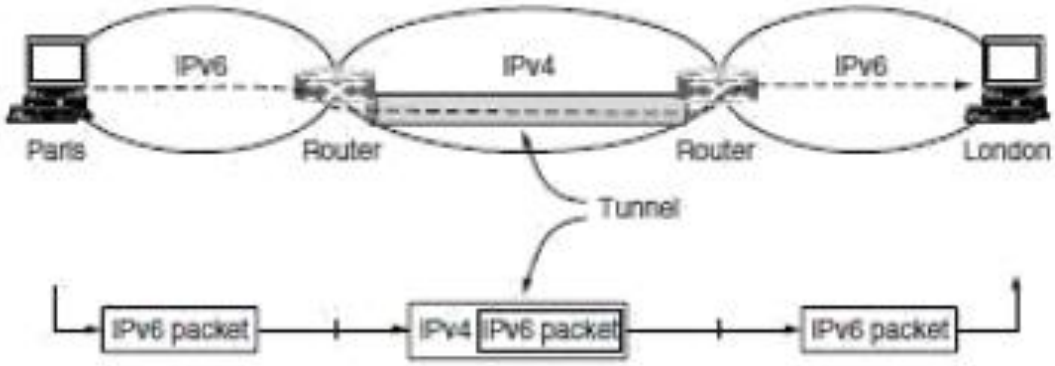
UNIT – IV
Internetworking

Until now, we have implicitly assumed that there is a single homogeneous network, with each machine using the same protocol in each layer. Unfortunately, this assumption is wildly optimistic. Many different networks exist, including PANs, LANs, MANs, and WANs. We have described Ethernet, Internet over cable, the fixed and mobile telephone networks, 802.11, 802.16, and more. Numerous protocols are in widespread use across these networks in every layer. In the following sections, we will take a careful look at the issues that arise when two or more networks are connected to form an **internetwork**, or more simply an **internet**.

It would be much simpler to join networks together if everyone used a single networking technology, and it is often the case that there is a dominant kind of network, such as Ethernet. Some pundits speculate that the multiplicity of technologies will go away as soon as everyone realizes how wonderful [fill in your favorite network] is. Do not count on it. History shows this to be wishful thinking. Different kinds of networks grapple with different problems, so, for example, Ethernet and satellite networks are always likely to differ. Reusing existing systems, such as running data networks on top of cable, the telephone network, and power lines, adds constraints that cause the features of the networks to diverge. Heterogeneity is here to stay. If there will always be different networks, it would be simpler if we did not need to interconnect them. This also is unlikely. Bob Metcalfe postulated that the value of a network with N nodes is the number of connections that may be made between the nodes, or $N2$ (Gilder, 1993). This means that large networks are much more valuable than small networks because they allow many more connections, so there always will be an incentive to combine smaller networks. The Internet is the prime example of this interconnection. (We will write Internet with a capital ‘‘I’’ to distinguish it from other internets, or connected networks.)The purpose of joining all these networks is to allow users on any of them to communicate with users on all the other ones. When you pay an ISP for Internet service, you may be charged depending on the bandwidth of your line, but what you are really paying for is the ability to exchange packets with any other host that is also connected to the Internet. After all, the Internet would not be very popular if you could only send packets to other hosts in the same city. Since networks often differ in important ways, getting packets from one network to another is not always so easy. We must address problems of heterogeneity, and also problems of scale as the resulting internet grows very large. We will begin by looking at how networks can differ to see what we are up against. Then we shall see the approach used so successfully by IP (Internet Protocol), the network layer protocol of the Internet, including techniques for tunneling through networks, routing in internetworks, and packet fragmentation.

Tunneling

Handling the general case of making two different networks interwork is exceedingly difficult. However, there is a common special case that is manageable even for different network protocols. This case is where the source and destination hosts are on the same type of network, but there is a different network in between. As an example, think of an international bank with an IPv6 network in Paris, an IPv6 network in London and connectivity between the offices via the IPv4 Internet. This situation is shown in Fig.



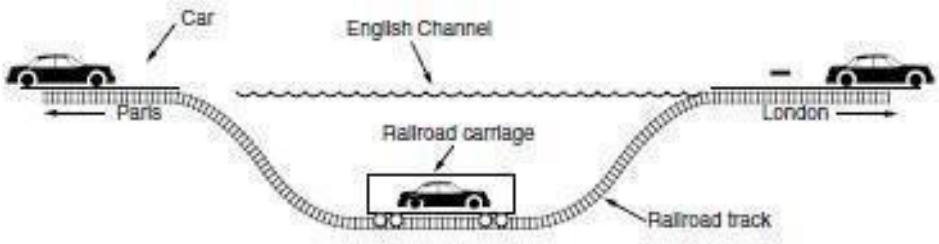
Tunneling a packet from Paris to London.

The solution to this problem is a technique called **tunneling**. To send an IP packet to a host in the London office, a host in the Paris office constructs the packet containing an IPv6 address in London, and sends it to the multiprotocol router that connects the Paris IPv6 network to the IPv4 Internet. When this router gets the IPv6 packet, it encapsulates the packet with an IPv4 header addressed to the IPv4 side of the multiprotocol router that connects to the London IPv6 network.

That is, the router puts a (IPv6) packet inside a (IPv4) packet. When this wrapped packet arrives, the London router removes the original IPv6 packet and sends it onward to the destination host.

The path through the IPv4 Internet can be seen as a big tunnel extending from one multiprotocol router to the other. The IPv6 packet just travels from one end of the tunnel to the other, snug in its nice box. It does not have to worry about dealing with IPv4 at all. Neither do the hosts in Paris or London. Only the multiprotocol routers have to understand both IPv4 and IPv6 packets. In effect, the entire trip from one multiprotocol router to the other is like a hop over a single link.

An analogy may make tunneling clearer. Consider a person driving her car from Paris to London. Within France, the car moves under its own power, but when it hits the English Channel, it is loaded onto a high-speed train and transported to England through the Chunnel (cars are not permitted to drive through the Chunnel). Effectively, the car is being carried as freight, as depicted in Fig. At the far end, the car is let loose on the English roads and once again continues to move under its own power. Tunneling of packets through a foreign network works the same way. Tunneling is widely used to connect isolated hosts and networks using other networks. The network that results is called an **overlay** since it has effectively been overlaid on the base network. Deployment of a network protocol with a new feature is a common reason, as our ‘‘IPv6 over IPv4’’ example shows. The disadvantage of tunneling is that none of the hosts on the network that are tunneled over can be reached because the packets cannot escape in the middle of the tunnel.



Tunneling a car from France to England

However, this limitation of tunnels is turned into an advantage with VPNs (Virtual Private Networks). A VPN is simply an overlay that is used to provide a measure of security.

Internetwork Routing

Routing through an internet poses the same basic problem as routing within a single network, but with some added complications. To start, the networks may internally use different routing algorithms. For example, one network may use link state routing and another distance vector routing. Since link state algorithms need to know the topology but distance vector algorithms do not, this difference alone would make it unclear how to find the shortest paths across the internet. Networks run by different operators lead to bigger problems. First, the operators may have different ideas about what is a good path through the network. One operator may want the route with the least delay, while another may want the most inexpensive route. This will lead the operators to use different quantities to set the shortest-path costs (e.g., milliseconds of delay vs. monetary cost). The weights will not be comparable across networks, so shortest paths on the internet will not be well defined. Worse yet, one operator may not want another operator to even know the details of the paths in its network, perhaps because the weights and paths may reflect sensitive information (such as the monetary cost) that represents a competitive business advantage. Finally, the internet may be much larger than any of the networks that comprise it. It may therefore require routing algorithms that scale well by using a hierarchy; even if none of the individual networks need to use a hierarchy. All of these considerations lead to a two-level routing algorithm. Within each network, an intra domain or interior gateway protocol is used for routing.(‘‘Gateway’’ is an older term for ‘‘router.’’) It might be a link state protocol of the kind we have already described. Across the networks that make up the internet, an inter domain or exterior gateway protocol is used. The networks may all use different intra domain protocols, but they must use the same inter domain protocol. In the Internet, the inter domain routing protocol is called BGP (Border Gateway Protocol).

We will there is one more important term to introduce. Since each network is operated independently of all the others, it is often referred to as an AS (Autonomous System). A good mental model for an AS is an ISP network. In fact, an ISP network may be comprised of more than one AS, if it is managed, or, has been acquired, as multiple networks. But the difference is usually not significant. The two levels are usually not strictly hierarchical, as highly suboptimal paths might result if a large international network and a small regional network were both abstracted to be a single network. However, relatively little information about routes within the networks is exposed to find routes across the internetwork.

This helps to address all of the complications. It improves scaling and lets operators freely select routes within their own networks using a protocol of their choosing. It also does not require weights to be compared across networks or expose sensitive information outside of networks. However, we have said little so far about how the routes across the networks of the internet are determined. In the Internet, a large determining factor is the business arrangements between ISPs. Each ISP may charge or receive money from the other ISPs for carrying traffic. Another factor is that if internetwork routing requires crossing international boundaries, various laws may suddenly come into play, such as Sweden’s strict privacy laws about exporting personal data about Swedish citizens from Sweden. All of these nontechnical factors are wrapped up in the concept of a routing policy that governs the way autonomous networks select the routes that they use. We will return to routing policies when we describe BGP.

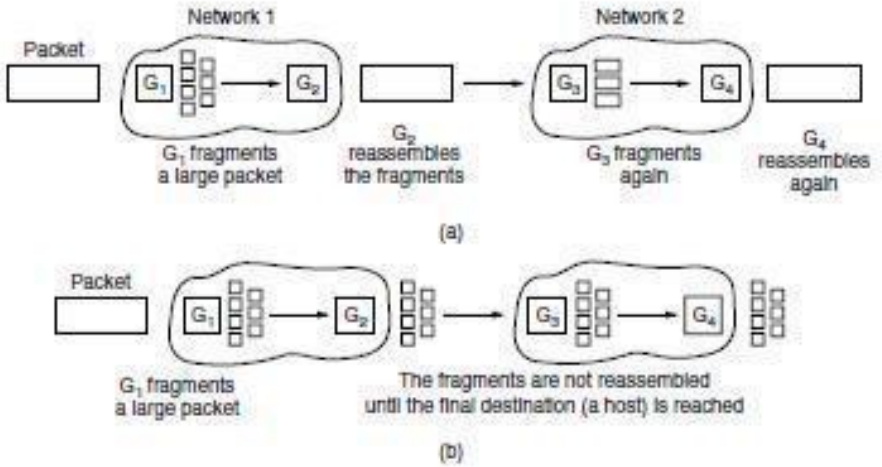
Packet Fragmentation

Each network or link imposes some maximum size on its packets. These limits have various causes, among them

1. Hardware (e.g., the size of an Ethernet frame).
2. Operating system (e.g., all buffers are 512 bytes).
3. Protocols (e.g., the number of bits in the packet length field).
4. Compliance with some (inter)national standard.
5. Desire to reduce error-induced retransmissions to some level.
6. Desire to prevent one packet from occupying the channel too long.

The result of all these factors is that the network designers are not free to choose any old maximum packet size they wish. Maximum payloads for some common technologies are 1500 bytes for Ethernet and 2272 bytes for 802.11. IP is more generous, allows for packets as big as 65,515 bytes. Hosts usually prefer to transmit large packets because this reduces packet overheads such as bandwidth wasted on header bytes. An obvious internetworking problem appears when a large packet wants to travel through a network whose maximum packet size is too small. This nuisance has been a persistent issue, and solutions to it have evolved along with much experience gained on the Internet. One solution is to make sure the problem does not occur in the first place. However, this is easier said than done. A source does not usually know the path a packet will take through the network to a destination, so it certainly does not know how small packets must be to get there. This packet size is called the Path MTU (Path Maximum Transmission Unit). Even if the source did know the path MTU, packets are routed independently in a connectionless network such as the Internet. This routing means that paths may suddenly change, which can unexpectedly change the path MTU. The alternative solution to the problem is to allow routers to break up packets into fragments, sending each fragment as a separate network layer packet. However, as every parent of a small child knows, converting a large object into small fragments is considerably easier than the reverse process. (Physicists have even given this effect a name: the second law of thermodynamics.) Packet-switching networks, too, have trouble putting the fragments back together again. Two opposing strategies exist for recombining the fragments back into the original packet. The first strategy is to make fragmentation caused by a ‘‘small packet’’ network transparent to any subsequent networks through which the packet must pass on its way to the ultimate destination. This option is shown in Fig (a). In this approach, when an oversized packet arrives at G1, the router breaks it up into fragments. Each fragment is addressed to the same exit router, G2, where the pieces are recombined. In this way, passage through the small-packet network is made transparent. Subsequent networks are not even aware that fragmentation has occurred.

Transparent fragmentation is straightforward but has some problems. For one thing, the exit router must know when it has received all the pieces, so either a count field or an ‘‘end of packet’’ bit must be provided. Also, because all packets must exit via the same router so that they can be reassembled, the routes are constrained. By not allowing some fragments to follow one route to the ultimate destination and other fragments a disjoint route, some performance may be lost. More significant is the amount of work that the router may have to do. It may need to buffer the fragments as they arrive, and decide when to throw them away if not all of the fragments arrive. Some of this work may be wasteful, too, as the packet may pass through a series of small packet networks and need to be repeatedly fragmented and reassembled.

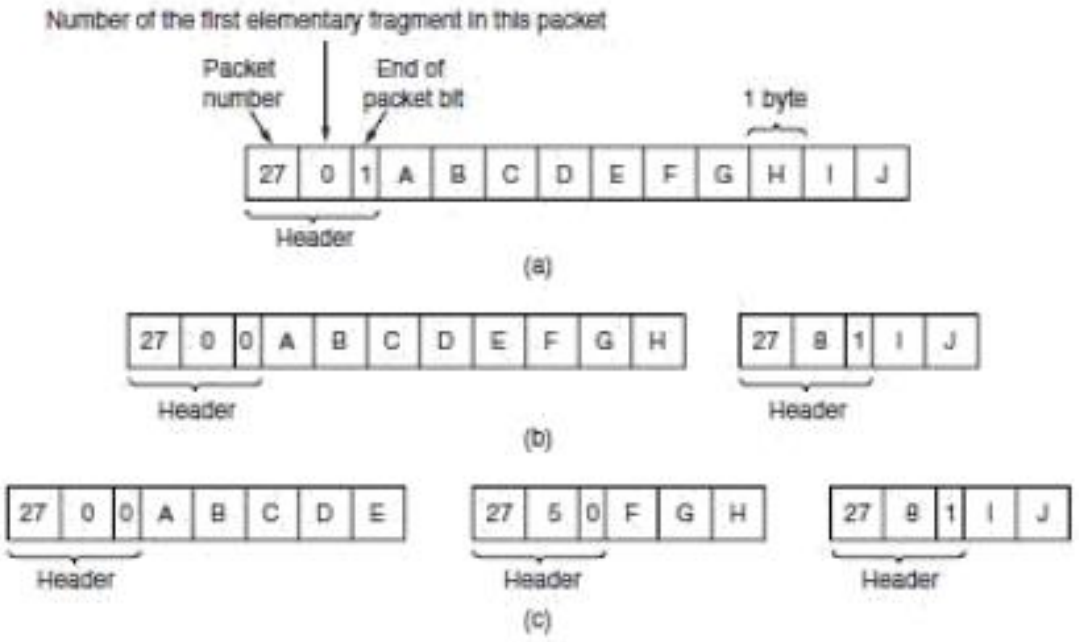


(a) Transparent fragmentation. (b) Nontransparent fragmentation.

The other fragmentation strategy is to refrain from recombining fragments at any intermediate routers. Once a packet has been fragmented, each fragment is treated as though it were an original packet. The routers pass the fragments, as shown in Fig. (b), and reassembly is performed only at the destination host. The main advantage of nontransparent fragmentation is that it requires routers to do less work. IP works this way. A complete design requires that the fragments be numbered in such a way that the original data stream can be reconstructed.

The design used by IP is to give every fragment a packet number (carried on all packets), an absolute byte offset within the packet, and a flag indicating whether it is the end of the packet. An example is shown in Fig. While simple, this design has some attractive properties. Fragments can be placed in a buffer at the destination in the right place for reassembly, even if they arrive out of order.

Fragments can also be fragmented if they pass over a network with a yet smaller MTU. This is shown in Fig. (c). Retransmissions of the packet (if all fragments were not received) can be fragmented into different pieces. Finally, fragments can be of arbitrary size, down to a single byte plus the packet header. In all cases, the destination simply uses the packet number and fragment offset to place the data in the right position, and the end-of-packet flag to determine when it has the complete packet. Unfortunately, this design still has problems. The overhead can be higher than with transparent fragmentation because fragment headers are now carried over some links where they may not be needed. But the real problem is the existence of fragments in the first place. Kent and Mogul (1987) argued that fragmentation is detrimental to performance because, as well as the header overheads, a whole packet is lost if any of its fragments are lost and because fragmentation is more of a burden for hosts than was originally realized.



Fragmentation when the elementary data size is 1 byte.

(a) Original packet, containing 10 data bytes.

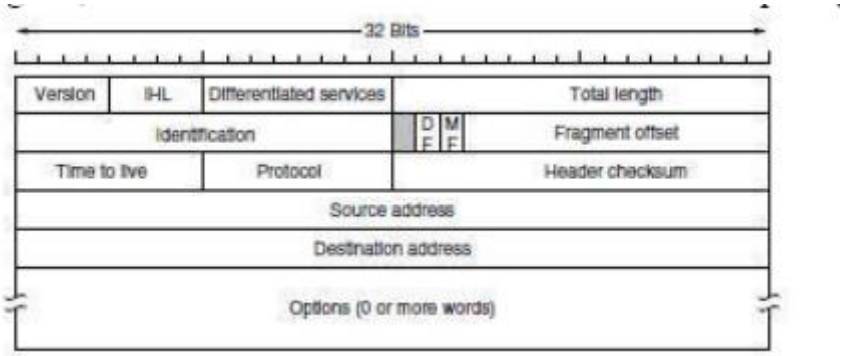
(b) Fragments after passing through a network with maximum packet size of 8 payload bytes plus header.

(c) Fragments after passing through a size 5 gateway.

This leads us back to the original solution of getting rid of fragmentation in the network, the strategy used in the modern Internet. The process is called path MTU discovery (Mogul and Deering, 1990). It works as follows. Each IP packet is sent with its header bits set to indicate that no fragmentation is allowed to be performed. If a router receives a packet that is too large, it generates an error packet, returns it to the source, and drops the packet. This is shown in Fig. When the source receives the error packet, it uses the information inside to refragment the packet into pieces that are small enough for the router to handle. If a router further down the path has an even smaller MTU, the process is repeated.

The IP Version 4 Protocol

An appropriate place to start our study of the network layer in the Internet is with the format of the IP datagrams themselves. An IPv4 datagram consists of a header part and a body or payload part. The header has a 20-byte fixed part and a variable-length optional part. The header format is shown in Fig. 5-46. 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, such as Intel x86 computers, 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.



The IPv4 (Internet Protocol) header.

The Version field keeps track of which version of the protocol the datagram belongs to. Version 4 dominates the Internet today, and that is where we have started our discussion. 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. In fact, IPv6, the next version of IP, was defined more than a decade ago, yet is only just beginning to be deployed. We will describe it later in this section. Its use will eventually be forced when each of China’s almost 231 people has a desktop PC, a laptop, and an IP phone. As an aside on numbering, IPv5 was an experimental real-time stream protocol that was never widely used. Since the header length is not constant, a field in the header, IHL, is provided to tell how long the header is, in 32-bit words. The minimum value is 5, which applies when no options are present. The maximum value of this 4-bit field is 15, which limits the header to 60 bytes, and thus the Options field to 40 bytes. For some options, such as one that records the route a packet has taken, 40 bytes is far too small, making those options useless. The Differentiated services field is one of the few fields that has changed its meaning (slightly) over the years. Originally, it was called the Type of service field. It was and still is intended to distinguish between different classes of service.

Various combinations of reliability and speed are possible. For digitized voice, fast delivery beats accurate delivery. For file transfer, error-free transmission is more important than fast transmission. The Type of service field provided 3 bits to signal priority and 3 bits to signal whether a host cared more about delay, throughput, or reliability. However, no one really knew what to do with these bits at routers, so they were left unused for many years. When differentiated services were designed, IETF threw in the towel and reused this field. Now, the top 6 bits are used to mark the packet with its service class; we described the expedited and assured services earlier in this chapter. The bottom 2 bits are used to carry explicit congestion notification information, such as whether the packet has experienced congestion; we described explicit congestion notification as part of congestion control earlier in this chapter. The Total length includes everything in the datagram—both header and data. The maximum length is 65,535 bytes.

At present, this upper limit is tolerable, but with future networks, larger datagram's may be needed. The Identification field 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, which is surprising, as available real estate in the IP header is extremely scarce. As an April fool's joke, Bellovin (2003) proposed using this bit to detect malicious traffic. This would greatly simplify security, as packets with the ''evil'' bit set would be known to have been sent by attackers and could just be discarded. Unfortunately, network security is not this simple. Then come two 1-bit fields related to fragmentation. DF stands for Don't Fragment. It is an order to the routers not to fragment the packet. Originally, it was intended to support hosts incapable of putting the pieces back together again. Now it is used as part of the process to discover the path MTU, which is the largest packet that can travel along a path without being fragmented. By marking the datagram with the DF bit, the sender knows it will either arrive in one piece, or an error message will be returned to the sender. MF stands for More Fragments. All fragments except the last one have this bit set. It is needed to know when all fragments of a datagram have arrived. The Fragment offset tells where in the current packet this fragment belongs.

All fragments except the last one in a datagram must be a multiple of 8 bytes, the elementary fragment unit. Since 13 bits are provided, there is a maximum of 8192 fragments per datagram, supporting a maximum packet length up to the limit of the Total length field. Working together, the Identification, MF, and Fragment offset fields are used to implement fragmentation as described in Sec. 5.5.5.The TtL (Time to live) field is a counter used to limit packet lifetimes. It was originally supposed to count time in seconds, allowing a maximum lifetime of 255 sec. It must be decremented on each hop and is supposed to be decremented multiple times when a packet is queued for a long time in a router. In practice, it just counts hops. When it hits zero, the packet is discarded and a warning packet is sent back to the source host. This feature prevents packets from wandering around forever, something that otherwise might happen if the routing tables ever become corrupted.

When the network layer has assembled a complete packet, it needs to know what to do with it. The Protocol field tells it which transport process to give the packet to. TCP is one possibility, but so are UDP and some others. The numbering of protocols is global across the entire Internet. Protocols and other assigned numbers were formerly listed in RFC 1700, but nowadays they are contained in an online database located at www.iana.org. Since the header carries vital information such as addresses, it rates its own checksum for protection, the Header checksum. The algorithm is to add up all the 16-bit half words of the header as they arrive, using one's complement arithmetic, and then take the one's complement of the result. For purposes of this algorithm, the Header checksum is assumed to be zero upon arrival. Such a checksum is useful for detecting errors while the packet travels through the network. Note that it must be recomputed at each hop because at least one field always changes (the Time to live field), but tricks can be used to speed up the computation. The Source address and Destination address indicate the IP address of the source and destination network interfaces. The Options field was designed to provide an escape to allow subsequent versions of the protocol to include information not present in the original design, to permit experimenters to try out new ideas, and to avoid allocating header bits to information that is rarely needed. The options are of variable length. Each begins with a 1-byte code identifying the option. Some options are followed by a 1-byte option length field, and then one or more data bytes. The Options field is padded out to a multiple of 4 bytes. Originally, the five options listed in Fig. were defined.

The Security option tells how secret the information is. In theory, a military router might use this field to specify not to route packets through certain countries the military considers to be ‘‘bad guys.’’ In practice, all routers ignore it, so its only practical function is to help spies find the good stuff more easily. The Strict source routing option gives the complete path from source to destination as a sequence of IP addresses.

It is most useful for system managers who need to send emergency packets when the routing tables have been corrupted, or for making timing measurements. The Loose source routing option requires the packet to traverse the list of routers specified, in the order specified, but it is allowed to pass through other routers on the way. Normally, this option will provide only a few routers, to force a particular path. For example, to force a packet from London to Sydney to go west instead of east, this option might specify routers in New York, Los Angeles, and Honolulu. This option is most useful when political or economic considerations dictate passing through or avoiding certain countries.

The Record route option tells each router along the path to append its IP address to the Options field. This allows system managers to track down bugs in the routing algorithms (‘‘Why are packets from Houston to Dallas visiting Tokyo first?’’). When the ARPANET was first set up, no packet ever passed through more than nine routers, so 40 bytes of options was plenty. As mentioned above, now it is too small. Finally, the Timestamp option is like the Record route option, except that in addition to recording its 32-bit IP address, each router also records a 32-bit timestamp. This option, too, is mostly useful for network measurement. Today, IP options have fallen out of favor. Many routers ignore them or do not process them efficiently, shunting them to the side as an uncommon case. That is, they are only partly supported and they are rarely used.

IP Version 6

IP has been in heavy use for decades. It has worked extremely well, as demonstrated by the exponential growth of the Internet. Unfortunately, IP has become a victim of its own popularity: it is close to running out of addresses. Even with CIDR and NAT using addresses more sparingly, the last IPv4 addresses are expected to be assigned by ICANN before the end of 2012. This looming disaster was recognized almost two decades ago, and it sparked a great deal of discussion and controversy within the Internet community about what to do about it. In this section, we will describe both the problem and several proposed solutions. The only long-term solution is to move to larger addresses. IPv6 (IP version 6) is a replacement design that does just that. It uses 128-bit addresses; a shortage of these addresses is not likely any time in the foreseeable future. However, IPv6 has proved very difficult to deploy. It is a different network layer protocol that does not really interwork with IPv4, despite many similarities. Also, companies and users are not really sure why they should want IPv6 in any case. The result is that IPv6 is deployed and used on only a tiny fraction of the Internet (estimates are 1%) despite having been an Internet Standard since 1998. The next several years will be an interesting time, as the few remaining IPv4 addresses are allocated. Will people start to auction off their IPv4 addresses on eBay? Will a black market in them spring up? Who knows? In addition to the address problems, other issues loom in the background. In its early years, the Internet was largely used by universities, high-tech industries, and the U.S. Government (especially the Dept. of Defense). With the explosion of interest in the Internet starting in the mid-1990s, it began to be used by a different group of people, often with different requirements. For one thing, numerous people with smart phones use it to keep in contact with their home bases. For another, with the impending convergence of the computer, communication, and entertainment industries, it may not be that long before every telephone and television set in the world is an Internet node, resulting in a billion machines being used for audio and video on demand. Under these circumstances, it became apparent that IP had to evolve and become more flexible. Seeing these problems on the horizon, in 1990 IETF started work on a new version of

IP, one that would never run out of addresses, would solve a variety of other problems, and be more flexible and efficient as well. Its major goals were:

1. Support billions of hosts, even with inefficient address allocation.
2. Reduce the size of the routing tables.
3. Simplify the protocol, to allow routers to process packets faster.
4. Provide better security (authentication and privacy).
5. Pay more attention to the type of service, particularly for real-time data.
6. Aid multicasting by allowing scopes to be specified.
7. Make it possible for a host to roam without changing its address.
8. Allow the protocol to evolve in the future.
9. Permit the old and new protocols to coexist for years.

The design of IPv6 presented a major opportunity to improve all of the features in IPv4 that fall short of what is now wanted. To develop a protocol that met all these requirements, IETF issued a call for proposals and discussion in RFC 1550. Twenty-one responses were initially received. By December 1992, seven serious proposals were on the table. They ranged from making minor patches to IP, to throwing it out altogether and replacing it with a completely different protocol. One proposal was to run TCP over CLNP, the network layer protocol designed for OSI. With its 160-bit addresses, CLNP would have provided enough address space forever as it could give every molecule of water in the oceans enough addresses (roughly 25) to set up a small network. This choice would also have unified two major network layer protocols. However, many people felt that this would have been an admission that something in the OSI world was actually done right, a statement considered Politically Incorrect in Internet circles. CLNP was patterned closely on IP, so the two are not really that different. In fact, the protocol ultimately chosen differs from
IP far more than CLNP does. Another strike against CLNP was its poor support for service types, something required to transmit multimedia efficiently.

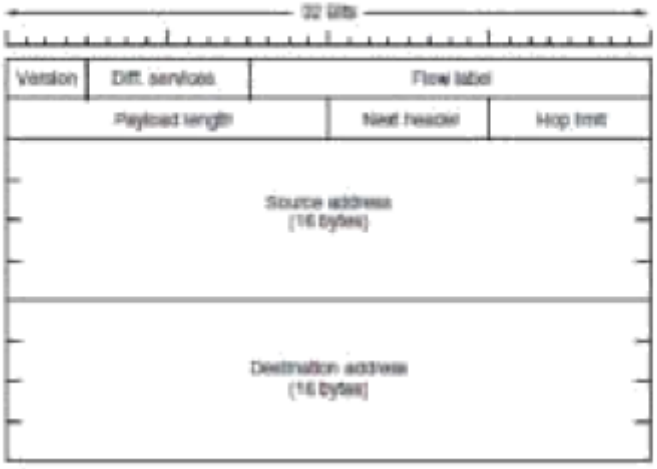
Three of the better proposals were published in *IEEE Network* (Deering, 1993; Francis, 1993; and Katz and Ford, 1993). After much discussion, revision, and jockeying for position, a modified combined version of the Deering and Francis proposals, by now called **SIPP** (**Simple Internet Protocol Plus**) was selected and given the designation **IPv6.** IPv6 meets IETF's goals fairly well. It maintains the good features of IP, discards or deemphasizes the bad ones, and adds new ones where needed. In general, IPv6 is not compatible with IPv4, but it is compatible with the other auxiliary Internet protocols, including TCP, UDP, ICMP, IGMP, OSPF, BGP, and DNS, with small modifications being required to deal with longer addresses. The main features of IPv6 are discussed below. More information about it can be found in RFCs 2460 through 2466.First and foremost, IPv6 has longer addresses than IPv4. They are 128 bits long, which solves the problem that IPv6 set out to solve: providing an effectively unlimited supply of Internet addresses. We will have more to say about addresses shortly. The second major improvement of IPv6 is the simplification of the header. It contains only seven fields (versus 13 in IPv4). This change allows routers to process packets faster and thus improves throughput and delay. We will discuss the header shortly, too. The third major improvement is better support for options. This change was essential with the new header because fields that previously were required are now optional (because they are not used so often). In addition, the way options are represented is different, making it simple for routers to skip over options not intended for them. This feature speeds up packet processing time.

A fourth area in which IPv6 represents a big advance is in security. IETF had its fill of newspaper stories about precocious 12-year-olds using their personal computers to break into banks and military bases all over the Internet.

There was a strong feeling that something had to be done to improve security. Authentication and privacy are key features of the new IP. These were later retrofitted to IPv4, however, so in the area of security the differences are not so great any more. Finally, more attention has been paid to quality of service. Various halfhearted efforts to improve QoS have been made in the past, but now, with the growth of multimedia on the Internet, the sense of urgency is greater.

The Main IPv6 Header

The IPv6 header is shown in Fig. 5-56. The *Version* field is always 6 for IPv6 (and 4 for IPv4). During the transition period from IPv4, which has already taken more than a decade, routers will be able to examine this field to tell what kind of packet they have. As an aside, making this test wastes a few instructions in the critical path, given that the data link header usually indicates the network protocol for de-multiplexing, so some routers may skip the check. For example, the Ethernet *Type* field has different values to indicate an IPv4 or an IPv6 payload. The discussions between the ‘‘Do it right’’ and ‘‘Make it fast’’ camps will no doubt be lengthy and vigorous.



The IPv6 fixed header (required)

The Differentiated services field (originally called Traffic class) is used to distinguish the class of service for packets with different real-time delivery requirements. It is used with the Differentiated service architecture for quality of service in the same manner as the field of the same name in the IPv4 packet. Also, the low-order 2 bits are used to signal explicit congestion indications, again in the same way as with IPv4.

The Flow label field 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 pseudo connection. For example, a stream of packets from one process on a certain source host to a process on a specific destination host might have stringent delay requirements and thus need reserved bandwidth. The flow can be set up in advance and given an identifier. When a packet with a nonzero Flow label shows up, all the routers can look it up in internal tables to see what kind of special treatment it requires. In effect, flows are an attempt to have it both ways: the flexibility of a datagram network and the guarantees of a virtual-circuit network. Each flow for quality of service purposes is designated by the source address, destination address, and flow number. This design means that up to 220 flows may be active at the same time between a given pair of IP addresses.

It also means that even if two flows coming from different hosts but with the same flow label pass through the same router, the router will be able to tell them apart using the source and destination addresses. It is expected that flow labels will be chosen randomly, rather than assigned sequentially starting at 1, so routers are expected to hash them. The Payload length field tells how many bytes follow the 40-byte header of Fig. The name was changed from the IPv4 Total length field because the meaning was changed slightly: the 40 header bytes are no longer counted as part of the length (as they used to be). This change means the payload can now be 65,535 bytes instead of a mere 65,515 bytes.

The Next header field lets the cat out of the bag. The reason the header could be simplified is that there can be additional (optional) extension headers. This field tells which of the (currently) six extension headers, if any, follow this one. If this header is the last IP header, the Next header field tells which transport protocol handler (e.g., TCP, UDP) to pass the packet to. The Hop limit field is used to keep packets from living forever. It is, in practice, the same as the Time to live field in IPv4, namely, a field that is decremented on each hop. In theory, in IPv4 it was a time in seconds, but no router used it that way, so the name was changed to reflect the way it is actually used. Next come the Source address and Destination address fields. Deering's original proposal, SIP, used 8-byte addresses, but during the review process many people felt that with 8-byte addresses IPv6 would run out of addresses within a few decades, whereas with 16-byte addresses it would never run out. Other people argued that 16 bytes was overkill, whereas still others favored using 20-byte addresses to be compatible with the OSI datagram protocol. Still another faction wanted variable-sized addresses. After much debate and more than a few words unprintable in an academic textbook, it was decided that fixed-length 16-byte addresses were the best compromise.

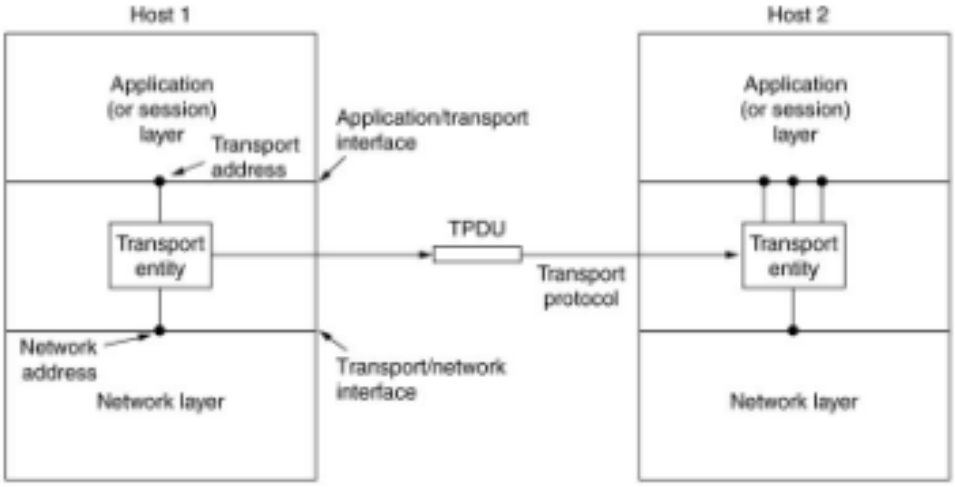
THE TRANSPORT LAYER

THE TRANSPORT SERVICE

Services Provided to the Upper Layers

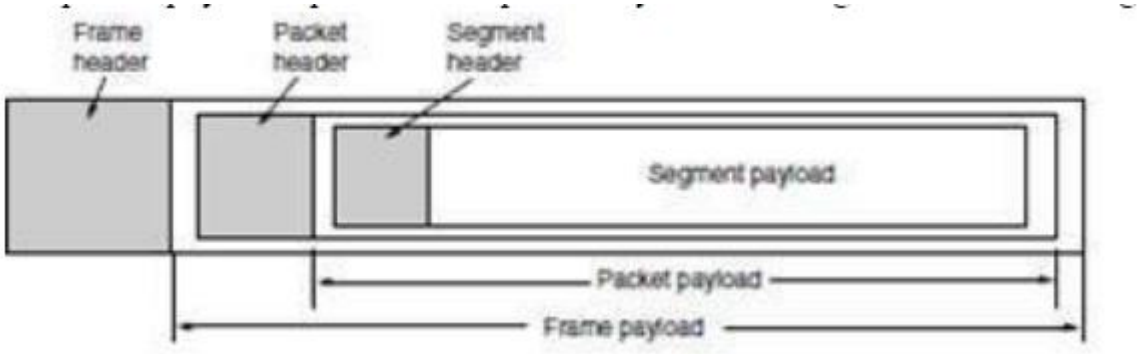
The ultimate goal of the transport layer is to provide efficient, reliable, and cost-effective data transmission service to its users, normally processes in the application layer. To achieve this, the transport layer makes use of the services provided by the network layer. The software and/or hardware within the transport layer that does the work is called the **transport entity**. The transport entity can be located in the operating system kernel, in a library package bound into network applications, in a separate user process, or even on the network interface card. The first two options are most common on the Internet. The (logical) relationship of the network, transport, and application layers is illustrated in Fig. there are also two types of transport service. The connection-oriented transport service is similar to the connection-oriented network service in many ways. In both cases, connections have three phases: establishment, data transfer, and release. Addressing and flow control are also similar in both layers. Furthermore, the connectionless transport service is also very similar to the connectionless network service. However, note that it can be difficult to provide a connectionless transport service on top of a connection-oriented network service, since it is inefficient to set up a connection to send a single packet and then tear it down immediately afterwards. The obvious question is this: if the transport layer service is so similar to the network layer service, why are there two distinct layers? Why is one layer not Problems occur, that's what? The users have no real control over the network layer, so they cannot solve the problem of poor service by using better routers or putting more error handling in the data link layer because they don't own the routers.

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, with no indication of what has happened to the data currently in transit, it can set up a new network connection to the remote transport entity. Using this new network connection, it can send a query to its peer asking which data arrived and which did not, and knowing where it was, pick up from where it left off.



Transport Service Primitives

To allow users to access the transport service, the transport layer must provide some operations to application programs, that is, a transport service interface. Each transport service has its own interface. In this section, we will first examine a simple (hypothetical) transport service and its interface to see the bare essentials. In the following section, we will look at a real example. The transport service is similar to the network service, but there are also some important differences. The main difference is that the network service is intended to model the service offered by real networks, warts and all. Real networks can lose packets, so the network service is generally unreliable. A quick note on terminology is now in order. For lack of a better term, we will use the term **segment** for messages sent from transport entity to transport entity.TCP, UDP and other Internet protocols use this term. Some older protocols used the ungainly name **TPDU** (**Transport Protocol Data Unit**). That term is not used much anymore now but you may see it in older papers and books the network entity similarly processes the packet header and then passes the contents of the packet payload up to the transport entity. This nesting is illustrated in Fig. 6-3



Connection Establishment

Establishing a connection sounds easy, but it is actually surprisingly tricky. At first glance, it would seem sufficient for one transport entity to just send a CONNECTION REQUEST segment to the destination and wait for a CONNECTION ACCEPTED reply. The problem occurs when the network can lose, delay, corrupt, and duplicate packets. This behavior causes serious complications. Imagine a network that is so congested that acknowledgements hardly ever get back in time and each packet times out and is retransmitted two or three times. Suppose that the network uses datagrams inside and that every packet follows a different route. Some of the packets might get stuck in a traffic jam inside the network and take a long time to arrive. That is, they may be delayed in the network and pop out much later, when the sender thought that they had been lost. The worst possible nightmare is as follows. A user establishes a connection with a bank, sends messages telling the bank to transfer a large amount of money to the account of a not-entirely-trustworthy person. Unfortunately, the packets decide to take the scenic route to the destination and go off exploring a remote corner of the network. The sender then times out and sends them all again. This time the packets take the shortest route and are delivered quickly so the sender releases the connection.

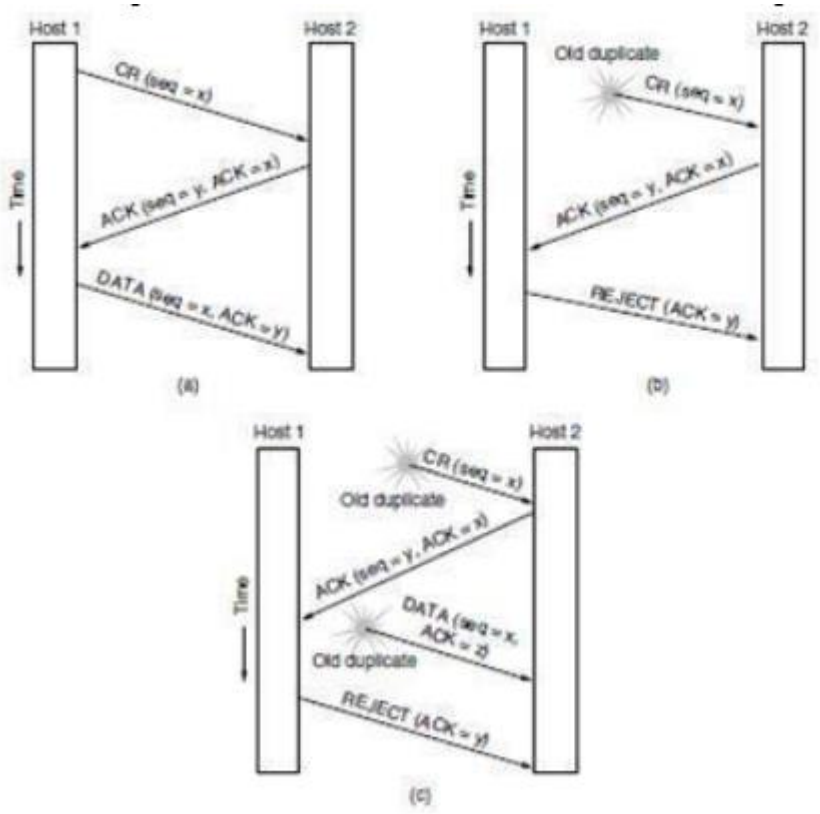
Packet lifetime can be restricted to a known maximum using one (or more) of the following techniques:

1. Restricted network design.
2. Putting a hop counter in each packet.
3. Time stamping each packet.

The first technique includes any method that prevents packets from looping, combined with some way of bounding delay including congestion over the (now known) longest possible path. It is difficult, given that internets may range from a single city to international in scope. The second method consists of having the hop count initialized to some appropriate value and decremented each time the packet is forwarded. The network protocol simply discards any packet whose hop counter becomes zero. The third method requires each packet to bear the time it was created, with the routers agreeing to discard any packet older than some agreed-upon time. This latter method requires the router clocks to be synchronized, which itself is a nontrivial task, and in practice a hop counter is a close enough approximation to age.

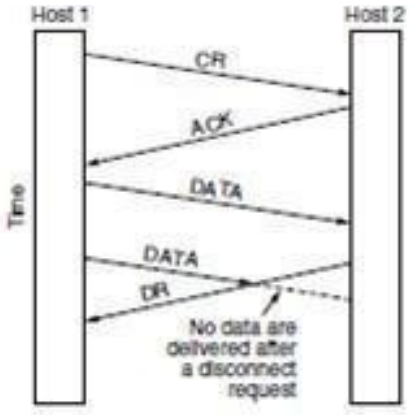
TCP uses this three-way handshake to establish connections. Within a connection, a timestamp is used to extend the 32-bit sequence number so that it will not wrap within the maximum packet lifetime, even for gigabit-per-second connections. This mechanism is a fix to TCP that was needed as it was used on faster and faster links.

It is described in RFC 1323 and called PAWS (Protection Against Wrapped Sequence numbers). Across connections, for the initial sequence numbers and before PAWS can come into play, TCP originally use the clock-based scheme just described. However, this turned out to have security vulnerability. The clock made it easy for an attacker to predict the next initial sequence number and send packets that tricked the three-way handshake and established a forged connection. To close this hole, pseudorandom initial sequence numbers are used for connections in practice.



Connection Release

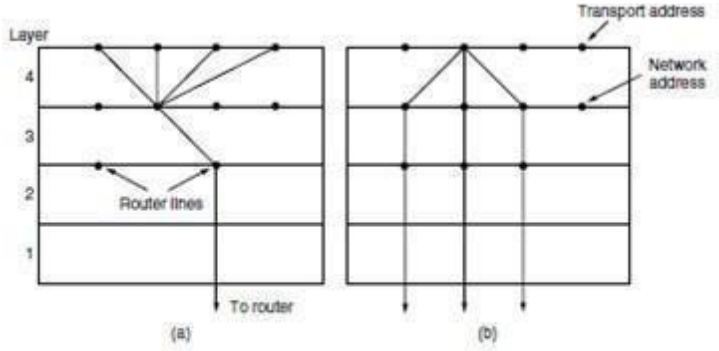
Releasing a connection is easier than establishing one. Nevertheless, there are more pitfalls than one might expect here. As we mentioned earlier, there are two styles of terminating a connection: asymmetric release and symmetric release Asymmetric release is the way the telephone system works: when one party hangs up, the connection is broken. Symmetric release treats the connection as two separate unidirectional connections and requires each one to be released separately. Asymmetric release is abrupt and may result in data loss. Consider the scenario of Fig. After the connection is established, host 1 sends a segment that arrives properly at host 2. Then host 1 sends another segment. Unfortunately, host 2 issues a DISCONNECT before the second segment arrives. The result is that the connection is released and data are lost.



Abrupt disconnection with loss of data.

Crash Recovery

If hosts and routers are subject to crashes or connections are long-lived (e.g., large software or media downloads) recovery from these crashes becomes an issue. If the transport entity is entirely within the hosts, recovery from network and router crashes is straightforward. The transport entities expect lost segments all the time and know how to cope with them by using retransmissions. A more troublesome problem is how to recover from host crashes. In particular, it may be desirable for clients to be able to continue working when servers crash and quickly reboot. To illustrate the difficulty, let us assume that one host, the client, is sending a long file to another host, the file server, using a simple Stop-and-wait protocol. The transport layer on the server just passes the incoming segments to the transport user, one by one. Partway through the transmission, the server crashes.



(a) Multiplexing. (b) Inverse multiplexing.

When it comes back up, its tables are reinitialized, so it no longer knows precisely where it was. In an attempt to recover its previous status, the server might send a broadcast segment to all other hosts, announcing that it has just crashed and requesting that its clients inform it of the status of all open connections. Each client can be in one of two states: one segment outstanding, *S1*, or no segments outstanding, *S0*. Based on only this state information, the client must decide whether to retransmit the most recent segment.

Introduction to ARP

ARP is a simple protocol that operates at the Data Link Layer (Layer 2) of the OSI model. It is responsible for resolving IP addresses to MAC addresses. This is necessary because IP addresses are assigned to devices on a network, but the data link layer uses MAC addresses to identify devices. When a device wants to send a packet to another device on the same LAN, it uses ARP to determine the MAC address of the destination device. When a device wants to send a packet to another device on the same LAN, it broadcasts an ARP request packet to all devices on the network. The packet contains the IP address of the destination device. The device with the matching IP address responds with its MAC address, and the sender can then use this information to send the packet to the destination device.

ARP (Address Resolution Protocol) is a communication protocol used to map a network address, such as an IP address, to a physical address, such as a MAC address. It is an essential component of the TCP/IP protocol suite, and it allows devices on a local area network (LAN) to communicate with each other. In this article, we will explore the different types of ARP, their uses, and examples of how they are implemented in a network.

Types of ARP

ARP Request

ARP requests are used to determine the MAC address of a device on a LAN. When a device wants to send a packet to another device, it broadcasts an ARP request packet containing the IP address of the destination device. The device with the matching IP address responds with its MAC address, and the sender can use this information to send the packet to the destination device.

Example − A device on a LAN wants to send a packet to another device with the IP address 192.168.1.100. It broadcasts an ARP request packet containing the IP address 192.168.1.100. The device with the IP address 192.168.1.100 responds with its MAC address, 00-11-22-33-44-55. The sender can now use this information to send the packet to the destination device.

ARP Reply

ARP replies are used to respond to ARP requests. When a device receives an ARP request packet containing its IP address, it responds with its MAC address. This allows the sender to send packets to the destination device.

Example: A device on a LAN receives an ARP request packet containing the IP address 192.168.1.100. It responds with its MAC address, 00-11-22-33-44-55. The sender can now use this information to send packets to the destination device.

ARP Cache

ARP cache is used to store recently resolved IP-MAC address mappings. When a device sends an ARP request, it stores the mapping in its ARP cache. This allows the device to quickly send packets to the same destination device without having to send an ARP request again.

Example − A device on a LAN sends an ARP request to determine the MAC address of the device with IP address 192.168.1.100. It receives a response with the MAC address 00-11-22-33-44-55. The device stores this mapping in its ARP cache. When it wants to send a packet to the same device again, it can quickly retrieve the MAC address from its ARP cache instead of sending another ARP request.

ARP Poisoning

ARP poisoning is a type of attack in which an attacker sends false ARP replies to a device, causing it to update its ARP cache with incorrect information. This can to intercept traffic intended for another device, or even redirect traffic to the attacker's device.

Example - An attacker sends false ARP replies to a device on a LAN, claiming that the attacker's device has the IP address 192.168.1.100 and the MAC address 00-11-22-33-44-55. The device updates its ARP cache with this information and sends packets intended for the device with IP address 192.168.1.100 to the attacker's device. The attacker can then intercept or redirect these packets.

Preventing ARP Poisoning

To prevent ARP poisoning, there are several techniques that can be used −

ARP Inspection − ARP inspection is a technique used to validate ARP requests and replies. It compares the ARP packets to a pre-configured list of allowed IP-MAC address mappings and discards any packets that do not match.

Port Security − Port security is a technique used to limit the number of MAC addresses that can be learned on a switch port. This can prevent an attacker from flooding the switch with false ARP replies.

ARP Spoofing Detection − ARP spoofing detection is a technique used to detect and alert on ARP spoofing attempts. This can be done by monitoring the ARP cache for changes, or by using a protocol analyzer to capture and analyze ARP packets.

Virtual Private LAN Service (VPLS) − VPLS is a technique used to segment a LAN into multiple virtual LANs. This can prevent an attacker from broadcasting false ARP replies to the entire LAN.

Dynamic Host Configuration Protocol

Dynamic Host Configuration Protocol (DHCP) is a network management protocol used to dynamically assign an IP address to nay device, or node, on a network so they can communicate using IP (Internet Protocol). DHCP automates and centrally manages these configurations. There is no need to manually assign IP addresses to new devices. Therefore, there is no requirement for any user configuration to connect to a DHCP based network. DHCP can be implemented on local networks as well as large enterprise networks. DHCP is the default protocol used by the most routers and networking equipment. DHCP is also called RFC (Request for comments).

How DHCP works

DHCP runs at the application layer of the TCP/IP protocol stack to dynamically assign IP addresses to DHCP clients/nodes and to allocate TCP/IP configuration information to the DHCP clients. Information includes subnet mask information, default gateway, IP addresses and domain name system addresses. DHCP is based on client-server protocol in which servers manage a pool of unique IP addresses, as well as information about client configuration parameters, and assign addresses out of those address pools.

The DHCP lease process works as follows:

- First of all, a client (network device) must be connected to the internet.
- DHCP clients request an IP address. Typically, client broadcasts a query for this information.
- DHCP server responds to the client request by providing IP server address and other configuration information. This configuration information also includes time period, called a lease, for which the allocation is valid.
- When refreshing an assignment, a DHCP clients request the same parameters, but the DHCP server may assign a new IP address. This is based on the policies set by the administrator.

Components of DHCP

When working with DHCP, it is important to understand all of the components. Following are the list of components:

- **DHCP Server:** DHCP server is a networked device running the DCHP service that holds IP addresses and related configuration information. This is typically a server or a router but could be anything that acts as a host, such as an SD-WAN appliance.
- **DHCP client:** DHCP client is the endpoint that receives configuration information from a DHCP server. This can be any device like computer, laptop, IoT endpoint or anything else that requires connectivity to the network. Most of the devices are configured to receive DHCP information by default.
- **IP address pool:** IP address pool is the range of addresses that are available to DHCP clients. IP addresses are typically handed out sequentially from lowest to the highest.
- **Subnet:** Subnet is the partitioned segments of the IP networks. Subnet is used to keep networks manageable.

- **Lease:** Lease is the length of time for which a DHCP client holds the IP address information. When a lease expires, the client has to renew it.
- **DHCP relay:** A host or router that listens for client messages being broadcast on that network and then forwards them to a configured server. The server then sends responses back to the relay agent that passes them along to the client. DHCP relay can be used to centralize DHCP servers instead of having a server on each subnet.

Benefits of DHCP

There are following benefits of DHCP:

Centralized administration of IP configuration: DHCP IP configuration information can be stored in a single location and enables that administrator to centrally manage all IP address configuration information.

Dynamic host configuration: DHCP automates the host configuration process and eliminates the need to manually configure individual host. When TCP/IP (Transmission control protocol/Internet protocol) is first deployed or when IP infrastructure changes are required.

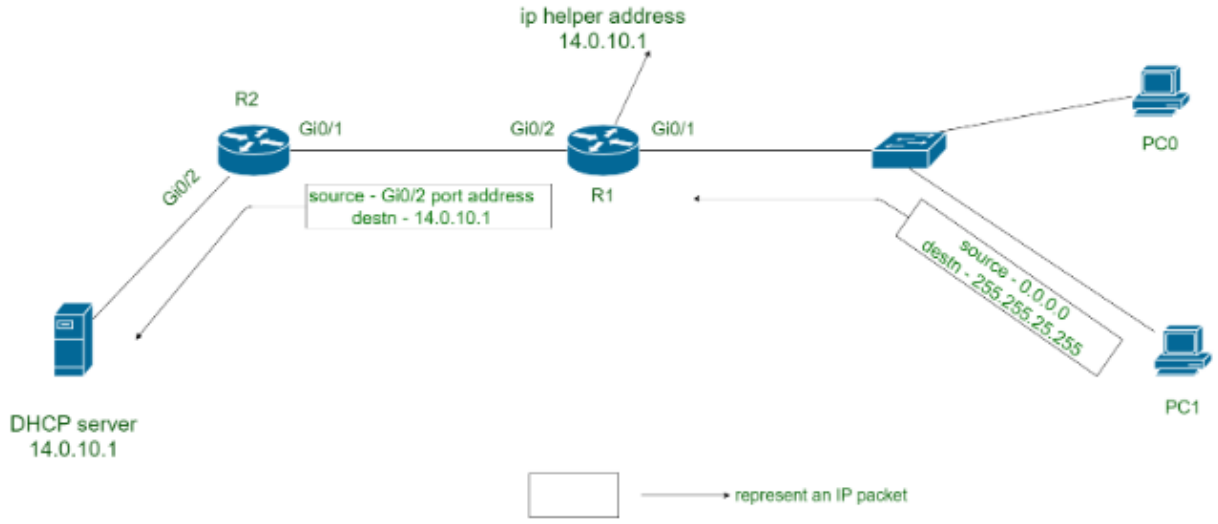
Seamless IP host configuration: The use of DHCP ensures that DHCP clients get accurate and timely IP configuration IP configuration parameter such as IP address, subnet mask, default gateway, IP address of DND server and so on without user intervention.

Flexibility and scalability: Using DHCP gives the administrator increased flexibility, allowing the administrator to move easily change IP configuration when the infrastructure changes.

Working of DHCP

The working of DHCP is as follows:

DHCP works on the Application layer of the TCP/IP Protocol. The main task of DHCP is to dynamically assigns IP Addresses to the Clients and allocate information on TCP/IP configuration to Clients. For more, you can refer to the Article Working of DHCP. The DHCP port number for the server is 67 and for the client is 68. It is a client-server protocol that uses UDP services. An IP address is assigned from a pool of addresses. In DHCP, the client and the server exchange mainly 4 DHCP messages in order to make a connection, also called the DORA process, but there are 8 DHCP messages in the process.

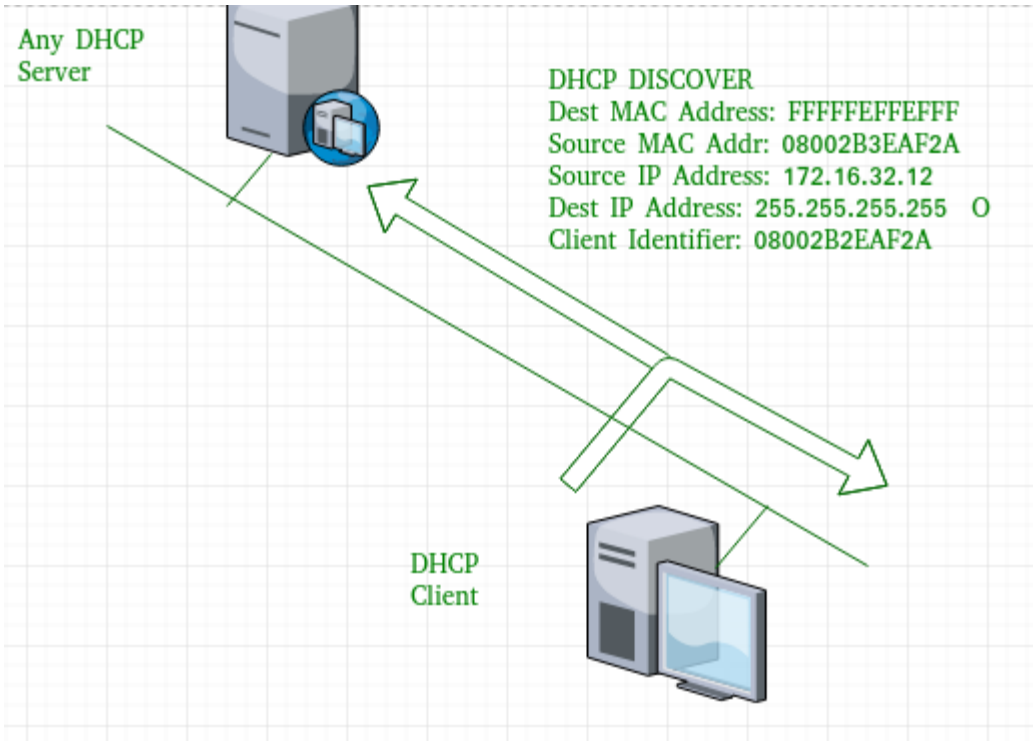


Working of DHCP

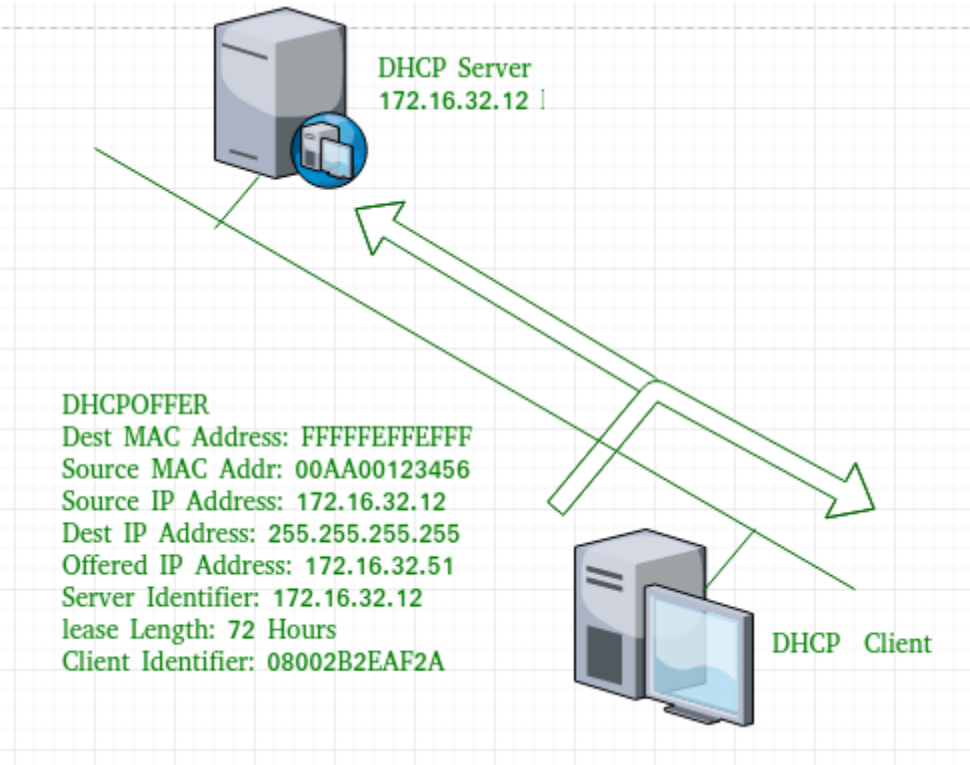
Working of DHCP

The 8 DHCP Messages:

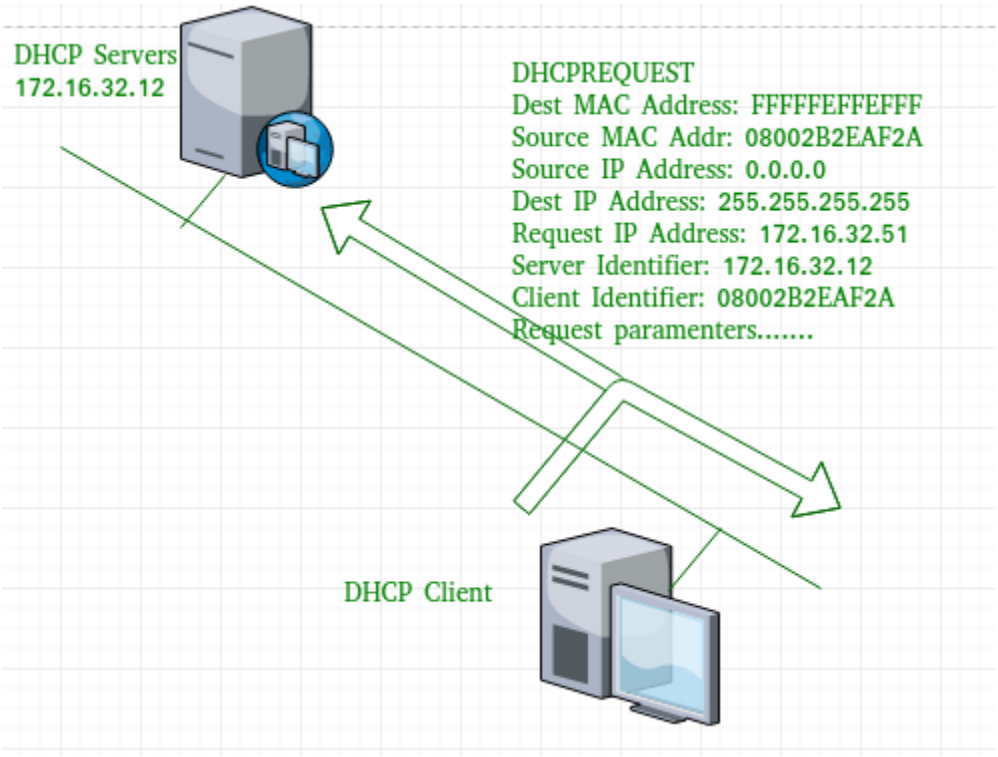
1. **DHCP discover message:** This is the first message generated in the communication process between the server and the client. This message is generated by the Client host in order to discover if there is any DHCP server/servers are present in a network or not. This message is broadcasted to all devices present in a network to find the DHCP server. This message is 342 or 576 bytes long As shown in the figure, the source MAC address (client PC) is 08002B2EAF2A, the destination MAC address(server) is FFFFFFFFFFFF, the source IP address is 0.0.0.0(because the PC has had no IP address till now) and the destination IP address is 255.255.255.255 (IP address used for broadcasting). As they discover message is broadcast to find out the DHCP server or servers in the network therefore broadcast IP address and MAC address is used.



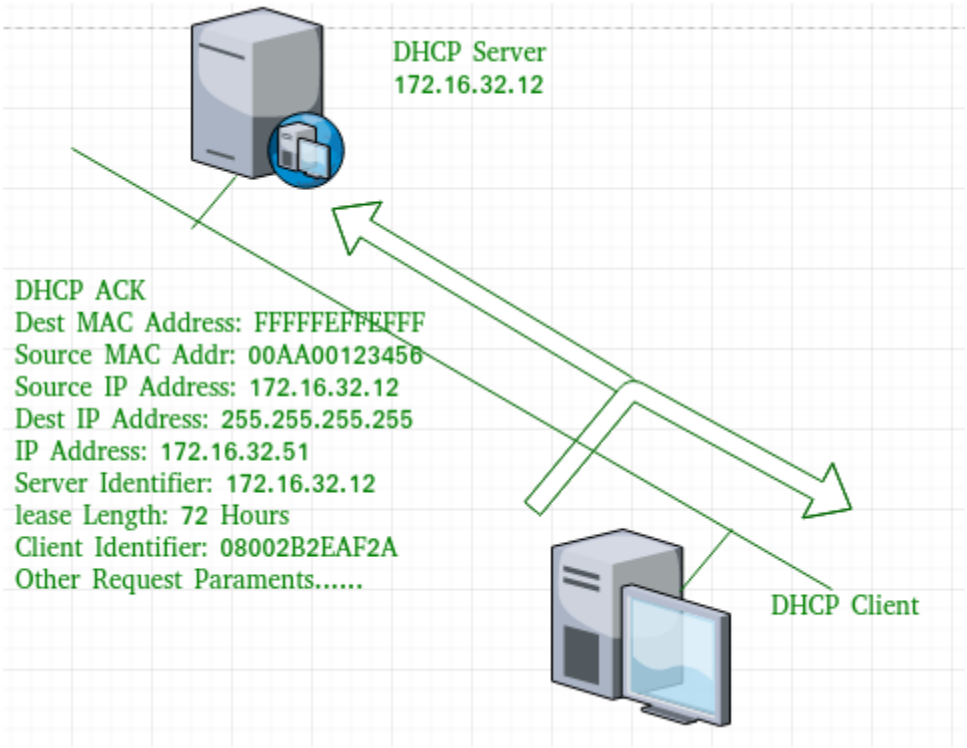
2. **DHCP offers a message:** The server will respond to the host in this message specifying the unleased IP address and other TCP configuration information. This message is broadcasted by the server. The size of the message is 342 bytes. If there is more than one DHCP server present in the network then the client host will accept the first DHCP OFFER message it receives. Also, a server ID is specified in the packet in order to identify the server. Now, for the offer message, the source IP address is 172.16.32.12 (server's IP address in the example), the destination IP address is 255.255.255.255 (broadcast IP address), the source MAC address is 00AA00123456, the destination MAC address is FFFFFFFFFFFF. Here, the offer message is broadcast by the DHCP server therefore destination IP address is the broadcast IP address and destination MAC address is FFFFFFFFFFFF and the source IP address is the server IP address and the MAC address is the server MAC address. Also, the server has provided the offered IP address 192.16.32.51 and a lease time of 72 hours (after this time the entry of the host will be erased from the server automatically). Also, the client identifier is the PC MAC address (08002B2EAF2A) for all the messages.



3. DHCP request message: When a client receives an offer message, it responds by broadcasting a DHCP request message. The client will produce a gratuitous ARP in order to find if there is any other host present in the network with the same IP address. If there is no reply from another host, then there is no host with the same TCP configuration in the network and the message is broadcasted to the server showing the acceptance of the IP address. A Client ID is also added to this message. Now, the request message is broadcast by the client PC therefore source IP address is 0.0.0.0(as the client has no IP right now) and destination IP address is 255.255.255.255 (the broadcast IP address) and the source MAC address is 08002B2EAF2A (PC MAC address) and destination MAC address is FFFFFFFFFFFF.



4. DHCP acknowledgment message: In response to the request message received, the server will make an entry with a specified client ID and bind the IP address offered with lease time. Now, the client will have the IP address provided by the server. Now the server will make an entry of the client host with the offered IP address and lease time. This IP address will not be provided by the server to any other host. The destination MAC address is FFFFFFFFFFFF and the destination IP address is 255.255.255.255 and the source IP address is 172.16.32.12 and the source MAC address is 00AA00123456 (server MAC address).



5. DHCP negative acknowledgment message: Whenever a DHCP server receives a request for an IP address that is invalid according to the scopes that are configured, it sends a DHCP Nak message to the client. Eg-when the server has no IP address unused or the pool is empty, then this message is sent by the server to the client.

6. DHCP decline: If the DHCP client determines the offered configuration parameters are different or invalid, it sends a DHCP decline message to the server. When there is a reply to the gratuitous ARP by any host to the client, the client sends a DHCP decline message to the server showing the offered IP address is already in use.

7. DHCP release: A DHCP client sends a DHCP release packet to the server to release the IP address and cancel any remaining lease time.

8. DHCP inform: If a client address has obtained an IP address manually then the client uses DHCP information to obtain other local configuration parameters, such as domain name. In reply to the DHCP inform message, the DHCP server generates a DHCP ack message with a local configuration suitable for the client without allocating a new IP address. This DHCP ack message is unicast to the client.

Reverse Address Resolution Protocol (RARP)

The Reverse Address Resolution Protocol (RARP) is a networking protocol that is used to map a physical (MAC) address to an Internet Protocol (IP) address. It is the reverse of the more commonly used Address Resolution Protocol (ARP), which maps an IP address to a MAC address.

RARP was developed in the early days of computer networking as a way to provide IP addresses to diskless workstations or other devices that could not store their own IP addresses. With RARP, the device would broadcast its MAC address and request an IP address, and a RARP server on the network would respond with the corresponding IP address.

While RARP was widely used in the past, it has largely been replaced by newer protocols such as DHCP (Dynamic Host Configuration Protocol), which provides more flexibility and functionality in assigning IP addresses dynamically. However, RARP is still used in some specialized applications, such as booting embedded systems and configuring network devices with pre-assigned IP addresses.

RARP is specified in RFC 903 and operates at the data link layer of the OSI model. It has largely been superseded by ARP and DHCP in modern networks, but it played an important role in the development of computer networking protocols and continues to be used in certain contexts. RARP is abbreviation of Reverse Address Resolution Protocol which is a protocol based on computer networking which is employed by a client computer to request its IP address from a gateway server's Address Resolution Protocol table or cache. The network administrator creates a table in gateway-router, which is used to map the MAC address to corresponding IP address. This protocol is used to communicate data between two points in a server. The client doesn't necessarily need prior knowledge the server identities capable of serving its request. Media Access Control (MAC) addresses requires individual configuration on the servers done by an administrator. RARP limits to the serving of IP addresses only. When a replacement machine is set up, the machine may or might not have an attached disk that may permanently store the IP Address so the RARP client program requests IP Address from the RARP server on the router. The RARP server will return the IP address to the machine under the belief that an entry has been setup within the router table.

Working of RARP

The RARP is on the Network Access Layer and is employed to send data between two points in a very network. Each network participant has two unique addresses:- IP address (a logical address) and MAC address (the physical address). The IP address gets assigned by software and after that the MAC address is constructed into the hardware.

The RARP server that responds to RARP requests, can even be any normal computer within the network. However, it must hold the data of all the MAC addresses with their assigned IP addresses. If a RARP request is received by the network, only these RARP servers can reply to it. The info packet needs to be sent on very cheap layers of the network. This implies that the packet is transferred to all the participants at the identical time. The client broadcasts a RARP request with an Ethernet broadcast address and with its own physical address. The server responds by informing the client its IP address.

How is RARP different from ARP

RARP	ARP
A protocol used to map a physical (MAC) address to an IP address	A protocol used to map an IP address to a physical (MAC) address
To obtain the IP address of a network device when only its MAC address is known	To obtain the MAC address of a network device when only its IP address is known
Client broadcasts its MAC address and requests an IP address, and the server responds with the corresponding IP address	Client broadcasts its IP address and requests a MAC address, and the server responds with the corresponding MAC address
MAC addresses	IP addresses
Rarely used in modern networks as most devices have a pre-assigned IP address	Widely used in modern networks to resolve IP addresses to MAC addresses
RFC 903 Standardization	RFC 826 Standardization
RARP stands for Reverse Address Resolution Protocol	ARP stands for Address Resolution Protocol
In RARP, we find our own IP address	In ARP, we find the IP address of a remote machine
The MAC address is known and the IP address is requested	The IP address is known, and the MAC address is being requested
It uses the value 3 for requests and 4 for responses	It uses the value 1 for requests and 2 for responses

Uses of RARP :
RARP is used to convert the Ethernet address to an IP address. It is available for the LAN technologies like FDDI, token ring LANs, etc.

Disadvantages of RARP :
The Reverse Address Resolution Protocol had few disadvantages which eventually led to its replacement by BOOTP and DHCP. Some of the disadvantages are listed below:

- The RARP server must be located within the same physical network.
- The computer sends the RARP request on very cheap layer of the network. Thus, it's unattainable for a router to forward the packet because the computer sends the RARP request on very cheap layer of the network.
- The RARP cannot handle the subnetting process because no subnet masks are sent. If the network is split into multiple subnets, a RARP server must be available with each of them.
- It isn't possible to configure the PC in a very modern network.
- It doesn't fully utilize the potential of a network like Ethernet.RARP has now become an obsolete protocol since it operates at low level. Due to this, it requires direct address to the network which makes it difficult to build a server.

Internet Control Message Protocol (ICMP)

Internet Control Message Protocol (ICMP) works in the network layer of the OSI model and the internet layer of the TCP/IP model. It is used to send control messages to network devices and hosts. Routers and other network devices monitor the operation of the network. When an error occurs, these devices send a message using ICMP. Messages that can be sent include "destination unreachable", "time exceeded", and "echo requests".

ICMP is a network layer protocol.

ICMP messages are not passed directly to the data link layer. The message is first encapsulated inside the IP datagram before going to the lower layer.

Types of ICMP messages

Information Messages − In this message, the sender sends a query to the host or router and expects an answer. For example, A host wants to know if a router is alive or not.

Error-reporting message − This message report problems that a router or a host (destination) may encounter when it processes an IP packet.

Query Message − It helps a router or a network manager to get specific information from a router or another host.

Category	Type	Message
Error-Reporting Messages	3	Destination unreachable
	4	Source quench
	11	Time Exceeded
	12	Parameter Problem
	5	Redirection
Query Message	8 or 0	Echo request or reply
	13 or 14	Timestamp request or reply
	17 or 18	Address mask request or reply
	10 or 9	Router Solicitation or advertisement

Source Quench − It requests to decrease the traffic rate of message sending from source to destination.

Time Exceeded − When fragments are lost in a network the fragments hold by the router will be dropped and then ICMP will take the source IP from the discarded packet and inform the source, that datagram is discarded due to the time to live field reaches zero, by sending time exceeded message.

Fragmentation Required − When a router is unable to forward a datagram because it exceeds the MTU of the next-hop network and the DF (Don't Fragment) bit is set, the router is required to return an ICMP Destination Unreachable message to the source of the datagram, with the Code indicating fragmentation is needed and DF (Don’t Fragment) set.

Destination Unreachable − This error message indicates that the destination host, network, or port number that is specified in the IP packet is unreachable. This may happen due to the destination host device is down, an intermediate router is unable to find a path to forward the packet, and a firewall is configured to block connections from the source of the packet.

Redirect Message − A redirect error message is used when a router needs to tell a sender that it should use a different path for a specific destination. It occurs when the router knows a shorter path to the destination.

ICMP Basic Error Message Format

A basic ICMP error message would have the following format −

Type − The type field identifies the type of the message.

Code − The code field in ICMP describes the purpose of the message.

Checksum − The checksum field is used to validate ICMP messages.