1. Explain broadcast routing algorithm.

Sending a packet to all destinations simultaneously is called broadcasting. In some applications, hosts need to send messages to many or all other hosts. For example, a service distributing weather reports, stock market updates, or live radio programs might work best by broadcasting to all machines and letting those that are interested read the data. Sending a packet to all destinations simultaneously is called broadcasting; various methods have been proposed for doing it.

One broadcasting method that requires no special features from the subnet is for the source to simply send a distinct packet to each destination. Not only is the method wasteful of bandwidth, but it also requires the source to have a complete list of all destinations.

Flooding is another obvious candidate. Although flooding is ill-suited for ordinary point-to-point communication, for broadcasting it might rate serious consideration, especially if none of the methods described below are applicable. The problem with flooding as a broadcast technique is the same problem it has as a point-to-point routing algorithm: it generates too many packets and consumes too much bandwidth.

A third algorithm is multidestination routing. If this method is used, each packet contains either a list of destinations or a bit map indicating the desired destinations. When a packet arrives at a router, the router checks all the destinations to determine the set of output lines that will be needed. (An output line is needed if it is the best route to at least one of the destinations.) The router generates a new copy of the packet for each output line to be used and includes in each packet only those destinations that are to use the line. In effect, the destination set is partitioned among the output lines. After a sufficient number of hops, each packet will carry only one destination and can be treated as a normal packet. Multidestination routing is like separately addressed packets, except that when several packets must follow the same route, one of them pays full fare and the rest ride free.

A fourth broadcast algorithm makes explicit use of the sink tree for the router initiating the broadcast—or any other convenient spanning tree for that matter. A spanning tree is a subset of the subnet that includes all the routers but contains no loops. If each router knows which of its lines belong to the spanning tree, it can copy an incoming broadcast packet onto all the spanning tree lines except the one it arrived on. This method makes excellent use of bandwidth, generating the absolute minimum number of packets necessary to do the job. The only problem is that each router must have knowledge of some spanning tree for the method to be applicable. Sometimes this information is available (e.g., with link state routing) but sometimes it is not (e.g., with distance vector routing).

The last broadcast algorithm is an attempt to approximate the behaviour of the previous one, even when the routers do not know anything at all about spanning trees. The idea, called reverse path forwarding, is remarkably simple once it has been pointed out. When a broadcast packet arrives at a router, the router checks to see if the packet arrived on the line that is normally used for sending packets to the source of the broadcast or not.

If so, there is an excellent chance that the broadcast packet itself followed the best route from the router and is therefore the first copy to arrive at the router. This being the case, the router forwards copies of it onto all lines except the one it arrived on. If, however, the broadcast packet arrived on a line other than the preferred one for reaching the source, the packet is discarded as a likely duplicate.

2. Explain multicast routing algorithm.

Sending a message to multiple receivers with a single send operation is called multicasting. Some applications require that widely-separated processes work together in groups, for example, a group of processes implementing a distributed database system. In these situations, it is frequently necessary for one process to send a message to all the other members of the group. If the group is small, it can just send each other member a point-to-point message. If the group is large, this strategy is expensive. Sometimes broadcasting can be used, but using broadcasting to inform 1000 machines on a million-node network is inefficient because most receivers are not interested in the message Sending a message to such a group is called multicasting, and its routing algorithm is called multicast routing.

Multicasting requires group management. Some way is needed to create and destroy groups, and to allow processes to join and leave groups. It is important that routers know which of their hosts belong to which groups. Either hosts must inform their routers about changes in group membership, or routers must query their hosts periodically. Either way, routers learn about which of their hosts are in which groups. Routers tell their neighbours, so the information propagates through the subnet.

To do multicast routing, each router computes a spanning tree covering all other routers. For example, in Fig. 2(a) there are two groups, 1 and 2. Some routers are attached to hosts that belong to one or both of these groups, as indicated in the figure. A spanning tree for the leftmost router is shown in Fig. 2(b).

When a process sends a multicast packet to a group, the first router examines its spanning tree and prunes it, removing all lines that do not lead to hosts that are members of the group. In our example, Fig. 2(c) shows the pruned spanning tree for group 1. Similarly, Fig. 2(d) shows the pruned spanning tree for group 2. Multicast packets are forwarded only along the appropriate spanning tree.

Various ways of pruning the spanning tree are possible. The simplest one can be used if link state routing is used and each router is aware of the complete topology, including which hosts belong to which groups. Then the spanning tree can be pruned, starting at the end of each path, working toward the root, and removing all routers that do not belong to the group in question.

With distance vector routing, a different pruning strategy can be followed. The basic algorithm is reverse path forwarding. However, whenever a router with no hosts interested in a particular group and no connections to other routers receives a multicast message for that group, it responds with a PRUNE message, telling the sender not to send it any more

multicasts for that group. When a router with no group members among its own hosts has received such messages on all its lines, it, too, can respond with a PRUNE message. In this way, the subnet is recursively pruned.

One potential disadvantage of this algorithm is that it scales poorly to large networks. Suppose that a network has n groups, each with an average of m members. For each group, m pruned spanning trees must be stored, for a total of mn trees.

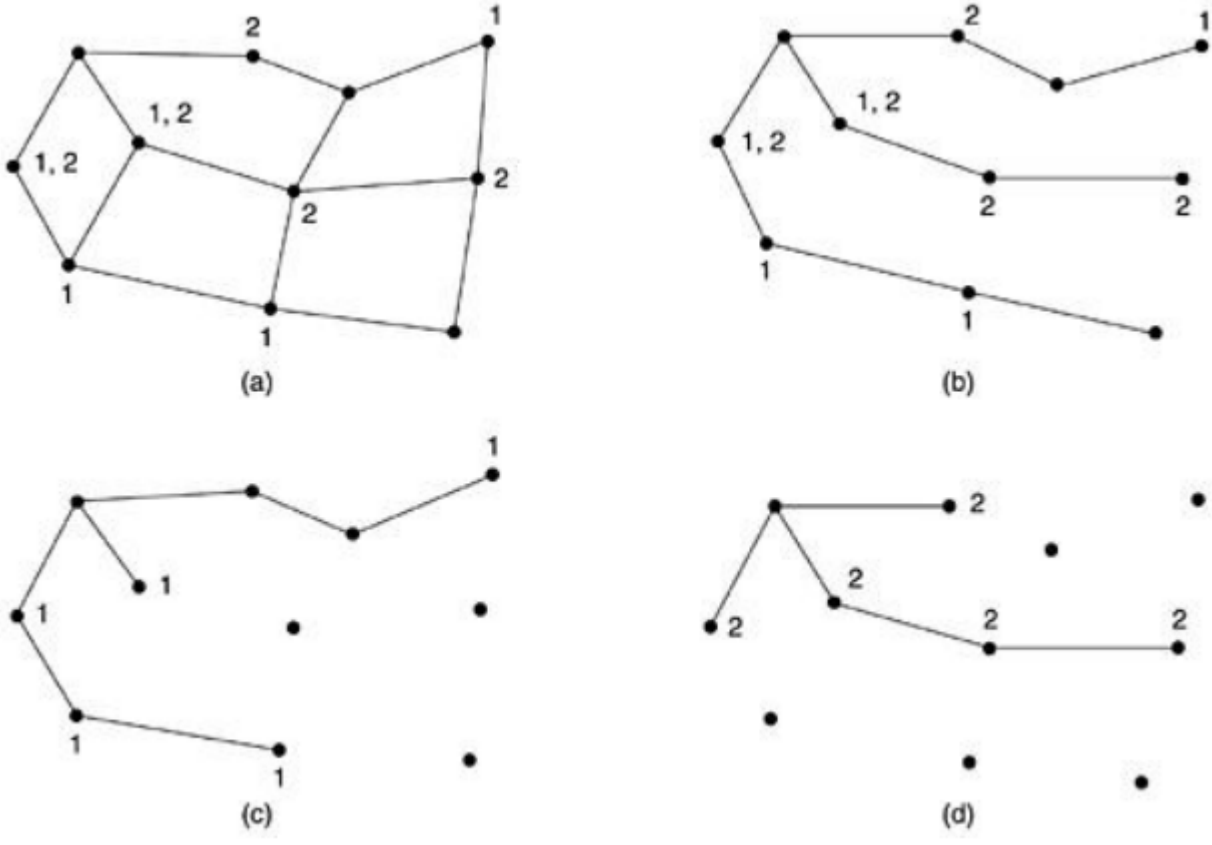


Fig.2 (a) A network. (b) A spanning tree for the leftmost router. (c) A multicast tree for group 1. (d) A multicast tree for group 2.

3. What is congestion.

Congestion is the state in which network performance decreases. When too many packets are present in (a part of) the subnet, performance degrades. This situation is called congestion. Figure 3 depicts the symptom. When the number of packets dumped into the subnet by the hosts is within its carrying capacity, they are all delivered (except for a few that are afflicted with transmission errors) and the number delivered is proportional to the number sent. However, as traffic increases too far, the routers are no longer able to cope and they begin losing packets. This tends to make matters worse. At very high traffic, performance collapses completely and almost no packets are delivered.

Congestion can be brought on by several factors. If all of a sudden, streams of packets begin arriving on three or four input lines and all need the same output line, a queue will

build up. If there is insufficient memory to hold all of them, packets will be lost. Adding more memory may help up to a point, but Nagle (1987) discovered that if routers have an infinite amount of memory, congestion gets worse, not better, because by the time packets get to the front of the queue, they have already timed out (repeatedly) and duplicates have been sent. All these packets will be dutifully forwarded to the next router, increasing the load all the way to the destination.

Slow processors can also cause congestion. If the routers' CPUs are slow at performing the bookkeeping tasks required of them (queueing buffers, updating tables, etc.), queues can build up, even though there is excess line capacity. Similarly, low-bandwidth lines can also cause congestion.

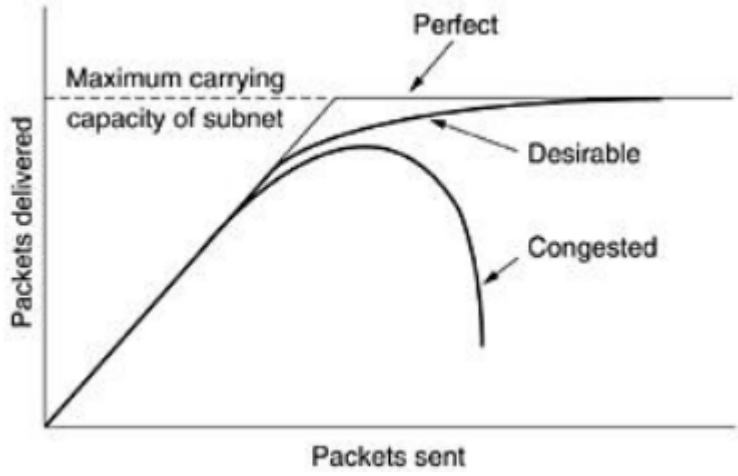


Fig.3 When too much traffic is offered, congestion sets in and performance degrades sharply

4. What are the congestion prevention policies.

The open loop systems are designed to minimize congestion in the first place, rather than letting it happen and reacting after the fact. They try to achieve their goal by using appropriate policies at various levels. In Fig.4 there are different data link, network, and transport policies that can affect congestion (Jain, 1990).

The retransmission policy is concerned with how fast a sender times out and what it transmits upon timeout. A jumpy sender that times out quickly and retransmits all outstanding packets using go back n will put a heavier load on the system than will a leisurely sender that uses selective repeat. Closely related to this is the buffering policy. If receivers routinely discard all out-of-order packets, these packets will have to be transmitted again later, creating extra load.

Acknowledgement policy also affects congestion. If each packet is acknowledged immediately, the acknowledgement packets generate extra traffic. However, if acknowledgements are saved up to piggyback onto reverse traffic, extra timeouts and

retransmissions may result. A tight flow control scheme (e.g., a small window) reduces the data rate and thus helps fight congestion.

At the network layer, the choice between using virtual circuits and using datagrams affects congestion since many congestion control algorithms work only with virtual-circuit subnets. Packet queueing and service policy relates to whether routers have one queue per input line, one queue per output line, or both. It also relates to the order in which packets are processed (e.g., round robin or priority based). Discard policy is the rule telling which packet is dropped when there is no space. A good policy can help alleviate congestion and a bad one can make it worse.

Layer	Policies
Transport	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy • Timeout determination
Network	• Virtual circuits versus datagram inside the subnet • Packet queueing and service policy • Packet discard policy • Routing algorithm • Packet lifetime management
Data link	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy

Fig. 4 Policies that affect congestion.

A good routing algorithm can help avoid congestion by spreading the traffic over all the lines whereas a bad one can send too much traffic over already congested lines. Finally, packet lifetime management deals with how long a packet may live before being discarded. If it is too long, lost packets may clog up the works for a long time, but if it is too short, packets may sometimes time out before reaching their destination, thus inducing retransmissions.

In the transport layer, the same issues occur as in the data link layer, but in addition, determining the timeout interval is harder because the transit time across the network is less predictable than the transit time over a wire between two routers. If the timeout interval is too

short, extra packets will be sent unnecessarily. If it is too long, congestion will be reduced but the response time will suffer whenever a packet is lost.

5. What are the general principles of congestion control.

Many problems in complex systems, such as computer networks, can be viewed from a control theory point of view. This approach leads to dividing all solutions into two groups: open loop and closed loop. Open loop solutions attempt to solve the problem by good design, in essence, to make sure it does not occur in the first place. Once the system is up and running, midcourse corrections are not made.

Tools for doing open-loop control include deciding when to accept new traffic, deciding when to discard packets and which ones, and making scheduling decisions at various points in the network. All of these have in common the fact that they make decisions without regard to the current state of the network.

In contrast, closed loop solutions are based on the concept of a feedback loop. This approach has three parts when applied to congestion control:

1. Monitor the system to detect when and where congestion occurs.
2. Pass this information to places where action can be taken.
3. Adjust system operation to correct the problem.

A variety of metrics can be used to monitor the subnet for congestion. Chief among these are the percentage of all packets discarded for lack of buffer space, the average queue lengths, the number of packets that time out and are retransmitted, the average packet delay, and the standard deviation of packet delay. In all cases, rising numbers indicate growing congestion.

The second step in the feedback loop is to transfer the information about the congestion from the point where it is detected to the point where something can be done about it. The obvious way is for the router detecting the congestion to send a packet to the traffic source or source announcing the problem.

6. Explain leaky bucket algorithm for traffic shaping.

The leaky bucket implementation is used to control the rate at which traffic is sent to the network. Leaky bucket implementation is same as bucket having a hole at the bottom. Imagine a bucket with a small hole in the bottom, as illustrated in Fig.6(a). No matter the rate at which water enters the bucket, the outflow is at a constant rate, r, when there is any water in the bucket and zero when the bucket is empty. Also, once the bucket is full, any additional water entering it spills over the sides and is lost (i.e., does not appear in the output stream under the hole).

The same idea can be applied to packets, as shown in Fig. 6(b). Conceptually, each host is connected to the network by an interface containing a leaky bucket, that is, a finite internal queue. If a packet arrives at the queue when it is full, the packet is discarded. In other words, if one or more processes within the host try to send a packet when the maximum number is already queued, the new packet is unceremoniously discarded. This arrangement can be built into the hardware interface or simulated by the host operating system. It was first proposed by Turner (1986) and is called the leaky bucket algorithm. In fact, it is nothing other than a single-server queueing system with constant service time.

The host is allowed to put one packet per clock tick onto the network. Again, this can be enforced by the interface card or by the operating system. This mechanism turns an uneven flow of packets from the user processes inside the host into an even flow of packets onto the network, smoothing out bursts and greatly reducing the chances of congestion.

When the packets are all the same size(e.g., ATM cells), this algorithm can be used as described. However, when variable-sized packets are being used, it is often better to allow a fixed number of bytes per tick, rather than just one packet. Thus, if the rule is 1024 bytes per tick, a single 1024-byte packet can be admitted on a tick, two 512-byte packets, four 256-byte packets, and so on. If the residual byte count is too low, the next packet must wait until the next tick.

Implementing the original leaky bucket algorithm is easy. The leaky bucket consists of a finite queue. When a packet arrives, if there is room on the queue it is appended to the queue; otherwise, it is discarded. At every clock tick, one packet is transmitted (unless the queue is empty).

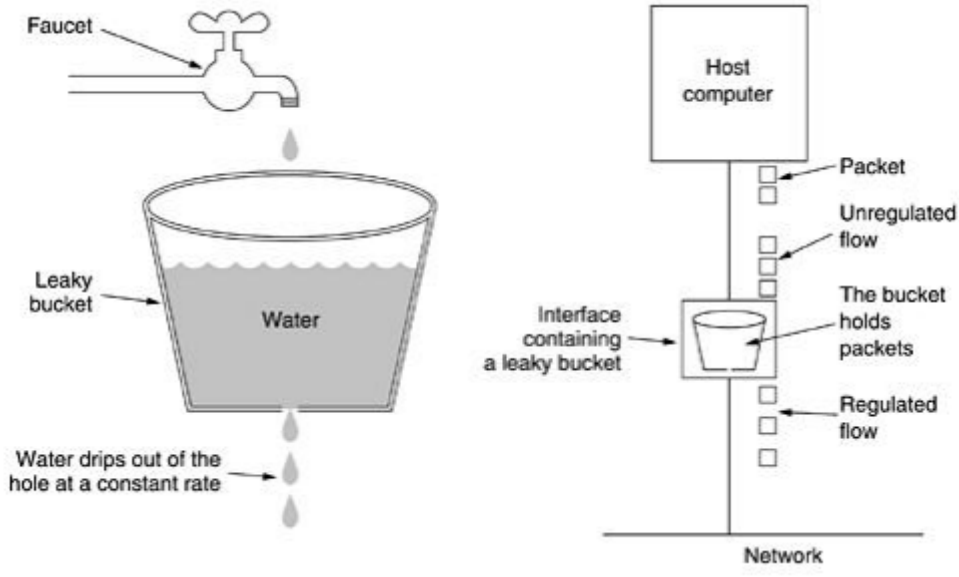


Fig 6. (a) A leaky bucket with water. (b) A leaky bucket with packets

The byte-counting leaky bucket is implemented almost the same way. At each tick, a counter is initialized to n. If the first packet on the queue has fewer bytes than the current

value of the counter, it is transmitted, and the counter is decremented by that number of bytes. Additional packets may also be sent, as long as the counter is high enough. When the counter drops below the length of the next packet on the queue, transmission stops until the next tick, at which time the residual byte count is reset and the flow can continue.

7. Explain RSVP protocol for congestion control.

RSVP stands for The Resource reSerVation Protocol. The main IETF protocol for the integrated services architecture is RSVP. This protocol is used for making the reservations; other protocols are used for sending the data. RSVP allows multiple senders to transmit to multiple groups of receivers, permits individual receivers to switch channels freely, and optimizes bandwidth use while at the same time eliminating congestion.

In its simplest form, the protocol uses multicast routing using spanning trees. Each group is assigned a group address. To send to a group, a sender puts the group's address in its packets. The standard multicast routing algorithm then builds a spanning tree covering all group members. The routing algorithm is not part of RSVP. The only difference from normal multicasting is a little extra information that is multicast to the group periodically to tell the routers along the tree to maintain certain data structures in their memories.

As an example, consider the network of Fig. 7(a). Hosts 1 and 2 are multicast senders, and hosts 3, 4, and 5 are multicast receivers. In this example, the senders and receivers are disjoint, but in general, the two sets may overlap. The multicast trees for hosts 1 and 2 are shown in Fig. 7(b) and Fig. 7(c), respectively.

To get better reception and eliminate congestion, any of the receivers in a group can send a reservation message up the tree to the sender. The message is propagated using the reverse path forwarding algorithm discussed earlier. At each hop, the router notes the reservation and reserves the necessary bandwidth. If insufficient bandwidth is available, it reports back failure. By the time the message gets back to the source, bandwidth has been reserved all the way from the sender to the receiver making the reservation request along the spanning tree.

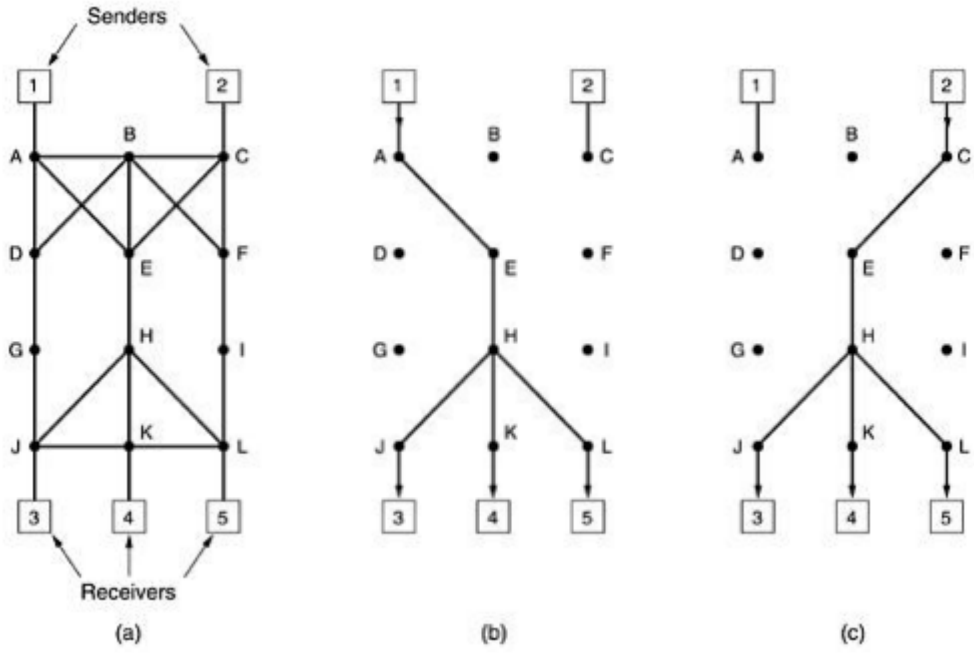


Fig 7. (a) A network. (b) The multicast spanning tree for host 1. (c) The multicast spanning tree for host 2.

8. Briefly discuss about concatenated virtual circuits.

(Or)

Briefly discuss about Connection oriented Internetworking

Two styles of internetworking are possible: a connection-oriented concatenation of virtual-circuit subnets, and a datagram internet style. In the past, most (public) networks were connection oriented (and frame relay, SNA, 802.16, and ATM still are). Then with the rapid acceptance of the Internet, datagrams became fashionable. With the growing importance of multimedia networking, it is likely that connection-orientation will make a come-back in one form or another since it is easier to guarantee quality of service with connections than without them.

In the concatenated virtual-circuit model, shown in Fig. 8, a connection to a host in a distant network is set up in a way similar to the way connections are normally established. The subnet sees that the destination is remote and builds a virtual circuit to the router nearest the destination network. Then it constructs a virtual circuit from that router to an external gateway(multiprotocol router). This gateway records the existence of the virtual circuit in its tables and proceeds to build another virtual circuit to a router in the next subnet. This process continues until the destination host has been reached.

Once data packets begin flowing along the path, each gateway relays incoming packets, converting between packet formats and virtual-circuit numbers as needed. Clearly, all data packets must traverse the same sequence of gateways. Consequently, packets in a flow are never reordered by the network.

The essential feature of this approach is that a sequence of virtual circuits is set up from the source through one or more gateways to the destination. Each gateway maintains tables telling which virtual circuits pass through it, where they are to be routed, and what the new virtual-circuit number is.

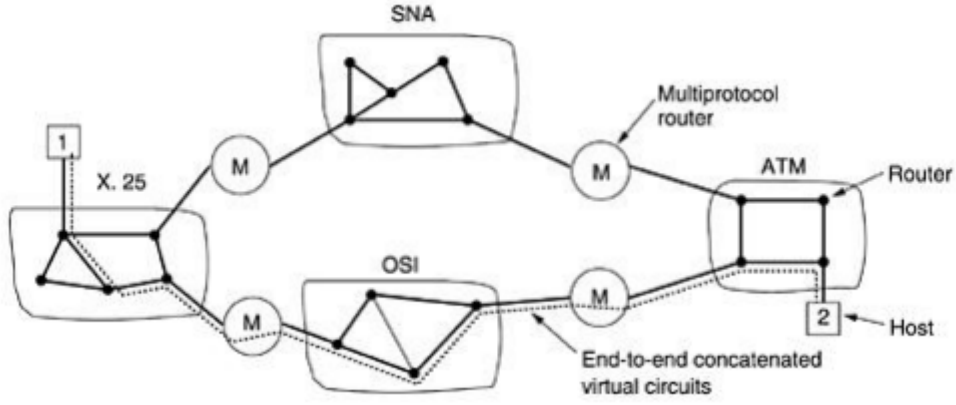


Fig 8. Internetworking using concatenated virtual circuits.

This scheme works best when all the networks have roughly the same properties. For example, if all of them guarantee reliable delivery of network layer packets, then barring a crash somewhere along the route, the flow from source to destination will also be reliable. Similarly, if none of them guarantee reliable delivery, then the concatenation of the virtual circuits is not reliable either. On the other hand, if the source machine is on a network that does guarantee reliable delivery but one of the intermediate networks can lose packets, the concatenation has fundamentally changed the nature of the service.

Concatenated virtual circuits are also common in the transport layer. In particular, it is possible to build a bit pipe using, say, SNA, which terminates in a gateway, and have a TCP connection go from the gateway to the next gateway. In this manner, an end-to-end virtual circuit can be built spanning different networks and protocols.

9. Briefly discuss about Connectionless Internetworking.

The datagram model is shown in Fig. 9. In this model, the only service the network layer offers to the transport layer is the ability to inject datagrams into the subnet and hope for the best. There is no notion of a virtual circuit at all in the network layer, let alone a concatenation of them. This model does not require all packets belonging to one connection to traverse the same sequence of gateways. In Fig. 9 datagrams from host 1 to host 2 are

shown taking different routes through the internetwork. A routing decision is made separately for each packet, possibly depending on the traffic at the moment the packet is sent. This strategy can use multiple routes and thus achieve a higher bandwidth than the concatenated virtual-circuit model. On the other hand, there is no guarantee that the packets arrive at the destination in order, assuming that they arrive at all.

The model of Fig. 9 is not quite as simple as it looks. For one thing, if each network has its own network layer protocol, it is not possible for a packet from one network to transit another one. One could imagine the multiprotocol routers actually trying to translate from one format to another, but unless the two formats are close relatives with the same information fields, such conversions will always be incomplete and often doomed to failure. For this reason, conversion is rarely attempted.

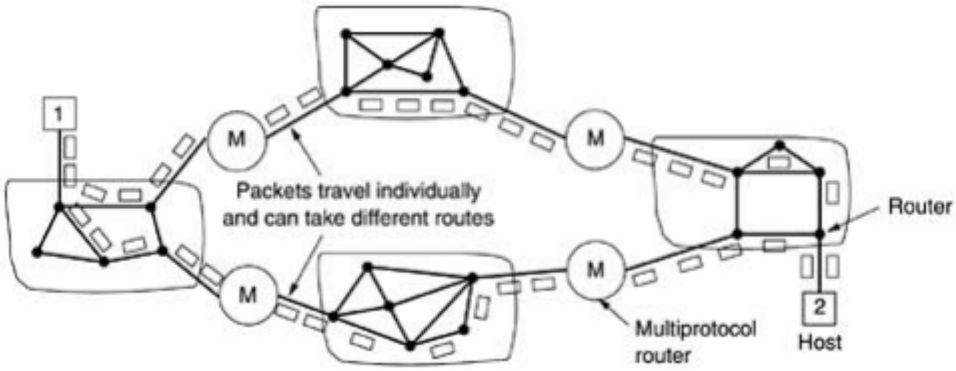


Fig 9. A connectionless internet.

A second, and more serious, problem is addressing. Imagine a simple case: a host on the Internet is trying to send an IP packet to a host on an adjoining SNA network. The IP and SNA addresses are different. One would need a mapping between IP and SNA addresses in both directions. Furthermore, the concept of what is addressable is different. In IP, hosts (actually, interface cards) have addresses. In SNA, entities other than hosts (e.g., hardware devices) can also have addresses. At best, someone would have to maintain a database mapping everything to everything to the extent possible, but it would constantly be a source of trouble.

Another idea is to design a universal "internet" packet and have all routers recognize it. This approach is, in fact, what IP is—a packet designed to be carried through many networks. Of course, it may turn out that IPv4 (the current Internet protocol) drives all other formats out of the market, IPv6 (the future Internet protocol) does not catch on, and nothing new is ever invented, but history suggests otherwise. Getting everybody to agree to a single format is difficult when companies perceive it to their commercial advantage to have a proprietary format that they control.

10. What is Tunneling?

Tunneling is a method of transmitting data that is intended for use only within a private network through a public network in such a way that the routing nodes in the public network are unaware that the transmission is a part of private network.

Handling the general case of making two different networks interwork is exceedingly difficult. However, there is a common special case that is manageable. 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 a TCP/IP-based Ethernet in Paris, a TCP/IP-based Ethernet in London, and a non-IP wide area network (e.g., ATM) in between, as shown in Fig. 10.1.

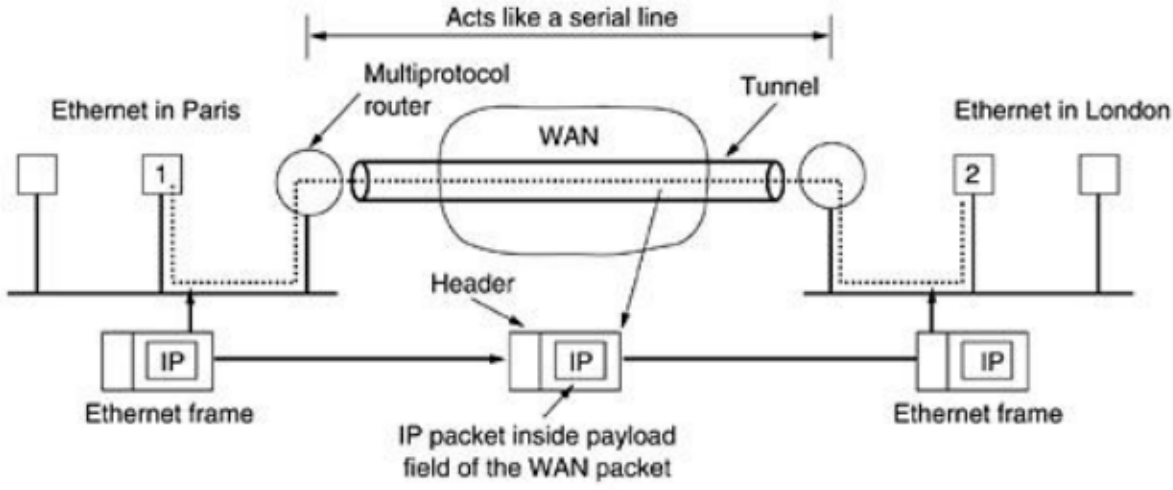


Fig 10.1. Tunneling a packet from Paris to London.

The solution to this problem is a technique called tunneling. To send an IP packet to host 2, host 1 constructs the packet containing the IP address of host 2, inserts it into an Ethernet frame addressed to the Paris multiprotocol router, and puts it on the Ethernet. When the multiprotocol router gets the frame, it removes the IP packet, inserts it in the payload field of the WAN network layer packet, and addresses the latter to the WAN address of the London multiprotocol router. When it gets there, the London router removes the IP packet and sends it to host 2 inside an Ethernet frame.

The WAN can be seen as a big tunnel extending from one multiprotocol router to the other. The IP 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 the WAN at all. Neither do the hosts on either Ethernet. Only the multiprotocol router has to understand IP and WAN packets. In effect, the entire distance from the middle of one multiprotocol router to the middle of the other acts like a serial line. 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 into 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. 10.2. 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.

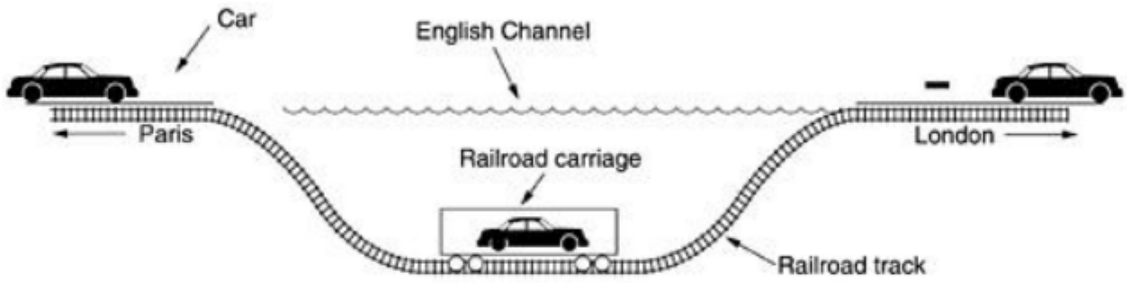


Fig 10.2. Tunneling a car from France to England.

11. Briefly discuss about Internetwork routing.

Routing through an internetwork is similar to routing within a single subnet, but with some added complications. Consider, for example, the internetwork of Fig. 11(a) in which five networks are connected by six (possibly multiprotocol) routers. Making a graph model of this situation is complicated by the fact that every router can directly access (i.e., send packets to) every other router connected to any network to which it is connected. For example, B in Fig. 11(a) can directly access A and C via network 2 and also D via network 3. This leads to the graph of Fig. 11(b).

Once the graph has been constructed, known routing algorithms, such as the distance vector and link state algorithms, can be applied to the set of multiprotocol routers. This gives a two-level routing algorithm: within each network an interior gateway protocol is used, but between the networks, an exterior gateway protocol is used ("gateway" is an older term for "router"). In fact, since each network is independent, they may all use different algorithms. Because each network in an internetwork is independent of all the others, it is often referred to as an Autonomous System (AS). A typical internet packet starts out on its LAN addressed to the local multiprotocol router (in the MAC layer header). After it gets there, the network layer code decides which multiprotocol router to forward the packet to, using its own routing tables. If that router can be reached using the packet's native network protocol, the packet is forwarded there directly. Otherwise it is tunneled there, encapsulated in the protocol required by the intervening network. This process is repeated until the packet reaches the destination network.

One of the differences between internetwork routing and intranetwork routing is that internetwork routing may require crossing international boundaries. Various laws suddenly come into play, such as Sweden's strict privacy laws about exporting personal data about Swedish citizens from Sweden. Another example is the Canadian law saying that data traffic originating in Canada and ending in Canada may not leave the country. This law means that

traffic from Windsor, Ontario to Vancouver may not be routed via nearby Detroit, even if that route is the fastest and cheapest.

Another difference between interior and exterior routing is the cost. Within a single network, a single charging algorithm normally applies. However, different networks may be under different managements, and one route may be less expensive than another. Similarly, the quality of service offered by different networks may be different, and this may be a reason to choose one route over another.

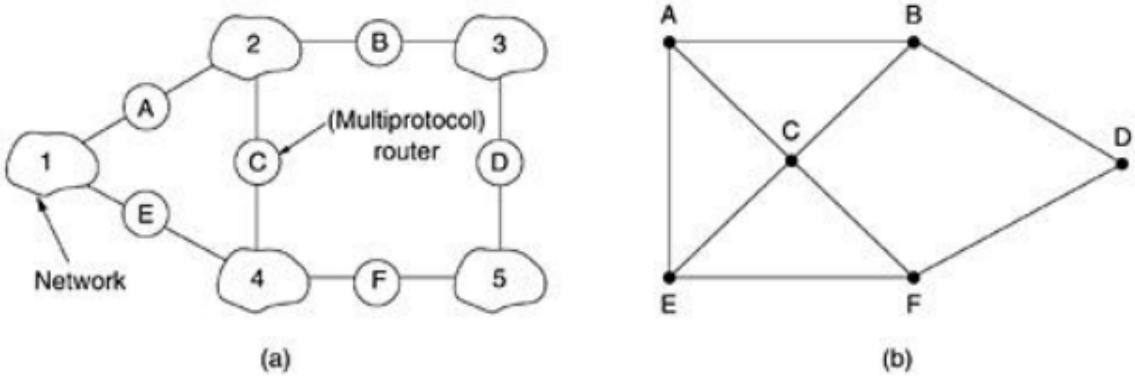


Fig.11 (a) An internetwork. (b) A graph of the internetwork.

12. What are the principles that are to be followed before designing the network layer in the Internet.

The principles that are to be followed before designing the network layer in the Internet are,

1**. Make sure it works**. Do not finalize the design or standard until multiple prototypes have successfully communicated with each other.

2. Keep it simple. The design must be as simple as possible by removing the unnecessary components from the layer

3. Make clear choices. If there are several ways of doing the same thing, choose one. Having two or more ways to do the same thing is looking for trouble. Standards often have multiple options or modes or parameters because several powerful parties insist that their way is best.

4. Exploit modularity. This principle leads directly to the idea of having protocol stacks, each of whose layers is independent of all the other ones. In this way, if circumstances that require one module or layer to be changed, the other ones will not be affected.

5. Expect heterogeneity. Different types of hardware, transmission facilities, and applications will occur on any large network. To handle them, the network design must be simple, general, and flexible.

6. Avoid static options and parameters. If parameters are unavoidable (e.g., maximum packet size), it is best to have the sender and receiver negotiate a value than defining fixed choices.

7. Look for a good design; it need not be perfect. Often the designers have a good design but it cannot handle some weird special case. Rather than messing up the design, the designers should go with the good design and put the burden of working around it on the people with the strange requirements.

8. Be strict when sending and tolerant when receiving. In other words, only send packets that rigorously comply with the standards, but expect incoming packets that may not be fully conformant and try to deal with them.

9. Think about scalability. If the system is to handle millions of hosts and billions of users effectively, no centralized databases of any kind are tolerable and load must be spread as evenly as possible over the available resources.

10. Consider performance and cost. If a network has poor performance or outrageous costs, nobody will use it.

13. Explain IP protocol.

IP protocol is a protocol of network layer whose main objective is to support internetworking. An IP datagram consists of a header part and a text part. The header has a 20-byte fixed part and a variable length optional part. The header format is shown in Fig. 13. It is transmitted in big-endian order: from left to right, with the high-order bit of the Version field going first.

The Version field keeps track of which version of the protocol the datagram belongs to. By including the version in each datagram, it becomes possible to have the transition between versions take years, with some machines running the old version and others running the new one.

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 that option useless.

The Type of service field is one of the few fields that has changed its meaning (slightly) over the years. It was and is still 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.

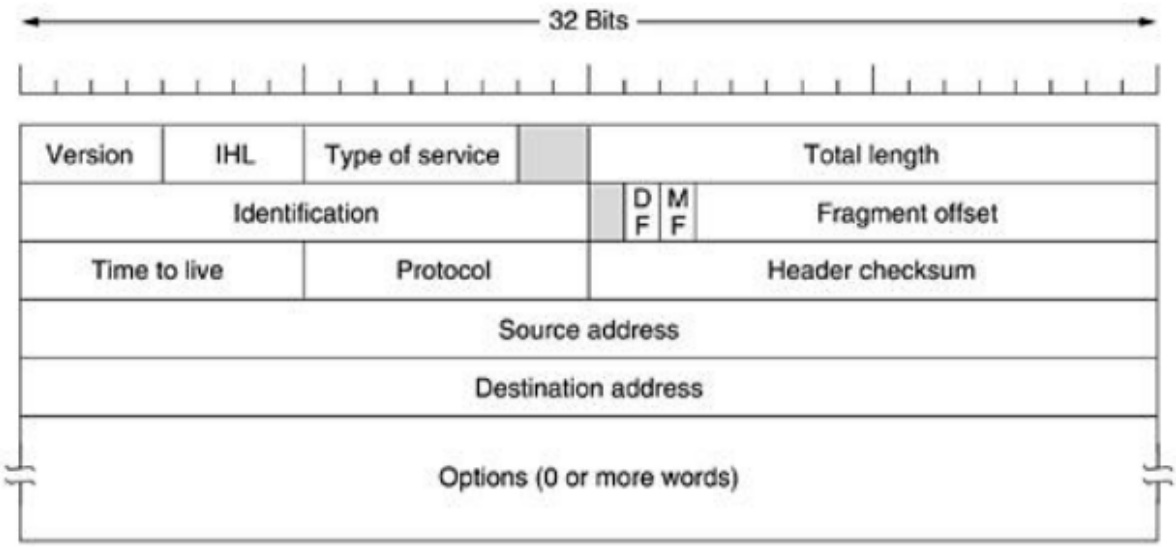


Fig13 The IPv4 (Internet Protocol) header.

Originally, the 6-bit field contained (from left to right), a three-bit Precedence field and three flags, D, T, and R. The Precedence field was a priority, from 0 (normal) to 7 (network control packet). The three flag bits allowed the host to specify what it cared most about from the set {Delay, Throughput, Reliability}. In theory, these fields allow routers to make choices between, for example, a satellite link with high throughput and high delay or a leased line with low throughput and low delay. In practice, current routers often ignore the Type of service field altogether.

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 gigabit networks, larger datagrams may be needed.

The Identification field is needed to allow the destination host to determine which datagram a newly arrived fragment belongs to. All the fragments of a datagram contain the same Identification value.

Next comes an unused bit and then two 1-bit fields. DF stands for Don't Fragment. It is an order to the routers not to fragment the datagram because the destination is incapable of putting the pieces back together again.

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 datagram 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, giving a maximum datagram length of 65,536 bytes, one more than the Total length field.

The Time to live field is a counter used to limit packet lifetimes. It is 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 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 datagrams from wandering around forever, something that otherwise might happen if the routing tables ever become corrupted.

When the network layer has assembled a complete datagram, it needs to know what to do with it. The Protocol field tells it which transport process to give it to. TCP is one possibility, but so are UDP and some others.

The Header checksum verifies the header only. Such a checksum is useful for detecting errors generated by bad memory words inside a router. The algorithm is to add up all the 16-bit halfwords 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. This algorithm is more robust than using a normal add.

The Source address and Destination address indicate the network number and host number. 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 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 four bytes.

14. Explain the ICMP protocol.

ICMP stands for Internet Control Message Protocol. The operation of the Internet is monitored closely by the routers. When something unexpected occurs, the event is reported by the ICMP, which is also used to test the Internet. About a dozen types of ICMP messages are defined. The most important ones are listed in Fig. 14. Each ICMP message type is encapsulated in an IP packet.

Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo	Ask a machine if it is alive
Echo reply	Yes, I am alive
Timestamp request	Same as Echo request, but with timestamp
Timestamp reply	Same as Echo reply, but with timestamp

Fig 14. The principal ICMP message types.

The DESTINATION UNREACHABLE message is used when the subnet or a router cannot locate the destination or when a packet with the DF bit cannot be delivered because a "small-packet" network stands in the way.

The TIME EXCEEDED message is sent when a packet is dropped because its counter has reached zero. This event is a symptom that packets are looping, that there is enormous congestion, or that the timer values are being set too low.

The PARAMETER PROBLEM message indicates that an illegal value has been detected in a header field. This problem indicates a bug in the sending host's IP software or possibly in the software of a router transited.

The SOURCE QUENCH message was formerly used to throttle hosts that were sending too many packets. When a host received this message, it was expected to slow down.

The REDIRECT message is used when a router notices that a packet seems to be routed wrong. It is used by the router to tell the sending host about the probable error.

The ECHO and ECHO REPLY messages are used to see if a given destination is reachable and alive. Upon receiving the ECHO message, the destination is expected to send an ECHO REPLY message back. The TIMESTAMP REQUEST and TIMESTAMP REPLY messages are similar, except that the arrival time of the message and the departure time of the reply are recorded in the reply. This facility is used to measure network performance.

15. Explain the RARP, BOOTP, and DHCP protocol.

ARP solves the problem of finding out which Ethernet address corresponds to a given IP address. Sometimes the reverse problem has to be solved: Given an Ethernet address, what is the corresponding IP address? In particular, this problem occurs when a diskless

workstation is booted. Such a machine will normally get the binary image of its operating system from a remote file server. But how does it learn its IP address?

The first solution devised was to use RARP (Reverse Address Resolution Protocol) (defined in RFC 903). This protocol allows a newly-booted workstation to broadcast its Ethernet address and say: My 48-bit Ethernet address is 14.04.05.18.01.25. Does anyone out there know my IP address? The RARP server sees this request, looks up the Ethernet address in its configuration files, and sends back the corresponding IP address.

Using RARP is better than embedding an IP address in the memory image because it allows the same image to be used on all machines. If the IP address were buried inside the image, each workstation would need its own image.

A disadvantage of RARP is that it uses a destination address of all 1s (limited broadcasting) to reach the RARP server. However, such broadcasts are not forwarded by routers, so a RARP server is needed on each network. To get around this problem, an alternative bootstrap protocol called BOOTP was invented. Unlike RARP, BOOTP uses UDP messages, which are forwarded over routers. It also provides a diskless workstation with additional information, including the IP address of the file server holding the memory image, the IP address of the default router, and the subnet mask to use. BOOTP is described in RFCs 951, 1048, and 1084.

A serious problem with BOOTP is that it requires manual configuration of tables mapping IP address to Ethernet address. When a new host is added to a LAN, it cannot use BOOTP until an administrator has assigned it an IP address and entered its (Ethernet address, IP address) into the BOOTP configuration tables by hand. To eliminate this error-prone step, BOOTP was extended and given a new name: DHCP (Dynamic Host Configuration Protocol). DHCP allows both manual IP address assignment and automatic assignment. It is described in RFCs 2131 and 2132. In most systems, it has largely replaced RARP and BOOTP.

Like RARP and BOOTP, DHCP is based on the idea of a special server that assigns IP addresses to hosts asking for one. This server need not be on the same LAN as the requesting host. Since the DHCP server may not be reachable by broadcasting, a DHCP relay agent is needed on each LAN, as shown in Fig. 15

To find its IP address, a newly-booted machine broadcasts a DHCP DISCOVER packet. The DHCP relay agent on its LAN intercepts all DHCP broadcasts. When it finds a DHCP DISCOVER packet, it sends the packet as a unicast packet to the DHCP server, possibly on a distant network. The only piece of information the relay agent needs is the IP address of the DHCP server.

An issue that arises with automatic assignment of IP addresses from a pool is how long an IP address should be allocated. If a host leaves the network and does not return its IP address to the DHCP server, that address will be permanently lost. After a period of time,

many addresses may be lost. To prevent that from happening, IP address assignment may be for a fixed period of time, a technique called leasing. Just before the lease expires, the host must ask the DHCP for a renewal. If it fails to make a request or the request is denied, the host may no longer use the IP address it was given earlier.

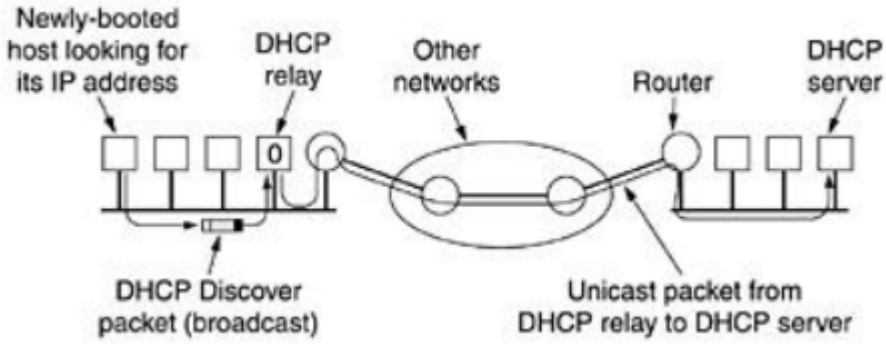


Fig 15. Operation of DHCP

16. What is Fragmentation? Explain different types of Fragmentation.

Each network 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 maximum packet size they wish. Maximum payloads range from 48 bytes (ATM cells) to 65,515 bytes (IP packets), although the payload size in higher layers is often larger.

An obvious problem appears when a large packet wants to travel through a network whose maximum packet size is too small. One solution is to make sure the problem does not occur in the first place. In other words, the internet should use a routing algorithm that avoids sending packets through networks that cannot handle them. However, this solution is no solution at all.

Basically, the only solution to the problem is to allow gateways to break up packets into fragments, sending each fragment as a separate internet packet. However, as every parent of a small child knows, converting a large object into small fragments is considerably easier

than the reverse process. 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. 16(a). In this approach, the small-packet network has gateways (most likely, specialized routers) that interface to other networks. When an oversized packet arrives at a gateway, the gateway breaks it up into fragments. Each fragment is addressed to the same exit gateway, where the pieces are recombined. In this way passage through the small-packet network has been made transparent. Subsequent networks are not even aware that fragmentation has occurred. ATM networks, for example, have special hardware to provide transparent fragmentation of packets into cells and then reassembly of cells into packets. In the ATM world, fragmentation is called segmentation; the concept is the same, but some of the details are different.

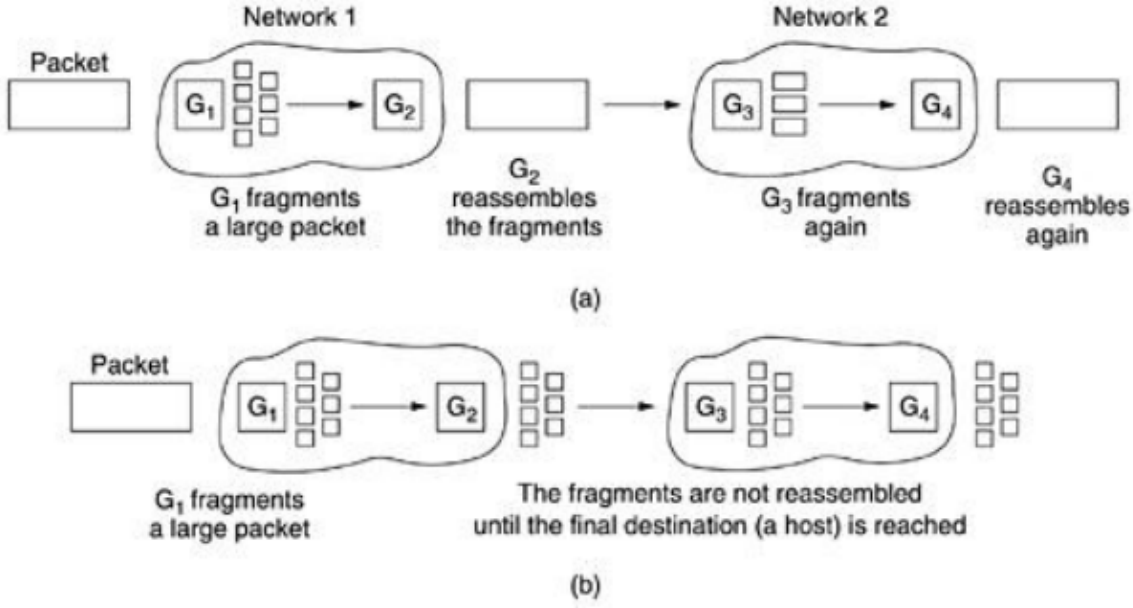


Fig 16. (a) Transparent fragmentation. (b) Nontransparent fragmentation

Transparent fragmentation is straightforward but has some problems. For one thing, the exit gateway must know when it has received all the pieces, so either a count field or an "end of packet" bit must be provided. For another thing, all packets must exit via the same gateway. By not allowing some fragments to follow one route to the ultimate destination and other fragments a disjoint route, some performance may be lost. ATM requires transparent fragmentation.

The other fragmentation strategy is to refrain from recombining fragments at any intermediate gateways. Once a packet has been fragmented, each fragment is treated as

though it were an original packet. All fragments are passed through the exit gateway (or gateways), as shown in Fig. 16(b). Recombination occurs only at the destination host. IP works this way.

Nontransparent fragmentation also has some problems. For example, it requires every host to be able to do reassembly. Yet another problem is that when a large packet is fragmented, the total overhead increases because each fragment must have a header. Whereas in the first method this overhead disappears as soon as the small-packet network is exited, in this method the overhead remains for the rest of the journey. An advantage of nontransparent fragmentation, however, is that multiple exit gateways can now be used and higher performance can be achieved. Of course, if the concatenated virtual-circuit model is being used, this advantage is of no use.

When a packet is fragmented, the fragments must be numbered in such a way that the original data stream can be reconstructed. One way of numbering the fragments is to use a tree. If packet 0 must be split up, the pieces are called 0.0, 0.1, 0.2, etc. If these fragments themselves must be fragmented later on, the pieces are numbered 0.0.0, 0.0.1, 0.0.2, . . . , 0.1.0, 0.1.1, 0.1.2, etc. If enough fields have been reserved in the header for the worst case and no duplicates are generated anywhere, this scheme is sufficient to ensure that all the pieces can be correctly reassembled at the destination, no matter what order they arrive in.

However, if even one network loses or discards packets, end-to-end retransmissions are needed, with unfortunate effects for the numbering system. Suppose that a 1024-bit packet is initially fragmented into four equal-sized fragments, 0.0, 0.1, 0.2, and 0.3. Fragment 0.1 is lost, but the other parts arrive at the destination. Eventually, the source times out and retransmits the original packet again. Only this time Murphy's law strikes and the route taken passes through a network with a 512-bit limit, so two fragments are generated. When the new fragment 0.1 arrives at the destination, the receiver will think that all four pieces are now accounted for and reconstruct the packet incorrectly.

A completely different (and better) numbering system is for the internetwork protocol to define an elementary fragment size small enough that the elementary fragment can pass through every network. When a packet is fragmented, all the pieces are equal to the elementary fragment size except the last one, which may be shorter. An internet packet may contain several fragments, for efficiency reasons. The internet header must provide the original packet number and the number of the (first) elementary fragment contained in the packet. As usual, there must also be a bit indicating that the last elementary fragment contained within the internet packet is the last one of the original packet.

This approach requires two sequence fields in the internet header: the original packet number and the fragment number. There is clearly a trade-off between the size of the elementary fragment and the number of bits in the fragment number. Because the elementary fragment size is presumed to be acceptable to every network, subsequent fragmentation of an internet packet containing several fragments causes no problem. The ultimate limit here is to

have the elementary fragment be a single bit or byte, with the fragment number then being the bit or byte offset within the original packet, as shown in Fig.16.1

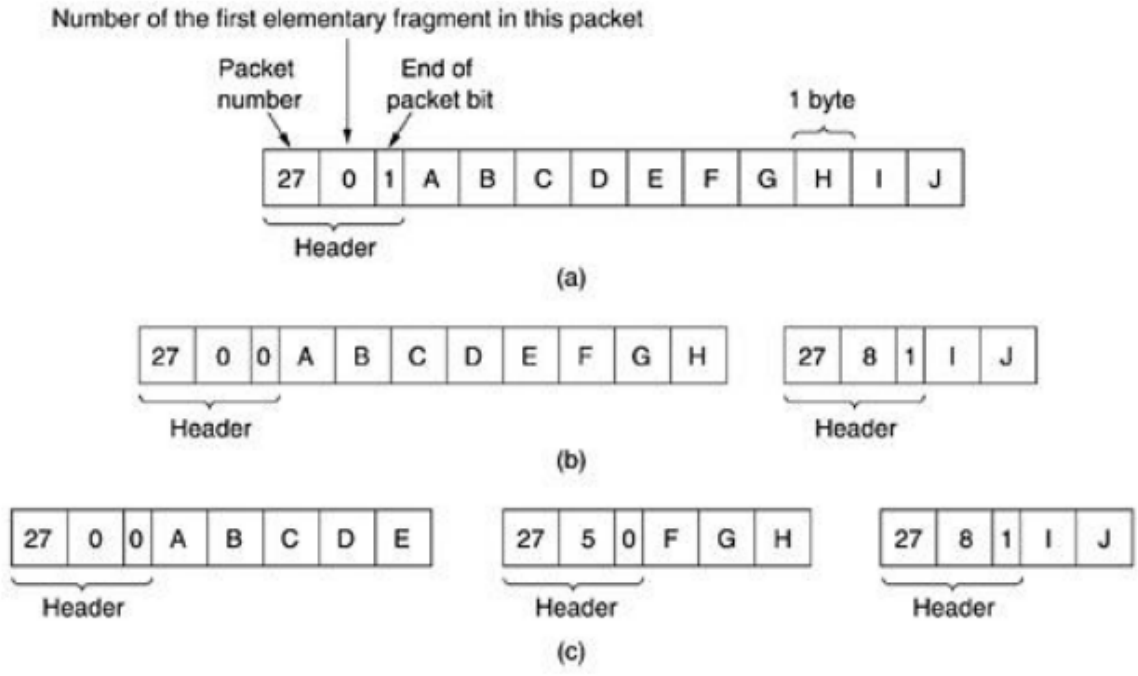


Fig 16.1. Fragmentation when the elementary data size is 1 byte. (a) Original packet, containing 10 data bytes. (b) Fragments after passing through a network with maximum packet size of 8 payload bytes plus header. (c) Fragments after passing through a size 5 gateway..

17. With an example, explain address resolution protocol.

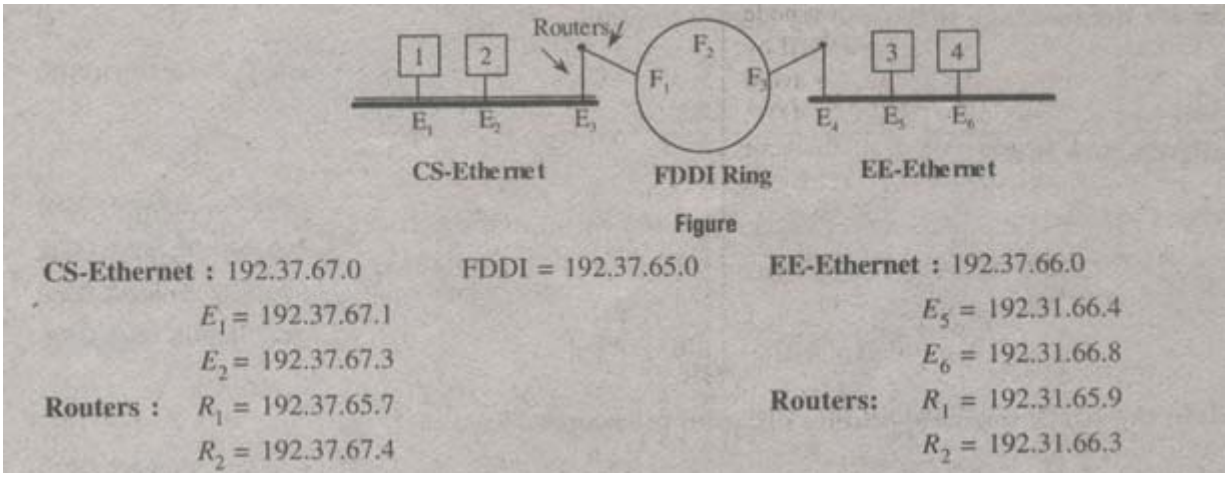
Address Resolution Protocol (ARP):

Address Resolution Protocol (ARP) is a protocol used by Internet Protocol (IP), specifically IPV4, to map IP network address to the hardware addresses used by a datalink protocol. The protocol operates below the network layer as a part of the interface between OSI network and OSI link layer.

The term address resolution refers to the process of finding an address of a computer in a network. The address is resolved using a protocol in which a piece of information is sent by a client process executing on the local computer to a server process executing on a remote computer. The information received by the server allows the server to uniquely identify the network system for which the address was required and therefore it provides the required address. The address resolution procedure is completed when the client receives a response from the server containing the required address. Thus, learning of 48-hit Ethernet address from 32-bit IP address is done with the help of Address Resolution Protocol (ARP).

Example:

Consider the following class C networks which are interconnected by using FDDI (Fibre Distributed Data Interface).



When host 1 on CS-Ethernet wants to send data to host 2 of CS-Ethernet, it should know the IP address of host 2, so it broadcasts a message asking for the name of the owner of this IP address. When host 2 receives the broadcast message it replies to host 1 by specifying it's ethernet address as E, Thus, the ethernet address is appended to the transmitting frame of host 1 by the data link layer. The frame is transmitted and i is received by host 2.

Thus, with the help of ARP the host can learn the physical address of the destination host when IP address is known.

Working of ARP:

When an incoming packet destined for a host machine on a particular Local Area Network (LAN) which arrives at gateway, the gateway asks the ARP program to find a physical host or MAC address that matches the IP address. The ARP program looks in the ARP cache and if it finds the IP address, provides it and sends it to the machine. If IP address is not found, ARP broadcasts a request packet in a special format to all machines on the LAN to see if any machine knows this LP address. If a machine recognizes the IP address as its own, it sends a reply. ARP updates the ARP cache for future reference and then sends the packet to the MAC address that replied.

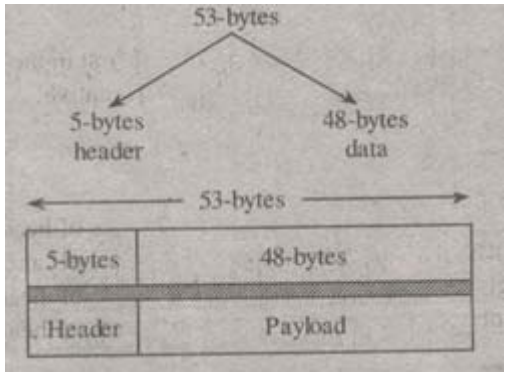
18. Give ATM cell format. Explain how this format is used in different ATM adaptation layers.

Asynchronous Transfer Mode (ATM)

The main goal of ATM is to transmit various types of data including video, IP traffic into one high-speed network. The protocols defining ATM are standardized by International Telecommunication Union (ITU-T), American National Standard Institute (ANSI). The functions of ATM are basically multiplexing and switching of ATM cells.

ATM Cell Format

ATM transmits, switches and multiplexes information in fixed length cells. The length of a cell 53bytes, consisting of 5-bytes cell header and 48bytes of data (payload).



The main idea for using small data cells is that it reduces the jitter i.e., delay variance during multiplexing of data streams, reduction of end-to-end round trip delays which are important when carrying voice traffic.

Header Format

ATM defines two different cell formats

1. User-Network interface (UNI)

2. Network-Network Interface (NNI)

Most ATM links use UNI cell format. The ATM header contains information about destination, types and priority of the cell.

Generic Flow Control (GFC)

The GFC field allows a multiplexer to control the rate of an ATM terminal. The GFC field is only available at the user-to-network interface. In the network-to-network interface these bits belong to the Virtual Path Identifier. GFC mechanism is generally used to reduce overloading of data over the network. It is a 4-bit field.

Virtual Path Identifier (VPI)

Virtual path identifier and virtual channel identifier hold the locally valid relative address of the destination. These fields may be changed within an ATM switch. VPI is an 8bit-field in user-to-network interface and 12-bit field in network-to-network interface. It is used as a field for routing over the network.

Virtual Channel Identifier (VCI)

VCI is a 16bit field in both UNI and NNI. Cells are used for routing the data from end to end-users.

Payload Type

The payload type marks whether the cell carries, user data, signaling data or maintenance information i.e., it specifies the type of information that is present in the information field. If the value of this field is 000, then the cell is not the last cell and if the value is 001, then it is the last cell, The 1-bit of a 3-bit field indicates user information, its value is 0, (the second-bit indicates if congestion is present or not, the third-bit is called the Service Data Unit (SDU), which is used to differentiate two different types of ATM SDU's the first-bit of a 3-bit field contain maintenance information, its value is 1. Therefore, this field is used to provide control information related to in-band

Cell Loss Priority (CLP)

It is a 1-bit field that indicates which cells should be discarded first in the case of congestion, A cell with value 0 is a high priority cell which is not discarded until there is no other alternative, A cell with value 1 is a low priority cell which is discarded within the network.

Header Error Control (HEC)

This field is used to perform a CRC check over the first four bytes of header. Only the header is error checked in the ATM layer. Error check for the user data is left to higher layer protocols and is performed on an end-to-end base. This 8-bit-CRC is used to correct single-bit header errors and detect multi-bit header errors. When multi-bit-errors are deleted, the current and the subsequent cells are dropped until a cell with no header error is found.

19. The header checksum only verifies the Integrity of IP header. Discuss the pros and cons of doing the checksum on the header part versus the entire packet

IP Header Checksum:

The header checksum field verifies the integrity of the header of the IP packet. The data part is not verified and is left to upper-layer protocols. If the verification process fails, the packet is simply discarded. To compute the header checksum, the sender first sets the header checksum field to 0 and then applies the internet checksum algorithm. When a router decrements the TTL field, the router must also recomputed the header checksum field.

The main reason for computing IP header is that transmission of packets takes place in accordance to the header information. Apart from this, header information is even required while transmitting data to the higher layers.

Pros of IP header checksum:

1. Header checksum prevents the packet from being delivered to the incorrect destination (i.e., the packet after undergoing IP checksum is received only by the intended destination).

2. The process of implementation within the node is simplified as each node requires less checksum bit.

3. It prevents the discarding of unnecessary packets.

Cons of Performing IP Header Checksum:

1. It checks only the integrity of IP header but not the entire packet i.e., the corrupted data is not detected.

2. The speed of computing header checksum is very slow since computation needs to be performed repeatedly at each hop. The reason for recomputation is that one or more fields always changes.

3. Every high-level protocol needs to add their respective checksum.

Pros of Performing Entire Packet Checksum:

1. Performing checksum on entire packet ensures that the payload (data) part of the packet remains unmodified and is free from single-bit errors.

2. Checksum on entire packet ensures that the packet is not intruded by the unauthorized users.

3. Computing checksum on entire packet helps in discarding the damaged or modified packets.

Cons of Performing Entire Packet Checksum:

1. Performing checksum on entire packet is time consuming. Therefore, the transmission speed is reduced.

2. If the payload contains more errors then the cost of computing checksum is also increased.

3. Checksum fails to detect multiple-bit errors.