;
2. flexible control over the way reservations are shared along branches of multicast delivery trees;
3. scalability to large multicast groups;
4. and the ability to preempt resources to accommodate advance reservations.

The RSVP protocol basically acts according to its name. An RSVP request specifies the level of resources to be reserved for some or all of the packets in a particular session. An application requests resources by specifying a flow specification, which describes the type of traffic anticipated (for example, average and peak bandwidths and level of burstiness), and a resource class specifying the type of service required (such as guaranteed delay). A filter specification is also specified, which determines the sources to which a given reservation applies.

RSVP mandates that a resource reservation be initiated by the receiver rather than the sender. While the sender knows the properties of the traffic stream it is transmitting, it has been found that the sender initiated reservation scales poorly for large, dynamic multicast delivery trees. Receiver initiated reservation

deals with this by having each receiver request a reservation appropriate to itself; differences among heterogeneous receivers are resolved within the network by RSVP. After learning sender's flow specification via a higher level "out of band mechanism", the receiver generates its own desired flow specification and propagates it to senders, making reservations in each router along the way.

RSVP itself uses a connectionless approach to multicast distribution. The reservation state is cached in the router and periodically refreshed by the end station. If the route changes, these refresh messages automatically install the necessary state along the new route.

RTP and RTCP information is simply data from the point of view of routers that move the packets to their destination. RSVP prioritises multimedia traffic and provides a guaranteed quality of service. Routers that have been upgraded to support RSVP can reserve carrying capacity for video and audio streams and prevent unpredictable delays that would interfere with their transmission.

7.10 TCP/IP Over Serial Lines – SLIP and PPP

When data is sent over a serial link using TCP/IP some point to point data link layer protocol is required to perform framing, error control and other data link layer functions.

There are two protocols namely, SLIP (Serial Line Internet Protocol) and PPP (Point to Point Protocol) that are used to communicate over dial up lines using IP.

7.10.1 SLIP – Serial Line Internet Protocol

Around 1984, Rick Adams implemented SLIP for 4.2 Berkeley Unix and Sun Microsystems workstations and released it to the world. It quickly caught on as an easy and reliable way to connect TCP/IP hosts and routers with serial lines.

The Serial Line Internet Protocol (SLIP), documented in RFC 1055[1], is a packet-framing protocol for relaying IP packets over dial-up lines. It defines an encapsulation mechanism that frames IP packets on a serial line. There is no support for dynamic address assignment, link testing, multiplexing different protocols over a single link, packet type identification, error detection/correction or compression mechanisms.

SLIP is commonly used on dedicated serial links and sometimes for dialup purposes, and is usually used with line speeds between 1200 bps and 19.2 Kbps. It is useful for allowing mixes of hosts and routers to communicate with one another (host-host, host-router and router- router are all common SLIP network configurations).

The SLIP protocol does not define any link control information that could be used to dynamically control the characteristics of a connection. Therefore, SLIP systems must assume certain link characteristics. Because of this limitation,

[1] Nonstandard for transmission of IP datagrams over serial lines: SLIP

SLIP can only be used when both hosts know each other's address, and only when IP datagrams are being transmitted.

SLIP sends the datagram across the serial line as a series of bytes, and it uses special characters to mark when a series of bytes should be grouped together as a datagram. SLIP defines two special characters for this purpose:

1. The SLIP "END" character, a single byte with the decimal value 192, is the character that marks the end of a datagram. When the receiving SLIP encounters the "END" character, it knows that it has a complete datagram that can be sent up to IP.

2. The SLIP "ESC" character, a single byte with the decimal value of 219, is used to "escape" the SLIP control characters.

If the sending SLIP encounters a byte value equivalent to either a SLIP "END" character or a SLIP "ESC" character in the datagram it is sending, it converts that character to a sequence of two characters. The two-character sequences are "ESC 220" for the "END" character, and "ESC 221" for the "ESC" character itself. When the receiving SLIP encounters these two-byte sequences, it converts them back to single-byte values. This procedure prevents the receiving SLIP from incorrectly interpreting a data byte as the end of the datagram. Here "ESC" refers to the SLIP escape character, and not the ASCII escape character.

7.10.2 PPP – The Point to Point Protocol

The Internet uses PPP (point to point protocol – RFCs 1661, 1662 and 1663) to carry traffic between routers and that between home users and ISPs (Internet Service Providers). PPP is designed to handle among other things, error detection, support for multiple protocols, IP address negotiation at connection time, and authentication.

In order to achieve these PPP has the following features:

1. A framing method that unambiguously detects the end of one frame and the start of the next one. The frame format also handles error detection.
2. A Link Control Protocol (LCP) for bringing lines up, testing them, negotiating options and bringing them down gracefully when they are no longer needed. LCP supports both synchronous and asynchronous circuit technologies and byte oriented and bit oriented encodings.
3. A different Network Control Protocol (NCP) for each network layer supported. This allows to negotiate network layer options in a way that is independent of the network layer protocol to be used.

The PPP frame format (see figure 26) closely resembles the HDLC (High level Data Link Control) frame format, the major difference being that PPP is character oriented whereas HDLC is bit oriented. In particular PPP uses byte stuffing on dial up lines so all frames are an integral number of bytes. PPP uses the same

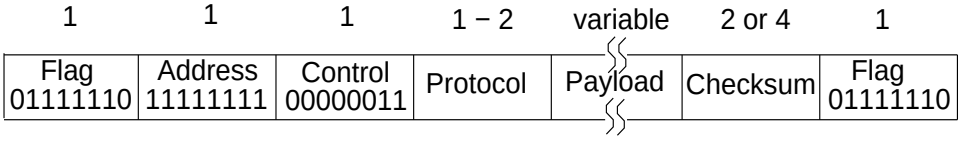


Figure 26: PPP Frame Format

HDLC *flags* which are byte stuffed if they occur within the payload field. The *address* field is always set to 11111111 to indicate that all stations are to accept the frame thus avoiding the need to assign data link addresses.

PPP does not support reliable transmission using sequence numbers and acknowledgements as the default. Hence the *control* field is set to 00000011 indicating unnumbered mode. In noisy environments such as wireless networks, RFC 1663 defines a reliable transmission using numbered mode that is rarely used.

The default LCP settings allow two parties to negotiate an option to just omit the transmission of Address and Control fields altogether thus saving 2 bytes per frame.

PPP defines codes for the *protocol* field. Protocol codes starting with a 0 bit are network layer protocols such as IP, IPX and OSI CLNP. Those starting with a 1 bit are used to negotiate other protocols including LCP and a different NCP for each network layer supported. The default size of the protocol field is 2 bytes. However, using LCP it can be negotiated down to 1 byte.

Each Network Control Protocol (NCP) on the other hand is specific to some network layer protocol and allows configuration requests to be made that are specific to that protocol.

In PPP the default *payload* length is 1500 bytes but can be negotiated up to a maximum length using LCP during line set up. The *checksum* field is normally 2 bytes (16 bits) but a 4 byte (32 bit) checksum can be negotiated.

Most Linux systems include both SLIP and PPP. However, on some Unix systems such as Solaris, PPP is included and SLIP is not. As a simple rule use PPP where you can and SLIP where you must.

8 IPv6 – The New Generation Internet Protocol

8.1 The Design of IPv6

IPv4 is a very robust design. If it had a major design flaw the Internet could not have been so successful. However IPv6 is not a simple derivative of IPv4, but a definitive improvement [19, 14, 24].

IPv6 is an evolutionary step forward from IPv4. Not only it fixes a number of shortcomings in IPv4 such as the limited number of available addresses, it also adds several improvements in such areas as routing and address auto configuration, authentication and privacy capabilities, and IP mobility. IPv6 is expected to eventually replace IPv4, but until such time both protocols will coexist. It is

therefore important that applications are developed to work transparently with both IPv4 and IPv6 during this transition period.

8.2 IPv6 Header Format

Compared to IPv4 header format (Figure 19 on page 19) which has 10 fixed header fields, 2 addresses and some options, the IPv6 header format shown in Figure 27 with only 6 fields and 2 addresses is in fact much simpler. Consequently the header processing is expected to be much more efficient in IPv6, thereby cutting down the processing and reducing the transit delays in internetworking devices.

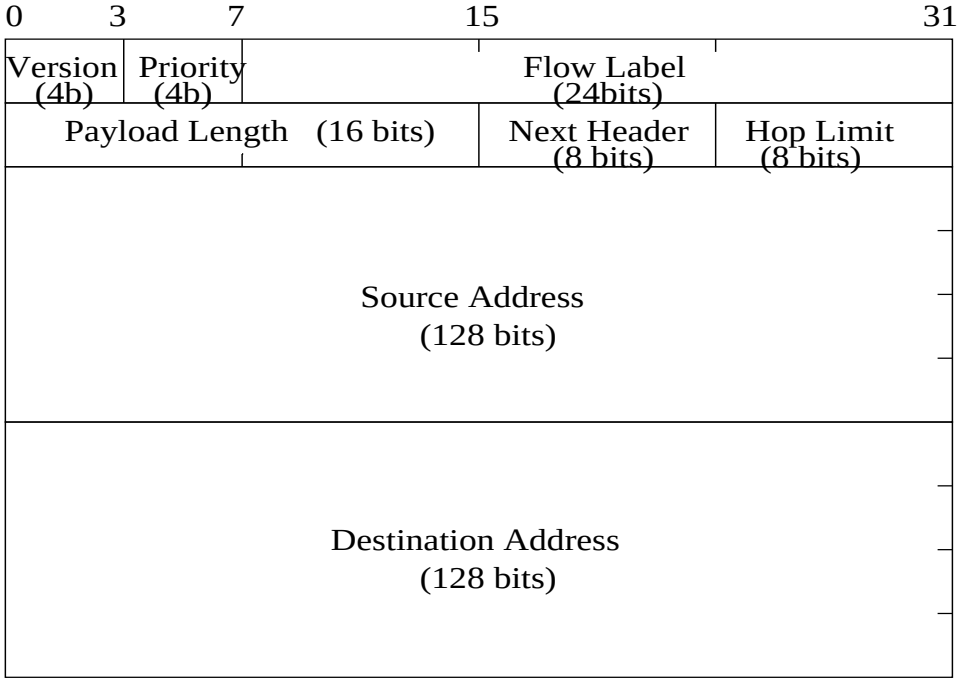


Figure 27: The IPv6 Header Format

8.3 Simplifications

The design of IPv6 has included three major simplifications (based on the experience gained in operating IPv4 for over 20 years!):

1. assign a fixed format to all headers
2. remove the header checksum
3. remove the hop by hop segmentation procedure.

IPv6 handles options in a different way (see Figure 28) compared to IPv4, ie. in the form of extension headers and hence there is no need for *header length field* in the IPv6 header.

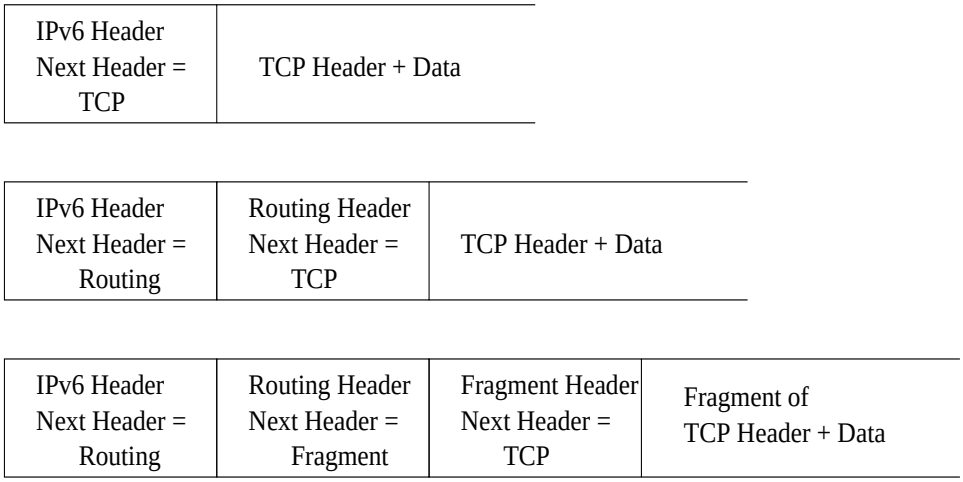


Figure 28: Chaining of IPv6 Headers

IPv6 currently defines six extension headers:

1. Hop by hop options header – special options requiring hop by hop processing
2. Routing header
3. Fragment header – fragmentation and reassembly
4. Authentication header – integrity and authentication
5. Encrypted security payload header – confidentiality
6. Destination options header – optional information to be examined by the destination node.

In order to improve the performance when handling subsequent option headers and the transport protocol which follows, IPv6 options are always an integer multiple of 8 octets long. This also helps to retain the alignment for subsequent headers.

The main advantage of removing header checksum is to diminish the cost of header processing by removing the need to check and update the checksum at each intermediate relay. This can however result in misrouted packets. Experience has shown that the risk is minimal since most encapsulation procedures include a packet checksum (eg. MAC procedure of IEEE 802.X networks, in the adaptation layers for ATM and in the framing procedure of the Point to Point Protocol for serial lines).

The last simplification is the removal of the Type of Service (TOS) field. It was found that although IPv4 provides TOS, this filed was hardly ever set by applications.

8.4 New Fields

The *Flow Label* and *Priority* have been included mostly to facilitate the handling of Real Time Traffic so as to ensure the proper treatment of high quality multimedia

communications in the new Internet. Flow labels will allow the stipulation of severe real time constraints, for example.

8.5 Special Services

In traditional packet switching, a packet is queued at a switch (Figure 29) until a line is available to carry it thereby giving rise to congestion and delay.

Figure 29: Traditional Packet Switching – One Queue Per Line

Assigning higher priority to the queue of real time packets over that of data packets is not enough. It is necessary to ensure that the *service rate* is equal to or higher than the *arrival rate* of packets. This can be achieved as follows:

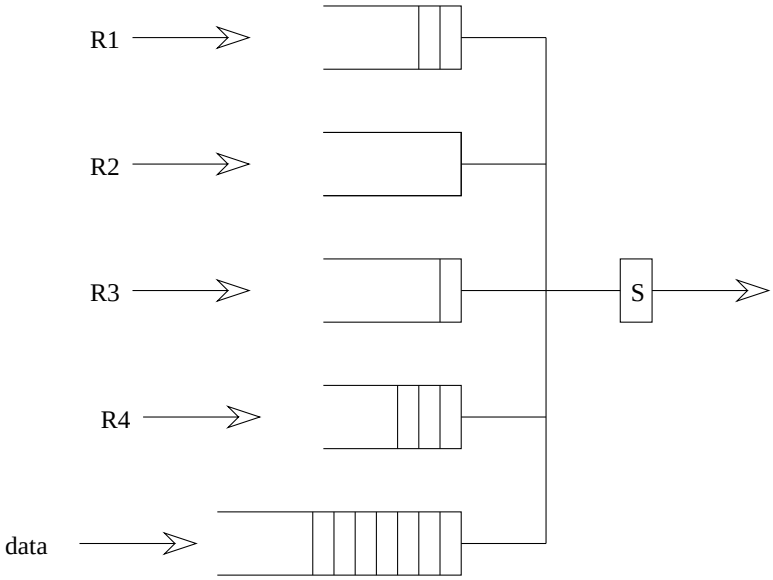


Figure 30: Real Time Streams and Data in Separate Queues

If each real time queue is serviced at a rate compatible with its requirements then it will never suffer from unpredictable queueing delays (see Figure 30). The data queue, however, will only be serviced on a *best effort basis*.

The proposers of IPv6 suggest that the *New Internet* is capable of providing all the services required by its users, including *Real Time Audio and Video*. It is their opinion that ATM is **"Another Terrible Mistake"**.

8.6 IPv6 Address Space

The address length of 128 bits gives rise to a total of 256 x 10^{36} different addresses in its address space. However due to inefficiencies in address allocation and

administration (expressed by H Factor[1] which has found generally to lie in the range 0.22 to 0.26), the new Internet is expected to support 10^{15} (quadrillion) hosts and 10^{12} (trillion) networks.

The notation for writing IPv6 addresses is (remember the dot notation in IPv4!):

FEDC:BA98:7654:3210:FEDC:BA98:7654:3210

The notation allows to skip leading zeros; for instance

0000 can be written as just 0 and 0032 can be written as 32. Further the notation also allows removing a 0 leaving the colons. Thus an address such as 1030:0:0:0:80:3210:2c15:417a would become 1030::80:3210:2c15:417a. The double colon notation can be used at the beginning or at the end of an address but only once.

In the interim period an IPv4 address will be written as an IPv6 address by prepending 12 octets of zeros giving 0:0:0:0:0:0:0:0:0:0:0:0:128.145.48.12. It is also allowed to write this as ::128.145.48.12.

8.7 Making IPv6 Compliant

IPv6 will coexist with IPv4 for a number of years after which all computers will run only IPv6. During this period it is suggested to have a dual protocol stack consisting of IPv6 and IPv4 (Figure 31) running on every host that uses both sets of protocols.

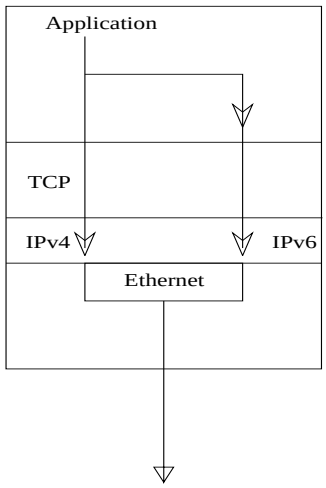


Figure 31: Typical Dual Stack Configuration

8.7.1 IPv6 Tunnelling

IPv6/IPv4 dual protocol stack will also be implemented in routers thus facilitating communication between two IPv6 compliant computers at the two ends of

[1] H Factor is defined by Huitema [19] as the ratio between log(number of addresses) and the number of bits in the address

the IPv4 based Internet [24]. This is known as IPv6 tunnelling (see Figure 32).

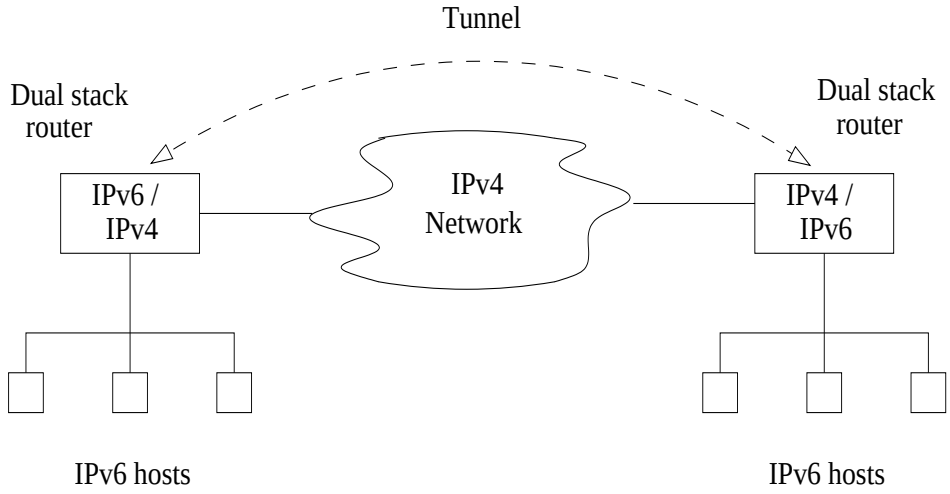


Figure 32: IPv6 Tunnelling over a IPv4 Network

9 Data Communication in Real Time

Packet switching techniques based on ITU-T X.25 and IP have been traditionally used for non Real Time data transfer. Since they do not guarantee packet sequence integrity and consistent latency times in delivery, they are inherently unsuitable for Real Time applications.

The following are essentially required for carrying Real Time data on existing networks.

1. Enough bandwidth for extremely dense audio and video traffic.
2. A transport protocol appropriate for the streaming requirements of real time data (RTP).
3. A protocol to reserve network bandwidth and assigning priorities for various types of traffic (RSVP).

9.1 RTP Data Transfer Protocol

A Real time Transport Protocol is therefore needed to provide end to end network transport functions suitable for applications communicating in real time. Such applications include transmission of interactive audio and video data or real time simulation data over multicast or unicast network services.

The largest (and the oldest) network which supports real time data communication is the telephone network. This falls into the category of circuit switched networks. They have been designed to support Plain Old Telephone Services (POTS). However in terms of a network protocol there is not much. Once the

network connection is established the communication process is largely in the hands of the two persons communicating with each other.

In view of this a Real time Transport Protocol (RTP) along with a profile for carrying audio and video over RTP were defined by the IETF in January 1996 [31, 32].

9.1.1 Characteristics of RTP

The Realtime Transport Protocol has the following characteristics.

- Payload type identification
- Sequence numbering
- Time stamping
- Delivery monitoring.

Real Time applications typically run RTP on top of UDP to make use of its multiplexing and checksum services (see Figure 33). Tailoring RTP to the application is accomplished through auxiliary profile and payload format specifications. A payload format defines the manner in which a particular payload, such as an audio or video encoding, is to be carried in RTP. A profile assigns payload type numbers for the set of payload formats that may be used in the application.

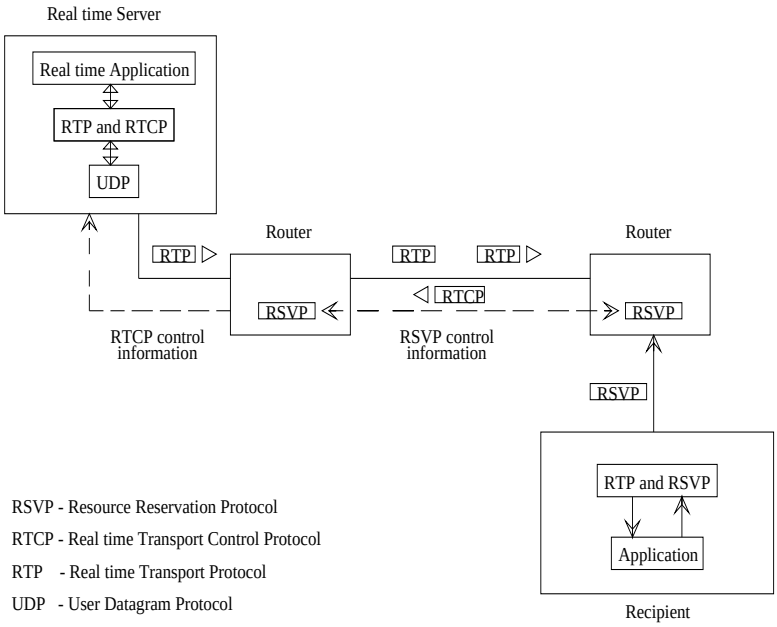


Figure 33: Real Time Application running RTP on top of UDP

However RTP is not limited to be used with UDP/IP. It can equally be used with other underlying network or transport protocols such as ATM or IPX. Moreover, RTP supports data transfer to multiple destinations using multicast distribution if provided by the underlying network, a feature which makes RTP ideal for multi party multimedia conferencing.

RTP is designed to work in conjunction with RTCP (Real time Transport Control Protocol) to monitor the quality of service. RTP delivers real time traffic with timing information for reconstruction as well as feedback on reception quality. The Resource Reservation Protocol (RSVP) is used to reserve network bandwidth and assign priority for various traffic types.

9.1.2 Definitions in RTP

The following definitions are extracts from the relevant RFCs [31, 32].

- RTP Payload

  The data transported by RTP in a packet, for example audio samples or compressed video data.

- RTP Packet

  A data packet consisting of the fixed RTP header, a possibly empty list of contributing sources, and the payload data. Typically one packet of the underlying protocol contains a single RTP packet, but several RTP packets may be contained if permitted by the encapsulation method.

- RTCP Packet

  A control packet consisting of a fixed header part similar to that of RTP data packets, followed by structured elements that vary depending upon the RTCP packet type. Typically multiple RTCP packets are sent together as a compound RTCP packet in a single packet of the underlying protocol. This is enabled by the length field of the fixed header of each RTPC packet.

- Port

  The "Abstraction" that transport protocols use to distinguish among multiple destinations within a given host computer (TCP/IP protocols identify ports using small positive integers. The transport selectors (TSEL) used by the OSI Transport layer are equivalent to ports). RTP depends on the lower layer protocol to provide some mechanism such as ports to multiplex the RTP and RTCP packets of a session.

- Transport Address

  The combination of a network address and the port that identifies a transport level end point, for example an IP address and a UDP port. Packets are transported from a source transport address to a destination transport address.

- RTP Session

  The association among a set of participants communicating using RTP. For each participant, the session is defined by a particular pair of destination transport addresses consisting of one network address and a port pair for RTP and RTCP. The destination transport address pair may be common for all participants, as in the case of IP multicast, or may be different for each, as in the case of individual unicast network addresses plus a common

port pair. In a multimedia session, each medium is carried in a separate RTP session with its own RTCP packets. The multiple RTP sessions are distinguished by different port number pairs and/or different multicast addresses.

- Synchronisation Source (SSRC)

  The source of a stream of RTP packets, identified by a 32 bit numeric SSRC identifier carried in the RTP header so as not to be dependent upon the network address. Examples of synchronisation sources include the sender of a stream of packets derived from a signal source such as a microphone, a camera or an RTP mixer. If a participant generates multiple streams in one RTP session, for example from separate video cameras, each must be identified as a different SSRC.

- Contributing Source (CSRC)

  A source of a stream of RTP packets that has contributed to the combined stream produced by an RTP mixer. The mixer inserts a list of the SSRC identifiers of the sources that contribute to the generation of a particular packet into the RTP header of that packet. This is called the CSRC list. An example application is audio conferencing where a mixer indicates all the talkers whose speech was combined to produce the outgoing packet, allowing the receiver to indicate the current talker, even though all the audio packets contain the same SSRC identifier (that of the mixer).

- End System

  An application that generates the content to be sent in RTP packets and/or consumes the content of received RTP packets.

- Mixer

  An intermediate system that receives RTP packets from one or more sources, possibly changes the data format, combines the packets in some manner and then forwards a new RTP packet. Since the timing among multiple input sources will not generally be synchronised, the mixer will make timing adjustments among the streams and generate its own timing for the combined stream. Thus all data packets originating from a mixer will be identified as having the mixer as their synchronisation source.

- Monitor

  An application that receives RTCP packets sent by participants in an RTP session, in particular the reception reports, and estimates the current quality of service, fault diagnosis and long term statistics.

9.1.3 RTP Fixed Header Fields

The RTP header format is shown in Figure 34. The first twelve octets are present in every RTP packet, while the list of CSRC identifiers is present only when inserted by a mixer.

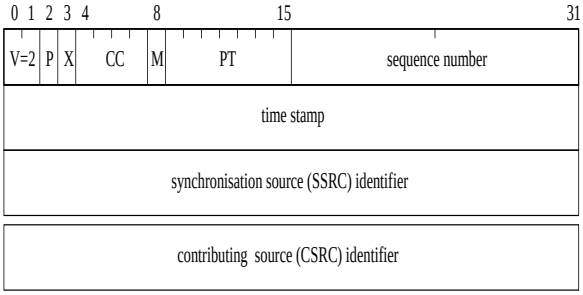


Figure 34: RTP Header Format

The RTP header provides the timing information necessary to synchronise and display audio and video data and to determine whether packets have been lost or arrive out of order. In addition, the header specifies the payload type, thus allowing multiple data and compression types. This is a key advantage over most proprietary solutions, which specify a particular type of compression and thus limit users' choice of compression schemes.

9.1.4 Multiplexing RTP Sessions

In RTP, multiplexing is provided by the destination transport address (network address and a port number) which define an RTP session.

9.1.5 Real time Transport Control Protocol (RTCP)

The operation of RTCP is based on the periodic transmission of control packets to all participants in the session, using the same distribution mechanism as the data packets. The underlying protocol must provide multiplexing of the data and control packets, for example using separate port numbers with UDP.

The primary function of RTCP is to furnish information on the quality of data distribution. This feedback is a critical part of RTP's use as a transport protocol, since applications can use it to control how they behave. The feedback is also important for diagnosing distribution faults. For instance, by monitoring reports from all data recipients, network managers can determine the spread of a problem. When used in conjunction with IP multicast, RTCP enables the remote monitoring and diagnosis.

In addition RTCP controls the rate at which participants in an RTP session transmit RTCP packets. In a session with a few participants, RTCP packets are sent at the maximum rate of one every five seconds whereas for a larger group, RTCP packets may be sent only once every 30 seconds. In other words, the more participants there are in a conference, the less frequently each participant sends RTCP packets. This makes RTCP scalable to accommodate tens of thousands of users.

9.2 Real Time Data Transfer using ATM

Audio and video applications generate lots of bits, and the traffic has to be streamed or transmitted continuously rather than in bursts. This is in contrast to conventional data types such as text, files and graphics, which are able to withstand short and inconsistent periods of delay between packet transmissions. What is needed then is a network capable of transporting both streaming and bursty data. ATM (Asynchronous Transfer Mode) is a transmission technique which has been designed primarily to achieve this [10, 38].

ATM is a connection oriented protocol designed to support high bandwidth, low delay (even services with predictable delay), packet like switching and multiplexing. The design of ATM ensures the capability to carry both stream traffic (such as voice and video) and bursty traffic (such as interactive data) with guaranteed QOS. It uses a fixed cell size for all types of traffic. In the case of stream traffic ATM guarantees the integrity of cell sequence which is essential for the successful delivery of such traffic.

ATM has grown out of the need for a worldwide standard to allow interchange of information regardless of the "end system" or type of information. Historically there have been separate methods used for the transmission of information among users on LANs and the users on WANs. This situation has been made more complex by the user's need for connectivity expanding from the LAN to MAN to WAN. ATM is a method to unify the communication of information on LANs and WANs.

ATM is the only technology based on standards, and has been designed from the beginning to accommodate the simultaneous transmission of data, voice and video. It is an easily scalable backbone which can be upgraded merely by adding more switches or links. ATM is switched instead of routed and therefore it is faster since not every IP packet at every node is examined to determine its destination. ATM LAN Emulation (LANE) allows transparent interconnection of "legacy" LANs based on Ethernet or FDDI technology, making the ATM backbone look like a fast Ethernet or FDDI to workstation applications. As more and more ATM nodes are deployed, the differences between local and wide area networks will disappear to form a seamless network based on one standard.

To use the limited bandwidth more efficiently, ATM uses circuit switching principles to give the users a full channel to themselves. Since these users do not use the full channel all the time, ATM uses statistical analysis to time division multiplex several users onto the same line. This allows each user to have all of the channel's bandwidth for the period of time in which it is needed.

To achieve this ATM uses two connection concepts; the *Virtual Channel (VC)* and the *Virtual Path (VP)*.

A virtual channel (also known as a virtual circuit) provides a logical connection between end users and is identified by a VCI (Virtual Channel Identifier) in the ATM header (see Figure 35). A virtual path defines a collection of virtual circuits traversing the same path in the network and is identified by a VPI (Virtual Path Identifier). The VPI emulates the functions of the trunk concept in circuit switching. Thus virtual paths define the cross connection functions across the

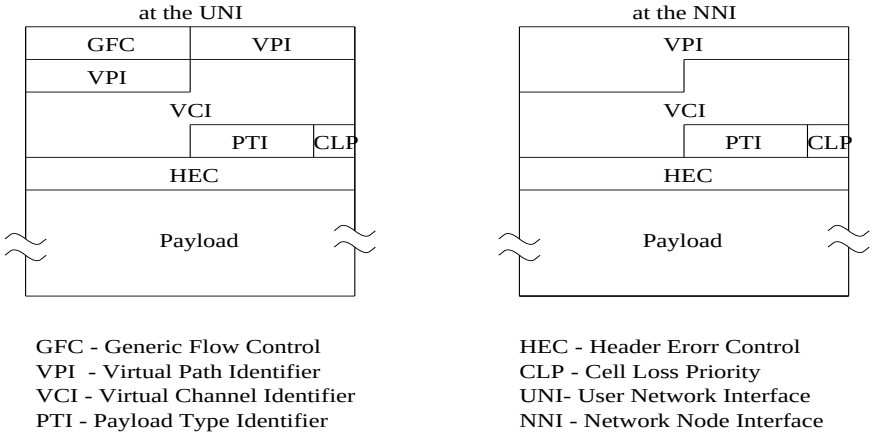


Figure 35: ATM Cell Formats

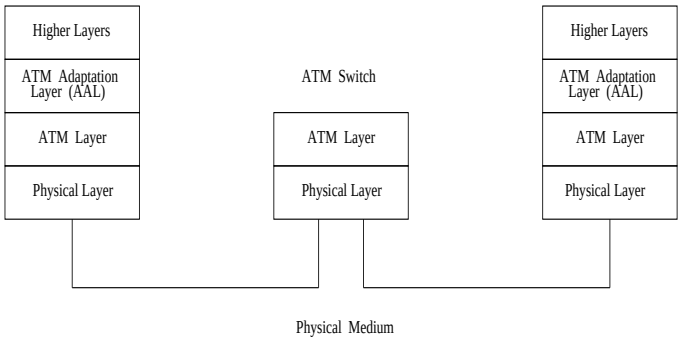


Figure 36: ATM Protocol Architecture

network, whereas virtual channels are concerned with switching and connection establishment functions. Virtual paths are statistically multiplexed on the physical link on a cell multiplexing basis.

GFC (Generic Flow Control) is used to control the amount of traffic entering the network.

VPI and VCI are used for routing. VPI will change from one node to the next when it travels through the ATM layer. VCI is predefined and usually remains the same throughout the duration of the transmission. PTI (Payload Type Identifier) is used to distinguish between cells that are carrying user data and those carrying control information. CLP is a single control bit which provides selective discard during network congestion and HEC is used to check header errors.

The ATM cell formats used at the UNI (User-Network Interface) and NNI (Network-Node Interface) are shown in Figure 35.

9.2.1 ATM Protocol Structure

Figure 36 shows the ATM layered architecture as described in ITU-T recommendation I.321 (1992). This is the basis on which the B-ISDN Protocol Reference Model has been defined.

- ATM Physical Layer

  The physical layer accepts or delivers payload cells at its point of access to the ATM layer. It provides for cell delineation which enables the receiver to recover cell boundaries. It generates and verifies the HEC field. If the HEC cannot be verified or corrected, then the physical layer will discard the errored cell. Idle cells are inserted in the transmit direction and removed in the receiving direction.

  For the physical transmission of bits, 5 types of transmission frame adaptations are specified (by the ITU and the ATM Forum). Each one of them has its own lower bound or upper bound for the amount of bits it can carry (from 12.5 Mbps to 10 Gbps so far).

  1. Synchronous Digital Hierarchy (SDH) $\geq$ 155 Mbps;
  2. Plesiochronous Digital Hierarchy (PDH) $\leq$ 34 Mbps;
  3. Cell Based $\geq$ 155 Mbps;
  4. Fibre Distributed Data Interface (FDDI) = 100 Mbps;
  5. Synchronous Optical Network (SONET) $\geq$ 51 Mbps.

  The actual physical link could be either optical or coaxial with the possibility of Unshielded Twisted Pair (UTP Category 3/5) and Shielded Twisted Pair (STP Category 5) in the mid range (12.5 to 51 Mbps).

- ATM Layer

  ATM layer mainly performs switching, routing and multiplexing. The characteristic features of the ATM layer are independent of the physical medium. Four functions of this layer have been identified.

  1. cell multiplexing (in the transmit direction)
  2. cell demultiplexing (at the receiving end)
  3. VPI/VCI translation
  4. cell header generation/extraction.

  This layer accepts or delivers cell payloads. It adds appropriate ATM cell headers when transmitting and removes cell headers in the receiving direction so that only the cell information field is delivered to the ATM Adaptation Layer.

  At the ATM switching/cross connect nodes VPI and VCI translation occurs. At a VC switch new values of VPI and VCI are obtained whereas at a VP switch only new values for the VPI field are obtained (see Figure 37). Depending on the direction, either the individual VPs and VCs are multiplexed into a single cell or the single cell is demultiplexed to get the individual VPs and VCs.

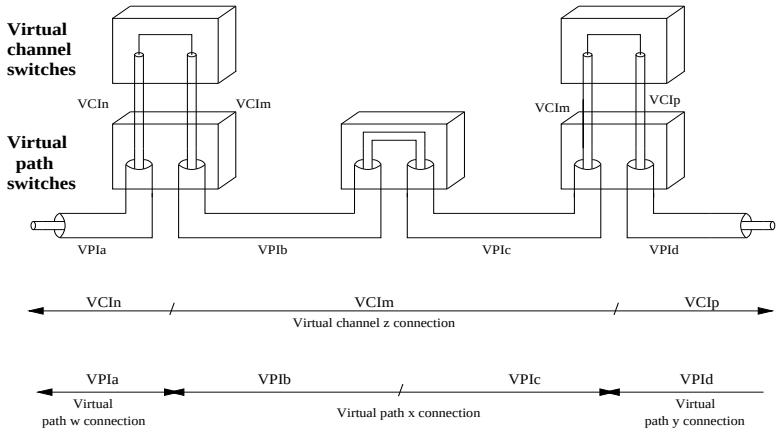


Figure 37: VC/VP Switching in ATM

- ATM Adaptation Layer (AAL)

  The ATM Adaptation Layer (AAL) is between ATM layer and the higher layers. Its basic function is the enhanced adaptation of services provided by the ATM layer to the requirements of the higher layers.

  This layer accepts and delivers data streams that are structured for use with user's own communication protocol. It changes these protocol data structures into ATM cell payloads when receiving and does the reverse when transmitting. It inserts timing information required by users into cell payloads or extracts from them. This is done in accordance with five AAL service classes defined as follows.

  1. AAL1 - Adaptation for Constant Bit Rate (CBR) services (connection oriented, 47 byte payload);
  2. AAL2 - Adaptation for Variable Bit Rate (VBR) services (connection oriented, 45 byte payload);
  3. AAL3 - Adaptation for Variable Bit Rate data services (connection oriented, 44 byte payload);
  4. AAL4 - Adaptation for Variable Bit Rate data services (connection less, 44 byte payload);
  5. AAL5 - Adaptation for signalling and data services (48 byte payload).

  In the case of transfer of information in real time, AAL1 and AAL2 which support connection oriented services are important. AAL4 which supports a connection less service was originally meant for data which is sensitive to loss but not to delay. However, the introduction of AAL5 which uses a 48 byte payload with no overheads, has made AAL3/4 redundant. Frame Relay and MPEG–2 (Moving Pictures Expert Group) video streaming are two services which will specifically use AAL5.

9.2.2 ATM Services

- CBR Service

  This supports the transfer of information between the source and destination at a constant bit rate. CBR service uses AAL1. A typical example is the transfer of voice at 64 Kbps over ATM. Another usage is for the transport of fixed rate video.

  This type of service over an ATM network is sometimes called circuit emulation (similar to a voice circuit on a telephone network).
- VBR Service

  This service is useful for sources with variable bit rates. Typical examples are variable bit rate audio and video.
- ABR and UBR Services

  The definition of CBR and VBR has resulted in two other service types called Available Bit Rate (ABR) services and Unspecified Bit Rate (UBR) services.

  ABR services use the instantaneous bandwidth available after allocating bandwidths for CBR and VBR services. This makes the bandwidth of the ABR service to be variable. Although there is no guaranteed time of delivery for the data transported using ABR services, the integrity of data is guaranteed. This is ideal to carry time insensitive (but loss sensitive) data such as in LAN-LAN interconnect and IP over ATM.

  UBR service, as the name implies, has an unspecified bit rate which the network can use to transport information relating to network management, monitoring, etc.

9.2.3 IP over ATM

The transmission of classical IP traffic over ATM can be accomplished as shown in figure 38.

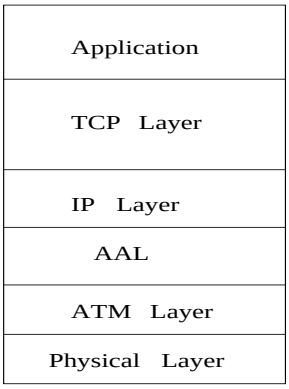


Figure 38: Transmission of IP over ATM

9.3 IP/TV – A Real life Example

IP/TV^{TM} is a client server application that multicasts live or pre-recorded digital video and audio streams in real time to an unlimited number of users over any IP based local or wide area network including the global Internet, using fully compliant TCP/IP protocol stacks supporting real time protocol components. It uses state-of-the-art Internet standards such as IP multicasting, RTP, RTCP and RSVP in its Flashware^{TM} software suite to provide high quality, synchronised audio/video information over existing packet switched networks simultaneous with current network data traffic.

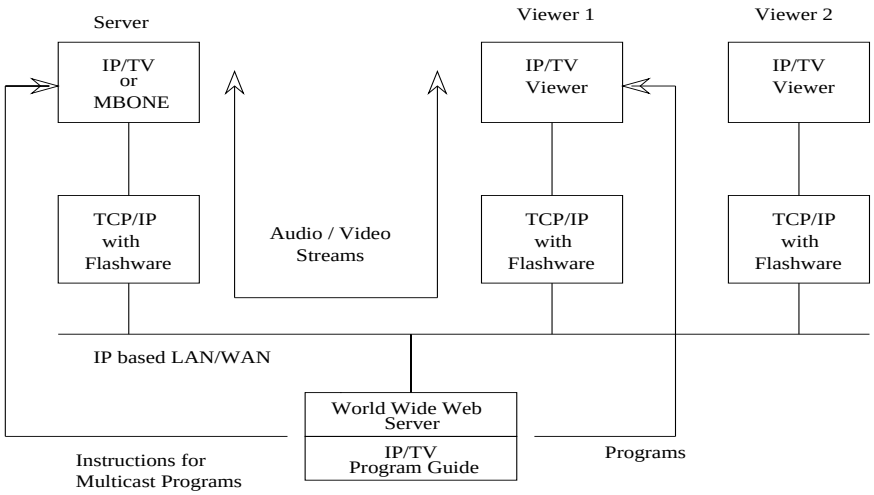


Figure 39: Real Time Audio/Video Over IP based LAN/WANs

IP/TV[1] consists of a Viewer, a Program Guide and a Server (Figure 39). The program guide shows a schedule of multicasts and can be accessed via a Web browser with HTTP (Hyper Text Transport Protocol). MBONE session information can be accessed with the Program Guide which controls the number of streams allowed on the network and the format of those streams, ie. audio only, audio and video or some other combination. The server delivers pre-recorded or live multimedia streams based on the Program Guide schedule and parameters such as start time and file name.

The IP/TV viewer, a tool for signing up for scheduled multicasts, is designed to provide VCR like controls as well as "channel changing" controls. With a software based codec (compliant with ITU–T video conferencing standard H.320/H.261) colour video running at a rate of 30 frames per second uses about 500 Kbps bandwidth. The use of IP multicasting helps to conserve network bandwidth by transmitting over the network a single data stream that can be picked up by any interested user. The use of RSVP provides the ability to reserve bandwidth on RSVP compliant routers, thereby giving priority to time dependent audio/video streams over less critical network traffic thus ensuring the desired Quality of Service (QOS).

[1] available with Flashware from Precept Software Inc.

9.4 Delivering Real Time Data to the Desk Top

9.4.1 Ethernet Based Solutions

Ethernet was invented by Dr Robert M Metcalfe in the 1970s at the Xerox PARC (Palo Alto Research Centre). This was at a time when a network bandwidth of 3 Mbps was found to be adequate to serve the needs of the users whose computers were connected to such networks. Formal specifications for Ethernet were published in 1980 by a multi vendor consortium that created the DEC-Intel-Xerox (DIX) standard and turned the 3 Mbps Ethernet into an open production quality Ethernet operating at 10 Mbps. The popularity of Ethernet made the LAN Standards Committee of the Institute of Electrical and Electronic Engineers (IEEE 802) to adopt and publish this as an IEEE standard in 1985. The IEEE specification which is based on the original DIX technology and provides an "Ethernet like" system, is called the IEEE 802.3 Carrier Sense Multiple Access/ Collision Detect (CSMA/CD) standard.

The original specification of 1985 specifies copper based transmission media, specially, thick (10base5) and thin (10base2) coaxial cables. Since then it has grown to include twisted pair (10/100baseT) and optical fibre (10/100baseFL). It is estimated that there are about 500 million Ethernet connections in the world today, making it the most widely deployed LAN technology.

However, the majority of today's computing is based on *client/server* and *peer-to-peer (p2p)* technologies. As more applications are moved from desktops to severs, and the number of desktop clients increases, the demand placed on a typical Ethernet operating at 10 Mbps tends to affect the application response time and impair the users' ability to effectively access, manipulate and transmit information. Moreover the high performance servers of today need maximum bandwidth and highly reliable network connections to make the most of their powerful processing capabilities.

Clients, on the other hand, require high speed, low cost connections to the high bandwidth networks. This asymmetric approach to client/server network design has led to the development of Fast Ethernet (IEEE 802.3u – 100base.. family) and Gigabit Ethernet (IEEE 802.3z – 1000base.. family) specifications giving a speed advantage of 10 and 100 times respectively, compared to the original Ethernet. Associated with the high bandwidth is also the switching technique, which makes the Ethernet and its variants to be appropriate for the delivery of real time data.

9.4.2 Signal Topology and Timing

The *signal topology* of the Ethernet is also known as the *logical topology* to distinguish it from the actual physical layout of the media cabling. The logical topology of an Ethernet provides a single channel (bus) that carries signals to all stations attached to it. Repeaters which amplify and re-time the signals can be used to link multiple Ethernet segments together to form large Ethernets.

The signal timing is based on the amount of time it takes to traverse the

complete media system from one end to the other and back. This is also called the round trip time, the maximum of which is strictly limited to ensure that every interface can hear all network signals within the specified amount of time provided in the Ethernet Medium Access Control (MAC) mechanism.

The longer a given network segment is the more time it takes for a signal to traverse it. Therefore configuration guidelines (rules) are provided to make sure that the round trip timing limits are met, no matter what combination of media segments are used in the network configuration.

However the expansion of Ethernet using repeaters is limited because of the signal timing restrictions. In order to meet the expansion needs of today, two kinds of hubs known as *repeater hubs* and *packet switching hubs* are available. The advantage of using a switching hub is that each port of the hub operates on *full duplex* mode and provides a connection to an Ethernet media system that operates as a separate Ethernet LAN, and the round trip timing rules for each LAN stop at the switching hub port. This in effect allows several individual LANs each supporting hundreds of computers to be linked together forming a much larger LAN, but still meeting the requirements of signal timing.

The use of switching hubs allows, in addition, to mix Ethernet links operating at 10 Mbps with links operating at higher link speeds and to operate either some or all of the links at higher bandwidths, namely 100 Mbps or 1000 Mbps.

9.4.3 High Performance Ethernets

Two approaches have been considered in making the Ethernet to operate at higher bandwidths. In the first approach, the original Ethernet system with the CSMA/CD MAC mechanism has been speeded up to 100 Mbps. This approach is therefore called 100baseT Fast Ethernet (IEEE 802.3u).

The second approach has been to create an entirely new MAC mechanism using hubs that control access to the medium using a **demand priority** mechanism. The design of the medium access control based on demand priority mechanism has allowed the transportation of token ring frames in addition to the standard Ethernet frames. Therefore it is called 100VG-AnyLAN (IEEE 802.12).

Switching invariably becomes central to the operation of high performance Ethernets. The backplane speed (in packets per second – PPS) which describes the capacity of the switch to move the data from the incoming ports to the outgoing ports and is an important parameter. Another factor which is important for the efficient functioning of a switch is the amount of buffer memory provided. In general, higher the number of ports the more buffer memory is needed.

9.4.4 Switched Ethernets

An Ethernet switch at its basic level can be thought of as a bridge with many ports and low latency. A network segment connected to each of the ports physically represent a separate Ethernet with its own repeater count and timing restrictions. Switches are useful in segmenting large networks for improved perfor-

mance and/or easy manageability. However, the many segments of the network operate as one single logical network.

The design of Ethernet switches gives rise to the following classification based on the type of packet forwarding technique used in the switch architecture. The packet switching type employed has an effect on latency which describes the delay in the switch. The latency has the greatest impact in environments where real time video and audio applications are supported.

- Store and Forward Switches

A store and forward switch stores each incoming frame in a buffer, checks for errors, and if the frame is good then forwards the frame to its destination port. The store and forward technique has the advantage that it prevents wasting network bandwidth by effectively blocking the damaged frames. However the disadvantage is that it increases the latency and therefore results in lower throughput in networks with few errors. Store and forward switches are useful in networks which may experience high error rates.

In general Store and Forward switching is likely to be the best choice when a network needs efficiency and stability.

- Cut Through Switches

In a cut through switch a frame is forwarded immediately upon receiving its destination address thus resulting in a very low latency in the switch. The disadvantage is that it propagates errors and therefore is only suitable for networks which experience a few occasional errors.

- Hybrid Switches

Hybrid switch is the result of an attempt to achieve the best of both, the Store & Forward and Cut Through techniques. In its normal operating mode a hybrid switch operates as a Cut Through switch constantly monitoring the rate at which damaged or invalid frames are forwarded. When the error rate is above a certain threshold then the switch reverts to Store and Forward mode and continues to operate in that mode until the error rate has fallen to an acceptable level before reverting to Cut Through mode.

The Hybrid switch has the performance advantage of a Cut Through switch when the error rate is low, and the error trapping ability of a Store and Forward switch when the error rates are high.

The above types are only applicable when the source and destination ports are all running at the same speed. If the switch has to perform a speed conversion, which is the case in many new network installations especially if a standard Ethernet is migrated to high performance Ethernet, then the switch must operate in the Store and Forward mode to cater to the differing link speeds.

9.4.5 Fast Ethernet

Fast Ethernet (IEEE 802.3u) operating at 100 Mbps is designed as the most direct and simple extension of 10baseT Ethernet operating at 10 Mbps. The Medium Access Control mechanism used in the Fast Ethernet is simply a scaled up version of the MAC technique used by 10baseT Ethernet. This makes Fast Ethernet (100baseT) to be similar to the conventional 10baseT, only faster. It uses the same reliable, robust and economical technology of 10baseT. The seamless compatibility between 10baseT and 100baseT allows easy migration to high speed network connections. The Fast Ethernet specification includes a mechanism for *auto negotiation* of the media speed. This allows the installation of a dual speed Ethernet interface which automatically detects and sets its link speed.

Although Fast Ethernet preserves the critical 100 m maximum UTP cable length from the hub to the desktop, the rules applicable to the 100 Mbps technology are different because of the scaling of the MAC interface.

100baseT Fast Ethernet is a natural evolution from the standard 10baseT Ethernet. Figure 40 shows the two in comparison.

Feature	10baseT Ethernet	100baseT Fast Ethernet
Speed	10Mbps	100Mbps
IEEE Standard	802.3	802.3
Media Access Protocol	CSMA/CD	CSMA/CD
Topology	Bus or Star	Star
Media support	Coax, UTP, Optical Fiber	UTP, STP, Optical Fiber
Hub to node distance (Maximum)	100 meters	100 meters
Media Interface	AUI	MII

Figure 40: Comparison between Ethernet and Fast Ethernet

Fast Ethernet also supports multiple media types. For 100baseT the same cabling installed for a 10baseT network can be used. Figure 41 shows the three media specifications supported by Fast Ethernet, namely 100baseTX (2 pairs of high quality cables), 100baseT4 (4 pairs of normal quality cables) and 100baseFX (optical).

- 100baseTX

The 100baseTX specification supports 100 Mbps transmission speed over two pairs of UTP Category 5 or STP Category 5. The RJ45 connector used for

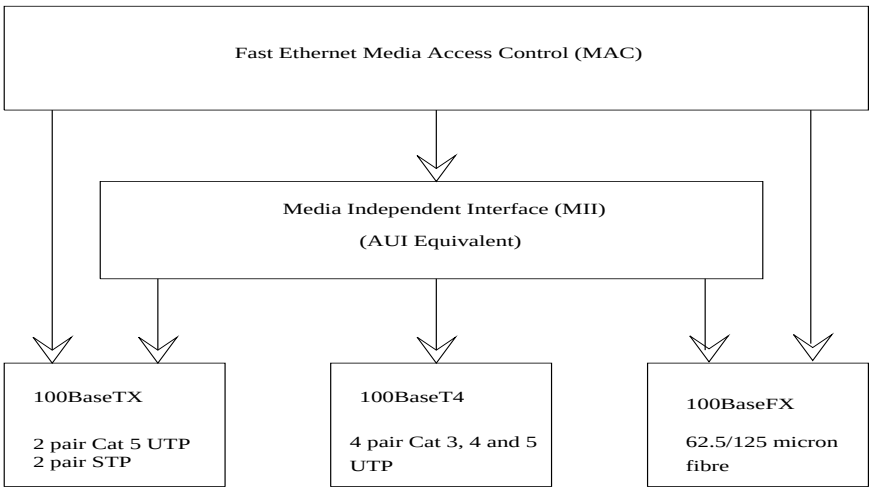


Figure 41: Fast Ethernet Media Support

100baseTX UTP is exactly the same as that used by 10baseT UTP. For STP wiring, 100baseTX also specifies the traditional DB-9 connector.

- 100baseT4

100baseT4 media uses four pairs of Category 3, 4 or 5 UTP wiring to carry data at 100 Mbps. The signalling scheme in 100baseT4 uses three pairs of wires for data and the fourth for collision detection. Because 100baseT4 can use Category 3 it enables migration to 100baseT4 without having to rewire.

100baseT4 also uses RJ45 connector.

- 100baseFX

100baseFX media specification defines 100 Mbps operation over two strands of 62.5/125 micron fibre and allows transmission over greater distances than UTP. The fibre optic connectors are the same as those defined for 10baseFX networks.

Fast Ethernet specification includes a Media Independent Interface (MII) which defines a standard interface between the CSMA/CD MAC layer and any of the three media specifications supported. This is much like the AUI connector for standard Ethernet. The MII defines a 40 pin connector to connect to the external transceivers.

The Fast Ethernet however does not support coaxial cabling largely due to its inability to support high data rates over the distances of interest. The ISO 11801 cabling standard is applicable to Fast Ethernet implementation.

9.4.6 Gigabit Ethernet

The ever increasing processing power in the computers and the deployment of power hungry applications on the networks must be matched by developing high

speed network connections to reduce traffic bottlenecks, improve overall performance and ultimately enhance the productivity of the users who use the network. The development of Gigabit Ethernet is clearly a solution to match the network infrastructure with the capability of todays desktop computers.

Gigabit Ethernet (IEEE 802.3z) is an extension of the highly successful 10 Mbps Ethernet and 100 Mbps Fast Ethernet standards and therefore is fully compatible with the huge installed Ethernet base. Gigabit Ethernet employs all the features of Ethernet specification including frame format, support for CSMA/CD protocol, full duplex transmission, flow control and management objects. This compatibility preserves investment in network administrator expertise and support staff training.

The enhancements in Gigabit Ethernet includes the support for fast optical fibre connections at the physical layer of the network and a MAC layer specification which sustains a tenfold increase in the MAC layer data rates. This is a key to its ability to support data intensive applications such as imaging and video conferencing.

The proliferation of Gigabit Ethernet will take place in phases. Initially Gigabit Ethernet will be used as backbone switch-to-switch connections. The next phase will be to deploy switch-to-server connectivity to boost access to critical server resources. This evolution will be driven by the increasing installation of PCs with 100 Mbps network interfaces. Finally as the desk top costs come down and user network demand increases, Gigabit Ethernet switches will enter the backbone and will take over the switch fabric.

The original IEEE 802.3z specification for 1000baseX comes in three flavours, namely:

- 1000baseSX – 850nm laser on multi mode fibre
- 1000baseLX – 1300nm laser on single and multi mode fibre
- 1000baseCX – short haul copper STP.

The 1000baseT (IEEE 802.3ab) standard specifies CAT-5 balanced copper transmission media in order to take advantage of the already installed base of UTP Category 5 cabling up to 100 m distances.

Recently the IEEE has setup the 802.3ae committee to draw up standards for CSMA/CD that is capable of operating at speeds up to 10 Gbps.

9.4.7 Wireless LANs

A wireless local area network (WLAN) uses radio frequency (RF) technology to transmit and receive data over the air. The Institute of Electrical and Electronics Engineers (IEEE) have established the IEEE 802.11 standard, which is the predominant standard for wireless LANs. Any LAN application, network operating system, or protocol including TCP/IP, will run on 802.11-compliant WLANs as they would over Ethernet.

The Table 1 on page 51 shows the IEEE 802.11 family of standards for Wireless LANs (WLANs).

802.11	2.4 GHz band, supports data rates of 1-2 Mbps
802.11a	High speed WLAN standard for 5 GHz band. Supports up to 54 Mbps
802.11b	WLAN standard for 2.4 GHz band. Supports up to 11 Mbps. Also known as WiFi.
802.11e	QoS requirements for all WLAN services
802.11f	Defines inter-access point communication to facilitate multi-vendor distributed WLANs.
802.11g	An additional modulation technique for 2.4 GHz band, intended to provide rates up to 54 Mbps
802.11h	Defines the spectrum management of the 5 GHz band for use in Europe and Asia Pacific.
802.11i	Addresses security weaknesses of both authentication and encryption protocols.

Table 1: IEEE 802.11 Wireless LAN Standards

The indoor operating ranges for Wireless LANs are typically 30 m at 11 Mbps and 90 m at 1 Mbps [13]. The typical outdoor operating ranges are much larger (120 m at 11 Mbps and 460 m at 1 Mbps).

WLANs can be configured in a variety of ways, but the configuration which truly illustrates the benefits of WLAN technology is when it is used to provide a networking Hot Spot.

A hotspot provides wireless LAN service, for free or for a fee, from a wide variety of public meeting areas, including coffee shops and airport lounges. There are currently thousands of hotspots worldwide and new access points are being added daily. To use hotspots, your notebook must be configured with Wi-Fi Certified (Wi-Fi Alliance is a non profit international organisation set up to certify interoperability of wireless local area network products based on IEEE 802.11 specification) technology so you can connect with other products. Wi-Fi Certified notebooks can send and receive data anywhere within the range of a wireless LAN base station.

The IEEE has two more committees developing standards for Broadband Wireless communication systems. IEEE 802.16 committee is working on Broadband Wireless Access Standards where as the IEEE 802.20 committee works on Mobile Broadband Wireless Access (MBWA) standards.

9.4.8 Bluetooth Technology

Bluetooth technology[1] is defined as the chip technology enabling seamless voice and data connections between a wide range of devices through short-range digital two-way radio. It is an open specification for short-range communications of data and voice between both mobile and stationary devices. For instance, it specifies how mobile phones, Wireless Internet Devices (**WID**s), computers and PDAs interconnect with each other, with computers, and with office or home phones. It also provides a Service Discovery Protocol (SDP) for discovering Bluetooth devices. Bluetooth is thus used for deploying wireless personal area networks in homes and offices, the so called Home Area Networks (HANs).

Bluetooth is an industry standard initially developed by Ericsson in the 1990s and backed by a group of electronics manufacturers including 3Com, IBM, Intel, Lucent Technologies, Microsoft, Motorola, Nokia and Toshiba. The development is now led by Bluetooth SIG (Special Interest Group) which published Bluetooth version 1.0 specification in July 1999 that allows any sort of electronic equipment – from computers and cell phones to keyboards and headphones – to make its own connections, without wires, cables or any direct action from a user. Bluetooth is intended to be a standard that works at two levels:

1. It provides agreement at the physical level – Bluetooth is a radio-frequency standard.
2. It also provides agreement at the next level up, where products have to agree on when bits are sent, how many will be sent at a time and how the parties in a conversation can be sure that the message received is the same as the message sent.

The companies belonging to the Bluetooth SIG, and there are more than 1,000 of them, want to let Bluetooth's radio communications take the place of wires for connecting peripherals, telephones and computers.

Bluetooth communicates at a frequency of 2.4 gigahertz, which has been set aside by international agreement for the use in Industrial, Scientific and Medical (ISM) devices. A number of devices that are already in use take advantage of this same radio-frequency band. Baby monitors, garage-door openers and the newest generation of cordless phones all make use of frequencies in the ISM band. Making sure that Bluetooth and these other devices do not interfere with one another has been a crucial part of the design process. In order to limit spectral emissions and interference, it provides three different power classes corresponding to 10, 20 and 100 m distance operation.

Originally Bluetooth aimed to provide low cost, low power connections between a variety of consumer products. As an example, a Bluetooth link between a laptop and a 3G cellular phone could enable the laptop computer to gain access

[1]Named after Harald Bluetooth who was king of Denmark in the late 900s. He managed to unite Denmark and part of Norway into a single kingdom and then introduced Christianity into Denmark. He left a large monument, the rune stone in Jelling, in memory of his parents. He was killed in 986 during a battle with his son, Svend Forkbeard.

to the Internet by means of the 3G packet data transmission. The work of IEEE 802.15 committee aims to standardise the physical and datalink layers of Wireless Personal Area Networks (WPANs) based on Bluetooth specification which specifies a complete system from the physical layer to the application layer. Although the Bluetooth SIG and IEEE 802.15.1 versions are not identical, it is hoped that they will soon converge to a single standard.

9.4.9 Future of Wireless Technology

Wireless Internet will drive four wireless applications namely, messaging, browsing, interacting and conversing. The emerging trend is for mobile networks to migrate to fully IP based 4G systems. These will use a new spectrum and packet switching technology to support data rates of 10+ Mbps. The 3G and 4G systems will provide multimedia services to users everywhere, while WLANs will provide broadband services in hotspots, and WPANs will connect personal devices together at very short distances.

9.5 ADSL – delivering RT multimedia to the home and small business

ADSL technology is the result of the recent advances in modem technology that converts existing twisted pair telephone subscriber lines into access paths for the transfer of multimedia and high speed digital data. Present day ADSL can sustain downstream transmission speeds up to 6 Mbps to a subscriber and in excess of 800 Kbps in both directions (duplex).

These transmission rates expand the existing access capacities by a factor of 100 or more (V.90 modem supports 56 Kbps nominally) without new cabling, thus transforming the existing public telephone infrastructure which was only used to deliver voice, text and low resolution graphics in the past to a powerful system capable of bringing multimedia, including full motion video, in quality assured real time, to the home and small business.

ADSL is seen as an interim solution which will play a crucial role in the next decade for the delivery of information in video and other multimedia formats. This is expected to pave the way for a full broad band service, which will take decades to reach all prospective customers. In this interim period ADSL will bring movies, television, video catalogues, remote CDROMs, corporate LANs and the global Internet into homes and small businesses.

An ADSL circuit consists of an ADSL modem at each end of a twisted pair telephone subscriber line creating three information channels as follows (see Figure 42):

- high speed downstream channel (up to 6 Mbps)
- medium speed duplex channel (~800 Kbps)
- POTS (Plain Old Telephone Service) or an ISDN channel (64 – 128 Kbps).

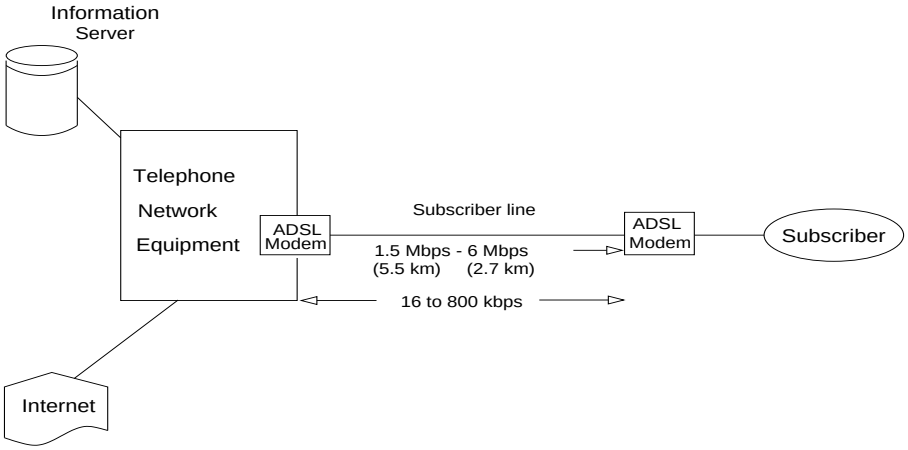


Figure 42: ADSL Channels

It is this asymmetric nature in the data transmission rates in the channels which gives rise to its name Asymmetric Digital Subscriber Line.

The downstream data rates of an ADSL channel depends on:

- length of copper line
- wire gauge
- presence of bridged taps
- cross coupled interference.

In transmitting digital compressed video as a real time signal, ADSL cannot use link level or network level error control procedures commonly found in other data communication systems due to the fact that they are generally based on error recovery by re–transmission best suited for non-real time services. ADLS modems therefore incorporate forward error correction.

Multiple channels are created by the ADSL modem by dividing available bandwidth of a telephone line in one of two ways (see Figure 43):

- FDM – Frequency Division Multiplexing
- Echo Cancellation.

In FDM one frequency band is assigned for upstream data and another for downstream data. The downstream path is then divided by FDM into one or more channels at the desired data rates.

Echo cancellation, a technology used in V.32 and V.34 modems, assigns the upstream band such that it overlaps the downstream band and separates the two by local echo cancellation.

With either technique ADSL splits a 4 KHz region for POTS at the DC end of the frequency spectrum and guarantees the availability of telephone channel even if ADSL fails.

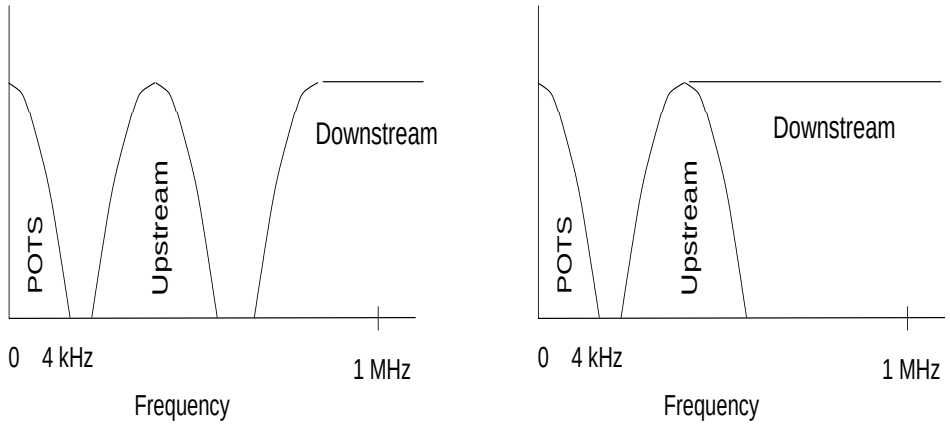


Figure 43: Creating ADSL Channels by (i) FDM and (ii) Echo Cancellation

9.5.1 ADSL Standardisation

At the present moment the following standards are applicable to ADSL:

1. ANSI T1.413 Issue I (1995) – up to 6.1 Mbps
2. ETSI Annex to T1.413 – to reflect European requirements
3. ANSI T1.413i2 (Issue II 1998) – includes multiplexed interfaces at user end and protocols for configuration and network management.
4. ITU–T G.dmt (G.992.1) and G.lite (G.992.2) (1999) – the latter allows the splitting to be done at the telephone network equipment end, at the expense of line speed.

The final step has helped moving towards vendor interoperability and service provider acceptance, further increasing ADSL deployment.

ATM Forum has recognised ADSL as a physical layer transmission protocol for UTP medium.

10 WAP – Wireless Application Protocol

WAP is designed as an application environment based on a set of communication protocols for wireless hand-held devices to enable manufacturer, vendor and technology independent access to the Internet and advanced telephony services. It bridges the gap between the mobile world and the Internet as well as corporate intranets and offers the ability to deliver an unlimited range of services to subscribers independent of their network service provider or type of terminal device. It is developed by mobile and wireless communication companies and includes a mini-browser, scripting language, access function and layered communication specification. This enables the mobile user to access the same wealth of information from a wireless device as they can from a desktop.

WAP is based on a secure specification that allows users to access information instantly via hand-held wireless devices such as:

- mobile phones
- pagers
- two way communication radios
- smart phones
- communicators, etc.

WAP supports most wireless network transmission technologies in use today, most notable being CDMA (Code Division Multiple Access), GSM (Global System for Mobile communications), TDMA (Time Division Multiple Access), HSCSD (High Speed Circuit Switched cellular Data), CDPD (Cellular Digital Packet Data), SMS (Short Messaging Service), GPRS (General Packet Radio Service), UMTS (Universal Mobile Telecommunication System), etc.

The operating systems that support WAP are many. Those specifically engineered for hand-held devices include:

- PalmOS
- Windows CE – Compact Edition
- J2ME – Java 2 Mobile Edition, etc.

Although WAP supports HTML and XML, the WML (Wireless Mark-up Language) is specifically designed for small screens of typically 150 x 150 pixel resolution and one hand navigation without a keyboard as opposed to using point-and-click action of a mouse. Like XML, WML is scalable and extensible as it allows users to define new markup tags.

WAP supports WMLScript (similar to JavaScript) but makes minimal demands on limited memory and CPU power of hand-held devices. Browsers that are specially designed to run on WAP devices and called micro-browsers are of small file size.

10.1 WAP Specification

The WAP specifications define a set of protocols to be used in application, session, transaction, security and transport layers. These are designed to enable operators, manufacturers, and application and content providers to meet the challenges in advanced wireless service provision.

WAP utilises standard Internet components such as XML, UDP (User Datagram Protocol), TLS (Transport Layer Security) and IP. These are optimised for the mobile hand-held device environment with its own unique constraints; low bandwidth, high latency and less connection stability. WAP utilises binary transmission for greater compression of data and WAP sessions are designed to cope with intermittent transmission coverage.

The WAP content is produced using WML and WMLScript and is scalable from two line display on a basic hand-held device to a full graphic screen on smart phones and communicators.

As WAP is based on a scalable layered architecture each layer can develop independently of others. This will also enable content providers to customise

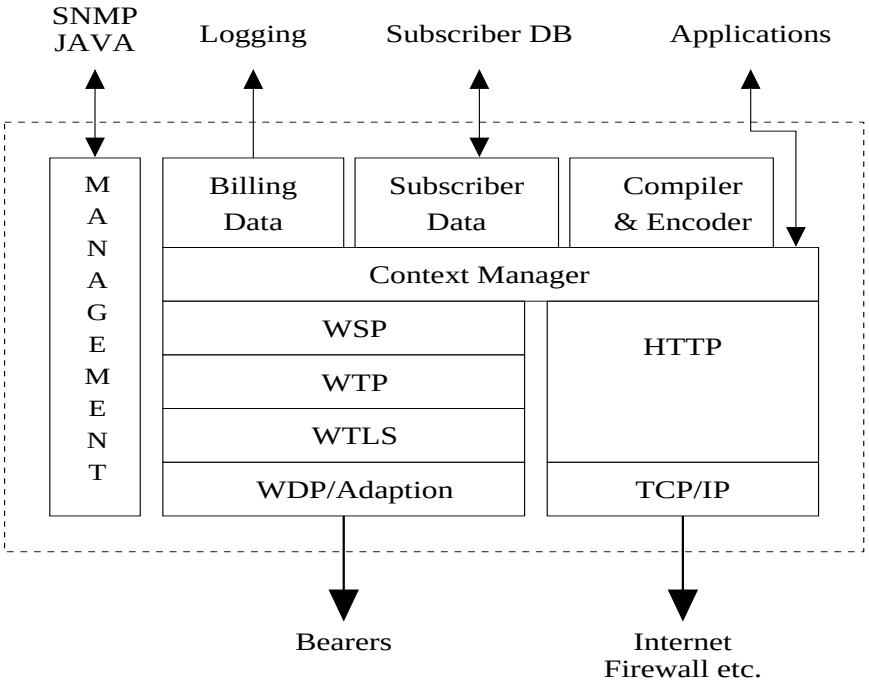


Figure 44: Layered Architecture of the WAP Gateway

content to match client expectations and differentiate themselves from their competitors with new, enhanced information services.

While WAP users benefit from easy, secure access to Internet based information such as unified messaging, banking, and entertainment, they will also enjoy considerable freedom of choice when selecting mobile devices and network operators.

10.2 Architecture of the WAP Gateway

Figure 44 shows the layered architecture of the WAP Gateway.

- HTTP Interface – This serves to retrieve from the Internet, the WAP content requested by the mobile device. WAP content (WML, WMLScript) is then converted into a compact binary byte code for efficient transmission over the air channel.

  The WAP microbrowser software resident within the mobile device interprets this byte code and displays the interactive WAP content.
- WSP – The WAP Session Protocol layer provides a light weight session to allow efficient exchange of data between applications. It supports *connection-oriented* for two way communication between the device and the network and *connection-less* for broadcasting or streaming the data to the device.
- WTP – The WAP Transport Protocol layer provides transaction support adding reliability to the datagram services provided by the WDP layer.

Technology	**Flow Rate**	**Operating Mode**	**Status**
GSM	9.6 Kbps	Circuit	in use
HSCSD	56 Kbps	High flow packets	in use
GPRS	112 Kbps	Packets	extension of GSM
UMTS	2 Mbps	High flow packets	not based on GSM technology. Changes required.

Table 2: Wireless Bearer Technologies

- WTLS – Wireless Transport Layer Security checks data integrity, performs client and server authentication and has encryption facilities to provide secure transport service required by secure applications such as e-commerce, e-banking, etc.
- WDP – The WAP Datagram Protocol is for the transport layer that sends and receives messages via any available bearer network.

The WAP gateway server can operate under two possible scenarios:

1. If the web server provides content in WML, the WAP gateway transmits this data directly to the WAP client.
2. If the web server delivers content in HTML, the WAP gateway server encodes HTML into WML before transmitting to the WAP client.

The present day wireless bearer technologies include GSM, HSCSD, SMS, GPRS and UMTS among others. Table 2 compares some of their important characteristics.

WAP was primarily developed for 2G mobile systems that provide slow data rates between 9.6 – 14.4 Kbps. The development of 3G wireless systems such as UMTS with data rates of 2 – 4 Mbps is expected to resolve the problem of limited bandwidth. With such systems it is natural that WAP will not be needed. However the WAP Forum argues that, even in 3G systems, bandwidth will play a crucial role as new applications will continue to demand higher bandwidths and data rates. In addition they can benefit from the ability to deal with small screen size, low power consumption, carrier independence, multi-device support and intermittent coverage.

Figure 45 shows the 'wapalised' ICTP homepage as it would appear on the screen of a WAP enabled mobile phone.

10.3 Mobile Web Services

Web services include well defined protocol interfaces through which services can be provided to users over the Internet. They are built on emerging technologies

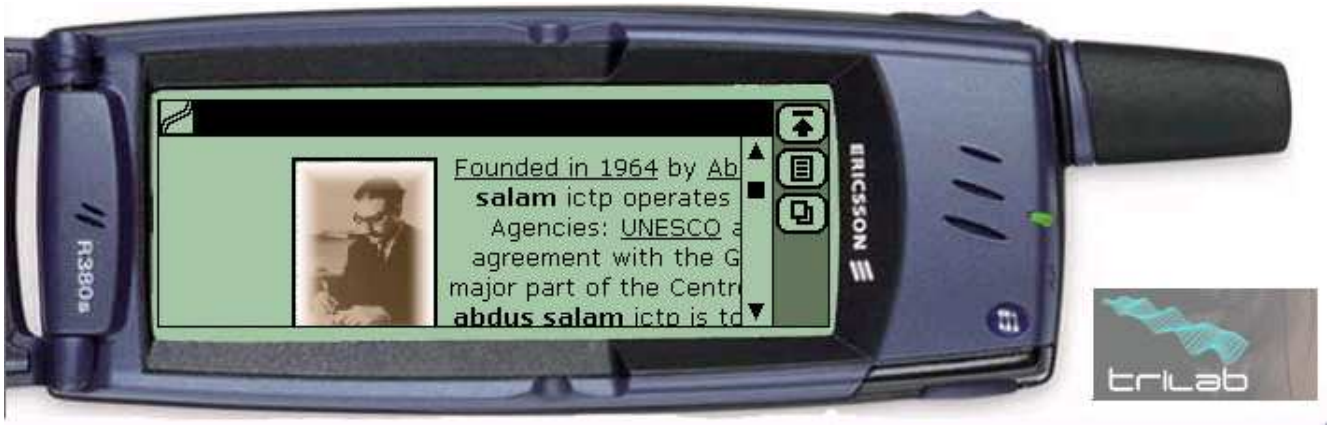


Figure 45: Content display on a WAP enabled mobile phone

such as XML, SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language) of the W3C (World Wide Web Consortium), UDDI (Universal Description, Discovery, and Integration) and HTTP.

Mobile web services provide content delivery, location discovery, user authentication, presence awareness, user profile management, terminal profile management and event notification. Initially wireless terminals are likely to access Mobile Web Services indirectly, through application servers which will manage the interaction with the required Web service.

11 Brief Introduction to Programming the Internet with Java™

This section is intended to illustrate the important principles of TCP/IP socket programming in Java and as such the examples are selected to illustrate them concisely. In this respect the code examples would fall seriously short of production quality code. Although Object Oriented design principles have been adhered to, large amounts of error handling code have been avoided thereby sacrificing robustness for brevity and clarity. For a more complete treatment of Java and object oriented programming the reader is referred to the standard text books [1, 11] and the Chapter on Introduction to Java Programming by Carlos Kavka.

The *java.net* class library package in Java[1] provides a variety of classes, interfaces and exception handling features useful in the implementation of networking applications. The low level APIs allow direct access to network protocols through TCP sockets and UDP datagrams. The high level APIs such as the `URL` and `URLConnection`, on the other hand, allow the faster development of network applications, with much less code to write, albeit at the expense of flexibility.

The classes in the *java.net* package can be classified into three broad categories, namely:

1. Web Classes – to access web documents via their URLs.
2. Raw Classes – to provide *stream based* interfaces to network hosts.
3. Extension Classes – to extend the web classes to cater for new document and content formats.

The following sections outline a number of relevant classes and some code segments to illustrate the use of Java APIs in network programming [5, 18, 20].

11.1 *InetAddress* Class

The `java.net.InetAddress` class represents an Internet Protocol (IP) address and can be used when creating `DatagramPacket` or `Socket` objects. The class does not have public constructor functions. Instead it supports three static methods which return one or more instances of `InetAddress`. The `InetAddress` class contains two fields namely, `hostName` (of type `String` object) and `address` (of type `int`) which are not public. The `hostName` field contains the fully qualified domain name of host, for eg. `www.ictp.trieste.it` and the `address` field contains the 32 bit IP address, for eg. *140.105.16.63* (this is IPv4. This representation will be different for a 16 byte IPv6 address). The `getLocalHost`, `getByName` and `getAllByName` methods must be used by applications to create a new `InetAddress` instance (see the Method Summary at `http://java.sun.com/j2se/1.`

There are other methods in the `java.net` package which return `InetAddress` objects. In Java 2 version 1.5 (Sun now calls this Java 5.0) these methods include:

[1] `http://java.sun.com/j2se/1.5/docs/api`

`DatagramPacket.getAddress()`,
`DatagramSocket.getInetAddress()`,
`DatagramSocket.getLocalAddress()`,
`MulticastSocket.getInterface()`,
`ServerSocket.getInetAddress()`,
`Socket.getInetAddress()`, and
`Socket.getLocalAddress()`.

The following code segments illustrate the use of some of the classes and methods available.

```
import java.net.*;

public class ICTPByName {
    public static void main (String [] args) {
        try {
            InetAddress ICTP_address =
                InetAddress.getByName("www.ictp.trieste.it");
            System.out.println (ICTP_address);
        }
        catch (UnknownHostExecption e) {
            System.out.println
                ("Could not find ICTP Web address");
        }
        catch (SecurityException e) {
            System.out.println
                ("A Security Manager exists - does not allow operation");
        }
    }
}
```

When the code above is executed successfully, it will give:

```
www.ictp.trieste.it/140.105.16.63
```

Creating InetAddress Objects

A string containing the dotted quad form of the IP address can be passed to `InetAddress.getByName()`.

The other two methods are:

`InetAddress.getAllByName(String hostName)` – returns an array containing all addresses;

`InetAddress.getLocalHost()`

All three methods throw `UnknownHostException`. The methods, except `getLocalHost`, also throw `SecurityException`.

InetAddress Class

`public String getHostName()` — returns the hostname of an `InetAddress` object as a string;
`public byte[] getAddress()` — returns the address as a byte array.

Beware!!! Type `byte` *in Java is signed! Addresses between 128 and 255 are returned as negative numbers!! Add 256 to the negative bytes!!!*

`public String getHostName()` — returns the address as a string.

Notice that there are no `setHostName()` and `setAddress()` methods. Java guarantees that packages out side *java.net* cannot change an `InetAddress` object's fields behind its back.

The `InetAddress` class has seen many changes in J2SE 1.4 in order to accommodate IPv6 support. `Inet4Address` and `Inet6Address` classes have been added as two subclasses of `InetAddress` to represent the 32 bit IPv4 addresses and the 128 bit IPv6 addresses respectively. However, in most common cases the IPv6 support is transparent to the developer who will continue to manipulate instances of `InetAddress`. All static methods in `InetAddress` return either an `Inet4Address` or `Inet6Address` depending on the resolution. Similarly all socket methods returning an `InetAddress`, such as `Socket.getInetAddress` will return an instance of `Inet4Address` or `Inet6Address`.

11.2 *URL* Class

This class is associated with a number of constructors which set up `URL` objects with the following general format:
a protocol, a host, a directory and a file name

This constructor throws a `MalformedURLException` if the format of the URL is incorrect. After successfully constructing the `URL` object Java language provides methods to determine its component parts such as protocol, port, host, etc.

HTTP provides a client with the ability to retrieve information from the server about a resource, such as *its size, when last modified* and *content type*. Access to this information is provided by the `URLConnection` class, an instance of which may be obtained from the `openConnection` method of an `URL` object. The `getDate`, `getExpiration` and `getLastModified` methods return dates in the form of the number of milliseconds since January 1, 1970.

java.net.URL Class

It is a final class hence cannot be subclassed.
It is Java's abstraction of the Uniform Resource Locator (URL).
`URL` object is not stored as a string. Rather it is treated in Java as an object with fields such as:

protocol
hostname
port
path
query string, *etc...*

each of which may be set independently.

Constructing URL Objects - Protocol Types

Six constructors available.
All of them throw `MalformedURLException`.
All VMs (Virtual Machines) support protocol types **HTTP** and **file**.
JDK supports **HTTP**, **file**, **FTP**, **mailto**.

Constructing a URL from a string

The following code segments illustrate the use of `URL` concepts.

```
try {
  URL u = new URL("http://mlab-14.ictp.trieste.it");
}

catch (MalformedURLException e) {
  System.out.println("Invalid String");
}
```

Constructing a URL from its component parts

```
try {

  // String protocol, String hostname, String filename

  URL u = new URL("http", "www.ictp.trieste.it", "/index.html");
}

catch (MalformedURLException e) {
  // All known virtual machines should recognise http!
}
```

The above assumes the default port which for HTTP is port 80.

For cases where the default port is not applicable the following allows the redefinition of a port number in the constructor:

```
try {

    // String protocol, String hostname, int port, String file

    URL u = new URl("http", "web.server.somewhere", 1221, "/myfile");
}

catch (MalformedURLException e) {
    System.err.println (e);
}
```

Splitting a URL object into its components

Java provides five public methods:

`getProtocol()` – returns the protocol of this URL;
`getHost()` – returns host name;
`getPort()` – returns the int port number or -1 if not specified;
`getFile()` – returns the file portion of the URL;
`getRef()` – returns the named anchor part of the URL.

Receiving Data from a URL

The following code segment illustrates how to receive data contained in a URL.

```
try {
  URL u = new URL("http://www.hamsterjam.com");
  InputStream is = URL.openStream(); // returns raw data
              // ASCII if text file
              // binary if an image file
  int c;
  while ((c = is.read()) != -1)
     System.out.write(c);
}
catch (IOException e) {
  System.err.println (e);
}
catch (MalformedURLException e) {
    System.err.println (e);
}
```

11.3 *Socket* and *ServerSocket* Classes

TCP is a protocol which allows reliable transport of stream data, of course at the expense of speed (due to increased processing time in handling error detection and correction) and complexity. Applications built using TCP/IP often use the *client-server* model of communication such as web page delivery or file transfer

using FTP. In this model a server waits (or *listens*) for a client to request a service. For standard applications such as HTTP and FTP these services are offered through *well known ports*; 80 for HTTP and 21 for FTP. Other services may be offered at any port number assigned by the server's operating system. On Linux, the port numbers associated with particular services are in */etc/services*.

A `Socket` object represents the Java version of a TCP connection. The sending and receiving paths are represented by an `InputStream` and an `OutputStream` in the socket.

When a `Socket` object is created a connection is made to a specified computer. When a `ServerSocket` connection is made the server on which it is made *listens* to clients and establishes a connection when it *hears* the connection request.

There are a number of constructors for the `Socket` and the most popular ones take the name of a host computer and a port number as arguments.

Socket Constructors

The following constructor creates a TCP socket to the specified port on the specified host and attempts to connect to the remote host.

```
try {
  Socket Google = new Socket ("www.google.com", 80);
  //  String hostname, int portnumber
  //  ...........
  //  send and receive data
  //  ...........
}
catch (UnknownHostExeption e) {
  System.err.println (e);
}
catch (IOExeption e) {
  System.err.println (‘‘An I/O error has occurred’’);
}
catch (SecurityException e) {
  System.out.println
       ("A Security Manager exists - does not allow operation");
}
```

The following is similar to the previous constructor, but uses the `InetAddress` object, rather than the hostname.

```
try {
  InetAddress GoogleAddress = InetAddress.getByName("www.google.com");
  Socket GoogleSocket = new Socket (GoogleAddress, 80);
  //
}
catch (UnknownHostExeption e) {
  System.err.println (e);
}
catch (IOExeption e) {
  System.err.println (e);
}
```

If the socket has to be created using a particular NIC (such as a 100 Mbps Ethernet card as opposed to a 10 Mbps card on a multihomed host), then:

```
try {
  InetAddress FEcard = InetAddress.getByName("FE.mymachine.com");
  Socket FEGoogleSocket = new Socket("www.google.com", 80, FEcard, 0);
  // String host, int port, InetAddress interface, int local port
  // '0' for local port means 'don't care'
  //
}
catch (UnknownHostExeption e) {
  System.err.println (e);
}
catch  (IOExeption e) {
  System.err.println (e);
}
```

The following constructor uses `InetAddress` class:

```
try {
  InetAddress ATMcard = InetAddress.getByName("ATM.mymachine.com");
  InetAddress Google  = InetAddress.getByNAme("www.google.com");
  Socket ATMGoogleSocket=new Socket(Google, 80, ATMcard, 0);
}
catch (UnknownHostExeption e) {
  System.err.println (e);
}
catch (IOExeption e) {
  System.err.println (e);
}
```

Getting Information about a Socket

Given a **socket** object the `getInetAddress()` method returns the IP address of the remote machine the socket is connected to or `null` if not connected.

Similarly the `getPort()` method returns the port to which socket is connected on the remote machine or 0 if it is not connected yet.

There are two ends to a socket. The `getLocalPort()` method returns the local port number to which the socket is bound or -1 if the socket is not bound.

11.4 *DatagramSocket* and *DatagramPacket* Classes

The reliable two way stream transport of TCP is not always required, specially when the amount of data to be transported is relatively small (eg. credit card verification or obtaining time from a NTP server), speed of communication is of prime importance or an application is run over a highly reliable network segment such as a local area network.

The two main classes in Java associated with potentially unreliable UDP communication are `DatagramSocket` and `DatagramPacket`. It is the very nature of

`DatagramSocket` that it is not associated with a particular receiving endpoint. It is simply used as a means of obtaining a UDP port from the local operating system in order to send and receive UDP packets.

The following code segments illustrate the use of some of the useful classes and methods available in the Java Networking API.

Example – Use of InetAddress class

```
/* prints the name/address pair (note!) of local host and the name of a
 * host specified and its IP address in dotted quad format.
 * also prints ALL known IP addresses of a host.  IP names are hard coded.
 */

import java.net.*;
class Hosts {

    public static void main (String args[]) {
        try {
            // useful to know to what host a client is connecting

            InetAddress localAddress = InetAddress.getLocalHost();
            System.out.println (localAddress);
            System.out.println ("==============================");

            // find out the address of a given host

            InetAddress hostAddress = InetAddress.getByName(
                "www.ictp.trieste.it");
            System.out.println ("My name is " + hostAddress.getHostName());
            System.out.println ("My address is " +
                 hostAddress.getHostAddress());
            System.out.println ("==============================");

            // list all registered addresses of the host

            InetAddress[] allAddresses = InetAddress.getAllByName(
                "www.mirror.ac.uk");
                System.out.println ("All known addresses for " +
                allAddresses[0].getHostName());
            for (int i=0; i<allAddresses.length; i++) {
                System.out.println (allAddresses[i]);
            }
        }
        catch (UnknownHostException e) {
            System.out.println ("Error: Cannot resolve");
        }
        catch (SecurityException e) {
            System.out.println
                ("A Security Manager exists - does not allow operation");
        }
    }
}
```

Example – Use of URL class

```
/*
 * This example illustrates the use of URL class to fetch
 * the contents of a web page pointed to by a given URL.
 * The URL of the web server is passed through command line.
 * MalformedURLException and IOException must be caught.
 */

import java.net.*;
import java.io.*;

class GetContents {

    public static void main (String args[]) throws Exception {
        if (args.length != 1) {
            System.out.println ("Usage: java GetContents <url>");
            System.exit(0);
        } // about usage

        String str, urlstring;

// check if url is entered with or without http://

        if (args[0].startsWith("http://")) {
            urlstring = args[0].substring(7);
        }
        else {
            urlstring = args[0];
        }

        URL webURL = new URL("http://"+urlstring);

        InputStream is = webURL.openStream();
        BufferedReader ds = new BufferedReader(new InputStreamReader(is));
        while (true) {
            str = ds.readLine();
            if (str == null) {
                break;
            }
            System.out.println(str.trim());
        }
    }
}
```

Example – Use of UDP DatagramPackets and DatagramSockets

```
/*
 * Illustrates the use of DatagramSocket and DatagramPacket to query the
 * Time Of Day on well known port number 13.
 */

import java.net.*;
import java.io.*;

class UDPDaytimeClient {
    public static void main (String args[]) throws
        UnknownHostException, SocketException, SocketTimeOutException {

        try {
            InetAddress host = InetAddress.getByName(args[0]);
            // get IP address of host
            DatagramSocket mySocket = new DatagramSocket();
            // create a datagram socket

            // send a dummy packet to the UDP daytime service
            // reply contains the response from the time server
            final int DaytimePort = 13;
            DatagramPacket query = new DatagramPacket(new byte[1],1,
                                                host,DaytimePort);

            mySocket.send (query);  // send query packet
            // get response
            byte[] r_buffer = new byte[100];
            DatagramPacket response = new DatagramPacket(r_buffer,
                                                r_buffer.length);
            mySocket.receive(response);  // display response
            String received = new String(response.getData());
            System.out.println ("The time of day received from port "
                 + response.getPort() + " of "
                 + response.getAddress().getHostName()
                 + " is: ");
            System.out.println (received);

            mySocket.close();  // close socket
        }
        catch (IOException e) {
            System.err.println ("System IO error");
            System.exit(1);
        }
    }
}
```

Example – Simple TCP Client

```
/*
 * TCP Client
 */

import java.io.*;
import java.net.*;

public class TCPEchoClient {
    public static void main (String args[]) throws IOException {

        Socket echoSocket = null;
        PrintWriter out = null;
        BufferedReader in = null;

        try { // use port number 4321
            echoSocket = new Socket (args[0], 4321);

            // enable automatic flush

            out = new PrintWriter (echoSocket.getOutputStream(), true);
            in = new BufferedReader (new InputStreamReader
                                     (echoSocket.getInputStream()));
        }
        catch (UnknownHostException e) {
            System.err.println ("Host unknown");
            System.exit(1);
        }
        System.out.println ("Server welcome is: " + in.readLine());
        out.println ("Hello");
        System.out.println ("Server Responded: " + in.readLine());
        out.println ("Hello again");
        System.out.println ("Server now responded: " +in.readLine());

        out.close();
        in.close();
        echoSocket.close();
    }
}
```

Example – Simple TCP Server

```
/*
 * TCP Server
 */

import java.io.*; import java.net.*; import java.util.*;

public class MyTCPServer {
   public static void main (String args []) throws IOException {

        ServerSocket tcpServerSocket = null;
        try {
            tcpServerSocket = new ServerSocket (4321);
        }
        catch (IOException e) {
            System.err.println ("Could not listen on specified port");
            System.exit(1);
        }
        while (true) {
            Socket clientSocket = null;
            try {
                System.out.println ("Server listening");
                clientSocket = tcpServerSocket.accept();
            }
            catch (IOException e) {
                System.err.println ("Accept failed");
                System.exit(1);
            }

            PrintWriter out = new PrintWriter (clientSocket.getOutputStream(),
                                               true);
            BufferedReader in = new BufferedReader (new InputStreamReader
                                    (clientSocket.getInputStream()));

            String inputLine, outputLine;
            out.println ("Welcome to the Echo Server");
            while ((inputLine = in.readLine()) != null) {
                if (inputLine.equals("Bye")) break;
                Date now = new Date (System.currentTimeMillis());
                out.println (now+": " + inputLine);
            }
            out.close(); in.close(); clientSocket.close();
        }
    }
}
```

12 Programming Multitasking Applications

The examples in the previous section demonstrated the features in Java that enable programming client server applications with sockets. These applications however handle one client at a time. If a client connects while another is already being served, the server will continue to serve the first client until it has finished and then serve the second client. This type of servers are called *iterative servers* because they handle clients sequentially. They work best where each client requires a short connection time to the server. However, if service time becomes large then the waiting time experienced by those clients in the queue may be unacceptable.

Threads in Java (covered in standard texts on Java and in the Chapter by Carlos Kavka) provides a convenient mechanism to develop server applications to handle many clients simultaneously allowing each connection to proceed independently without interfering with other connections.

There are two approaches to developing multitasking or concurrent applications using threads:

1. *thread per client* – a new thread is spawned to handle each client connection.
2. *thread pool* – a fixed number of pre-spawned threads work together to serve client connections.

They are independent of the type of client-server protocol (TCP or UDP) that is used.

12.1 Thread per Client

In this type of server, a new thread is created to handle each connection from the client. The server typically executes a loop that runs forever, listening for connection requests on a specified port. When connection requests are made by clients the server repeatedly accepts them by spawning a new thread to handle each connection.

12.2 Thread Pool

The idea of using a Thread Pool is to reduce the overheads created by thread per client approach. When a thread is created it consumes system resources such as CPU time. Each thread has its own data structures that consume system memory. In addition, scheduling and context switching, create extra overhead. As more threads are created, the system resources consumed by these threads increase causing the system to spend more and more time in thread management rather than serving client requests.

One solution to this problem is to limit the total number of threads that is allowed to be spawned and reuse them. Instead of spawning a new thread for

each connection request, the server creates a thread pool with a fixed number of threads and client requests are assigned to a thread from this pool. When the thread finishes servicing the client it returns to the pool so that it can be reassigned to a new request. Connection requests from clients that arrive when all threads in the pool are busy are queued until a thread becomes available.

13 Summary

TCP/IP is a time tested and a proven technology which is now universally available under almost all operating system platforms. In the recent times TCP/IP has been used in transporting time critical streaming data such as video and audio over the global Internet. A new version of Internet Protocol, IPv6, is being formulated taking care of the shortcomings of IPv4 making it more suitable for real time data communication applications.

Real time Transport Protocol (RTP), together with a host of other protocols facilitate the transfer of real time data streams over existing LANs and WANs based on the Internet Protocol (IPv4) technology.

The overall success and acceptability of Ethernet as a LAN technology has led to its enhancement to high performance platforms operating at 100 Mbps (Fast Ethernet) and 1 – 10+ Gbps (Gigabit Ethernet). These two offer the users the familiar Ethernet operating environment with the ability to deliver high bandwidth real time data such as video and imaging to the desktop. In the last few years however, there has been a noticeable drive to develop wireless technology to support computing in the personal and home environment as well as mobile and as a replacement to the local area and wide area connectivity based on wired technologies. As a result there are medium to high bandwidth wireless solutions in the market today slowly revolutionising the way not only homes and offices but entire cities, regions or even countries are interconnected.

Java programming environment provides a rich collection of features to enable network programmers to easily develop sophisticated data communication applications for time critical data over distributed data communication networks.

Over the years the Internet and its TCP/IP protocol suite have been used for such diverse applications for which types neither the Internet nor its protocols were designed in the first place. In the last decade the Internet and its TCP/IP protocol architecture have become the basis for the implementation of GRID Infrastructure [12, 2], aimed at delivering computational power to applications on demand. These applications, although belong to diverse areas of scientific research on the one hand as well as business and commercial on the other, have one thing in common. They both demand enormous CPU performance and storage capacities, that cannot be satisfied by the use of a single or multiple supercomputers of our time for their successful execution.

The concept of GRID infrastructure is to bring together CPUs, sensors, instruments, storage devices, and such like scattered all over the world into one pool that can then be shared over the Internet. Already there are hundreds of national and international GRID infrastructure development projects in motion. In the short term the Internet has been, and will further be, enhanced by developing the middleware components required to perform GRID specific computational tasks. These collaborations have already provided an opportunity to scientists hitherto unavailable, in order to tackle some of the hardest computational problems known to them including earth quake prediction, disaster

control and management, genome mapping, and weather prediction, to name a few.

One salient point that makes the GRID effort different from most other high performance computing projects is that the GRID is available to anyone who wants to use it. In the true spirit of the Internet all the software, documentation and literature are available in the public domain as open source. There is no doubt that this brings a new and exciting computing paradigm to the scientists in the developing world [21].

References

[1] Barnes D J, ***Object-Oriented Programming with Java; An Introduction***, Prentice Hall, 2000.

[2] Berman F, Fox G C and Hey A J G, (Eds), ***Grid Computing – Making the Global Infrastructure a Reality***, Wiley, 2003.

[3] Black U, ***Computer Networks, Protocols, Standards and Interface (2nd Edition)***, Prentice Hall, 1993.

[4] Black U, ***TCP/IP and Related Protocols (2nd Edition)***, McGraw Hill, 1995.

[5] Calvert K L and Donahoo M J, ***TCP/IP Sockets in Java: Practical Guide for Programmers***, Morgan Kaufmann, 2002.

[6] Comer D, ***Internetworking with TCP/IP: Principles, Protocols and Architecture***, Prentice Hall, 1988.

[7] Comer D, ***Internetworking with TCP/IP - Vol I: Principles, Protocols and Architecture (2nd Edition)***, Prentice Hall, 1991.

[8] Comer D and Stevens D L, ***Internetworking with TCP/IP - Vol II: Design, Implementation, and Internals***, Prentice Hall, 1991.

[9] Comer D, ***Computer Networks and Internets with Internet Applications (4th Edition)***, Prentice hall, 2004.

[10] De Prycker M, ***ATM Solutions for Broadband ISDN (3rd Edition)***, Prentice Hall, 1995.

[11] Deitel H M and Deitel P J, ***Java: How to Program (5th Edition)***, Prentice Hall, 2003.

[12] Foster I and Kesselman C, (Eds),***The GRID 2: Blueprint for a New Computing Infrastructure (2nd Edition)***, Morgan Kaufmann, 2004.

[13] Furht B and Ilyas M, (Eds), ***Wireless Internet Handbook – Technologies, Standards and Applications***, CRC Press, 2003.

[14] Goncalves M and Niles K, ***IPv6 Networks***, McGraw-Hill, 1998.

[15] Goncalves M, ***Voice Over IP Networks***, McGraw-Hill, 1999.

[16] Hagen S, ***IPv6 Essentials***, O'Reilly, 2002.

[17] Handel R, Huber M N, and Schroder S, **ATM Networks - Concepts, Protocols, Applications**, Addison-Wesley, 1994.

[18] Harold E R, ***Java Network Programming (2nd Edition)***, O'Reilly, 2000.

[19] Huitema C, ***IPv6: The New Internet Protocol***, Prentice Hall, 1996.

[20] Ince D & Freeman A, ***Programming the Internet with Java***, Addison-Wesley, 1997.

[21] Induruwa A S and Induruwa S D, ***eScience - A 21st century vision for eSri Lanka***, pp151-158, Proc. 22nd National IT Conference, Colombo, Sri Lanka, 2003.

[22] Jain R, ***FDDI Handbook - High Speed Networking using Fibre and Other Media***, Addison-Wesley, 1994.

[23] Keshav S, ***An Engineering Approach to Computer Networking - ATM Networks, the Internet, and the Telephone Network***, Addison-Wesley, 1997.

[24] Miller P E & Miller M A, ***Implementing IPv6 (2nd Edition)***, Hungry Minds, 2000.

[25] Partridge C, ***Gigabit Networking***, Addison Wesley, 1994.

[26] RFC[1] 1075, ***Distance Vector Multicasting Routing Protocol***, November 1988.

[27] RFC 1112, ***Host Extensions for IP Multicasting***, August 1989.

[28] RFC 1585, ***MOSPF – Analysis and Experience***, March 1994.

[29] RFC 1700, ***Assigned Numbers***, October 1994.

[30] RFC 1883, ***Internet Protocol, Version 6 (IPv6) Specification***, April 1996.

[31] RFC 1889, ***RTP: A Transport Protocol for Real Time Applications***, January 1996.

[32] RFC 1890, ***RTP Profile for Audio and Video Conferences with Minimal Control***, January 1996.

[33] RFC 2205, ***Resource Reservation Protocol***, September 1997.

[34] RFC 2236, ***IGMP – Internet Group Management Protocol (Version 2)***, November 1997.

[35] RFC 2362, ***Protocol Independent Multicast***, June 1998.

[36] RFC 3232, ***Assigned Numbers: RFC 1700 is Replaced by an On-line Database***, January 2002.

[1] all RFCs are available from `ftp://rs.internic.net/rfc`

[37] Smith P, ***Frame Relay - Principles and Applications***, Addison Wesley, 1993.

[38] Stallings W, ***ISDN and Broadband ISDN with Frame Relay and ATM (4th Edition)***, Prentice Hall, 1998.

[39] Stallings W, ***Local and Metropolitan Area Networks (6th Edition)***, Prentice Hall, 2000.

[40] Stallings W, ***Wireless Communications and Networks***, Prentice Hall, 2001.

[41] Stallings W, ***High Speed Networks and Internets: Performance and Quality of Service (2nd Edition)***, Prentice Hall, 2002.

[42] Stallings W, ***Data and Computer Communications (7th Edition)***, Prentice Hall, 2003.

[43] Stallings W, ***Computer Networking with Internet Protocols***, Prentice Hall, 2003.

[44] Tanenbaum A S, ***Computer Networks (4th Edition)***, Prentice Hall, 2003.

[45] Wilder F, ***A Guide to the TCP/IP Protocol Suite***, Artech House, 1993.