The diagram below illustrates four “routers” on the “Internet backbone” along with two companies that have “gateways” for their internal networks onto the Internet. Any message that is sent over these networks from one “host” computer to another host computer will be broken-up into some number of “packets” where each packet will be transmitted independently from the source host to the destination host. Each packet will contain a “header” and a “payload” where the header indicates where the packet came from and the destination of the packet.

By design, the Internet contains redundant paths such that there are usually multiple possible paths, or routes, for a packet to travel from its source to its destination. In the diagram below, as an example, a packet traveling from host computer #1 in company X to host computer #2 in company Y could travel from router A to router B and then to router C or it could travel from router A to router D and then to router C to reach the gateway into company Y.

Whenever a router receives a packet, it examines its “routing table” to determine which “path”, or paths (also known as “links”), from that router is/are appropriate for forwarding the packet to its destination. Each time a packet travels over a link, it is known as a “hop”.

The diagram below also illustrates that company Y has broken-up its internal network into two “subnets”. The networks within both company A and company B are likely to be local-area networks (LANs).
Every device (e.g., computer) connected to the Internet (including computers connected to the Internet through an ISP (Internet Service Provider)) must have an Internet address so that packets can be properly directed to/from that device. Each addressable device on a network (e.g., a computer connected to the Internet) is technically known as a ‘node’ on that network. Most nodes on the Internet use 32-bit addresses as defined by Internet Protocol version 4 (IPv4) where the 32-bit address represents a numerical value (not ASCII codes).

Since 32 bits can express 4G different values, 32-bit IP addresses allow for up to 4G nodes on the Internet. This appears to allow for a large number of nodes on the Internet – however, many of the larger companies/organizations that need access to the Internet have reserved relatively large ranges of addresses to allow for their future expansion. As such, there are not as many addresses available for future use as one might think. Internet Protocol version 6 (IPv6) allows for 128-bit addresses – but IPv6 is not yet in widespread use. IPv6 was designed so as to allow for IPv4 and IPv6 to coexist on the Internet at the same time.

When one thinks of web sites on the Internet one typically thinks in terms of the ‘name’ of each website (e.g., Yahoo.com) where these website names are technically known as “domain names”. Each of these domain names corresponds to a specific IP address. When you attempt to access a website by name, through your ISP, the ISP’s DNS (domain name system) server does “forward” translation of the domain name to an IP address (i.e., “name-to-number” translation) so that the packets containing your request can be sent onto the Internet. This translation is required since each packet’s “header” contains the destination’s IP address – not the destination’s domain name.

When a response from the website arrives at your ISP, the ISP’s DNS server does “reverse” translation (i.e., “number-to-name”) to convert the packet’s originating IP address to a website name (domain name). Your ISP’s DNS server must regularly update its translation tables from Top-Level Domain Root Servers (or simply, Root Servers) that are on the Internet.

Companies that allow customers to register “domain names” (e.g., MyPlace.com) for their use must be accredited with ICANN (Internet Corporation for Assigned Names and Numbers).

The 32-bit IP addresses of IPv4 are expressed as four ‘octets’ (an octet is 8 bits) where the octets are separated by periods (known as “dotted-quad notation”). The value of each octet is usually expressed in base ten (i.e., 0 – 255) (then known as “dotted-decimal notation”). An example IPv4 address would be:

172.57.165.212

Like most ‘addresses’, IPv4 addresses have the following overall characteristic: The high-order bits of the address determine the specific company/organization and the low-order bits of the address determine the specific node within the company/organization. The dividing line between the high-order bits of the address (that specify the company/organization) and the low-order bits of the address (that specify the specific node within the company/organization) is determined by how large the address range the company/organization reserved for itself. These address ranges are assigned to companies/organizations by ARIN (American Registry for Internet Numbers).

The high-order bits of the IP address that determine the specific company/organization are generally known as the “network ID”. The remaining lower-order bits then determine the specific node within the addressed company/organization and they are generally known as the “host ID”.
When companies/organizations want to acquire a range of IP addresses for the devices on their network, they usually purchase address ranges that are available in three distinct categories: Class A, Class B, and Class C (address ranges that are more “granular” than these three classes can now be purchased). These three classes are summarized below:

► A Class A address range allows for up to \(2^{24} - 2 = 16,777,214\) host IDs (i.e., 24 bits are used for the host ID portion of the IP address)

► A Class B address range allows for up to \(2^{16} - 2 = 65,534\) host IDs (i.e., 16 bits are used for the host ID portion of the IP address)

► A Class C address range allows for up to \(2^8 - 2 = 254\) host IDs (i.e., 8 bits are used for the host ID portion of the IP address)

Note: The “\(- 2\)” is due to two host ID addresses being reserved for special use. Specifically, if the host ID bits of the IP address are all ‘0’s, then this is considered the overall “network address” of the network and, by agreement, it should not address a specific host on that network. If the host ID bits of the IP address are all ‘1’s, then this is considered the address for sending a “broadcast” message to all the devices on that network.

The IP address space was initially divided-up for these three classes by assigning certain values for the first octet (most-significant octet) of the IP address to individual classes as follows:

► Class A addresses: First octet is any value from 1 to 127 (this allows for 127 Class A address ranges (or simply, 127 Class A addresses or 127 Class A network IDs))

Examples: 
- 1\(^{st}\) Class A address range: 1.0.0.0 to 1.255.255.255;
- 2\(^{nd}\) Class A address range: 2.0.0.0 to 2.255.255.255;
- 3\(^{rd}\) Class A address range: 3.0.0.0 to 3.255.255.255;

- ... 
- 127\(^{th}\) Class A address range: 127.0.0.0 to 127.255.255.255

(Notice that the 1\(^{st}\) octet stays the same value for any given Class A address range – which leaves 24 bits (the 2\(^{nd}\), 3\(^{rd}\), and 4\(^{th}\) octets) for the subnet/host ID address)

► Class B addresses: First octet is any value from 128 to 191 (and the second octet then also becomes part of the Network ID – this allows for \((191 - 128 + 1)\cdot256 = 16,384\) Class B addresses)

Examples: 
- 1\(^{st}\) Class B address range: 128.0.0.0 to 128.255.255.255;
- 2\(^{nd}\) Class B address range: 128.1.0.0 to 128.1.255.255;
- 3\(^{rd}\) Class B address range: 128.2.0.0 to 128.2.255.255;

- ... 
- 256\(^{th}\) Class B address range: 128.255.0.0 to 128.255.255.255;
- 257\(^{th}\) Class B address range: 129.0.0.0 to 129.0.255.255;
- 258\(^{th}\) Class B address range: 129.1.0.0 to 129.1.255.255;

- ... 
- 16,384\(^{th}\) Class B address range: 191.255.0.0 to 191.255.255.255

(Notice that the 1\(^{st}\) and 2\(^{nd}\) octets stay the same value for any given Class B address range – which leaves 16 bits (the 3\(^{rd}\) and 4\(^{th}\) octets) for the subnet/host ID address)
Class C addresses: First octet is any value from 192 to 223 (and the second and third octets then also become part of the Network ID — this allows for \((223 - 192 + 1) \times 256 \times 256 = 2,097,152\) Class C addresses)

Examples:
- 1st Class C address range: 192.0.0.0 to 192.0.0.255;
- 2nd Class C address range: 192.0.1.0 to 192.0.1.255;
- 3rd Class C address range: 192.0.2.0 to 192.0.2.255;
- \(\ldots\)
- 256th Class C address range: 192.255.0.0 to 192.255.255.255;
- 257th Class C address range: 192.1.0.0 to 192.1.0.255;
- 258th Class C address range: 192.1.1.0 to 192.1.1.255;
- \(\ldots\)
- 512th Class C address range: 192.255.0.0 to 192.255.255.255
- 513th Class C address range: 192.2.0.0 to 192.2.0.255
- \(\ldots\)
- Last Class C address range: 223.255.255.0 to 223.255.255.255

(Notice that the 1st, 2nd, and 3rd octets stay the same value for any given Class C address range — which leaves 8 bits (the 4th octet) for the subnet/host ID address)

The dividing line between the high-order address bits (the network ID) and the low-order address bits (the host ID) is commonly defined using a “subnet mask.” Subnet masks are important to routers since they allow a router to determine which portion of a packet’s destination IP address is addressing a specific company/organization and with this information the router can use its “routing table” (which contain routes to various companies/organizations) to best direct the packet onward.

The subnet mask consists of four octets organized like an IPv4 address where the value of each octet is expressed in base ten. However, to create the subnet mask, or to fully understand a specific subnet mask, the octets must be expressed in binary. The individual bits in the subnet mask have the following characteristic:

Each bit in the IPv4 address for a particular company/organization that must be a specific value (a ‘0’ or ‘1’) to access that company/organization will have a bit value of ‘1’ in the corresponding bit position in the subnet mask (i.e., all the high-order address bits that specify the company/organization will have bit values of ‘1’ in the corresponding bit positions in the subnet mask)

And,

Each bit in the IPv4 address for a particular company/organization that is used to address a specific node within that company/organization will have a bit value of ‘0’ in the corresponding bit position in the subnet mask (i.e., the low-order address bits that specify a particular node within the company/organization will have bit values of ‘0’ in the corresponding bit positions in the subnet mask)

The standard (default) subnet masks for the three classes of IP address ranges are as follows:

- **Class A default subnet mask:** 255.0.0.0
  (in binary this subnet mask is: 11111111.00000000.00000000.00000000 which indicates that the high-order 8 bits of the IP address are used to address the specific company/organization and the low-order 24 bits of the IP address are used to designate a specific node within that company/organization)
► **Class B default subnet mask:** 255.255.0.0  
(in binary this subnet mask is: 11111111.11111111.00000000.00000000 which indicates that the high-order 16 bits of the IP address are used to address the specific company/organization and the low-order 16 bits of the IP address are used to designate a specific node within that company/organization)

► **Class C default subnet mask:** 255.255.255.0  
(in binary this subnet mask is: 11111111.11111111.11111111.00000000 which indicates that the high-order 24 bits of the IP address are used to address the specific company/organization and the low-order 8 bits of the IP address are used to designate a specific node within that company/organization)

As an example, assume you are told the IP address for a specific node within a particular company/organization is:

**172.57.165.212**

Also assume you are told that the subnet mask for this particular company/organization is:

**255.255.0.0**

From the subnet mask you then realize that this particular company/organization has a Class B IP address where the beginning IP address of the company/organization is **172.57.0.0** and the particular host ID within the company/organization is the **165.212** portion of the IP address.

When a subnet mask for a particular company/organization is one of the above listed "default" subnet masks, it is termed a “**classful**” subnet mask (or classful IP address).

Companies and organizations that have a moderate to large number of nodes within their network tend to create “sub-networks” such that their computer network is divided-up into multiple sub-networks where there are “routers” for communications between these various sub-networks (which are known simply as “**subnets**”). The various hosts are then distributed among the subnets.

Subnets create an additional level of hierarchy within the overall network such that IP addressing must, in a subnet environment, be able to designate the following:

► Which company/organization is being addressed  
► Which subnet within that company/organization is being addressed  
► Which host within that subnet of the company/organization is being addressed

When companies/organizations create subnets within their network (i.e., **subnetting**), they use some number of the high-order bits of the host-ID portion of their IP address to designate a particular subnet. As an example, assume the following:

A company/organization has a Class C IP address (which, as a ‘classful' IP address allows for 256 host IDs (in reality, 256 – 2 = 254) due to the 8 bits available for the host ID). However, the company/organization decides that it wants to allow for up to 8 subnets. As such, three of the high-order host-ID IP-address bits must now be devoted to addressing a particular subnet within the company/organization. This leaves 8 – 3 = 5 bits for the actual host ID resulting in 2^5 = **32 host IDs on each of 8 subnets** (for a total of 32•8 = 256 host IDs among all 8 subnets).
The subnet mask for this situation should include the subnet addressing bits as part of the portion of the IP address that indicates which bits address the particular company/organization. As such, the subnet mask should be constructed (initially in binary) as follows:

\[
\begin{array}{cccccccccccccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & = & 255.255.255.224 \\
\end{array}
\]

24 high-order address bits to address the company/organization

5 bits to address the host

Note that 11100000 (binary) = 128 + 64 + 32 = 224 (base 10)

From the binary representation of the subnet mask we can see that the high-order 24 bits of the IP address specify this particular company/organization and that the low-order 8 bits of the IP address for this particular company/organization specify a particular subnet and host within that subnet. This type of subnet mask is termed a “classless” subnet mask (and not “classful”).

Note that there are only certain values that can be used as a subnet mask value within a given octet. These values are: 255, 254, 252, 248, 240, 224, 192, 128, and 0. These specific values are a result of the fact that the 8 bits of an octet, when used as a subnet mask, must either be all ‘1’s, all ‘0’s, or have some number of leading ‘1’s followed by trailing ‘0’s. The binary/decimal values are given below:

- 11111111 (binary) = 255 (base 10)
- 11111110 (binary) = 254 (base 10)
- 11111100 (binary) = 252 (base 10)
- 11111000 (binary) = 248 (base 10)
- 11100000 (binary) = 240 (base 10)
- 11000000 (binary) = 224 (base 10)
- 10000000 (binary) = 192 (base 10)
- 00000000 (binary) = 0 (base 10)

Another method for conveying the same information about which bits in an IP address are for addressing the company/organization/subnet (i.e., network ID and subnet) and which bits address a particular host within that organization is known as “CIDR-slash” notation where CIDR means Classless Inter-Domain Routing. Below is an example of CIDR-slash notation:

184.13.152.0 / 22  The 22 signifies that the high-order 22 bits of the IP address are the network ID/subnet address (and, therefore, the low-order 10 bits (i.e., 32 – 22 = 10) address the host)

Note that the dividing line between the network ID/subnet addressing and the host ID addressing is in the third octet (i.e., the host ID addressing is the low-order 10 bits of the IP address). To illustrate this, the IP address will be re-expressed with the third octet being in binary:

<table>
<thead>
<tr>
<th>Network ID/ Subnet Address (22 bits)</th>
<th>0 0 0</th>
<th>Host ID (10 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>184.13.1001100 (binary) = 128 + 16 + 8 = 152 (base 10)</td>
<td>0 0 0 0</td>
<td>10011000 (binary) = 128 + 64 + 32 = 224 (base 10)</td>
</tr>
</tbody>
</table>
**Packets:**

Most all computer networks, including the Internet, divide a message into individual “packets” and then transmit the individual packets, one-by-one through the network. In fact, the individual packets that comprise a message may follow different paths (often known as “links”) through the network to their destination when there are multiple links between the source of the packet and the packet’s destination.

When a network “router” receives a packet, the router then looks at its “routing table” (properly known as a Routing Information Table (RIT)) to determine all the appropriate links from that router for forwarding the packet onward to, or at least towards, its destination. Each of these links, to and from a router, is connected to the router through a “port” where any given port on a router will be connected, though a link, to another device, such as another router, a switch, a host, etc. The router then transmits the packet onto the least busy port among the acceptable ports.

Any given network (or subnet within a network) can transport a packet only if the size of the packet does not exceed that network’s **MTU** (maximum transmission unit) size (typically a value between 128 bytes and 10,000 bytes). When a router receives a packet and wants to forward that packet onto a particular port (based upon its routing table), it must first verify that the packet size does not exceed the MTU of that port’s network link. If the packet size exceeds the MTU of that link, then the router must “fragment” the packet into two or more packets such that none of the new packets exceeds the MTU of that link.

There are two main ‘protocols’ for sending packets from an application program on one node of the Internet to an application program on another node of the Internet. These two protocols are known as **Transmission Control Protocol** (TCP) and **User Datagram Protocol** (UDP). Note that IP (Internet Protocol) only deals with transferring packets from one node on the Internet to another node on the Internet – IP does not deal with “talking” directly to an application program. The Internet itself is a “connectionless” network (i.e., packets from one node to another node do not necessarily follow the same links from the source node to the destination node and, therefore, the packets may arrive out-of-order).

However, most application programs, that communicate over the Internet, expect the data that is sent over the Internet to arrive at the destination application program in the order that the data was sent from the source application. **TCP** is a protocol that, in effect, establishes a “connection” (technically known as a “socket”) from an application program on one node of the Internet to an application program on another node of the Internet. The TCP layer deals with re-assembling the received packets into their original order and requesting the re-transmission of any missing or damaged packets. TCP implements the “transport layer” in the OSI seven-layer model for network communications. For each TCP packet transmitted over a network, the final destination of the packet must send an “acknowledgment” packet back to the initial source of the packet so that the source knows that the packet has been properly received.

A few application programs, on the other hand, are receiving information on a “real-time” basis (e.g., VoIP (Voice over Internet Protocol), streaming video, and on-line Internet gaming) and cannot tolerate the time delay involved in re-ordering any packets that arrive out-of-order or requesting the re-transmission of any missing or damaged packets. These types of application programs use the User Datagram Protocol (UDP) which discards packets that have been received significantly out-of-order as well as damaged packets. UDP also implements the “transport layer” in the OSI seven-layer model for network communications, but for application programs that can/must tolerate missing or damaged data. For UDP-type packets, the final destination for any packet does not send an “acknowledgment” packet back to the initial source of the packet (unlike TCP-type packets).
Both TCP and UDP are often referred to as TCP/IP (TCP over Internet Protocol) and UDP/IP.

Each packet consists of a “header” and a “payload”. The payload contains the data for a portion of an overall message that is to be sent from a source node on the network to a destination node on the network (any given message may require numerous packets for communicating the entire message).

Although the above statement is, in general, true, the reality is more complex. When any given node on the network is communicating over the network (e.g., a computer being used for Internet browsing), there may be multiple application programs on that computer that are being executed at the same time (i.e., multitasking) where each application wishes to use the network (e.g., several windows of Internet Explorer where each one is viewing a different website). Whenever that computer sends a request to a website for information, the information that comes back must be directed to the appropriate application window. This requires that the network communications from this computer include not only a “return IP address” for the information requested, but also a “return application window” identification so that when the information is returned, the computer knows which window to place the information into.

The ISO (International Standards Organization) developed the OSI (Open Systems Interconnection) Seven-Layer Model to break-up the network communications activity into seven distinct “layers” so as to create distinct solutions to the situation described above as well as additional situations. As such, the packetizing process might appear to be somewhat complex.

The process of preparing a message for transmission over a network using TCP/IP (or UDP/IP) is as follows:

► When an application program wants to send a message over the network, the message is given to the networking software and then...
► The “Layer 4” portion of the networking software (the “transport layer”) breaks the message up into a sequence of multi-byte ‘chunks’ (each ‘chunk’ will become a packet)
► The “Layer 4” portion of the networking software will create a TCP packet which contains a header and a payload (the TCP packet’s payload will be the first ‘chunk’ of the message) (the header of the TCP packet identifies the source application program, the destination application program, the packet sequence identifier, and other information)
► The “Layer 4” portion of the networking software then hands the TCP packet just created to the “Layer 3” portion of the networking software (the “network layer”) which then creates an IP packet, which contains a header and a payload (the IP packet’s payload will be the entire TCP packet (header and payload) just created) (the header for the IP packet identifies the source node’s IP address, the destination node’s IP address, the packet identification number, and other information)
► The “Layer 3” portion of the networking software then hands the IP packet just created to the “Layer 2” portion of the networking software (the “data-link layer”) which then creates a packet for transmission (e.g., an Ethernet packet (or “frame”) for a LAN), which contains a header and a payload (the payload will be the entire IP packet (header and payload) just created) (the header for the data-link packet identifies the source node’s MAC (medium access control) address (also known as a NIC (network interface card) address) and the destination node’s MAC address where these MAC addresses must be appropriate for the source and destination of each individual hop that the packet travels (the source and destination MAC addresses must be updated by each router that handles the packet))
► The “Layer 2” portion of the networking software then hands the data-link packet just created to the “Layer 1” portion of the networking software/hardware (the “physical layer”) and the packet is physically transmitted
The above process is repeated for each ‘chunk’ of the message until the entire message is transmitted.

The above process can be summarized as follows: A TCP packet is "encapsulated" within an IP packet where the IP packet is then encapsulated within a data-link packet for actual transmission on the network’s physical media.

The concept of **encapsulating** a TCP packet within an IP packet is illustrated below:

![IP Packet Diagram](image)

An IP packet’s payload can carry a TCP packet (or a UDP or ICMP packet).

The concept of **encapsulating** an IP packet within an Ethernet packet is illustrated below:

![Ethernet Packet Diagram](image)

An Ethernet packet’s payload can carry an IP packet (which can carry a TCP packet).

The header for an **Internet Protocol (IP) packet** (which is considered a “Layer 3 header”) contains the following information:

- Internet Protocol version number for the packet (‘4’ for IPv4 or ‘6’ for IPv6)
- IP address of the packet’s source
- IP address of the packet’s destination
- Header length (number of 32-bit words (usually 5))
- Type-of-Service (TOS) (desired QoS (Quality of Service) for the packet (i.e., how routers should ‘queue’ (i.e., prioritize) this packet))
- Size of the datagram (header plus payload, in bytes)
- Packet identification number (for re-assembly of packets, in order, at the destination)
- Packet-fragmentation flags (2 bits – “Don't fragment” (or “Fragmenting is okay”) flag (set by the source of packet) and “More fragments” flag (set by any router that fragments the packet and is used to re-assemble fragments at the destination for a fragmented packet))
- Fragmentation offset (if a packet is fragmented, this field indicates where this fragment fits into the original packet) (if a packet is fragmented into multiple packets by a router, all the fragments (packets) will have the same “Packet identification number” as the original packet and the “Fragmentation offset” field will allow the destination to put the fragments into proper order – and the “More fragments” flag will allow the destination to know when it has received all of the fragments (i.e., the final fragment packet will have the “More fragments” flag false))
- Time-to-Live (TTL) (maximum number of router “hops” allowed for the packet – each time the packet is handled by a router, the router decrements this value by 1 and if the value is then equal to zero, the router discards the packet and then sends an ICMP-type packet (see below) to the original source of the packet to inform the source that the packet was discarded (it is assumed that if a large number of router hops has occurred then the packet can’t find its destination and is going in loops))
- Type of protocol, known as SAP (Service Access Point), which indicates whether the packet protocol is:
  1. TCP (Transmission Control Protocol) (used for most Internet communications – TCP creates a “connection” whereby the packets of a message are ultimately presented to the receiving application program “in order” with no missing or incorrect data)
  2. UDP (User Datagram Protocol) (used for VoIP (Voice over Internet Protocol), streaming video, on-line Internet gaming, and some router-to-router communications – UDP is a “connectionless” “real time” communications protocol whereby the packets of a message, that are presented to an application program may have missing data)
  3. ICMP (Internet Control Message Protocol) (used for reporting back to the source that there was a problem delivering a packet to its destination as well as for other uses)
- Checksum for the header (used to detect any bit errors in the header)
- Options field (an optional field, of variable length, can be added, if needed (not normally used))

The IP header is 20 bytes in length (assuming the Options field is absent). The Options field can lengthen the header to a maximum of 60 bytes.

The header for a Transmission Control Protocol (TCP) packet (which is considered a “Layer 4 header”) specifies a number of items including:

- The specific application program (known as a “port” and expressed as a numerical value) that is sending the packet
- The specific application program (“port”) for which the packet is destined at the destination IP address (which is specified in the IP header)
- A packet “sequence number” (for re-assembling the packets, in order, at the destination)
- A “window size” (the size of the buffer used for receiving packets – this value is set by the receiving node and the transmitting node sees the value when the receiver acknowledges packet receipt (this implements “flow control” whereby the destination can control the rate at which packets arrive for the source))
- A checksum on the entire packet (including the payload)

Unlike TCP, UDP does not create a “connection” (i.e., a guarantee that the packets will be passed to the application program in order) nor does it guarantee the delivery of all the data or even the correctness of the data delivered. Accordingly, a UDP header has fewer fields as compared to a TCP header.
An IP packet can contain, within its payload, a TCP-type packet, a UDP-type packet, or an ICMP-type packet (Internet Control Message Protocol). The following is a partial list of the circumstances under which a router, gateway, or host would send an ICMP-type packet:

- When a packet is received by a router, or gateway, or host and it is then determined that the packet cannot “reach” its intended subnet, or host, or application program, then the router, gateway, or host sends an ICMP-type packet back to the original source of the packet indicating that the subnet, host, or application program was “unreachable”

- When a packet is received by a router, if the TTL (Time-to-Live) value within the header of the IP packet becomes a value of 0 after being decremented by the router, then the router discards the packet and sends an ICMP-type packet back to the original source of the packet indicating that the packet was discarded due to the TTL value of the packet being 0 (described as “time-to-live exceeded in transit”)

- If an ICMP packet is received by the router/gateway/host to which it is addressed – and the ICMP packet indicates a request for a ‘ping’ (i.e., “Echo request” or, in effect, “Are you there?”), the router/gateway/host will send an ICMP-type packet back to the original source indicating a ‘ping’ (i.e., “Echo reply” or, in effect, “Yes, I am here”)

The gateways for some companies/organizations are programmed to not respond to unsolicited ICMP-type packets, and, therefore, will not answer ‘ping’ requests

- If a packet is received by a router and the packet size exceeds the MTU (Maximum Transmission Unit) size of the path onto which the router wants to forward the packet and the “Don’t Fragment” flag is ‘true’ in the IP header of the packet, then the router discards the packet and sends an ICMP-type packet back to the original source of the packet indicating that the packet was discarded due to a need to fragment the packet but with the packet’s “Don’t fragment” flag being ‘true’, fragmentation is prevented (i.e., “packet exceeds MTU and fragmentation prevented”)

There are several other types of ICMP-type packet communications, but the above examples are some of the more commonly used types of ICMP communications.

Whenever a router/gateway/host sends an ICMP packet back to some source, the ICMP packet will contain the IP-packet header and a portion of the payload of the packet it received that is prompting the need to send an ICMP packet back to the source. As an example, assume the TTL becomes 0 for a certain packet while in transit through the network. The router/gateway that sends the ICMP packet back to the source will include the IP header and a portion of the payload (specifically the TCP or UDP header) so that the source of the packet can identify which of the packets it transmitted failed to arrive at its destination.

The last example of ICMP packets given above (“packet exceeds MTU and fragmentation prevented”) is worthy of some additional explanation. To minimize the amount of overhead when sending large messages, each packet should be as large as possible so as to minimize the number of packets to be transmitted (each packet contains a header which is considered overhead). However, if the packets are sufficiently large so as to require fragmentation while in transit to their destination, there will be time lost due to the fragmentation process (and reassembly process). Furthermore, if any of the fragments are lost or damaged while in transit, the entire original packet must be retransmitted (which will again be fragmented at the point in transit where the packet size is too large).
As such, the ideal situation is to set the packet size to be equal to the smallest MTU size among all the hops necessary to transmit the packet from its source to its destination. The process to determine this value is termed “path-MTU discovery”. The source of the packets knows the MTU of the immediate path from itself (by inquiring the Layer 1 networking software). The source of the packets also knows the MTU of the immediate path connected to the destination host for the packets – it knows this based on the “handshake” communications that occur when a TCP connection is initially established and the destination host “announces” its MSS (Maximum Segment Size, i.e., maximum amount of data desired per packet), which is usually the MTU of the network connection to the destination host.

Knowing the MTU of the source host’s network connection and the MTU (MSS) of the destination host’s network connection, the source host sends a packet to the destination host where the packet size is equal to the smaller of the two MTU values and the “Don’t fragment” flag is ‘true’. If no ICMP packet comes back to the source indicating that some router along the path had to discard the packet, then subsequent communications will continue using this packet size. But if an ICMP packet does come back indicating the packet was discarded by some router due to a need to fragment the packet (but fragmentation is not allowed), then the source will try again using a smaller packet size. At some point in this trial-and-error process, the source will accomplish “path-MTU discovery” and all subsequent communications to this destination host will be performed using this packet size.

Note that, in general, packets can follow different paths through large networks such as the Internet (especially at noticeably different points in time) and, as such, the path-MTU between two hosts on the Internet can be different at different points in time.

**Packet Routing:**

Earlier in this document it was noted that when a router receives a packet, the router looks at the destination IP address for the packet and then looks at its routing table to determine the acceptable path(s) for forwarding the packet onward to its destination. Note that a router actually receives a data-link packet – which contains an IP packet as its payload. Therefore, a router must ‘dig into’ each data-link packet it receives so as to obtain the layer-3 header information (in the encapsulated IP packet) which contains the destination IP address.

It has been noted that data-link packets are transmitted at the physical layer and that each data-link packet contains, in its header, the source MAC address and destination MAC address. A MAC address is a unique 48-bit address that is hardwired into each NIC (Network Interface Card – the electronics that creates a connection from a host to a network). One might assume that the source MAC address in a data-link packet is the MAC address of the NIC that is originating the message and that the destination MAC address is the NIC address for the final destination of the packet. However, if this were true, then the source and destination MAC addresses would be performing the same function as the source and destination IP addresses (in the encapsulated IP header) – which they are not. Instead, the source and destination MAC addresses, in a data-link packet, must be the appropriate source and destination addresses for each individual “hop” of the packet.

As a packet travels from its source to its destination, it will generally travel through one or more routers. Each time a router receives a data-link packet, it “re-writes” the source MAC address in the packet to correspond to the MAC address of the port onto which it is going to forward the packet. The router also re-writes the destination MAC address to correspond to the MAC address of the node at the other end of the path onto which the packet is to be forwarded. As such, the MAC addresses in a data-link packet, at any particular point in time, correspond to the source and destination for that particular hop of the packet.
After re-writing the source and destination MAC addresses in preparation for the next hop, the router 'digs' into the payload of the data-link packet to access the encapsulated IP packet where it decrements the TTL (Time-to-Live) value by 1 (if the TTL = 0 then the packet is discarded and the router sends an ICMP-type packet back to the original source notifying it that the packet was discarded). If the TTL ≠ 0 then the router re-calculates and re-writes the checksum value for the encapsulated IP-packet header (due to the TTL value changing). The packet is then transmitted onto the desired path for its next hop – assuming the packet does not exceed the MTU (maximum transmission unit) size for that path.

If the packet exceeds the MTU size for the desired path, then the packet must be fragmented. Note that data-link packets (which are at the physical level) cannot be directly fragmented – only IP packets can be fragmented. As such, if a data-link packet exceeds the MTU of a path a router wishes to use, the router must take the IP packet out of the data-link packet, then fragment the IP packet into multiple IP packets, and, finally, create a data-link packet for each of the IP-packet fragments just created.

The diagram below illustrates a data-link packet (e.g., an Ethernet packet). Note that this packet illustrates a TCP (or UDP) packet (layer 4) encapsulated within an IP packet (layer 3) which is then encapsulated within the data-link packet (layer 2). Not all of the header information is shown for the TCP (or UDP), IP, or data-link packets – only the more important header information needed for packet processing by a router is shown.

The diagram on the next page uses the above representation for a data-link packet to illustrate a router receiving a packet, then processing the packet, and, if appropriate, forwarding the packet onward towards its final destination.
Key Points in Diagram:

1. The diagram above shows a packet traveling on a link to Router A. Then it shows the same packet within Router A where it is processed. Finally, it shows the packet traveling from Router A on its next hop (or being discarded).

2. The diagram does not specifically illustrate the situation where a packet must be fragmented for the next hop (in which case there would be two or more packets exiting Router A for each packet arriving into Router A).

3. Note that the packets illustrated do not show all of the header information for layer 2, 3, or 4.

Router’s Packet-Processing Algorithm:

1. **Step 1:** Decrement the packet’s TTL value (in the header of the encapsulated IP packet (i.e., layer 3 header)) by 1 and then:
   - **IF** TTL = 0 **THEN** discard the packet and send an ICMP packet to the source of the packet (the IP source address is in the layer 3 header) indicating the packet was discarded and processing of this packet is finished **ELSE** go to step 2

2. **Step 2:** Re-compute and then update the checksum value in the layer 3 header of the packet (due to the change in the TTL value)

3. **Step 3:** Using the Routing Information Table (RIT), the router determines the best port to use for forwarding the packet onward to its destination

4. **Step 4:** IF Packet size ≤ MTU of path connected to port for forwarding the packet **THEN** go to step 6 (bypass step 5)

5. **Step 5:** IF the “Don’t fragment” flag (in the layer 3 header) is ‘true’ **THEN** discard the packet and send an ICMP packet to the source of the packet (the IP source address is in the layer 3 header) indicating the packet was discarded and processing of this packet is finished **ELSE** (the “Don’t fragment” flag (in the layer 3 header) is ‘false’) fragment the encapsulated IP packet into multiple IP packets (and create a data-link packet for each fragment) such that the Packet size ≤ MTU of path connected to port for forwarding the packet

6. **Step 6:** The router changes the Source MAC address and Destination MAC address values in the data-link packet (layer 2 header) (or packets, if fragmentation occurred) from the values appropriate for the packet’s hop to this router to the values appropriate for the next hop of the packet (e.g., the received packet shown in the example above has a Destination MAC address of 47 which needs to be changed to a value of 97 (the MAC address of the port on the next router that is to receive this packet) and the Source MAC address needs to be changed from 12 to 49 for the next hop)

7. **Step 7:** The packet (or packets, if fragmentation occurred) is/are transmitted onward for the next hop

Key to Abbreviations:

**Router:**
- **RIT:** Routing Information Table
- **NIC:** Network Interface Card
- **MAC:** MAC Address
- **MTU:** Maximum Transmission Unit size

**Packet:**
- **S MAC:** Source MAC Address
- **D MAC:** Destination MAC Address
- **S IP:** Source IP Address
- **D IP:** Destination IP Address
- **TTL:** Time-to-Live value
- **CS:** Checksum value
- **S P:** Source Port Address
- **D P:** Destination Port Address
Public vs. Private IP Addresses:

In the previous discussion the IP address of any given node in the network becomes part of the header information of any IP packet sent from that node onto the network, including IP packets destined for the Internet through the company’s/organization’s gateway.

However, it is possible to assign “private” IP addresses within a company’s/organization’s network and then have the company’s/organization’s gateway change the source IP address (in an IP packet) to a ‘real’ IP address whenever an internal host wants to send a packet to a destination outside the company/organization (e.g., the Internet). When a response to a message is received by the gateway, the gateway will then change the real IP address in the received IP packet to the private IP address and then forward the packet to the internal host. The only direct benefit of doing this is to ‘hide’ the true IP address of internal hosts within the company/organization so as to help prevent outside computers from directly initiating transfers to/from an internal host.

In this situation the gateway must maintain a translation table from ‘real’ IP addresses (i.e., public IP addresses) to the corresponding private IP addresses that are appropriate within the company/organization. This translation process is, in general, called NAT (Network Address Translation).

It is actually possible for a company/organization to have fewer public IP addresses than private IP addresses and yet allow all the internal hosts to operate simultaneously on the Internet. In fact, it is possible for a company/organization to operate using just one public IP address and yet support a large number of internal hosts operating simultaneously on the Internet (approximately 64k hosts are possible). When the term NAT is used, it generally refers to this specific situation.

To perform this feat, a ‘trick’ is used. TCP packets contain a source port address and a destination port address. These port addresses are numerical values that identify the specific application program on the source computer that wishes to communicate to a specific application program on a destination computer. If there are multiple copies of the same application program executing on the source host (e.g., multiple copies of Internet Explorer), there will be a unique source port address for each ‘window’ of the application program (so as to be able to direct returned information (packets from websites) to the appropriate window of the application).

The TCP packet header allows for 16 bits to identify the source port address as well as 16 bits to identify the destination port address. This means that TCP packets are able to distinguish up to 65,536 different application programs (simultaneously accessing the network) on a given host. It is unlikely that any given host would have more than ≈10 application programs accessing the network simultaneously. As such, some of the bits in the TCP’s source port address can really be used as, in effect, a host address. For this scheme to work, all of the hosts, routers, and the gateway of the network must be aware that the TCP header is really being used to identify individual hosts as well as application programs within each host that are accessing the network.