INF3190

Veronika Heimsbakk
veronahe@student.matnat.uio.no

June 3, 2014

Contents

List of Figures

List of Tables

1 Introduction

This document is a summary of the text book, lectures and own notes in the course INF3190 - Data communication at the Department of informatics, University of Oslo. It sure will contain a lot of flaws and lack some descriptions on certain topics. Mainly used for own repetition before the exam.

2 Network Basics

2.1 Network Components

Data transfer from end system to end system END-SYSTEM (ES) also known as Data Terminal Equipment (DTE) is for example terminal, computers, telephones. Data Circuit terminating Equipment (DCE) is for example modem, multiplexer, repeater. Routers is Data Switching Exchange (DSE).

2.2 Network Structures

Point-to-point channels net = multitude of cable and radio connections often also called a *network*. A cable that always connects two nodes (e.g. telephone).

Broadcasting channels Systems share one communication channel. One sends, all other listen. Wide area use as radio, TV, computer communication. Local area use for local networks.

2.3 Network Types

See table 1.

2.4 OSI Reference Model

The ISO Open Systems Interconnection (OSI) Reference Model (table 2) is a model for layered communication systems. It defines fundamental concepts and terminology, and seven layers and their functionalities.

3 Physical Layer

The physical layer provides the following features:

Mechanical size of plugs, allocation of pins etc. ISO 4903.

Electrical voltage etc. CCITT X.27/V.11.

Distance between processors	**CPU location (on/in..)**	**Example**
<= 0.1m	Boards	multiprocessor systems
1 m	Systems	body/sensor/storage area network
10 m 100 m 1 km	Rooms Buildings Campuses	LAN
10 km	Cities	MAN
100 km 1000 km >=10000 km	National Continents (intern) Planets	WAN

Table 1: Example of network types.

7	Application layer
6	Presentation layer
5	Session layer
4	Transport layer
3	Network layer
2	Data link layer
1	Physical layer

Table 2: The reference model for Open Systems Interconnection (OSI).

Functional definition of switching functions; pin allocation (data, control, timing, ground). CCITT X.24.

Procedural rules for using switching functions. CCITT X.21.

This to initiate, maintain and terminate physical connections between DTE and DCE.

The physical layer ensures the transfer of a transparent bit-stream between data link layer entities. A physical connection permits transfer of a bit-stream in duplex or semi-duplex mode.

3.1 Bit Rate and Baud Rate

Baud rate is the measure of number of symbols (characters) transmitted per unit of time. Where signal speed is the number of signals per second. This changes in amplitude, frequency, phase. Each symbol normally consist

of a number of bits – so the baud rate will only be the same as the bit rate when there is one bit per symbol.

Bit rate is the number of bits transferred per second (bps). Bit rate signal speed may be higher than baud rate because one signal value may transfer several bits.

3.2 Operating Modes

Transfer directions

- Simplex communication: data is always transferred into one direction only.
- Half-duplex/semi-duplex communication: data is transferred into both direction, but never simultaneously.
- Full-duplex communication: data may flow simultaneously in both directions.

Serial and parallel transmission In parallel transmissions the signals are transmitted simultaneously over several channels. And in serial transmission the signals are transmitted sequentially over one channel.

3.2.1 Synchronous transmission

Definition The point in time at which the bit exchange occurs is predefined by a regular clock pulse (requires synchronization). Whereby the clock pulse lasts as long as the transmission of a series of multiple characters takes.

3.2.2 Asynchronous transmission

Definition Clock pulse fixed for the duration of a signal. Termination is marked by a stop signal (bit) or number of bits per signal.

4 Data Link Layer

The data link layer takes the packets it gets from the network layer and encapsulates them into **frames** for transmission. Each frame contains a frame header, a payload field for holding the packet and a frame trailer. On the source machine is an entity, a process, in the network layer that hands some bits to the data link layer for transmission to the destination. The job of the data link layer is to transmit the bits to the destination machine so they can be handed over to the network layer there.

The data link layer can be designed to offer various services. Some may be:

- Unacknowledged connectionless service.
- Acknowledge connectionless service.
- Acknowledge connection-oriented service.

Unacknowledged connectionless service consists of having the source machine send independent frames to the destination machine without having the destination machine acknowledge them. Ethernet is an example of this.

When acknowledged connectionless service is offered, there are still no logical connections used, but each frame sent is individually acknowledged. WiFi is an example of this.

Connection-oriented services is the most sophisticated service that the data link layer may offer the network layer. With this service, a connection is established to the destination machine before it starts to send data.

4.1 Framing

To provide service to the network layer, the data link layer use the service provided to it by the physical layer.

The bit stream received by the data link layer is not guaranteed to be error free. Some bits may have different values and the number of bits received may be less than, equal to, or more than the number of bits transmitted. It is up to the data link layer to detect and correct errors.

The usual approach is for the data link layer to break up the bit stream into discrete frames, compute a short token called a checksum for each frame, and include the checksum in the frame when it is transmitted. When the frame arrives at the destination, the checksum is recomputed.

4.2 Error control

For unacknowledged connectionless service it might be fine if the sender just kept outputting frames without regard to whether they were arriving properly. The usual way to ensure reliable delivery is to provide the sender with some feedback about what is happening at the other end of the line. The protocol calls for the receiver to send back special control frames bearing positive or negative acknowledgements about the incoming frames.

An additional complication comes from the possibility that hardware troubles may case a frame to vanish completely. If the acknowledgement frame is lost, the sender will not know how to proceed. This possibility is dealt with by introducing timers into the data link layer. When the sender transmits a frame, it generally also starts a timer. The timer is set to expire after an interval long enough for the frame to reach the destination,

be processed there, and have the acknowledgement propagate back to the sender.

4.3 Flow control

Two kinds of flow control are commonly used. **Feedback-based flow control**, the receiver sends back information to the sender giving it permission to send more data, or at least telling the sender how the receiver is doing.

Rate-based flow control, the protocol has a built-in mechanism that limits the rate at which senders may transmit data, without using feedback from the receiver.

5 Medium Access Control Sublayer

Network links can be divided into two categories: those using point-to-point connections and those using broadcast channels.

In any broadcast network, the key issue is how to determine who gets to use the channel when there is competition for it. Broadcast channels are also referred to as **multi-access channels** or **random access channels**.

The protocols used to determine who goes next on a multi-access channel belong to a sublayer of the data link layer called the **MAC (Medium Access Control)** sublayer.

5.1 Channel Allocation

5.1.1 Static Channel Allocation

The traditional way of allocating a single channel among multiple competing users is to chop up its capacity by using one of the multiplexing schemes such as FDM (Frequency Division Multiplexing).

5.1.2 Assumptions for Dynamic Channel Allocation

1. **Independent Traffic.** The model consist of N independent **stations** (e.g., computers), each with a program or user that generates frames for transmission. The expected number of frames generated in an interval of length Δt is $\lambda \Delta t$, where λ is a constant (the arrival rate of new frames). Once a frame has been generated, the station is blocked and does nothing until the frame has been successfully transmitted.

2. **Single Channel.** A single channel is available for all communication. All stations can transmit on it and all can receive from it. The stations are assumed to be equally capable, though protocols may assign different roles.

3. **Observable Collisions.** If two frames are transmitted simultaneously, they overlap in time and the resulting signal is garbled. This is a **collision**. All stations can detect that a collision has occurred. A collided frame must be transmitted again later. No errors other than those generated by collisions occur.

4. **Continuous or Slotted Time.** Time may be assumed continuous, in which case frame transmission can begin at any instant. Alternatively, time may be slotted into discrete intervals. Frame transmissions must then begin at the start of a slot. A slot may contain 0, 1, or more frames, corresponding to an idle slot, a successful transmission, or a collision.

5. **Carrier Sense or No Carrier Sense.** With the carrier sense assumption, stations can tell if the channel is in use before trying to use it. No station will attempt to use the channel while it is sensed as busy. If there is no carrier sense, stations cannot sense the channel before trying to use it. They just go ahead and transmit. Only later can they determine whether the transmission was successful.

5.2 Multiple Access Protocols

5.2.1 ALOHA

Let users transmit whenever they have data to be sent. Colliding frames will be damaged. Sender then just wait a random amount of time and sends again. Systems in which multiple users share a common channel in a way that can lead to conflicts are known as **contention** systems.

The throughput of the ALOHA system is maximized by having a uniform frame size rather that by allowing variable-length frames. Whenever two frames try to occupy the channel at the same time, there will be a collision. Both frames will be destroyed and both will have to be transmitted later.

5.2.2 Carrier Sense Multiple Access Protocols

Protocols in which stations listen for a carrier (i.e., a transmission) and act accordingly are called **carrier sense protocols**.

1-persistent CSMA When a station has data to send, it first listens to the channel to see if anyone else is transmitting at that moment. If the channel is idle, the stations sends its data. Otherwise, if the channel is busy, the station just waits until it becomes idle. Then the stations transmit a frame. If collision occurs, the station waits a random amount of time and starts all over again. The protocol is called 1-persistent because the station transmit with a probability of 1 when it finds the channel idle.

Non-persistent CSMA A station sense the channel when it wants to send a frame, and if no one else is sending, the station begins doing so itself. However, if the channel is already in use, the station does not continually sense it for the purpose of seizing it immediately upon detecting the end of the previous transmission. Instead, it waits a random period of time and then repeats the algorithm.

P-persistent CSMA When a station becomes ready to send, it senses the channel. If it is idle, it transmits with a probability p. With a probability $q = 1-p$, it defers until the next slot. If that slot is idle, it either transmits or defers again, with probabilities p and q. This process is repeated until either the frame has been transmitted or another stations has begun transmitting. The unlucky stations acts as if there had been a collision (waits a random amount of time and starts again).

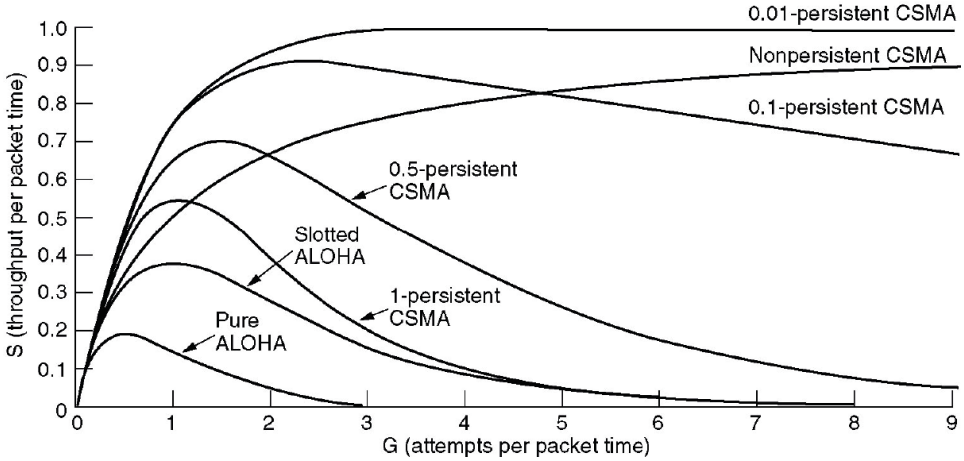

Figure 1: Comparison of the channel utilization versus load for various random access protocols.

CSMA with Collision Detection Persistent and non-persistent CSMA protocols are an improvement over ALOHA because they ensure that no station begins to transmit while the channel is busy. However, if two stations sense the channel to be idle and begin transmitting at the same time, their signals will still collide. **CSMA/CD (CSMA with Collision Detection)** is the basis of the classic Ethernet LAN. Collision detection is an analogue process. The implications are that a received signal must not be tiny compared to the transmitted signal and that the modulation must be chosen to allow collisions to be detected. This does not work on wireless.

5.3 Wireless LANs

5.3.1 The 802.11 MAC Sublayer Protocol

Radios are nearly always half duplex, meaning that they cannot transmit and listen for noise bursts at the same time on a single frequency. With Ethernet, a station just waits until the ether goes silent and then starts transmitting. If it does not receive a noise burst back while transmitting the first 64 bytes, the frame has almost assuredly been delivered correctly. With wireless, this collision detection mechanism does not work.

Trying to avoid collisions is done with the protocol **CSMA/CA (CSMA with Collision Avoidance)**. This protocol is conceptually similar to Ethernet's CSMA/CD, with channel sensing before sending and exponential back-off after collisions. However, a station that has a frame to send starts with a random back-off. It does not wait for a collision. The station waits until the channel is idle, by sensing that there is no signal for a short period of time (called the DIFS), and counts down idle slots, pausing when frames are sent. It sends its frame when the counter reaches 0. If the frame gets through, the destination immediately sends a short acknowledgement. Lack of an acknowledgement is inferred to indicate an error. In this case, the sender doubles the backoff period and tries again.

6 Network Layer

The network layer is responsible for packet forwarding including routing through intermediate routers. The network later provides the functional and procedural means of transferring variable-length data sequences from a source to destination host via one or more network, while maintaining the quality of service functions. To achieve its goals, the network layer must know about the topology of the network and choose appropriate paths though it. It must also take care when choosing routes to avoid overloading some of the communication lines and routers while leaving others idle. And it is up to the network layer to handle communication when source and destination are in different networks.

6.1 Design Issues

6.1.1 Services provided to the Transport Layer

The network layer provides services to the transport layer at the network layer/transport layer interface. The services need to have the following goals in mind:

1. The services should be independent of the router technology.

2. The transport layer should be shielded from the number, type, and topology of the routers present.

3. The network addresses made available to the transport layer should use a uniform numbering plan, even across LANs and WANs.

This gives a lot of freedom in writing detailed specifications of the services. There is a discussion centred on whether the network layer should provide connection-oriented or connectionless service.

The connectionless service-solution (represented by the Internet community) says that the routers job is moving packets around and nothing else. This concludes with that the network layer should contain the primitives `send packet` and `receive packet` and little else. No packet ordering and flow control should be done, because the hosts are going to do that anyway. This is an example of the **end-to-end argument**, a design that has been very influential in shaping the Internet.

On the other hand, the telephone companies, argues that the network should provide reliable, connection-oriented service. In this view, quality of service is the dominant factor.

However, today connectionless services have grown tremendously in popularity.

6.1.2 Comparison of Virtual-Circuit and Datagram Networks

If connectionless service is offered, packets are injected into the network individually and routed independently of each other. In this context packets are called **datagrams** and the network is called a **datagram network**. If connection-oriented service is uses, a path from the source router all the way to the destination router must be established before any data packets can be sent. This connection is called a **virtual circuit (VC)**, and the network is called a **virtual-circuit network**.

6.2 Routing Algorithms

The **routing algorithm** is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on. If the network uses datagrams, this decision must be made anew for every arriving data packet since the best route may have changed since last time. If the network uses virtual circuit internally, routing decisions are made only when a new virtual circuit is being set up. Thereafter, packets just follow the already established route. The latter case is sometimes called **session routing** since a route remains in force for an entire session (e.g., while logged in over a VPN).

One can think of a router as having two processes inside it. One that handles each packet as it arrives, looking up the outgoing line to use for it

Issue	**Datagram network**	**Virtual-circuit network**
Circuit set-up	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed though the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

Table 3: Comparison of datagram and virtual-circuit networks.

in the routing tables. This process is **forwarding**. The other process is responsible for filling in and updating the routing tables.

Routing algorithms can be grouped into two major classes: nonadaptive and adaptive. **Nonadaptive algorithms** do not base their routing decisions on any measurements or estimates of the current topology and traffic. Instead, the choice of the route to use to get from I to J is computed in advance, offline, and downloaded to the routers when the network is booted. Also called **static routing**. Because it does not respond to failures, static routing is mostly useful for situations in which the routing choice is clear.

Adaptive algorithms change their routing decisions to reflect changes in the topology, and sometimes changes in the traffic as well. These **dynamic routing** algorithms differ in where they get their information when they change the routers and what metric is used for optimization.

6.2.1 Shortest Path Algorithm

In graph theory, the shortest path problem is the problem of finding a path between two vertices in a graph such that the sum of the weights of its constituent edges is minimized.

Dijkstra's algorithm Dijkstra's original algorithm without min-priority queue runs in time $O(|V|^2)$, where $|V|$ is the number of vertices.

```
1  function Dijkstra(Graph, source):
2    for each vertex v in Graph:
3      dist[v] := infinity;
4      previous[v] := undefined;
5    end for
6
7    dist[source] := 0;
8    Q := the set of all nodes in Graph;
9
10   while Q is not empty:
11     u := vertex in Q with smallest distance in dist[];
12     remove u from Q;
13     if dist[u] = infinity:
14       break;
15     end if
16
17     for each neighbour v of u:
18       alt := dist[u] + dist_between(u,v);
19       if alt < dist[v]:
20         dist[v] := alt;
21         previous[v] := u;
22         decrease-key v in Q;
23       end if
24     end for
25   end while
26   return dist[], previous[];
27 end function
```

6.2.2 Distance Vector Routing

A **distance vector routing** algorithm operates by having each router maintain a table giving the best known distance to each destination and which link to use to get there. These tables are updated by exchanging information with the neighbours. Eventually, every router knows the best link to reach each destination.

The most common distance vector routing algorithm is the **Bellman-Ford** algorithm. This was the original ARPANET routing algorithm and was also used in the Internet under the name RIP.

In distance vector routing, each router maintains a routing table indexed by, and containing one entry for each router in the network. This entry has two parts: the preferred outgoing line to use for that destination and an estimate of the distance to that destination.

6.2.3 Link State Routing

Distance vector routing was used in the ARPANET until 1979, when it was replaced by link state routing. The idea behind link state routing can be stated as five parts:

1. Discover its neighbours and learn their network addresses.
2. Set the distance or cost metric to each of its neighbours.
3. Construct a packet telling all it has just learned.
4. Send this packet to and receive packets from all other routers.
5. Compute the shortest path to every other router.

The complete topology is distributed to every router. Then Dijkstra's algorithm can be run at each router to find the shortest path to every other router.

6.2.4 Broadcast Routing

Sometimes it is needed to send messages to many or all other hosts. Sending packet to all destinations simultaneously is called **broadcasting**. There are various methods to achieve this.

One is called **multidestination routing**, in which each packet contains either a list of destinations or a bit map indicating the desired destinations. When a packet arrived at a router, the router checks all the destinations to determine the set of output lines that will be needed. The router generated a new copy of the packet for each output line to be used and includes in each packet only those destinations that are to use the line.

Another one is **reverse path forwarding**. When a broadcast packet arrives at a router, the router checks to see if the packet arrived on the link that is normally used for sending packets toward the source of the broadcast. If so, there is an chance that the broadcast packet itself followed the best route from the router and is therefore the first copy to arrive.

6.2.5 Multicast Routing

When we need a way to send messages to well-defined groups that are numerically large in size, but small compared to the network as a whole we can use **multicasting**.

All multicasting schemes require some way to create and destroy groups and to identify which routers are members of a group. If the group is dense, broadcast is a good start because it efficiently gets the packet to all parts of the network. But broadcast will reach some routers that are not members of the group, which is wasteful. A solution is to prune the broadcast spanning tree by removing links that do not lead to members.

6.2.6 Anycast Routing

In **anycast**, a packet is delivered to the nearest member of a group. Schemes that find these paths are called **anycast routing**. This is useful when any node will do. For example, anycast is used in the Internet as a part of DNS.

6.3 Congestion Control Algorithms

Too many packets present in the network causes packet delay and loss that degrades performance. This situation is called **congestion**. The most effective way to control congestion is to reduce the load that the transport layer is placing on the network. This requires the network and transport layers to work together.

Unless the network is well designed, it may experience a **congestion collapse**, in which performance plummets as the offered load increases beyond the capacity. This can happen because packets can be sufficiently delayed inside the network that they are no longer useful when they leave the network.

The difference between congestion control and flow control is that congestion control has to do with making sure the network is able to carry the offered traffic. Flow control, in contrast, relates to the traffic between a particular sender and a particular receiver.

6.3.1 Approaches to Congestion Control

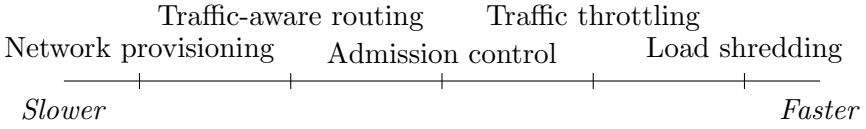

Figure 2: Time line of approaches to congestion control.

The most basic way to avoid congestion is to build a network that is well matched to the traffic it carries. If there is a low-bandwidth link on the path along which most traffic is directed, congestion is likely. Sometimes resources can be added dynamically when there is serious congestion (turning on spare routers or enabling lines that are normally used only for backup). This is

called **provisioning** and happens on a time scale of months, driven by long-term traffic trends.

To make the most of the existing network capacity, routers can be tailored to traffic patterns that change during the day as network users wake and sleep in different time zones. Routers may be changes to shift traffic away from heavily used paths by changing the shortest path weights. This is called **traffic-aware routing**.

However, sometimes it is not possible to increase capacity. Then you have to decrease the load to beat congestion. In a virtual-circuit network, new connection can be refused if they would cause the network to become congested. This is called **admission control**.

6.4 The Network Layer in the Internet

Top 10 principles from RFC 1958 – Architectural Principles of the Internet.

1. **Make sure it works.** Do not finalize the design or standard until multiple prototypes have successfully communicated with each other.
2. **Keep it simple.** When in doubt, use the simplest solution.
3. **Make clear choices.** If there are several ways of doing the same thing, choose one.
4. **Exploit modularity.** This principle leads directly to the idea of having protocol stacks, each of whose layers is independent of all the other ones. In this way, if one module or layer need to be changes, the other ones will not be affected.
5. **Expect heterogeneity.** Different types of hardware, transmission facilities, and applications will occur on any large network. To handle then, the network design must be simple, general, and flexible.
6. **Avoid static operations and parameters.** If parameters are unavoidable, it is best to have the sender and receiver negotiate a value rather than defining fixed choices.
7. **Look for a good design; it need not be perfect.**
8. **Be strict when sending and tolerant when receiving.** Send only packets that rigorously comply with the standards, but expect incoming packets that may not be correct and try to deal with them.
9. **Think about scalability.**
10. **Consider performance and cost.**

The glue that holds the whole Internet together is the network layer protocol, **IP (Internet Protocol)**. IP was designed from the beginning with internetworking in mind.

Communication in the Internet works as follows. The transport layer takes data streams and breaks them up so that they be sent as IP packets. In theory, packets can be up to 64 KB each, but in practice they are usually not more than 1500 bytes (due to Ethernet frames). IP routers forward each packet through the Internet, along a path from one router to the next, until the destination is reached. At the destination, the network layer hands the data to the transport layer, which gives it to the receiving process.

6.4.1 The IPv4 Protocol

An IPv4 diagram consists of a header part and a body or payload part. The bits are transmitted from left to right and top to bottom, with the high-order bit of the *Version* field going first. This is a "big-endian" network byte order. On little-endian machines, a software conversion is required on both transmission and reception. In retrospect, little endian would have been a better choice, but at the time IP was designed, no one knew it would come to dominate computing.

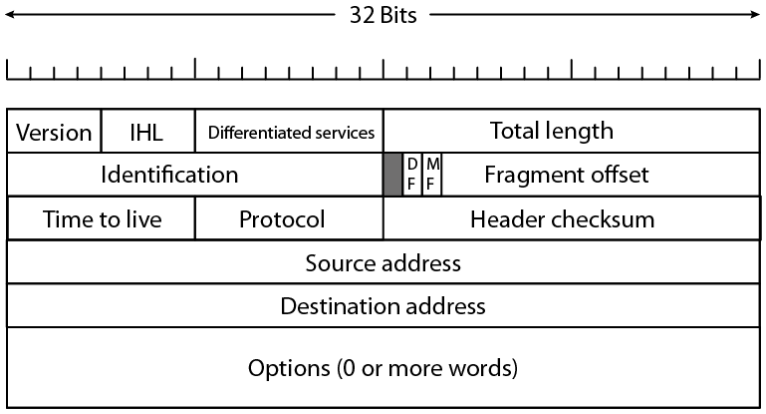

Figure 3: The IPv4 header

Version This field keeps track of which version of the protocol it belongs to. By including the version at the start of each datagram, it becomes possible to have a transition between versions over a long period of time.

IHL Since the header length is not constant, a field in the header, IHL, is provided to tell how long the header is. Minimum value is 5, which applies when no options are present. Maximum is 15, which limits the header to 60 bytes.

Differentiated services The top 6 bits are used to mark the packet with its service class. The bottom 2 bits are used to carry explicit congestion

notification information, such as whether the packet has experienced congestion.

Total length Includes everything in the datagram—both header and data. Maximum length is 65,535 bytes.

Identification This is needed to allow the destination host to determine which packet a newly arrived fragment belongs to. All the fragments of a packet contain the same identification value.

Next comes an unused bit

DF This stands for Don't Fragment. It is an order to the routers not to fragment the packet.

MF This stands for More Fragments. All fragments except the last one have this bit set.

Fragment offset This tells where in the current packet this fragment belongs.

Time to live A counter used to limit packet lifetimes. Supposed to count in seconds, allowing maximum lifetime of 225 sec. When it hits zero, the packet is discarded and a warning packer is sent back to the source host.

Protocol Tells which transport process to give the packet to. TCP is one possibility, so are UDP and some others.

Header checksum The algorithm to add up all the 16-bit halfwords of the header as they arrive, using one's complement arithmetic, and then take the one's complement of the result. Is assumed to be zero upon arrival. Useful for detecting errors while the packet travels through the network.

Source address IP address of source network interface.

Destination address IP address of the destination network interface.

Options See table 4.

IP Addresses A defining feature of IPv4 is its 32-bits addresses. It refers to a network interface, so if a host is on two networks, it must have two IP addresses.

Each 32-bit address is comprised of a variable-length network portion in the top bits and a host portion in the bottom bits. The network portion has the same value for all hosts on a single network, such as an Ethernet LAN.

Option	Description
Security	Specifies how secret the datagram is.
Strict source routing	Gives the complete path to be followed.
Loose source routing	Gives a list of routers not to be missed.
Record route	Makes each router append its IP address.
Timestamp	Makes each router append its address and timestamp.

Table 4: Some of the IP options.

This means that a network corresponds to a contiguous block of IP address space. This block is called **prefix**.

IP addresses are written in **dotted decimal notation**. Each of the 4 bytes is written in decimal from 0 to 255. The 32-bit hexadecimal address 80D00297 is written as 128.208.2.151.

The key advantage of prefixes is that routers can forward packets based on only the network portion of the address. The host portion does not matter to the routers because all hosts on the same network will be sent in the same direction. This makes routing tables much smaller than they would otherwise be.

A disadvantage is that if addresses are assigned to networks in too large blocks, there will be many addresses that are allocated but not in use.

6.4.2 The IPv6 Protocol

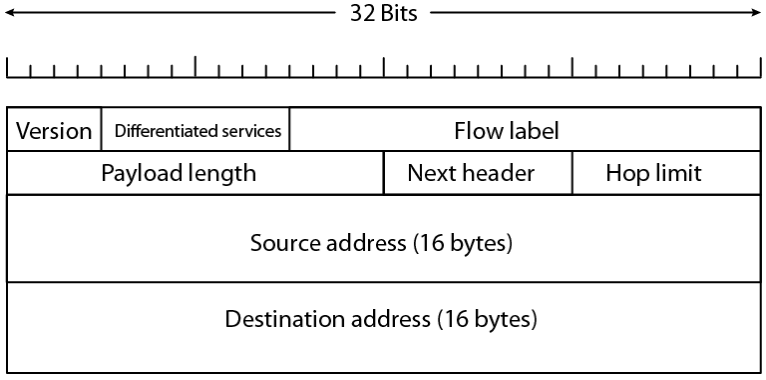

Figure 4: The IPv6 fixed header (required).

Version Is always 6 for IPv6, as it is always 4 for IPv4.

Differentiated services Same as for IPv4.

Flow label Provides a way for a source and destination to mark groups of packets that have the same requirements and should be treated in

the same way by the network, forming a pseudoconnection. Each flow for quality of service purposes is designated by the source address, destination address, and flow number. This design means that up to 2^{20} flows may be active at the same time between a given pair of IP addresses.

Payload length This field tells how many bytes follow the 40-byte header. The 40 header bytes are no longer counted as part of the length (as they used to). This change means the payload can now be 65,545 byes instead of 65,515 bytes.

Next header Tells which transport protocol handler (e.g., TCP, UDP) to pass the packet to.

Hop limit Same as the *Time to live* field in IPv4.

Source address IP address of source network interface.

Destination address IP address of the destination network interface.

A new notation has been devised for writing 16-byte addresses. They are written as eight groups of four hexadecimal digits with colons between the groups:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Since many addresses will have many zeros, three optimizations have been authorized. Leading zeros within a group can be omitted. One of more groups of 16 zero bits can be replaced with a pair of colons. Then it becomes:

8000::123:4567:89AB:CDEF

And IPv4 addresses can be written as a pair of colons and an old dotted decimal number:

::192.31.20.46

There is a lot of 16-bytes addresses. 2^{128} of them. Which is approximately 3×10^{38}.

Extension Headers Some of the missing IPv4 fields are occasionally still needed, so IPv6 introduce the concept of optional **extension headers**.

7 Transport Layer

The transport layer builds on the network layer to provide data transport from a process on a source machine to a process on a destination machine with a desired level of reliability that is independent of the physical networks currently in use.

Extension header	Description
Hop.by-hop options	Miscellaneous information for routers
Destination options	Additional information for the destination
Routing	Loose list of routers to visit
Fragmentation	Management of datagram fragments
Authentication	Verification of the sender's identity
Encrypted security payload	Information about the encrypted contents

Table 5: IPv6 extension headers.

7.1 Services Provided to the Upper Layers

The ultimate goal of the transport layer is to provide efficient, reliable, and cost-effective data transmission service to its users, normally processes in the application layer. The software and/or hardware within the transport layer that does the work is called the **transport entity**.

The transport code runs entirely on the users' machines, but the network layer mostly run on the routers, which are operated by the carrier. The users have no real control over the network layer. To achieve that, the only possibility is to put on top of the network layer another layer that improves the quality of the service. If, in a connectionless network, packets are lost or mangled, the transport entity can detect the problem and compensate for it by using retransmissions. If, in a connection-oriented network, a transport entity is informed halfway through a long transmission that its network connection has been abruptly terminated, it can set up a new network connection to the remote transport entity.

7.2 Socket Primitives

Primitive	Meaning
`SOCKET`	Create a new communication endpoint.
`BIND`	Associate a local address with a socket.
`LISTEN`	Announce willingness to accept connections; give queue size.
`ACCEPT`	Passively establish an incoming connection.
`CONNECT`	Actively attempt to establish a connection.
`SEND`	Send some data over the connection.
`RECEIVE`	Receive some data from the connection.
`CLOSE`	Release the connection.

Table 6: The socket primitives for TCP.

7.3 Transport Protocols

The transport service is implemented by a **transport protocol** used between the two transport entities.

7.3.1 Addressing

The method normally used is to define transport address to which process can listen for connection requests. In the Internet, these endpoints are called **ports**. At the transport layer they are called **TSAP (Transport Service Access Point)**. The analogous endpoints in the network layer is called **NSAPs (Network Service Access Points)**. IP addresses are examples of NSAPs.

Application processes, both clients ans servers, can attach themselves to a local TSAP to establish a connection to a remote TSAP. These connections run through NSAPs on each host. The purpose of having TSAPs is that in some networks, each computer has a single NSAP, so some way is needed to distinguish multiple transport endpoints that share that NSAP.

While stable TSAP addresses work for a small number of key services that never change (e.g., web server), user processes, in general, often want to talk to other user processes that do not have TSAP addresses that are known in advance, or that may exist for only a short time. To handle this situation, an alternative scheme can be used. In this scheme, there is a special process called a **portmapper**. To find the TSAP address corresponding to a given service name, such as "BitTorrent", a user sets up a connection to the portmapper. The user then sends a message specifying the service name, and the portmapper sends back the TSAP address. Then the user releases the connection with the portmapper and establishes a new one with desired service.

7.4 The Internet Transport Protocols: UDP

UDP (User Datagram Protocol)[1] is a connectionless protocol. It does almost nothing beyond sending packets between applications, letting applications build their own protocols on top as needed.

UDP transmit **segments** consisting of an 8-byte header (figure 5) followed by the payload.

The two **ports** identifies the endpoints within the source and destination machines. When a UDP packet arrives its payload is handed to the process attached to the destination port. This attachment occurs when the `BIND` primitive or similar is used. The main value of UDP over using raw IP is the addition of the source and destination ports.

[1] UDP is described in RFC 768.

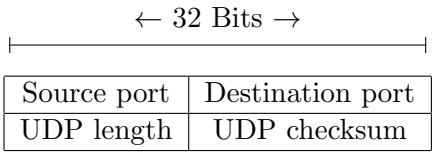

Figure 5: The UDP header.

By copying the *Source port* field from the incoming segment into the *Destination port* field of the outgoing segment, the process sending the reply can specify which process on the sending machine is to get it. The *UDP length* field includes the 8-byte header and the data. The minimum length is 8 bytes, the maximum length is 65,515 bytes. Which is lower than the largest number that will fit in 16 bits because of the size limit on IP packets. An optional *Checksum* is also provided for extra reliability. It checksums the header, the data and a conceptual IP pseudo-header.

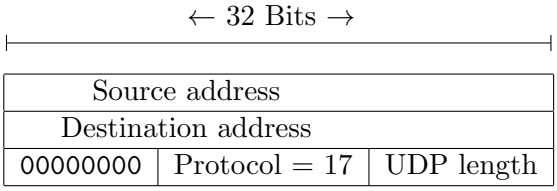

Figure 6: The IPv4 pseudo-header included in the UDP checksum.

UDP is not a very clever choice of protocol. However, an application that used UDP is for example DNS (Domain Name System).

7.5 The Internet Transport Protocols: TCP

TCP (Transmission Control Protocol)[2] was specifically designed to provide a reliable end-to-end byte stream over an unreliable internetwork.

Each machine supporting TCP has a TCP transport entity, either a library procedure, a user process, or most commonly part of the kernel. In all cases, it manages TCP streams and interfaces to the IP layer. A TCP entity accepts user data streams from local processes, breaks them up into pieces not exceeding 64 KB, and sends each piece as a separate IP datagram. When datagrams containing TCP data arrive at a machine, they are given to the TCP entity, which reconstructs the original byte streams.

7.5.1 TCP Service Model

TCP service is obtained by both the sender and the receiver creating end-points, called **sockets**. Each socket has the host IP address and a 16-bit

[2]Formally defined in RFC 793.

number local to that host, called **port**. A port is the TCP name for TSAP.

A socket may be used for multiple connections at the same time. Two or more connections may terminate at the same socket.

Port numbers below 1024 are reserved for standard services. They are called **well-known ports**, see table 7.

Port	Protocol	Use
20,21	FTP	File transfer
22	SSH	Remote login
25	SMTP	E-mail
80	HTTP	World Wide Web
110	POP-3	Remote e-mail access
143	IMAP	Remote e-mail access
443	HTTPS	Secure Web (HTTP over SSL/TLS)

Table 7: Some assigned ports.

7.5.2 The TCP Header

A key feature of TCP is that every byte on a TCP connection has its own 32-bit sequence number. Separate 32-bit sequence numbers are carried on packets for the sliding window position in one direction and for acknowledgements in the reverse direction.

The sending and receiving TCP entities exchange data in the form of segments. A **TCP segment** consist of a fixed 20-byte header (plus an optional part) followed by zero or more data bytes. There are two limits that restrict the segment size. Each segment, including the header, must fit in the 65,515-byte IP payload. And each link has an **MTU (Maximum Transfer Unit)**. Each segment must fit in the MTU at the sender and receiver so that it can be sent and received in a single, unfragmented packet.

Source/Dest port Identifies the local end points of the connection.

Sequence/ACK number Both are 32-bits because every byte of data is numbered in a TCP stream.

Data offset Tells how many 32-bit words are contained in the TCP header.

Reserved A 4-bit field that is not used.

Flags *CWR* and *ECE* are used to signal congestion, when *ECN (Explicit Congestion Notification)* is used. *CWR* is set to signal *Congestion Window Reduced* from the TCP sender to the TCP receiver so that it knows the sender has slowed down and can stop sending the *ECN-Echo*. *URG* is set to 1 if the *Urgent pointer* is in use. The *Urgent*

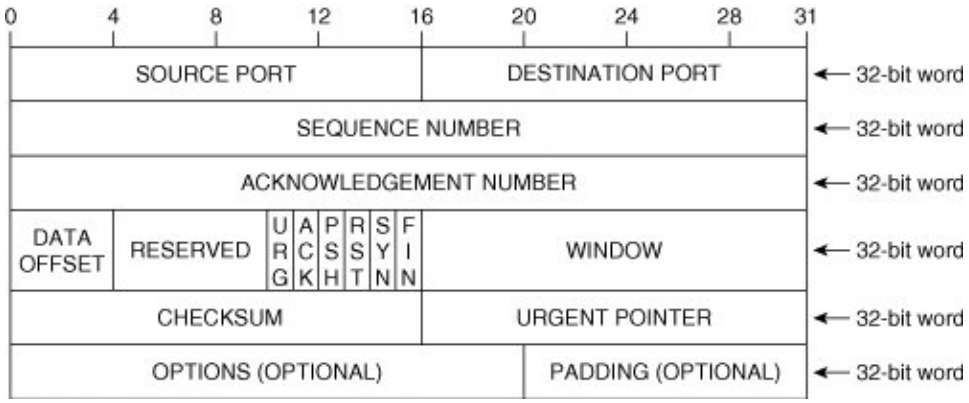

Figure 7: The TCP header.

pointer is used to indicate a byte offset from the current sequence number at which urgent data are to be found. The *ACK* is set to 1 to indicate that *Acknowledgement number* is valid. *PSH* indicates pushed data. *RST* is used to abruptly reset a connection that has become confused due to a host crash or some other reason. *SYN* is used to establish connections. *FIN* is used to release a connection.

Window size This field tells how many bytes may be sent starting at the byte acknowledgement.

Checksum Provided for extra reliability. It checksums the header, the data, and a conceptual pseudo-header in the same way as UDP.

Options Provides a way to add extra facilities.

7.5.3 TCP Connection Establishment

Connections are established in TCP by means of the three-way handshake. To establish a connection, one side (e.g. server), passively waits for an incoming connection by executing the `LISTEN` and `ACCEPT` primitives, wither specifying a specific source or nobody in particular.

The other side (e.g. client), executes a `CONNECT` primitive, specifying the IP address and port to which it wants to connect, the maximum TCP segment size it is willing to accept, and optionally some user data. The `CONNECT` primitive sends a TCP segment with the `SYN` bit on and `ACK` bit off and waits for a response.

When this segment arrives, the TCP entity checks to see if there is a process that has done a `LISTEN` on the port given in *Destination port* field. If there is a process listening, that process is given the incoming TCP segment.

7.5.4 TCP Connection Management Modelling

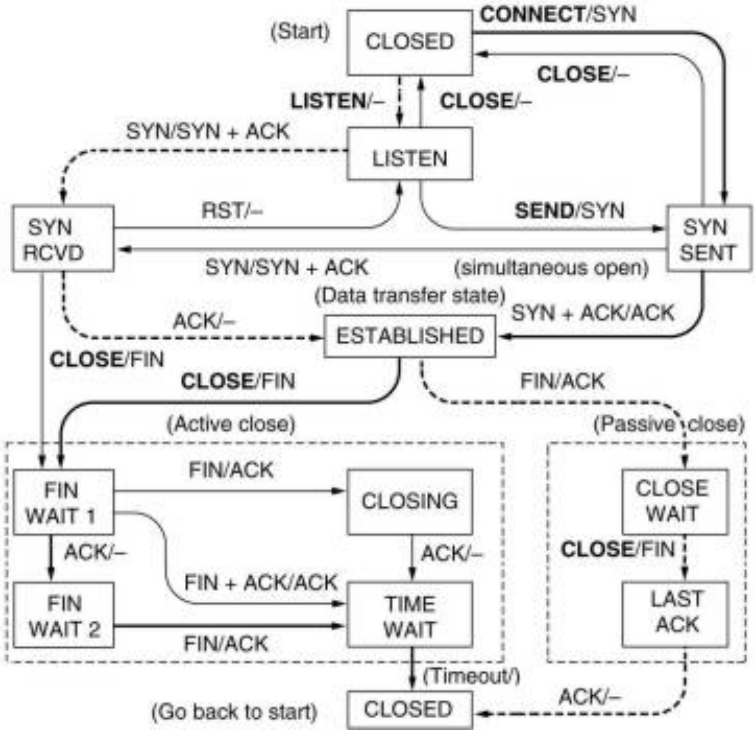

Figure 8: TCP connection management finite state machine. The heavy solid line is the normal path for a client. The heavy dashed line is the normal path for a server. The light lines are unusual events. Each transition is labelled by the event causing it and the action resulting from it, separated by a slash.

7.6 The Real-Time Transport Protocol

RTP (Real-time Transport Protocol)[3] is a generic real-time transport protocol for multiple applications. There are two aspects of real-time transport. The first is the protocol for transporting audio and video data in packets. The second is the processing that takes place to play out the audio and video at the right time. The functions fit into the protocol stack as shown in figure 9.

The basic function of RTP is to multiplex several real-time data streams onto a single stream of UDP packets. The UDP stream can be both unicasting and multicasting. Because RTP just uses normal UDP, its packets are not treated specially by the routers unless some IP features are enabled.

[3]Described in RFC 3550.

State	Description
CLOSED	No connection is active or pending.
LISTEN	The server is waiting for an incoming call.
SYN RCVD	A connection request has arrived, wait for ACK.
SYN SENT	The application has started to open a connection.
ESTABLISHED	The normal data transfer state.
FIN WAIT 1	The application has said it is finished.
FIN WAIT 2	The other side has agreed to release.
TIMED WAIT	Wait for all packets to die off.
CLOSING	Both sides have tried to close simultaneously.
CLOSE WAIT	The other side has initiated a release.
LAST ACK	Wait for all packets to die off.

Table 8: The states used in the TCP connection management finite state machine.

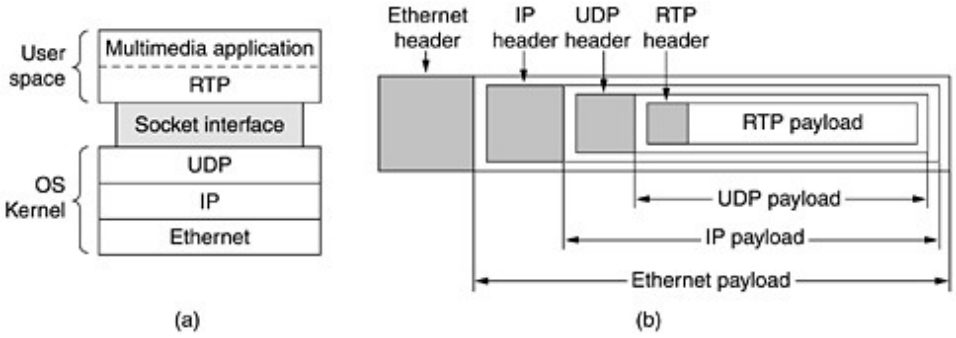

Figure 9: (a) The position of RTP in the protocol stack. (b) Packet nesting.

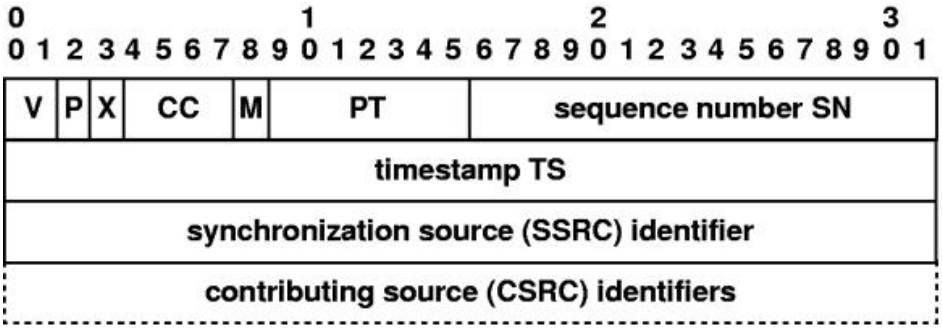

Figure 10: The RTP header.

Figure 10 shows the RTP header. It consists of three 32-bit words and potentially some extensions.

Version Version of RTP. Is already at 2.

P This bit indicates that the packet has been padded to a multiple of 4 bytes.

X This bit indicated that an extension header is present.

CC Tells how many contributing sources are present, from 0 to 15.

M This bit is an application-specific marker bit. Can be used to mark the start of a video frame etc.

Payload type Tells which encoding algorithm has been used (e.g., uncompressed 8-bit audio, MP3, etc.).

Sequence number This is just a counter that is incremented on each RTP packet sent.

Timestamp This is produced by the stream's source to note when the first sample in the packet was made.

Synchronization source identifier Tells which stream the packet belongs to. The method used to multiplex and demultiplex multiple data streams onto a singe stream of UDP packets.

Contributing source identifiers Are used when mixers are present in the studio. Streams being mixed are listed here.

7.7 RTCP—The Real-time Transport Control Protocol

This protocol handles feedback, synchronization, and the user interface. It does not transport any media samples.

The first function can be used to provide feedback on delay, variation in delay or jitter, bandwidth, congestion, and other network properties to the sources. This information can be used by the encoding process to increase the data rate (better quality) when the network is functioning well and to cut back the data rate when there is trouble in the network.

8 Application Layer

8.1 DNS—The Domain Name System

The essence of DNS[4] is the invention of a hierarchical, domain-base naming scheme and a distributed database system for implementing this naming scheme. It is primarily used for mapping host names and e-mail destinations to IP addresses but can also be used for other purposes.

[4]Defined in RFC 1034 and 1035.

8.1.1 The DNS Name Space

The Internet is divided into over 200 top-level **domains**, where each domain covers many hosts. Each domain is partitioned into subdomains, and these are further partitioned and so on. All these domains can be represented as a tree (see figure 11).

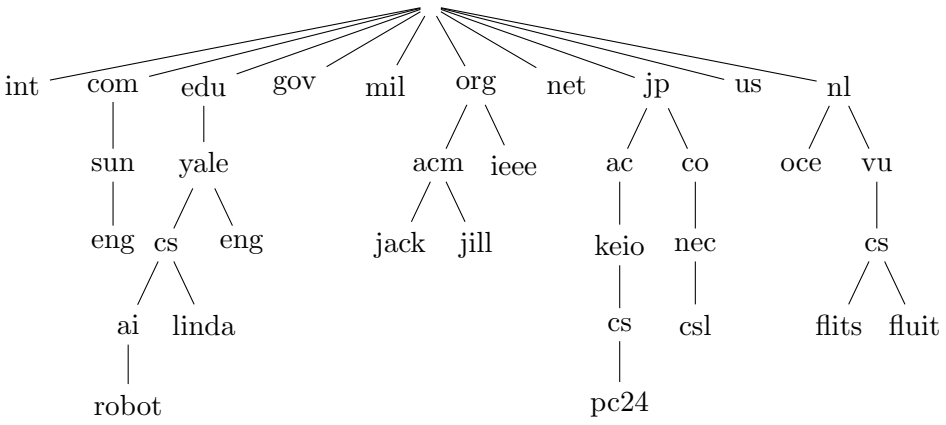

Figure 11: A portion of the Internet domain name space. Here int to net is generic, and jp to nl are countries.

The leaves of the tree represent domains that have no subdomains. A leaf may contain a single host, or it may represent a company and contain thousands of hosts.

The top-level domains come in two flavours: generic and countries. The original generic domains where *com* (commercial), *edu* (educational institution), *gov* (the U.S. Federal Government, *int* (certain international organizations), *mil* (the U.S. armed forces), *net* (network providers), and *org* (non-profit organizations). The country domains include one entry for every country, as defined in ISO 3166.

Each domain id named by the path upward from it to the root. The components are separated by periods. Domain names can be either absolute or relative. An absolute domain name always ends with a period (e.g., *eng.sun.com.*), whereas a relative one does not.

8.2 The Simple Mail Transfer Protocol

Within the Internet, e-mail is delivered by having the source machine establish a TCP connection to port 25 of the destination machine. Listening to this port is an e-mail daemon that speaks SMTP (Simple Mail Transfer Protocol). This daemon accepts incoming connections and copies messages from them into the appropriate mailboxes. SMTP is a simple ASCII protocol.

Even though the SMTP protocol is well defined, a few problems can still arise. Some older implementations cannot handle messages exceeding 64 KB. Another problem is if the client and server have different time outs, one of them may give up while the other is still busy. And in rare situations, infinite mailstorms can be triggered.

8.3 The World Wide Web

The Client Side

1. The browser determines the URL (by seeing what was selected).
2. The browser asks DNS for the IP address of *www.itu.org*.
3. DNS replies with 156.106.192.32.
4. The browser makes a TCP connection to port 80 on 156.106.192.32.
5. It then sends over a request asking for file */home/index.html*.
6. The *www.itu.org* server sends the file */home/index.html*.
7. The TCP connection is released.
8. The browser displays all the text in */home/index.html*.
9. The browser fetches and displays all images in this file.

The Server Side

1. Accept a TCP connection from a client (a browser).
2. Get the name of the file requested.
3. Get the file (from disk).
4. Return the file to the client.
5. Release the TCP connection.

8.3.1 HTTP—The HyperText Transfer Protocol

The transfer protocol used throughout the World Wide Web is **HTTP (HyperText Transfer Protocol)**. It specifies what message clients may send to servers and what responses they get back in return. Each interaction consists of one ASCII request, followed by one RFC 822 MIME-like response. All clients and all servers must obey this protocol.

Connections HTTP 1.1. supports **persistent connections**. With them, it is possible to establish a TCP connection, send a request and get a response, and then send additional requests and get additional responses. By amortizing the TCP setup and release over multiple requests, the relative overhead due to TCP is much less per request. It is also possible to pipeline requests, that is, send request 2 before the response to request 1 has arrived.

Method	Description
`GET`	Request to read a Web page.
`HEAD`	Request to read a Web page's header.
`PUT`	Request to store a Web page.
`POST`	Append to a named resource (e.g., a Web page).
`DELETE`	Remove the Web page.
`TRACE`	Echo the incoming request.
`CONNECT`	Reserved for future use.
`OPTIONS`	Query certain options.

Table 9: The built-in HTTP request methods.

Methods

8.4 WAP—The Wireless Application Protocol

The basic idea is to use the existing digital wireless infrastructure. Users can literally call up a WAP gateway over the wireless link and send Web page requests to it. The gateway then checks its cache for the page requested. If present, it sends it; if absent, it fetches it over the wired Internet.

8.5 SIP—The Session Initiation Protocol

This protocol describes how to set up Internet telephone calls, video conferences, and other multimedia connections. SIP is a single module, but it has been designed to interwork well with existing Internet applications.

SIP can establish two-party sessions (ordinary telephone call), multiparty sessions (everyone can hear and speak), and multicast sessions (one sender, many receivers). The sessions may contain audio, video, or data, the latter being useful for multiplayer real-time games. SIP just handles setup, management, and termination of sessions. Other protocols, such as RTP/RTCP, are used for data transport. SIP is an application layer protocol and can run over UDP and TCP.

References

[1] Andrew S. Tanenbaum, *Computer Networks*. Pearson Education International, 4th edition, 2003.