[Unit 2: Internet Protocol Overview]

Internet Technology (CSC-402)

Jagdish Bhatta

Central Department of Computer Science & Information Technology

Tribhuvan University

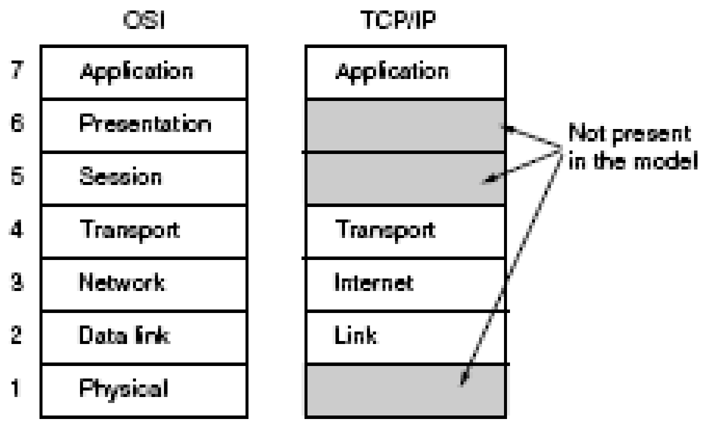
Internet Protocol Suite:

The **Internet protocol suite** is the set of communications protocols used for the Internet and similar networks, and generally the most popular protocol stack for wide area networks. It is commonly known as **TCP/IP**, because of its most important protocols: Transmission Control Protocol (TCP) and Internet Protocol (IP), which were the first networking protocols defined in this standard.

TCP/IP and the IP Layer overview:

The **Internet protocol suite** is the set of communications protocols used for the Internet and similar networks, and generally the most popular protocol stack for wide area networks. It is commonly known as **TCP/IP**, because of its most important protocols: Transmission Control Protocol (TCP) and Internet Protocol (IP), which were the first networking protocols defined in this standard

TCP/IP provides end-to-end connectivity specifying how data should be formatted, addressed, transmitted, routed and received at the destination. It has four abstraction layers, each with its own protocols



From lowest to highest, the layers are:

1. The link layer (commonly Ethernet) contains communication technologies for a local network.
2. The internet layer (IP) connects local networks, thus establishing internetworking.
3. The transport layer (TCP) handles host-to-host communication.
4. The application layer (for example HTTP) contains all protocols for specific data communications services on a process-to-process level (for example how a web browser communicates with a web server).

It loosely defines a four-layer model, with the layers having names, not numbers, as follows:

Application layer (process-to-process): This is the scope within which applications create user data and communicate this data to other processes or applications on another or the same host. The communications partners are often called *peers*. This is where the "higher level" protocols such as SMTP, FTP, SSH, HTTP, etc. operate.

The Application Layer combines a few services that the OSI separates into three layers. **These services are end user-related: authentication, data handling, and compression.** This is where email, Web browsers, telnet clients, and other Internet applications get their connections.

The application layer contains the higher-level protocols used by most applications for network communication. Examples of application layer protocols include the File Transfer Protocol (FTP) and the Simple Mail Transfer Protocol (SMTP). **Data coded according to application layer protocols are then encapsulated into one or (occasionally) more transport layer protocols (such as TCP or UDP), which in turn use lower layer protocols to effect actual data transfer. Since the IP stack defines no layers between the application and transport layers, the application layer must include any protocols that act like the OSI's presentation and session layer protocols.** This is usually done through libraries.

Transport layer (host-to-host): The transport layer constitutes the networking regime between two network hosts, either on the local network or on remote networks separated by routers. The transport layer provides a uniform networking interface that hides the actual topology (layout) of the underlying network connections. **This is where flow-control, error-correction, and connection protocols exist, such as TCP**. This layer deals with opening and maintaining connections between Internet hosts.

The transport layer establishes host-to-host connectivity, meaning it handles the details of data transmission that are independent of the structure of user data and the logistics of exchanging information for any particular specific purpose. **Its responsibility includes end-to-end message transfer independent of the underlying network, along with error control, segmentation, flow control, congestion control, and application addressing (port numbers). End to end message transmission or connecting applications at the transport layer can be categorized as connection-oriented, implemented in TCP, or connectionless, implemented in UDP.**

The layer simply establishes a basic data channel that an application uses in its task-specific data exchange. For this purpose the layer establishes the concept of the port, a numbered logical construct allocated specifically for each of the communication channels an application needs. For many types of services, these port numbers have been standardized so that client computers may address specific services of a server computer without the involvement of service announcements or directory services.

Internet layer (internetworking): The internet layer has the task of exchanging datagrams across network boundaries. **It is therefore also referred to as the layer that establishes internetworking, indeed, it defines and establishes the Internet. This layer defines the**

addressing and routing structures used for the TCP/IP protocol suite. The primary protocol in this scope is the Internet Protocol, which defines **IP addresses**. Its function in routing is to transport datagrams to the next IP router that has the connectivity to a network closer to the final data destination.

The internet layer has the responsibility of sending packets across potentially multiple networks. Internetworking requires sending data from the source network to the destination network. This process is called routing.

In the Internet protocol suite, the Internet Protocol performs two basic functions:

- ***Host addressing and identification*: This is accomplished with a hierarchical addressing system (see IP address).**
- ***Packet routing*: This is the basic task of sending packets of data (datagrams) from source to destination by sending them to the next network node (router) closer to the final destination.**

The internet layer only provides an unreliable datagram transmission facility between hosts located on potentially different IP networks by forwarding the transport layer datagrams to an appropriate next-hop router for further relaying to its destination. With this functionality, the internet layer makes possible internetworking, the interworking of different IP networks, and it essentially establishes the Internet. The Internet Protocol is the principal component of the internet layer, and it defines two addressing systems to identify network hosts computers, and to locate them on the network. The schemes are IPV4 and IPV6.

Link layer: This layer defines the networking methods within the scope of the local network link on which hosts communicate without intervening routers. This layer describes the protocols used to describe the local network topology and the interfaces needed to affect transmission of Internet layer datagrams to next-neighbor hosts (the OSI data link layer).

This is the lowest component layer of the Internet protocols, as TCP/IP is designed to be hardware independent. As a result TCP/IP is able to be implemented on top of virtually any hardware networking technology.

The link layer is used to move packets between the Internet layer interfaces of two different hosts on the same link. The processes of transmitting and receiving packets on a given link can be controlled both in the software device driver for the network card, as well as on firmware or specialized chipsets. These will perform data link functions such as adding a packet header to prepare it for transmission, then **actually transmit the frame over a physical medium.** The TCP/IP model includes specifications of translating the network addressing methods used in the Internet Protocol to data link addressing, such as Media Access Control (MAC), however all other aspects below that level are implicitly assumed to exist in the link layer, but are not explicitly defined.

Transmission Control Protocol (TCP):

Transmission Control Protocol (TCP) is a protocol that provides a reliable stream delivery and connection service to applications. TCP uses sequenced acknowledgment and is able to retransmit packets as needed.

The protocol corresponds to the transport layer of TCP/IP suite. TCP provides a communication service at an intermediate level between an application program and the Internet Protocol (IP). That is, when an application program desires to send a large chunk of data across the Internet using IP, instead of breaking the data into IP-sized pieces and issuing a series of IP requests, the software can issue a single request to TCP and let TCP handle the IP details. TCP is utilized extensively by many of the Internet's most popular applications, including the World Wide Web (WWW), E-mail, File Transfer Protocol, Secure Shell, peer-to-peer file sharing, and some streaming media applications.

Transmission Control Protocol accepts data from a data stream, segments it into chunks, and adds a TCP header creating a TCP segment. The TCP segment is then encapsulated into an Internet Protocol (IP) datagram. A TCP segment is "the packet of information that TCP uses to exchange data with its peers**. A TCP segment consists of a segment *header* and a *data* section. The TCP header contains 10 mandatory fields, and an optional extension field. The data section follows the header. Its contents are the payload data carried for the application**. The length of the data section is not specified in the TCP segment header.

TCP protocol operations may be divided into three phases. **Connections must be properly established in a multi-step handshake process (*connection establishment*) before entering the *data transfer* phase. After data transmission is completed, the *connection termination* closes established virtual circuits and releases all allocated resources.**

The TCP header appears as follows:

TCP Header

Offsets	Octet	0								1								2								3							
Octet	**Bit**	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	**0**	Source port																Destination port															
4	**32**	Sequence number																															
8	**64**	Acknowledgment number (if ACK set)																															
12	**96**	Data offset				Reserved 0 0 0			NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
16	**128**	Checksum																Urgent pointer (if URG set)															
20 ...	**160 ...**	Options (if Data Offset > 5, padded at the end with "0" bytes if necessary) ...																															

- **Source port (16 bits)** – identifies the sending port
- **Destination port (16 bits)** – identifies the receiving port
- **Sequence number (32 bits)** – has a dual role:
  - If the SYN flag is set (1), then this is the initial sequence number. The sequence number of the actual first data byte and the acknowledged number in the corresponding ACK are then this sequence number plus 1.
  - If the SYN flag is clear (0), then this is the accumulated sequence number of the first data byte of this packet for the current session.
- **Acknowledgment number (32 bits)** – if the ACK flag is set then the value of this field is the next sequence number that the receiver is expecting. This acknowledges receipt of all prior bytes (if any). The first ACK sent by each end acknowledges the other end's initial sequence number itself, but no data.
- **Data offset (4 bits)** – specifies the size of the TCP header in 32-bit words. The minimum size header is 5 words and the maximum is 15 words thus giving the minimum size of 20 bytes and maximum of 60 bytes, allowing for up to 40 bytes of options in the header. This field gets its name from the fact that it is also the offset from the start of the TCP segment to the actual data.
- **Reserved (3 bits)** – for future use and should be set to zero
- **Flags (9 bits) (i.e. Control bits)** – contains 9 1-bit flags
  - NS (1 bit) – ECN-nonce concealment protection (added to header by RFC 3540).

- CWR (1 bit) – Congestion Window Reduced (CWR) flag is set by the sending host to indicate that it received a TCP segment with the ECE flag set and had responded in congestion control mechanism (added to header by RFC 3168).
- ECE (1 bit) – ECN-Echo indicates

- If the SYN flag is set (1), that the TCP peer is ECN capable.
- If the SYN flag is clear (0), that a packet with Congestion Experienced flag in IP header set is received during normal transmission (added to header by RFC 3168).

- URG (1 bit) – indicates that the Urgent pointer field is significant
- ACK (1 bit) – indicates that the Acknowledgment field is significant. All packets after the initial SYN packet sent by the client should have this flag set.
- PSH (1 bit) – Push function. Asks to push the buffered data to the receiving application.
- RST (1 bit) – Reset the connection
- SYN (1 bit) – Synchronize sequence numbers. Only the first packet sent from each end should have this flag set. Some other flags change meaning based on this flag, and some are only valid for when it is set, and others when it is clear.
- FIN (1 bit) – No more data from sender

- **Window size (16 bits)** – the size of the *receive window*, which specifies the number of bytes (beyond the sequence number in the acknowledgment field) that the sender of this segment is currently willing to receive
- **Checksum (16 bits)** – The 16-bit checksum field is used for error-checking of the header and data
- **Urgent pointer (16 bits)** – if the URG flag is set, then this 16-bit field is an offset from the sequence number indicating the last urgent data byte
- **Options (Variable 0-320 bits, divisible by 32)** – The length of this field is determined by the data offset field. **Options have up to three fields: Option-Kind (1 byte), Option-Length (1 byte), Option-Data (variable).** The Option-Kind field indicates the type of option, and is the only field that is not optional. Depending on what kind of option we are dealing with, the next two fields may be set: the Option-Length field indicates the total length of the option, and the Option-Data field contains the value of the option, if applicable. For example, an Option-Kind byte of 0x01 indicates that this is a No-Op option used only for padding, and does not have an Option-Length or Option-Data byte following it. An Option-Kind byte of 0 is the End Of Options option, and is also only one byte. An Option-Kind byte of 0x02 indicates that this is the Maximum Segment Size option, and will be followed by a

byte specifying the length of the MSS field (should be 0x04). Note that this length is the total length of the given options field, including Option-Kind and Option-Length bytes. So while the MSS value is typically expressed in two bytes, the length of the field will be 4 bytes (+2 bytes of kind and length). In short, an MSS option field with a value of 0x05B4 will show up as (0x02 0x04 0x05B4) in the TCP options section.

- **Padding** – The TCP header padding is used to ensure that the TCP header ends and data begins on a 32 bit boundary. The padding is composed of zeros

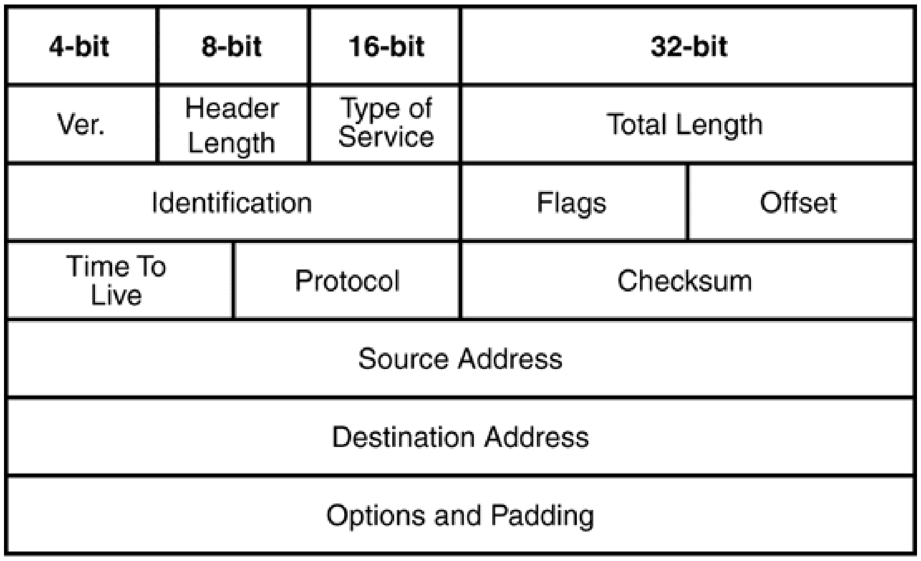
Internet Protocol (IP):

The **Internet Protocol** (**IP**) is the principal communications protocol used for relaying datagrams (also known as network packets) across an internetwork using the Internet Protocol Suite. Responsible for routing packets across network boundaries, it is the primary protocol that establishes the Internet. **IP manages how packets are delivered to and from servers and clients. IP defines datagram structures that encapsulate the data to be delivered. It also defines addressing methods that are used to label the datagram source and destination.**

The Internet Protocol is responsible for addressing hosts and routing datagrams (packets) from a source host to the destination host across one or more IP networks. For this purpose the Internet Protocol defines an addressing system that has two functions: identifying hosts and providing a logical location service. This is accomplished by defining standard datagrams and a standard addressing system.

Each datagram has two components, a header and a payload. The IP header is tagged with the source IP address, destination IP address, and other meta-data needed to route and deliver the datagram. The payload is the data to be transported. This process of nesting data payloads in a packet with a header is called encapsulation.

The IP header appears as follows:



4-bit	8-bit	16-bit	32-bit
Ver.	Header Length	Type of Service	Total Length
Identification		Flags	Offset
Time To Live	Protocol		Checksum
Source Address			
Destination Address			
Options and Padding			

Each field contains information about the IP packet that it carries.

Version Number
The version number indicates the version of IP that is in use for this packet. IP version 4 (Ipv4) is currently in widespread use.

Header Length
The header length indicates the overall length of the header. The receiving machine then knows when to stop reading the header and start reading data.

Type of Service
Mostly unused, the Type of Service field indicates the importance of the packet in a numerical value. Higher numbers result in prioritized handling.

Total Length
Total length shows the total length of the packet in bytes. The total packet length cannot exceed 65,535 bytes or it will be deemed corrupt by the receiver.

Identification
If there is more than one packet (an invariable inevitability), the identification field has an identifier that identifies its place in line, as it were. Fragmented packets retain their original ID number.

Flags
The first flag, if set, is ignored. If the DF (Do Not Fragment) flag is set, under no circumstances can the packet be fragmented. If the MF (More Fragments) bit is turned on (1), there are packet fragments to come, the last of which is set to off (0).

Offset

If the Flag field returns a 1 (on), the Offset field contains the location of the missing piece(s) indicated by a numerical offset based on the total length of the packet.

Time To Live (TTL)

Typically 15 to 30 seconds, TTL indicates the length of time that a packet is allowed to remain in transit. If a packet is discarded or lost in transit, an indicator is sent back to the sending computer that the loss occurred. The sending machine then has the option of resending that packet.

Protocol

The protocol field holds a numerical value indicating the handling protocol in use for this packet.

Checksum

The checksum value acts as a validation checksum for the header.

Source Address

The source address field indicates the address of the sending machine.

Destination Address

The destination address field indicates the address of the destination machine.

Options and Padding

The Options field is optional. If used, **it contains codes that indicate the use of security, strict or loose source routing, routing records, and timestamping.** If no options are used, the field is called *padded* and contains a 1. Padding is used to force a byte value that is rounded. Following indicates the bit counts for the options available;

Table: Bit Counts

Class	**Number**	**Option**
0	0	End of option list
0	2	Military security
0	3	Loose source

Class	Number	Option
		routing
0	7	Routing record*
0	9	Strict source routing
2	4	Timestamping

**This option adds fields.*

IP Address:

An **Internet Protocol address** (**IP address**) is a numerical label assigned to each device (e.g., computer, printer) participating in a computer network that uses the Internet Protocol for communication. An IP address serves two principal functions: host or network interface identification and location addressing. Its role has been characterized as follows: "*A name indicates what we seek. An address indicates where it is. A route indicates how to get there.*"

The designers of the Internet Protocol defined an IP address as a 32-bit number and this system, known as Internet Protocol Version 4 (IPv4), is still in use today. However, due to the enormous growth of the Internet and the predicted depletion of available addresses, a new version of IP (IPv6), using 128 bits for the address, was developed in 1995. IPv6 was standardized as RFC 2460 in 1998, and its deployment has been ongoing since the mid-2000s.

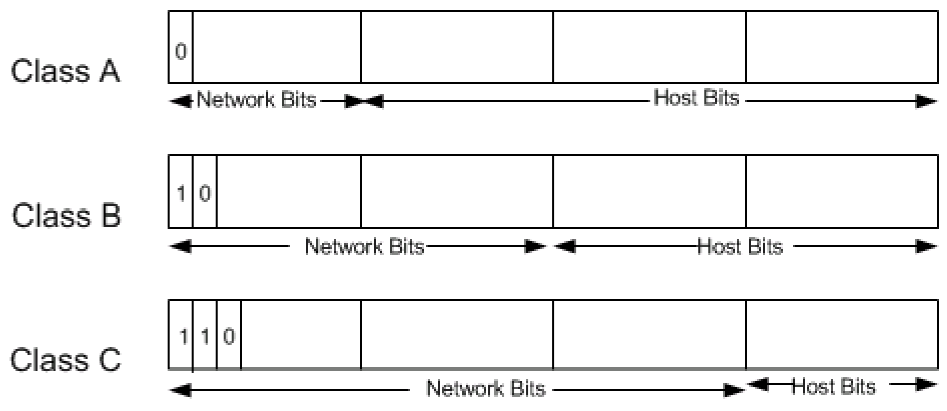
IP addresses are binary numbers, but they are usually stored in text files and displayed in human-readable notations, such as 172.16.254.1 (for IPv4), and 2001:db8:0:1234:0:567:8:1 (for IPv6). Two versions of the Internet Protocol (IP) are in use: IP Version 4 and IP Version 6. Each version defines an IP address differently. Because of its prevalence, the generic term *IP address* typically still refers to the addresses defined by IPv4. The gap in version sequence between IPv4 and IPv6 resulted from the assignment of number 5 to the experimental Internet Stream Protocol in 1979, which however was never referred to as IPv5.

IPV4:

The IP address space is divided into different network classes. The system defined five classes, Class A, B, C, D, and E. **The Classes A, B, and C had different bit lengths for the new network identification. The rest of an address was used as previously to identify a host within a network, which meant that each network class had a different capacity to address hosts. Class D was allocated for multicast addressing and Class E was reserved for future applications.**

Class A networks are intended mainly for use with a few very large networks, because they provide only 8 bits for the network address field. Class B networks allocate 16 bits, and Class C networks allocate 24 bits for the network address field. Class C networks only provide 8 bits for the host field, however, so the number of hosts per network may be a limiting factor. In all three cases, the left most bit(s) indicate the network class. IP addresses are written in dotted decimal format; for example, 34.0.0.1.

Address Formats for Class A, B, and C IP Networks



In IPv4 an address consists of 32 bits which limits the address space to 4294967296 (2^{32}) possible unique addresses. IPv4 reserves some addresses for special purposes such as private networks (~18 million addresses) or multicast addresses (~270 million addresses).

IPv4 addresses are canonically represented in dot-decimal notation, which consists of four decimal numbers, each ranging from 0 to 255, separated by dots, e.g., 172.16.254.1. Each part represents a group of 8 bits (octet) of the address. In some cases of technical writing, IPv4 addresses may be presented in various hexadecimal, octal, or binary representations.

IPV6:

An **Internet Protocol Version 6 address** (**IPv6 address**) is a numerical label that is used to identify a network interface of a computer or other network node participating in anIPv6-enabled computer network. IPv6 implements a new addressing system that allows for far more addresses to be assigned than with IPv4.

IPv6 does not implement interoperability features with IPv4, but essentially creates a parallel, independent network. Exchanging traffic between the two networks requires special translator gateways, but this is not generally required, since most computer operating systems and software implement both protocols for transparent access to both networks, either natively or using a tunneling protocol like 6to4, 6in4, or Teredo.

IPv6 is the successor to the Internet's first addressing infrastructure, Internet Protocol version 4 (IPv4). In contrast to IPv4, which defined an IP address as a 32-bit value, IPv6 addresses have a size of 128 bits. Therefore, IPv6 has a vastly enlarged address space compared to IPv4.

IPv6 addresses, as usually represented in human-readable notation, consist of eight groups of four hexadecimal digits separated by colons, for example;

2001:0db8:85a3:0042:0000:8a2e:0370:7334.

IPv6 addresses are classified by the primary addressing and routing methodologies common in networking: unicast addressing, anycast addressing, and multicast addressing.

- **A unicast address** identifies a single network interface. The Internet Protocol delivers packets sent to a unicast address to that specific interface.
- **An anycast address** is assigned to a group of interfaces, usually belonging to different nodes. A packet sent to an anycast address is delivered to just one of the member interfaces, typically the *nearest* host, according to the routing protocol’s definition of distance. Anycast addresses cannot be identified easily, they have the same format as unicast addresses, and differ only by their presence in the network at multiple points. Almost any unicast address can be employed as an anycast address.
- **A multicast address** is also used by multiple hosts, which acquire the multicast address destination by participating in the multicast distribution protocol among the network routers. A packet that is sent to a multicast address is delivered to all interfaces that have joined the corresponding multicast group.

An IPv6 address consists of 128 bits. Addresses are classified into various types for applications in the major addressing and routing methodologies: unicast, multicast, and anycast networking. In each of these, various address formats are recognized by logically dividing the 128 address bits into bit groups and establishing rules for associating the values of these bit groups with special addressing features.

Unicast and anycast address format

Unicast and anycast addresses are typically composed of two logical parts: a 64-bit network prefix used for routing, and a 64-bit interface identifier used to identify a host's network interface.

General unicast address format (routing prefix size varies)

bits	48 (or more)	16 (or fewer)	64
field	*routing prefix*	*subnet id*	*interface identifier*

The *network prefix* (the *routing prefix* combined with the *subnet id*) is contained in the most significant 64 bits of the address. The size of the routing prefix may vary; a larger prefix size means a smaller subnet id size. The bits of the *subnet id(entifier)* field are available to the network administrator to define subnets within the given network. The 64-bit *interface identifier* is either automatically generated from the interface's MAC address using the modified EUI-64 format, obtained from a DHCPv6 server, automatically established randomly, or assigned manually.

A link-local address is also based on the interface identifier, but uses a different format for the network prefix.

Link-local address format

bits	10	54	64
field	*prefix*	*zeroes*	*interface identifier*

The *prefix* field contains the binary value 1111111010. The 54 zeroes that follow make the total network prefix the same for all link-local addresses, rendering them non-routable.

Multicast address format

Multicast addresses are formed according to several specific formatting rules, depending on the application.

General multicast address format

bits	8	4	4	112
field	*prefix*	*flg*	*sc*	*group ID*

The *prefix* holds the binary value 11111111 for any multicast address.

Currently, 3 of the 4 flag bits in the *flg* field are defined; the most-significant flag bit is reserved for future use.

Multicast address flags

bit	flag	Meaning when 0	Meaning when 1
8	*reserved*	*reserved*	*reserved*
9	R (Rendezvous)	Rendezvous point not embedded	Rendezvous point embedded
10	P (Prefix)	Without prefix information	Address based on network prefix
11	T (Transient)	Well-known multicast address	Dynamically assigned multicast address

The 4-bit scope field (*sc*) is used to indicate where the address is valid and unique.

There are special multicast addresses, like Solicited Node.

Solicited-Node multicast address format

bits	8	4	4	79	9	24
field	*prefix*	*flg*	*sc*	*zeroes*	*ones*	*unicast address*

The *sc(ope)* field holds the binary value 0010 (link-local). Solicited-node multicast addresses are computed as a function of a node's unicast or anycast addresses. A solicited-node multicast address is created by copying the last 24 bits of a unicast or anycast address to the last 24 bits of the multicast address.

Unicast-prefix-based multicast address format

bits	8	4	4	4	4	8	64	32
field	*prefix*	*flg*	*sc*	*res*	*riid*	*plen*	*network prefix*	*group ID*

Link-scoped multicast addresses use a comparable format.

IPv4 and IPv6 Header Structure:

IPV4 Header Structure:

The IPv4 packet header consists of 14 fields, of which 13 are required. The 14th field is optional (red background in table) and aptly named: options. The fields in the header are packed with the most significant byte first (big endian), and for the diagram and discussion, the most significant bits are considered to come first (MSB 0 bit numbering).

The most significant bit is numbered 0, so the version field is actually found in the four most significant bits of the first byte, for example.

bit offset	0–3	4–7	8–13	14-15	16–18	19–31
0	Version	Internet Header Length	Differentiated Services Code Point	Explicit Congestion Notification	Total Length	
32	Identification				Flags	Fragment Offset
64	Time to Live		Protocol		Header checksum	
96	Source IP Address					
128	Destination IP Address					
160	Options (if Header Length > 5)					

Version : The first header field in an IP packet is the four-bit version field. For IPv4, this has a value of 4 (hence the name IPv4).

Internet Header Length (IHL): The second field (4 bits) is the Internet Header Length (IHL), which is the number of 32-bit words in the header. Since an IPv4 header may contain a variable number of options, this field specifies the size of the header (this also coincides with the offset to the data). The minimum value for this field is 5 (RFC 791), which is a length of 5×32 = 160 bits = 20 bytes. Being a 4-bit value, the maximum length is 15 words (15×32 bits) or 480 bits = 60 bytes.

Differentiated Services Code Point (DSCP): Originally defined as the Type of Service field, this field is now defined by RFC 2474 for Differentiated services (DiffServ). New technologies are emerging that require real-time data streaming and therefore make use of the DSCP field. An example is Voice over IP (VoIP), which is used for interactive data voice exchange.

Explicit Congestion Notification (ECN): This field is defined in RFC 3168 and allows end-to-end notification of network congestion without dropping packets. ECN is an optional feature that is only used when both endpoints support it and are willing to use it. It is only effective when supported by the underlying network.

Total Length: This 16-bit field defines the entire packet (fragment) size, including header and data, in bytes. The minimum-length packet is 20 bytes (20-byte header + 0 bytes data) and the maximum is 65,535 bytes — the maximum value of a 16-bit word. The largest datagram that any host is required to be able to reassemble is 576 bytes, but most modern hosts handle much larger packets. Sometimes subnetworks impose further restrictions on the packet size, in which case datagrams must be fragmented. Fragmentation is handled in either the host or router in IPv4.

Identification: This field is an identification field and is primarily used for uniquely identifying fragments of an original IP datagram. Some experimental work has suggested using the ID field for other purposes, such as for adding packet-tracing information to help trace datagrams with spoofed source addresses.

Flags; A three-bit field follows and is used to control or identify fragments. They are (in order, from high order to low order):

- bit 0: Reserved; must be zero.
- bit 1: Don't Fragment (DF)
- bit 2: More Fragments (MF)

If the DF flag is set, and fragmentation is required to route the packet, then the packet is dropped. This can be used when sending packets to a host that does not have sufficient resources to handle fragmentation. It can also be used for Path MTU Discovery, either automatically by the host IP software, or manually using diagnostic tools such as ping or traceroute.

For unfragmented packets, the MF flag is cleared. For fragmented packets, all fragments except the last have the MF flag set. The last fragment has a non-zero Fragment Offset field, differentiating it from an unfragmented packet.

Fragment Offset: The fragment offset field, measured in units of eight-byte blocks, is 13 bits long and specifies the offset of a particular fragment relative to the beginning of the original unfragmented IP datagram. The first fragment has an offset of zero. This allows a maximum offset of $(2^{13} – 1) \times 8 = 65,528$ bytes, which would exceed the maximum IP packet length of 65,535 bytes with the header length included $(65,528 + 20 = 65,548$ bytes).

Time To Live (TTL): An eight-bit time to live field helps prevent datagrams from persisting (e.g. going in circles) on an internet. This field limits a datagram's lifetime. It is specified in seconds, but time intervals less than 1 second are rounded up to 1. In practice, the field has become a hop count—when the datagram arrives at a router, the router decrements the TTL field by one. When the TTL field hits zero, the router discards the packet and typically sends a ICMP Time Exceeded message to the sender.

The program traceroute uses these ICMP Time Exceeded messages to print the routers used by packets to go from the source to the destination.

Protocol: This field defines the protocol used in the data portion of the IP datagram. The Internet Assigned Numbers Authority maintains a list of IP protocol numbers which was originally defined in RFC 790.

Header Checksum: The 16-bit checksum field is used for error-checking of the header. When a packet arrives at a router, the router calculates the checksum of the header and compares it to the checksum field. If the values do not match, the router discards the packet. Errors in the data field must be handled by the encapsulated protocol. Both UDP and TCP have checksum fields.

When a packet arrives at a router, the router decreases the TTL field. Consequently, the router must calculate a new checksum. RFC 1071 defines the checksum calculation:

The checksum field is the 16-bit one's complement of the one's complement sum of all 16-bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero.

For example, consider Hex 4500003044224000800600008c7c19acae241e2b (20 bytes IP header):

Step 1.

4500 + 0030 + 4422 + 4000 + 8006 + 0000 + 8c7c + 19ac + ae24 + 1e2b = 2BBCF (16-bit sum)

Step 2.

2 + BBCF = BBD1 = 1011101111010001 (1's complement 16-bit sum)

Step 3.

~BBD1 = 0100010000101110 = 442E (1's complement of 1's complement 16-bit sum)

To validate a header's checksum the same algorithm may be used - the checksum of a header which contains a correct checksum field is a word containing all zeros (value 0):

2BBCF + 442E = 2FFFD. 2 + FFFD = FFFF. the 1'S of FFFF = 0.

Source address: This field is the IPv4 address of the sender of the packet. Note that this address may be changed in transit by a network address translation device.

Destination address: This field is the IPv4 address of the receiver of the packet. As with the source address, this may be changed in transit by a network address translation device.

Options: The options field is not often used. Note that the value in the IHL field must include enough extra 32-bit words to hold all the options (plus any padding needed to ensure that the header contains an integral number of 32-bit words). The list of options may be terminated with an EOL (End of Options List, 0x00) option; this is only necessary if the end of the options would not otherwise coincide with the end of the header.

IPV6 Header Structure:

IPv6 specifies a new packet format, designed to minimize packet header processing by routers. Because the headers of IPv4 packets and IPv6 packets are significantly different, the two protocols are not interoperable. However, in most respects, IPv6 is a conservative extension of IPv4. Most transport and application-layer protocols need little or no change to operate over IPv6; exceptions are application protocols that embed internet-layer addresses, such as FTP and NTPv3, where the new address format may cause conflicts with existing protocol syntax.

The fixed header of an IPv6 packet consists of its first 40 octets (320 bits). It has the following format:

Fixed header format

0								1								2								3							
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Version				*Traffic Class*								*Flow Label*																			
Payload Length																*Next Header*								*Hop Limit*							
Source Address																															
Destination Address																															

Version **(4 bits):** The constant 6 (bit sequence 0110).

Traffic Class **(8 bits):** The bits of this field hold two values. The 6 most-significant bits are used for DSCP, which is used to classify packets. The remaining two bits are used for ECN; priority values subdivide into ranges: traffic where the source provides congestion control and non-congestion control traffic.

***Flow Label* (20 bits):** Originally created for giving real-time applications special service. Flow Label specifications and minimum requirements are described, and first uses of this field are emerging.

***Payload Length* (16 bits):** The size of the payload in octets, including any extension headers. The length is set to zero when a *Hop-by-Hop* extension header carries a Jumbo Payload option.

***Next Header* (8 bits):** Specifies the type of the next header. This field usually specifies the transport layer protocol used by a packet's payload. When extension headers are present in the packet this field indicates which extension header follows. The values are shared with those used for the IPv4 protocol field, as both fields have the same function

***Hop Limit* (8 bits):** Replaces the time to live field of IPv4. This value is decremented by one at each intermediate node visited by the packet. When the counter reaches 0 the packet is discarded.

***Source Address* (128 bits):** The IPv6 address of the sending node.

***Destination Address* (128 bits):** The IPv6 address of the destination node(s).

Internet RFCs:

In computer network engineering, a **Request for Comments** (**RFC**) is a memorandum published by the Internet Engineering Task Force (IETF) describing methods, behaviors, research, or innovations applicable to the working of the Internet and Internet-connected systems. It began in 1969 as a set of working notes about ARPAnet research and development.

Memos in the **Requests for Comments (RFC)** document series contain technical and organizational notes about the Internet. They cover many aspects of computer networking, including protocols, procedures, programs, and concepts, as well as meeting notes, opinions, and sometimes humor.

RFCs are numbered (roughly) consecutively, and these numbers provide a single unique label space for all RFCs. RFCs are published online through a number of repositories, and there is an online index of RFCs.

The most common meaning for the word *standard* on the Internet is probably 'current (i.e. non-obsoleted) RFC'. This isn't quite as rigorous a concept as it may sound. ***RFC***s are *an enumerated series of documents issued by the IETF*, varying greatly in their nature and status. Some of them are often called standards - this may apply even to RFCs which explicitly state that they do not define a standard of any kind! - but according to the official terminology, only a few of them have been designated as ***Internet standards***. An Internet standard has, in addition to an RFC number, an STD number, which does not change even if the RFC number is changed; for example, the IP protocol is defined by STD 5 which is currently RFC 791.

RFC Categories

Each RFC has a "category" or "status" designation. The possible categories are:

- STANDARD, DRAFT STANDARD, PROPOSED STANDARD

  These are *standards-track* documents, official specifications of the Internet protocol suite defined by the Internet Engineering Task Force (IETF) and its steering group the IESG.

- BEST CURRENT PRACTICE

  These are official guidelines and recommendations, but not standards, from the IETF.

- INFORMATIONAL, EXPERIMENTAL

  These non-standards documents may originate in the IETF or may be independent submissions.

- HISTORIC

  These are former standards that have been actively deprecated.