Subject: Computer Networks [2140709]

UNIT – 4 – NETWORK LAYER

Q.1 What is Network Service Models? Explain various Network Service Models available. Also explain services provided by it.

Ans

- "***The network service model defines the characteristics of end-to-end transport of packets between sending and receiving end systems***." That means,
  - ✓ When the transport layer at a sending host passes down the packets to the network layer, can the transport layer be sure that the packets will be delivered to their destination?
  - ✓ When multiple packets are sent from the sending host, will they be delivered to the receiving host in the same order in which they are sent?
  - ✓ Will the amount of time between sending of two sequential packets be the same as the time between their reception?
  - ✓ Will the network provide any feedback about congestion in the network?
  - ✓ What are the properties of the channel connecting the transport layers of the sending and receiving hosts?

- The answers to all above questions are decided by the "*Network Service Model"* provided by the network layer.
- Let's now consider some possible services that the network layer could provide when the transport layer at the sending host sends a packet
  - ✓ *Guaranteed delivery*. This service guarantees that the packet will, sooner or later, arrive at its destination.
  - ✓ *Guaranteed delivery with bounded delay.* This service not only guarantees delivery of the packet, but it delivers the packet within a specific time period (for example, within 100 msec).

- Let's now consider some possible services provided to a flow of packets(more than one packet) between a given source and destination,
  - ✓ *In-order packet delivery*. This service guarantees that packets arrive at the destination in the same order in which they were sent.
  - ✓ *Guaranteed minimal bandwidth*. This network-layer service sets an equal/even bit rate for the entire transmission channel(for example,1Mbps) from host to destination. As long as the sending host transmits bits (as part of packets) at a rate below the specified bit rate, then no packet is lost and each packet arrives within a specific time period(for example, within 40 msec).
  - ✓ *Guaranteed maximum jitter*. This service guarantees that the amount of time between the transmission of two successive packets at the sender is equal to the amount of time between their reception at the destination (or that this spacing changes by no more than some specified value).
  - ✓ *Security services*. Using a secret session key known only by a source and

destination host, the network layer in the source host could encrypt the data of all packets being sent to the destination host. The network layer in the destination host would then be responsible for decrypting the data. With such a service, secrecy and security would be provided to all transport-layer segments (TCP and UDP) between the source and destination hosts. In addition to this, the network layer could provide data integrity and source authentication services.

- The Internet's network layer provides a single service, known as *best-effort service*. Here, sometimes best effort might also mean no service at all. With best-effort service, :
  - ✓ Timing between packets is not guaranteed to be preserved.
  - ✓ Packets are not guaranteed to be received in the same order in which they were sent.
  - ✓ Eventual delivery of transmitted packets is not guaranteed.
- To avoid such problems, ATM network is available as an alternative to Internet's best effort model. Two of the more important ATM service models are:
- ***Constant bit rate (CBR) ATM network service.*** This was the first ATM service model to be standardized. It arose early interest of the telephone companies in ATM.
- The CBR was suitable for carrying real-time, constant bit rate audio and video traffic.
- The goal of CBR service is conceptually simple—to provide a flow of packets (known as cells in ATM terminology) with a virtual pipe which has constant properties, as if a dedicated fixed-bandwidth transmission link existed between sending and receiving hosts.
- With CBR service, a flow of ATM cells is carried across the network in such a way that:
  - ✓ A cell's end-to-end delay is guaranteed to be less than a particular value.
  - ✓ The variability in the cells end-to-end delay (that is, the jitter) is guaranteed to be less than a particular value.
  - ✓ And, the fraction of cells that are lost or delivered late are less than a specified value.
  - These values are decided by the cells that are lost or delivered late are all guaranteed to be less than a particular value sending host and the ATM network when the CBR connection is first established.

- ***Available bit rate (ABR) ATM network service.*** With the Internet offering so called best-effort service, ATM's ABR might best be characterized as being a slightly-better-than-best-effort service.
- As with the Internet service model, cells may be lost under ABR service. Unlike in the Internet, however, cells cannot be reordered (although they may be lost), and a minimum cell transmission rate (MCR) is guaranteed to a connection using ABR service.
- If the network has enough free resources at a given time, a sender may also be able

to send cells successfully at a higher rate than the MCR.

- ATM ABR service can provide feedback to the sender (in terms of a congestion notification bit, or an explicit rate at which to send) that controls how the sender adjusts its rate between the MCR and an allowable peak cell rate.

Q.2 Explain Virtual-Circuit Networks. OR
Explain working of Virtual-Circuit Networks and describe its various phases.

Ans

- These network-layer connections are called virtual circuits (VCs).A VC consists of
  1) A path (that is, a series of links and routers) between the source and destination hosts,
  2) VC numbers, one number for each link along the path, and
  3) Entries in the forwarding table in each router along the path.
- A packet belonging to a virtual circuit will carry a VC number in its header. Because a virtual circuit may have a different VC number on each link, each router on the transmission path must replace the VC number of each packet passing through it with a new VC number. The new VC number is obtained from the forwarding table.
- Consider the network shown in Figure 1. The numbers next to the links of R1 in Figure 1 are the link interface numbers.

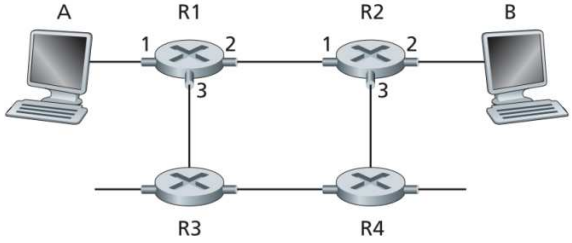


Figure 1 : A simple virtual circuit network

- Suppose now that Host A requests that the network establish a VC between itself and Host B. Suppose also that the network chooses the path A-R1-R2-B and assigns VC numbers 12, 22, and 32 to the three links in this path for this virtual circuit.
- In this case, when a packet in this VC leaves Host A, the value in the VC number field in the packet header is 12; when it leaves R1, the value is 22; and when it leaves R2, the value is 32.
- How does the router decide the new VC number for a packet passing through the router? For a VC network, each router's forwarding table includes VC number translation; for example, the forwarding table in R1 might look something like this:

Incoming Interface	Incoming VC #	Outgoing Interface	Outgoing VC #
1	12	2	22
2	63	1	18
3	7	2	17
1	97	3	87
...	...	...	...

- Whenever a new VC is established across a router, an entry is added to the forwarding table. Similarly, whenever a VC terminates, the related entries in each table along its path are removed.
- There are basically three phases in a virtual circuit:
- ***VC setup.*** During the setup phase,
  1) The sender's transport layer contacts the network layer, specifies the receiver's address, and waits for the network to set up the VC.
  2) The network layer decides the path between sender and receiver, that is, the series of links and routers through which all packets of the VC will travel.
  3) The network layer also decides the VC number for each link along the path.
  4) Finally, the network layer adds an entry in the forwarding table in each router along the path.

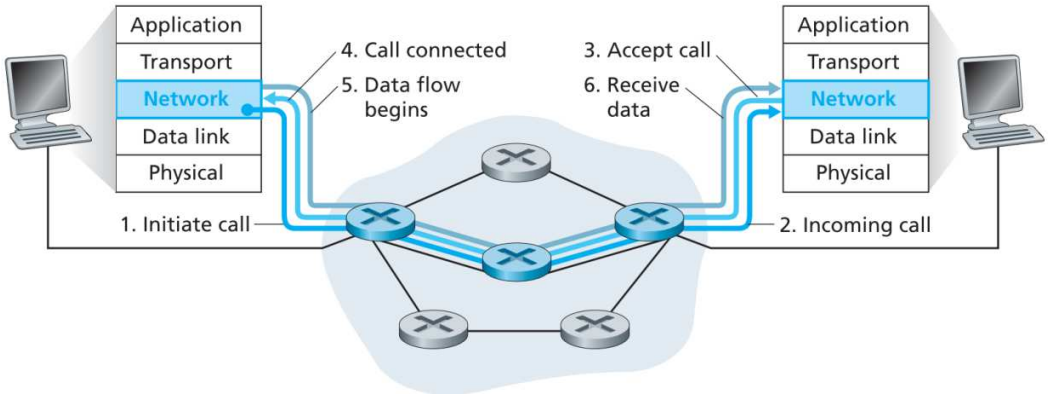


Figure 2 : Virtual-circuit setup

- During VC setup, the network layer may also reserve resources (for example, bandwidth) along the path of the VC.
- ***Data transfer.*** As shown in Figure 2, once the VC has been established, packets can begin to flow along the VC.
- ***VC teardown.*** This is initiated when the sender (or receiver) informs the network layer that it wants to terminate the VC.
- The network layer will then typically inform the end system on the other side of the network of the call termination (i.e it'll inform the sender if the receiver requests to terminate the VC and it'll inform the receiver if the sender requests to terminate the VC) and update the forwarding tables in each of the packet routers on the path to

indicate that the VC no longer exists.

Q.3 Why a packet doesn't just keep the same VC each of the links along its route? Explain.

Ans
- There are two reasons for this:
- First, changing the number from link to link reduces the length of the VC field in the packet header.
- Second, and more importantly, VC setup is considerably simplified by permitting a different VC number at each link along the path of the VC. Specifically, with multiple VC numbers available, each link in the path can choose a VC number, independent of the VC numbers chosen at other links along the path.
- If a common VC number were required for all links along the path, the routers would have to exchange and process a large number of messages to agree on a common VC number (e.g., one that is not being used by any other existing VC at these routers) to be used for a connection.

Q.4 What is the difference between VC setup at the network layer and connection setup at the transport layer?

Ans
- Connection setup at the transport layer involves only the two end systems. During transport-layer connection setup, the two end systems alone determine the parameters (for example, initial sequence number and flow-control window size) of their transport-layer connection. Although the two end systems are aware of the transport-layer connection, the routers within the network are completely unaware to it.
- On the other hand, in a VC network layer, routers along the path between the two end systems are involved in VC setup, and each router is fully aware of all the VCs passing through it.

Q.5 Describe high-level view of generic router architecture. **OR**
What's Inside a Router? Explain its components.

Ans
- A high-level view of generic router architecture is shown in Figure 3. Four router components can be identified:
- ***Input ports.*** An input port performs several main functions. It performs the physical layer function of terminating an incoming physical link at a router; this is shown in the leftmost box of the input port and the rightmost box of the output port in Figure 3.
- An input port also performs link-layer functions needed to operate with the link layer at the other side of the incoming link; this is represented by the middle boxes in the input and output ports.

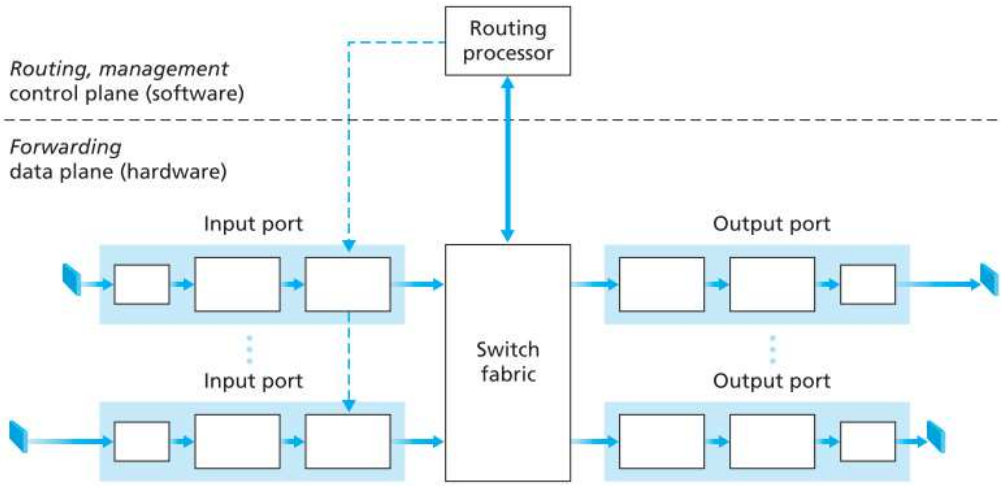


Figure 3 : Router architecture

- The lookup function is also performed at the input port; this will occur in the rightmost box of the input port. Here the forwarding table is consulted to determine the router output port to which an arriving packet will be forwarded through the switching fabric.
- Control packets (for example, packets carrying routing protocol information) are forwarded from an input port to the routing processor. Note that the term port here referring to the physical input and output router interfaces—is different from the software ports associated with network applications and sockets discussed in application layer and transport layer.
- ***Switching fabric.*** The switching fabric connects the router's input ports to its output ports. This switching fabric is within the router- a network inside a network router!
- ***Output ports.*** An output port stores packets received from the switching fabric and transmits these packets on the outgoing link by performing the necessary link-layer and physical-layer functions.
- When a link is bidirectional (that is, carries traffic in both directions), an output port will typically be paired with the input port for that link on the same line card (a printed circuit board containing one or more input ports, which is connected to the switching fabric).
- ***Routing processor.*** The routing processor executes the routing protocols, maintains routing tables and attached link state information, and computes the forwarding table for the router.
- It also performs the network management functions.

Q.6 Justify *"Router's forwarding functions are almost always implemented in hardware whereas control functions are usually implemented in software."*

Ans

- There is a difference between a router's forwarding and routing functions. A router's input ports, output ports, and switching fabric together constitute the **forwarding function** and are almost always implemented in hardware, as shown in Figure 3. These forwarding functions are sometimes collectively known as the **router forwarding plane**.
- Let us see why a hardware implementation is needed, then software. Consider that with a 10 *Gbps* input link and a 64-byte IP datagram, the input port has only 51.2 *ns* to process the datagram before another datagram arrives. If N ports are combined on a line card (as is often done in practice), the datagram-processing pipeline must operate N times faster, which is far too fast for software implementation.
- Forwarding plane hardware can be implemented by using a router vendor's own hardware designs, or by using hardware designs made using purchased merchant-silicon chips. *E.g., as sold by companies such as Intel and Broadcom*.
- While the forwarding plane operates at the *nanosecond* time scale, a router's control functions—executing the routing protocols, responding to attached links that go up or down, and performing management functions—operate at the millisecond or second timescale. These router control plane functions are usually implemented in software and execute on the routing processor.

Q.7 Explain router's *Input Processing* functionality in details.

Ans

- Detailed view of input processing is given in Figure 4. The input port's line termination function and link-layer processing implement the physical and link layers for that individual input link.

- The lookup performed in the input port is central to the router's operation—The router decides the output port to which on arriving packet will be forwarded via the switching fabric. The forwarding table is calculated and updated by the routing processor, with a shadow copy stored at each input port.

- As a shadow copy of the forwarding table is made at each input port, the forwarding decisions can be made locally at each input port, without invoking the centralized routing processor on a per-packet basis and thus avoiding a blockage at a single point within the router.

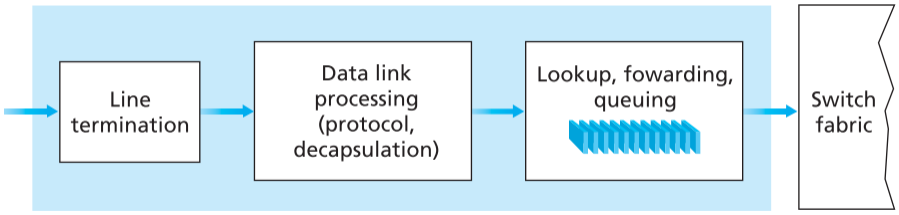


Figure 4: Input port processing

- In the forwarding table, we just search through the forwarding table looking for the longest prefix match. But at Gigabit transmission rates, this lookup must be performed in nanoseconds

- Thus, not only lookup must be performed in hardware, but techniques other than a simple linear search through a large table are needed.

- Special attention must also be paid to memory access times, resulting in designs with inbuilt on-chip DRAM and faster SRAM (used as a DRAM cache) memories. Ternary Content Address Memories (TCAMs) are also often used for lookup. With a TCAM, a 32-bit IP address is presented to the memory, which returns the content of the forwarding table entry for that address in essentially constant time. The Cisco 8500 has a 64K CAM for each input port.

- Once a packet's output port has been determined via the lookup, the packet can be sent into the switching fabric. A packet can be temporarily blocked from entering the switching fabric if packets from other input ports are currently using the fabric. A blocked packet will be queued at the input port and then will cross the fabric, later on.
- Although "lookup" is the most important action in input port processing, many other actions must be taken:

i. Physical- and link-layer processing must occur.
ii. The packet's version number, checksum and time-to-live field—must be checked and the last two fields must be rewritten.
iii. Counters used for network management (such as the number of IP datagrams received) must be updated.

- In concluding statement *"input port processing by noting that the input port steps of looking up an IP address ("match") then sending the packet into the switching fabric ("action") is a specific case of a more general "match plus action" abstraction.*

Q.8 Explain Switching process in router. OR Explain switching fabric in router. OR Explain switching process in router with all its switching techniques using diagram.

Ans.

- The switching fabric is the most important part of a router. Through this fabric the packets are actually switched (that is, forwarded) from an input port to an output port. Switching can be done in a number of ways, as shown in Figure 5:
- ***Switching via memory.*** The simplest, earliest routers were traditional computers. Switching between input and output ports was done under direct control of the CPU (routing processor).
- Input and output ports functioned as traditional I/O devices in a traditional operating system. An input port with an arriving packet first signaled the routing processor through an interrupt. The packet was then copied from the input port into processor memory.
- The routing processor then obtained the destination address from the header, found the appropriate output port in the forwarding table, and copied the packet to the output port's buffers.
- In this case, if the memory bandwidth is such that B packets per second can be written into, or read from memory, then the overall forwarding throughput (the total rate at which packets are transferred from input ports to output ports) must be less than B/2.
- Note also that two packets cannot be forwarded at the same time, even if they have different destination ports, since only one memory read/write can be done at a time.

- Many modern routers switch through memory. A major difference from early routers, however, is that the lookup of the destination address and the storing of the packet into the appropriate memory location are performed by processing on the input line cards.
- In some ways, routers that switch via memory look very much like shared-memory multiprocessors, because of the processing on a line card switching (writing) packets into the memory of the appropriate output port. *Cisco's Catalyst 8500 series switches [Cisco 8500 2012] forward packets via a shared memory*.

- ***Switching via a bus.*** In this approach, an input port transfers a packet directly to the output port over a shared bus, without interference by the routing processor. This is typically done by making the input port pre-pend a switch-internal label (header) to the packet indicating the local output port to which this packet is being transferred and transmitting the packet onto the bus.
- The packet is received by all output ports, but only the port that matches the label will keep the packet.
- The label is then removed at the output port, as this label is only used within the switch to cross the bus.
- If multiple packets arrive to the router at the same time, each at a different input port, only one packet can cross the bus at a time, and others will have to wait. Because every packet must cross the single bus, the switching speed of the router is limited to the bus speed.
- Nonetheless, switching via a bus is often enough for routers that operate in small local area and enterprise networks. *The Cisco 5600 [Cisco Switches 2012] switches packets over a 32 Gbps backplane bus.*

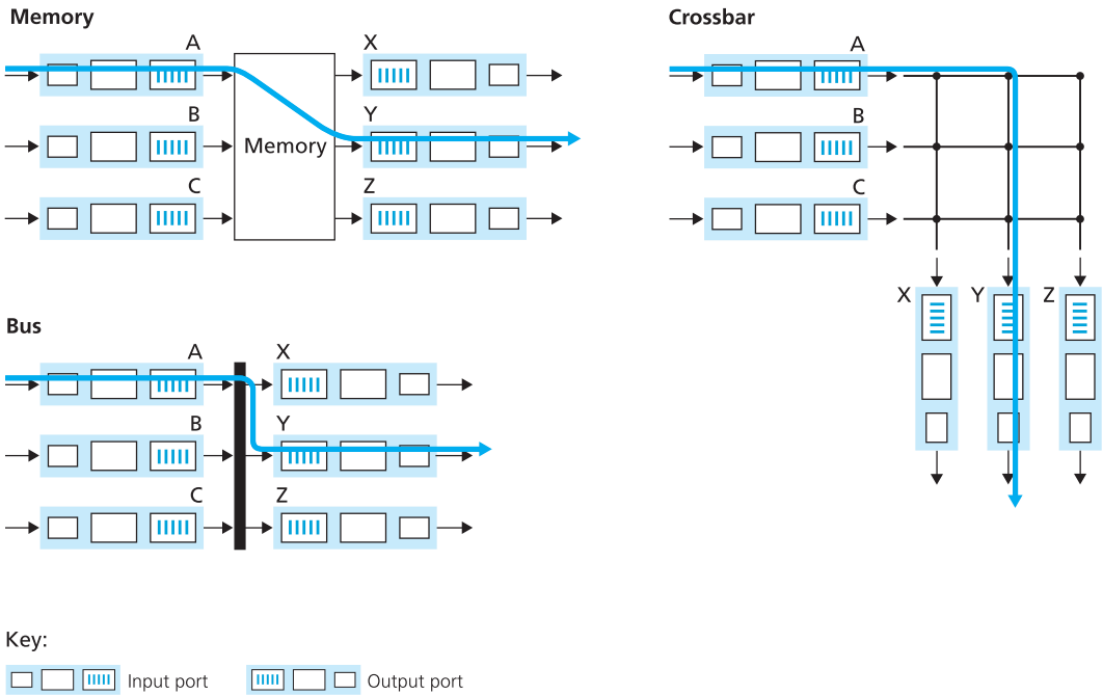


Figure 5: Three switching techniques

- ***Switching via an interconnection network***. One way to overcome the bandwidth limitation of a single, shared bus is to use a more advanced interconnection network, such as those that have been used in the past to interconnect processors in a multiprocessor computer architecture.
- A crossbar switch is an interconnection network consisting of 2N buses that connect N input ports to N output ports, as shown in Figure 5. Each vertical bus intersects each horizontal bus at a cross point, which can be opened or closed at any time by the switch fabric controller (whose logic is part of the switching fabric itself).
- When a packet arrives from port A and needs to be forwarded to port Y, the switch controller closes the cross point at the intersection of buses A and Y, and port A then sends the packet on its bus, which is picked up (only) by bus Y. Note that a packet from port B can be forwarded to port X at the same time, since the A-to-Y and B-to-X packets use different input and output buses.
- Thus, unlike the previous two switching methods, crossbar networks are capable of forwarding multiple packets in parallel.
- However, if two packets from two different input ports are destined to the same output port, then one will have to wait at the input, since only one packet can be sent over any given bus at a time.
- More advanced interconnection networks use multiple stages of switching elements to allow packets from different input ports to go towards the same output port at the same time through the switching fabric. *Cisco 12000 family switches [Cisco 12000 2012] use an interconnection network.*

Q.9 Explain Output Processing in router and also discuss queuing process in it.

Ans

- Output port processing, shown in Figure 6, takes the packets that have been stored in the output port's memory and transmits them to the output link. This includes selecting and de-queuing packets for transmission, and performing the needed link - layer and physical-layer transmission functions.

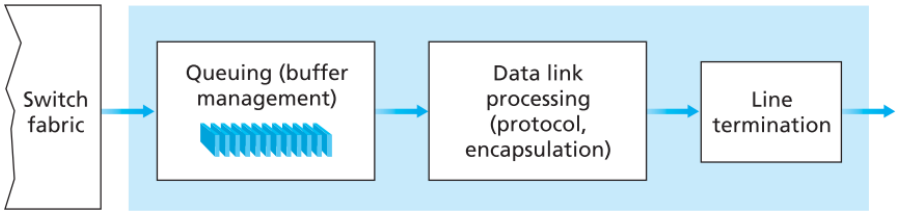


Figure 6: Output port processing

Queueing process:

- If we consider input and output port functionality and the configurations shown in Figure 6, it's clear that packet queues may form at both the input ports and the output ports.
- The location and amount of queuing (either at the input port queues or the output port queues) will depend on the traffic load, the relative speed of the switching fabric, and the line speed.
- As these queues grow large, the router's memory can eventually get over and packet loss will occur when no memory is available to store arriving packets.
- Many times packets are "lost within the network" or "dropped at a router." It is here, at these queues within a router, where such packets are actually dropped and lost.
- Suppose that the input and output line speeds (transmission rates) all have an equal transmission rate of R_{line} packets per second. There are N input ports and N output ports. To further simplify the discussion, let's assume that all packets have the same fixed length, and the packets arrive to input ports in a synchronous manner. That is, the time to send a packet on any link is equal to the time to receive a packet on any link, and during such an interval of time, either zero or one packet can arrive on an input link.
- Define the switching fabric transfer rate R_{switch} as the rate at which packets can be moved from input port to output port. If $R_{switch}$is N times faster than R_{line}, then very less queuing will occur at the input ports.
- This is because even in the worst case, where all N input lines are receiving packets, and all packets are to be forwarded to the same output port, each batch of N packets (one packet per input port) can be passed through the switch fabric before the next batch arrives.
- At the output ports, suppose that R_{switch} is still N times faster than R_{line}. Once again, packets arriving at each of the N input ports are supposed to go to the same output port. In this case, in the time it takes to send a single packet to the outgoing link, N new packets will arrive at this output port.

- Since the output port can transmit only a single packet in a unit of time (the packet transmission time), the N arriving packets will have to queue (wait) for transmission over the outgoing link. Then N more packets can possibly arrive during the transmission time of just one of the N packets that had just previously been queued. And so on.
- Eventually, the number of queued packets can grow large enough to finish the available memory at the output port, in which case packets are dropped.

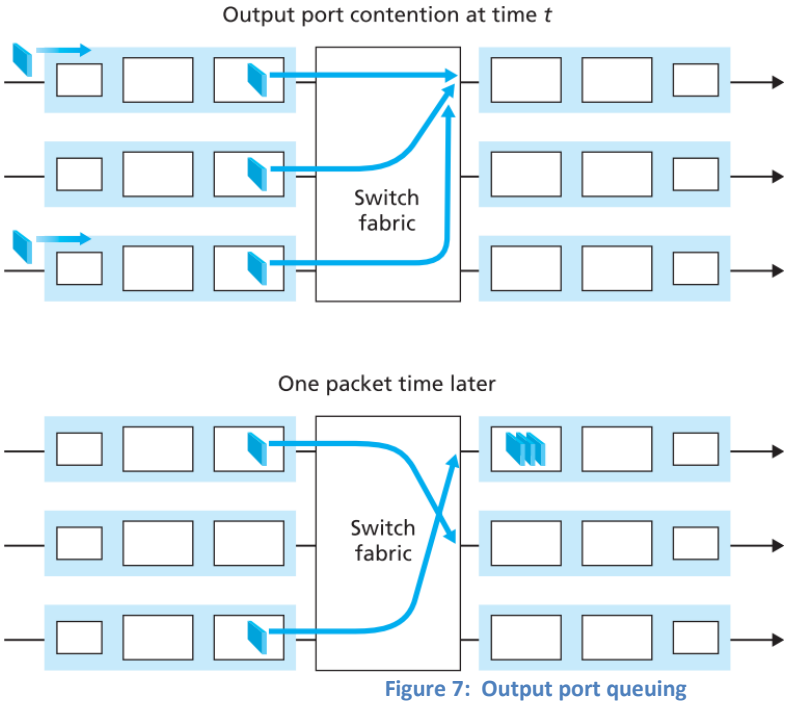


Figure 7: Output port queuing

- Output port queuing is showed in Figure 7. At time t, a packet has arrived at each of the incoming input ports and every packet is to be sent to the uppermost outgoing port. Assuming equal line speeds and a switch operating at three times the line speed, one time unit later (that is, in the time needed to receive or send a packet), all three original packets have been transferred to the outgoing port and are queued awaiting transmission.
- In the next time unit, one of these three packets will have been transmitted to the outgoing link. In our example, two new packets have arrived at the incoming side of the switch; one of these packets is destined for this uppermost output port.

Q.10 What is the consequence of output port queuing? How *quality-of-service guarantees in this situation? Also explain head-of-the-line (HOL) blocking at input port.*

- One of the results of output port queuing is that a ***packet scheduler*** at the output port must choose one packet among those queued for transmission.

- This selection might be done on a simple basis, such as ***first-come-first-served (FCFS)*** scheduling, or a more sophisticated scheduling rule such as ***weighted fair queuing (WFQ)***, which shares the outgoing link equally among the different end-to-end connections that have packets queued for transmission. Packet scheduling plays an important role in providing ***quality-of-service guarantees***.
- Similarly, if there is not enough memory to store an incoming packet, a decision must be made to either drop the arriving packet (a policy known as ***drop-tail***) or remove one or more already-queued packets to make memory available for the newly arrived packet.
- In some cases, it may be advantageous to drop (or mark the header of) a packet before the buffer is full in order to provide a congestion (blockage) signal to the sender. A number of packet-dropping and marking policies (which collectively have become known as ***active queue management (AQM) algorithms***) have been proposed and studied.
- One of the most widely studied and used AQM algorithms is the ***Random Early Detection (RED) algorithm***.
- Under RED, a weighted average is maintained for the length of the output queue. If the average queue length is less than a minimum limit, min_{th} , when a packet arrives, the packet is admitted to the queue. If the queue is full or the average queue length is greater than a maximum threshold, $max_{th,}$ when a packet arrives, the packet is marked or dropped. Finally, if the packet arrives to find an average queue length in the interval $[min_{th,}, max_{th}]$, the packet is marked or dropped with a probability that is some function of the average queue length, min_{th}, and max_{th}.
- A number of probability based marking/dropping functions have been proposed, and various versions of RED have been analytically modeled, simulated, and/or implemented.
- If the switch fabric is not fast enough to transfer all arriving packets through the fabric without delay, then packet queuing can also occur at the input ports, as packets must join input port queues to wait their turn to be transferred through the switching fabric to the output port. To understand an important effect of this queuing, consider a crossbar switching fabric and suppose that (1) all link speeds are identical, (2) that one packet can be transferred from any one input port to a given output port in the same amount of time it takes for a packet to be received on an input link, and (3) packets are moved from a given input queue to their desired output queue in an FCFS manner. Multiple packets can be transferred in parallel, as long as their output ports are different.
- However, if two packets at the front of two different input queues are to be sent to the same output queue, then one of the packets will be blocked and must wait at the input queue—the switching fabric can transfer only one packet to a given output port at a time.

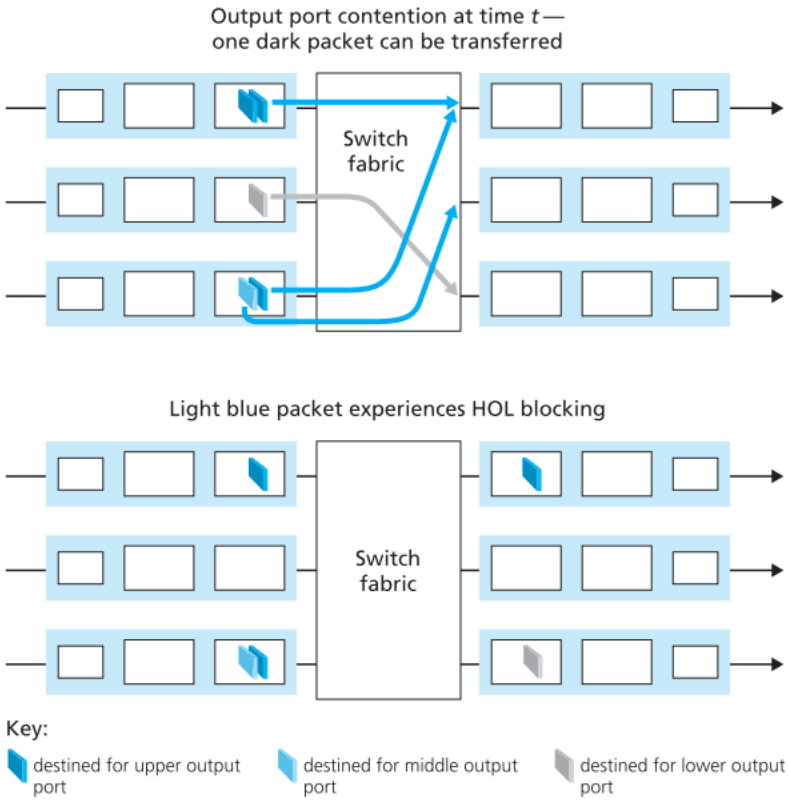


Figure 8: HOL blocking at an input queued switch

- Figure 8 shows an example in which two packets (darkly shaded) at the front of their input queues are to be sent to the same upper-right output port. Suppose that the switch fabric chooses to transfer the packet from the front of the upper-left queue.
- In this case, the darkly shaded packet in the lower-left queue must wait. Apart from this, the lightly shaded packet that is queued behind that packet in the lower-left queue must also wait, even though there is no contention(disagreement; in this case, traffic or congestion) for the middle-right output port (the destination for the lightly shaded packet). This phenomenon is known as ***head-of-the-line (HOL) blocking*** in an input-queued switch—a queued packet in an input queue must wait for transfer through the fabric (even though its output port is free) because it is blocked by another packet at the head of the line.

Q.11 Draw IPv4 datagram header format and explain its key field.

Ans

- IPv4 datagram format is shown in Figure 9. An IP datagram has a total of 20 bytes of header (assuming no options). The key fields in the IPv4 datagram are the following:

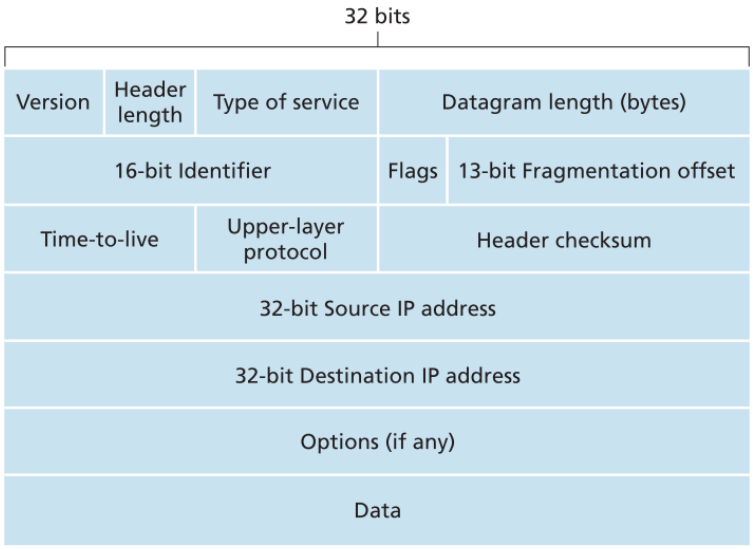


Figure 9: IPv4 datagram format

- ***Version number.*** These 4 bits specify the IP protocol version of the datagram. By looking at the version number, the router can determine how to interpret the remainder of the IP datagram. Different versions of IP use different datagram formats. The datagram format for the current version of IP, IPv4, is shown in Figure 9.
- ***Header length.*** Because an IPv4 datagram can contain a variable number ofoptions (which are included in the IPv4 datagram header), these 4 bits are neededto determine where in the IP datagram the data actually begins. Most IP datagrams do not contain options, so the typical IP datagram has a 20-byte header.
- ***Type of service.*** The type of service (**TOS**) bits were included in the IPv4 headerto allow different types of IP datagrams (for example, datagrams particularlyrequiring low delay, high throughput, or reliability) to be distinguished from eachother. For example, it might be useful to distinguish real-time datagrams (such asthose used by an IP telephony application) from non-real-time traffic (for example, FTP). The specific level of service to be provided is a policy issue determined by the router's administrator.
- ***Datagram length.*** This is the total length of the IP datagram (header plus data), measured in bytes. Since this field is 16 bits long, the theoretical maximum size of the IP datagram is 65,535 bytes. However, datagrams are rarely larger than 1,500 bytes.
- ***Identifier, flags, fragmentation offset***. These three fields have to do with so-called IP fragmentation, a topic we will consider in depth shortly. Interestingly, the new version of IP, IPv6, does not allow for fragmentation at routers.
- ***Time-to-live.*** The time-to-live (**TTL**) field is included to ensure that datagrams do not circulate forever (due to, for example, a long-lived routing loop) in the network. This field is decremented by one each time the datagram is processed by a router. If the TTL field reaches 0, the datagram must be dropped.
- ***Protocol***. This field is used only when an IP datagram reaches its final destination. The value of this field indicates the specific transport-layer protocol to which the data portion of this IP datagram should be passed. For example, a value of 6 indicates that

the data portion is passed to TCP, while a value of 17 indicates that the data is passed to UDP. For a list of all possible values, Note that the protocol number in the IP datagram has a role that is analogous to the role of the port number field in the transport layer segment. The protocol number is the glue that binds the network and transport layers together, whereas the port number is the glue that binds the transport and application layers together. Link-layer frame also has a special field that binds the link layer to the network layer.

- ***Header checksum.*** The header checksum aids a router in detecting bit errors in a received IP datagram. The header checksum is computed by treating each 2 bytes in the header as a number and summing these numbers using 1s complement arithmetic. The 1s complement of this sum, known as the Internet checksum, is stored in the checksum field. A router computes the header checksum for each received IP datagram and detects an error condition if the checksum carried in the datagram header does not equal the computed checksum. Routers typically discard datagrams for which an error has been detected. Note that the checksum must be recomputed and stored again at each router, as the TTL field, and possibly the options field as well, may change. An interesting discussion of fast algorithms for computing the Internet checksum is. A question often asked at this point is, why does TCP/IP perform error checking at both the transport and network layers? There are several reasons for this repetition. First, note that only the IP header is checksummed at the IP layer, while the TCP/UDP checksum is computed over the entire TCP/UDP segment. Second, TCP/UDP and IP do not necessarily both have to belong to the same protocol stack. TCP can, in principle, run over a different protocol (for example, ATM) and IP can carry data that will not be passed to TCP/UDP.
- ***Source and destination IP addresses.*** When a source creates a datagram, it insertsits IP address into the source IP address field and inserts the address of theultimate destination into the destination IP address field. Often the source hostdetermines the destination address via a DNS lookup.
- ***Options***. The options fields allow an IP header to be extended. Header options were meant to be used rarely—hence the decision to save overhead by not including the information in options fields in every datagram header. However, the mere existence of options does complicate matters—since datagram headers can be of variable length, one cannot determine a priori where the data field will start. Also, since some datagrams may require options processing and others may not, the amount of time needed to process an IP datagram at a router can vary greatly. These considerations become particularly important for IP processing in high-performance routers and hosts. For these reasons and others, IP options were dropped in the IPv6 header.
- ***Data (payload).***The data field of the IP datagram contains the transport-layer segment (TCP or UDP) to be delivered to the destination. However, the data field can carry other types ofdata, such as ICMP messages.

Q.12 Explain IP Datagram Fragmentation. OR Explain network layer fragmentation in detail.

Ans

- Each hardware technology specifies the maximum amount of data that a frame can carry. This limit is termed as **maximum transmission unit (MTU).**
- Every hardware component is designed in such a way that it does not accept or transfer the data more than MTU limit.
- So a datagram must be less than or equal to MTU, or it cannot be encapsulated for transmission.
- Internet service model is a collection of networks which consists of different hardware technologies which has different MTU values.
- For Eg. An Ethernet frame can carry upto 1500 bytes of data whereas some intermediate network elements can carry upto 576 bytes of data (MTU).
- To overcome this issue two ways can be figured out:
    a) Make sure that all IP datagrams are small enough to fit inside each and every hardware's MTU.
    b) Provides a means in which packets can be divided into small pieces called **fragments** and then reassembled whenever required to form original datagram.
- Solution (a) may be practically impossible because internet is network of networks and packet switching technique is used for communication. So we cannot trace a path to be followed by packet because packet can change path depending upon various situations such as congestion etc.
- So solution (b) turns out be a better choice.
- Fragmentation occurs in router when it receives a datagram that it wants to forward over a network that has a MTU that is smaller than the received datagram.
- At destination node reassembly of fragments is performed to form an original datagram.
- Fragments of same datagram contain a same identifier which is unique among all datagrams.
- By inspecting the identifier field the destination host recognize that they are the part of these datagram.
- Each fragment carries source address, destination address, identification number offset and flag bit.
- The fragments can be lost or arrive out of order, to overcome it the last fragment of original datagram have flag bit set to zero whereas other fragments carry flag bit as one indicating that remaining fragments are arriving and it also helps to order out the fragments to bulid an original datagram.

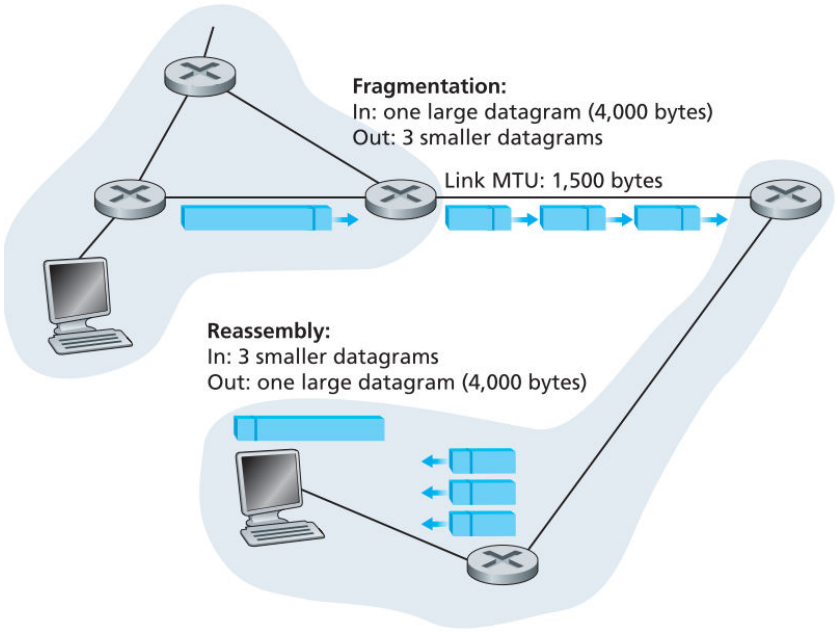


Figure 10: IP fragmentation and reassembly

- Figure 10 illustrates an example. A datagram of 4,000 bytes (20 bytes of IP header plus 3,980 bytes of IP payload) arrives at a router and must be forwarded to a link with an MTU of 1,500 bytes.
- This implies that the 3,980 data bytes in the original datagram must be allocated to three separate fragments (each of which is also an IP datagram). Suppose that the original datagram is stamped with an identification number of 777. The characteristics of the three fragments are shown in Figure 11. The values in Figure 11 reflect the requirement that the amount of original payload data in all but the last fragment be a multiple of 8 bytes, and that the offset value be specified in units of 8-byte chunks.

Fragment	Bytes	ID	Offset	Flag
1st fragment	1,480 bytes in the data field of the IP datagram	identification = 777	offset = 0 (meaning the data should be inserted beginning at byte 0)	flag = 1 (meaning there is more)
2nd fragment	1,480 bytes of data	identification = 777	offset = 185 (meaning the data should be inserted beginning at byte 1,480. Note that 185 · 8 = 1,480)	flag = 1 (meaning there is more)
3rd fragment	1,020 bytes (= 3,980–1,480–1,480) of data	identification = 777	offset = 370 (meaning the data should be inserted beginning at byte 2,960. Note that 370 · 8 = 2,960)	flag = 0 (meaning this is the last fragment)

Figure 11 : IP fragments

- The offset field tells a receiver where data fragments belongs to an original datagram.
- Therefore, a receiver uses identifier, flags and offset fields to reassemble the fragments.

Q.13 What is DHCP? And why it is called plug-and-play protocol?

Ans

- Each user must be properly configured with an IP address, subnet mask, default router to establish a communication between internet and end system.
- To common user it does not make any sense so it becomes a burden for network administrator. A network administrator has to manually configure to establish communication.
- **Dynamic Host Configuration Protocol (DHCP)** provide the mechanism that allows an arbitrary computer to join a new network and obtain IP address automatically. The concept is been termed as plug and play protocol.
- DHCP allows a computer to join a network end obtain its configuration without requiring an administrator to make manual changes.
- In addition to host IP address assignment, DHCP also allows a host to learn additional information, such as its subnet mask, the address of its first-hop router (often called the default gateway), and the address of its local DNS server.
- DHCP is also enjoying widespread use in residential Internet access networks and in wireless LANs, where hosts join and leave the network frequently.

Q.14 What is DHCP and explain its 4-step process in detail.

Ans

- DHCP is a client-server protocol. A client is typically a newly arriving host wanting to obtain network configuration information, including an IP address for itself.
- For a newly arriving host, the DHCP protocol is a four-step process, as shown in Figure 12.

DHCP server discovery. The first task of a newly arriving host is to find a DHCP server with which client/server can interact. This is done using a DHCP discover message, DHCP discover message is sent by DHCP client with the destination IP address 255.255.255.255 and a source address of 0.0.0.0 the message contains unique transaction id so client can relate the response from the server to this request.

- The DHCP client passes the IP datagram to the link layer, which then broadcasts this frame to all nodes attached to the subnet.

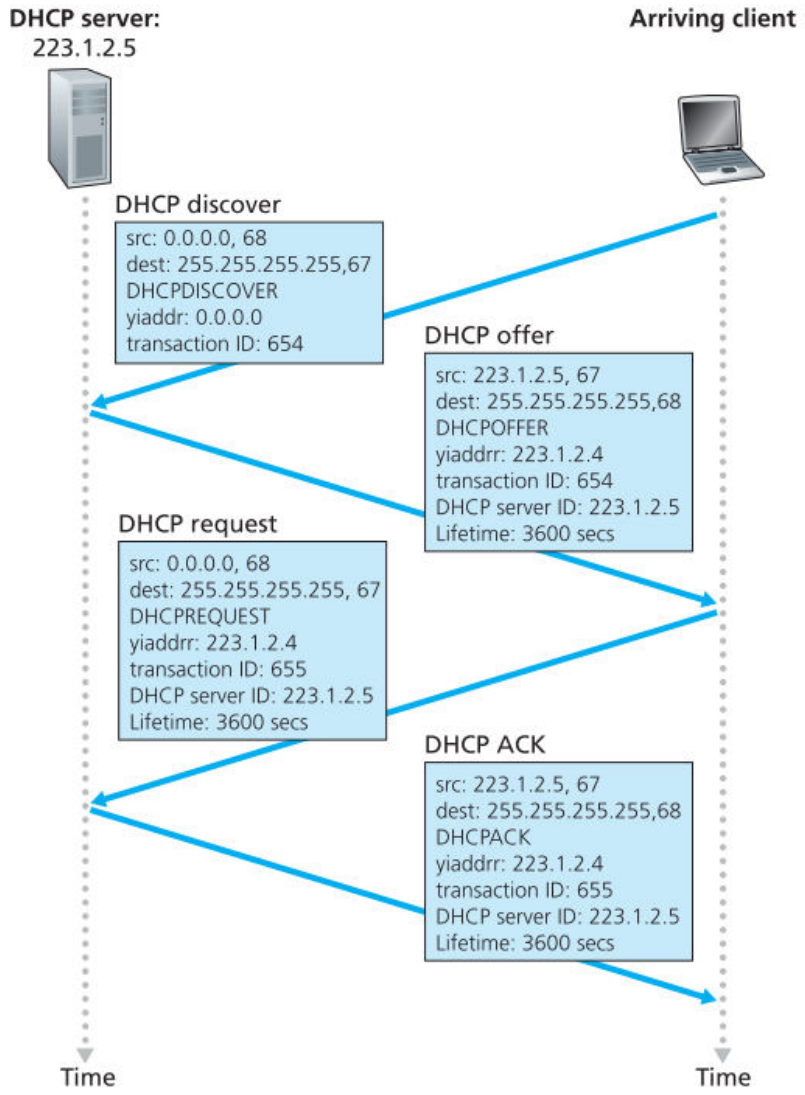


Figure 12 : DHCP client-server interaction

- ***DHCP server offer(s).*** This is returned by DHCP server running in site server which contains the same transaction id that was in discover message, the proposed IP address of client, the IP address mask and a lease time indicating the duration of time the IP address is valid.
- Since several DHCP servers can be present on the subnet, the client may find itself in the enviable position of being able to choose from among several offers.
- ***DHCP request.*** This is the response to the DHCP offer message and contains the same parameters as they were present in the DHCP offer message.
- The newly arriving client will choose from among one or more server offers and respond to its selected offer with a DHCP request message, echoing back the configuration parameters.

- ***DHCP ACK.*** This is returned by the server and contains the same parameters indicating that the interaction is complete and client can use DHCP allocated IP

address for given duration.

- Since a client may want to use its address beyond the lease's expiration, DHCP also provides a mechanism that allows a client to renew its lease on an IP address. The value of DHCP's plug-and-play capability is clear, considering the fact that the alternative is to manually configure a host's IP address.

Q.15 Explain the working of Network Address Translation (NAT).

Ans

- As internet grew and IP address became scarce a mechanism discovered called NAT.
- The basic idea behind NAT is to provide an illusion, because a single IP address cannot be assigned to multiple computers-as conflict arises.
- Some IP addresses are reserved for sole purpose of private & intra-enterprise communications. These addresses are known as private IP addresses defined in RFC 1918.
- Each end system in a organization is been assigned a IP Address from private IP Pool.
- When any of the the end system requests a communication to internet with source IP of it, but private IP cannot be used for global communication. So a NAT enabled router is installed which acts as an interface between home network and internet, Router has its own unique IP over internet. So every request from organization gets transferred to NAT Router. NAT enabled maintains a table of source IP & destination IP of each and every request.
- NAT router eliminates private IP as source IP and its own IP as source IP address whereas, destination IP remains untouched.
- The request gets transferred to destination IP address and the response generated is received by NAT router which examines the table and forwards the response to end system.
- Figure 13 shows the operation of a ***NAT-enabled router***. The NAT-enabled router, residing in the home, has an interface that is part of the home network on the right of Figure 13.

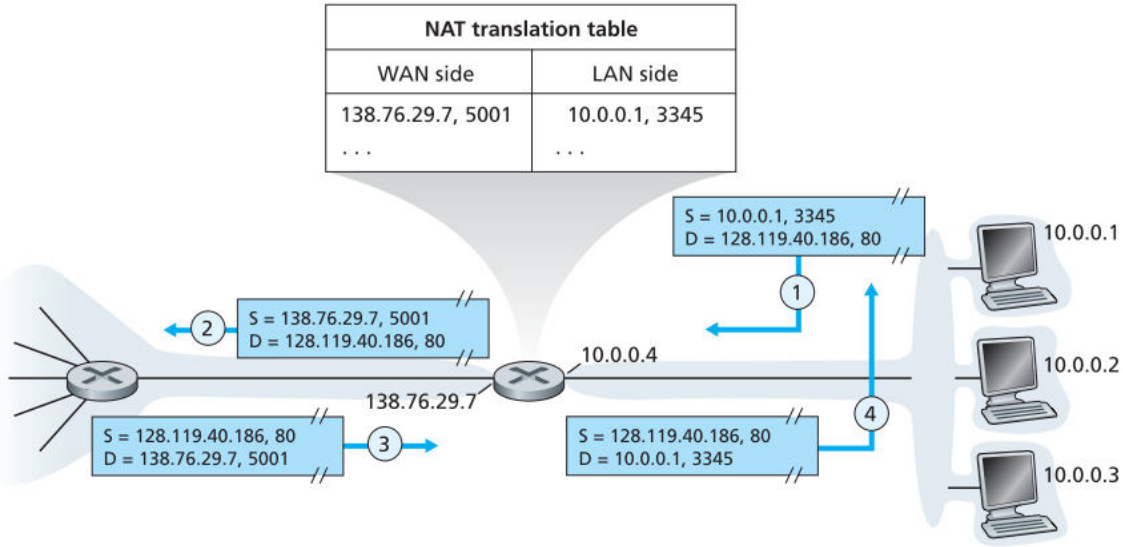


Figure 13: Network address translation

- Addressing within the home network is exactly as we have seen above—all four interfaces in the home network have the same subnet address of 10.0.0/24. The address space 10.0.0.0/8 is one of three portions of the IP address space that is reserved in for a private network or a ***realm*** with private addresses, such as the home network in Figure 13.
- ***A realm with private addresses refers to a network whose addresses only have meaning to devices within that network.***
- To see why this is important, consider the fact that there are hundreds of thousands of home networks, many using the same address space, 10.0.0.0/24.
- Devices within a given home network can send packets to each other using 10.0.0.0/24 addressing. However, packets forwarded beyond the home network into the larger global Internet clearly cannot use these addresses (as either a source or a destination address) because there are hundreds of thousands of networks using this block of addresses. That is, the 10.0.0.0/24 *addresses can only have meaning within the given home network*.
- *But if private addresses only have meaning within a given network, how is addressing handled when packets are sent to or received from the global Internet, where addresses are necessarily unique? The answer lies in understanding NAT.*
- The *NAT-enabled router does not look like a router to the outside world.* Instead the NAT router behaves to the outside world as a single device with a single IP address. In Figure 13, all traffic leaving the home router for the larger Internet has a source IP address of 138.76.29.7, and all traffic entering the home router must have a destination address of 138.76.29.7.
- In essence, the NAT-enabled router is hiding the details of the home network from the outside world. But where the home network computers get their addresses and where the router gets its single IP address. Often, the answer is the same—DHCP!
- The router gets its *address from the ISP's DHCP server, and the router runs a DHCP server to provide addresses to computers within the NAT-DHCP-router-controlled home network's address space.*
- When datagrams arrives at the NAT router from the WAN have the same destination IP address (specifically, that of the WAN-side interface of the NAT router), then router uses a NAT translation table at the NAT router, and to include port numbers as well as IP addresses in the table entries to determine internal host to which it should forward a given datagram.
- Consider the example in Figure 13. Following sequence of steps are going to execute,
  a) Suppose a user sitting in a home network behind host 10.0.0.1 requests a Web page on some Web server (port 80) with IP address 128.119.40.186.
  b) The host 10.0.0.1 assigns the (arbitrary) source port number 3345 and sends the datagram into the LAN.
  c) The NAT router receives the datagram, generates a new source port number 5001 for the datagram, replaces the source IP address with its WAN-side IP address 138.76.29.7, and replaces the original source port number 3345 with the new source port number 5001. When generating a new source port number, the NAT router can select any source port number that is not currently in the NAT

translation table. ***(Note that because a port number field is 16 bits long, the NAT protocol can support over 60,000 simultaneous connections with a single WAN-side IP address for the router!)***

d) NAT in the router also adds an entry to its NAT translation table.
e) The Web server, unaware that the arriving datagram containing the HTTP request has been manipulated by the NAT router, responds with a datagram whose destination address is the IP address of the NAT router, and whose destination port number is 5001.
f) When this datagram arrives at the NAT router, the router indexes the NAT translation table using the destination IP address and destination port number to obtain the appropriate IP address (10.0.0.1) and destination port number (3345) for the browser in the home network.
g) The router then rewrites the datagram's destination address and destination port number, and forwards the datagram into the home network.

Q.16 What is Link-State (LS) Routing Algorithm? Write Link-State (LS) Routing Algorithm.

Ans

- In a link-state algorithm, the network topology and all link costs are known, that is, available as input to the LS algorithm. In practice this is accomplished by having each node broadcast link-state packets to all other nodes in the network.
- Result of the nodes' broadcast is that all nodes have an identical and complete view of the network. Each node can then run the LS algorithm and compute the same set of least-cost paths as every other node.
- Dijkstra's algorithm computes the least-cost path from one node (the source, which we will refer to as u) to all other nodes in the network.
- Dijkstra's algorithm is iterative and has the property that after the k^{th} iteration of the algorithm, the least-cost paths are known to k destination nodes, and among the least-cost paths to all destination nodes, these k paths will have the k smallest costs. Let us define the following notation:
  - ✓ *D(v):* cost of the least-cost path from the source node to destination v as of this iteration of the algorithm.
  - ✓ *p(v):* previous node (neighbor of v) along the current least-cost path from the source to *v*.
  - ✓ *N':* subset of nodes; *v* is in *N'* if the least-cost path from the source to *v* is definitively known.
- The global routing algorithm consists of an initialization step followed by a loop. The number of times the loop is executed is equal to the number of nodes in the network. Upon termination, the algorithm will have calculated the shortest paths from the source node u to every other node in the network.

Link-State (LS) Algorithm for Source Node u:

```
1  Initialization:
2  N' = {u}
3  for all nodes v
```

```
4    if v is a neighbor of u
5       then D(v) = c(u,v)
6    else D(v) =  ∞
7
8  Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for each neighbor v of w and not in N':
12           D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14       least path cost to w plus cost from w to v */
15  until N'= N
```

-

Q.17 Write and explain the working of Link-State (LS) Routing Algorithm.

Ans

- Let us define the following notation:
  - ✓ *D(v):* cost of the least-cost path from the source node to destination v as of this iteration of the algorithm.
  - ✓ *p(v):* previous node (neighbor of v) along the current least-cost path from the source to *v*.
  - ✓ *N':* subset of nodes; *v* is in *N'* if the least-cost path from the source to *v* is definitively known.

Link-State (LS) Algorithm for Source Node u:

```
1  Initialization:
2  N' = {u}
3  for all nodes v
4    if v is a neighbor of u
5       then D(v) = c(u,v)
6    else D(v) =  ∞
7
8  Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for each neighbor v of w and not in N':
12           D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14       least path cost to w plus cost from w to v */
15  until N'= N
```

- As an example, let's consider the network in Figure 14 and compute the least-cost paths from u to all possible destinations. A tabular summary of the algorithm's computation is shown in Table 15, where each line in the table gives the values of the algorithm's variables at the end of the iteration. Let's consider the few first steps in detail.

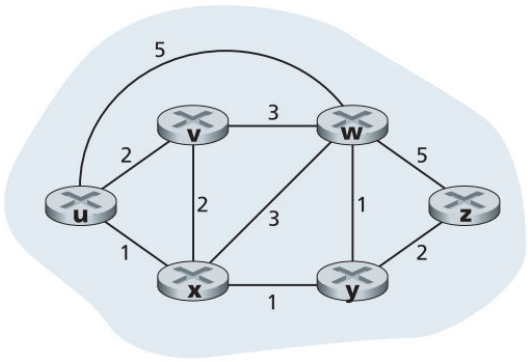


Figure 14: Abstract graph model of a computer network

step	N′	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

Figure 15: Running the link-state algorithm on the network in Figure 14

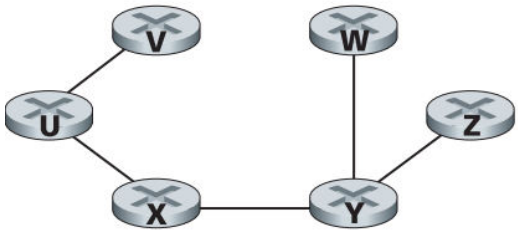
1.) In the initialization step, the currently known least-cost paths from u to its directly attached neighbors, v, x, and w, are initialized to 2, 1, and 5, respectively. Note in particular that the cost to w is set to 5 (even though we will soon see that a lesser-cost path does indeed exist) since this is the cost of the direct (one hop) link from u to w. The costs to y and z are set to infinity because they are not directly connected to u.

2.) In the first iteration, we look among those nodes not yet added to the set N and find that node with the least cost as of the end of the previous iteration. That node is x, with a cost of 1, and thus x is added to the set N' . Line 12 of the LS algorithm is then performed to update D(v) for all nodes v, yielding the results shown in the second line (Step 1) in Table 4.3. The cost of the path to v is unchanged. The cost of the path to w (which was 5 at the end of the initialization) through node x is found to have a cost of 4. Hence this lower-cost path is selected and w's predecessor along the shortest path from u is set to x. Similarly, the cost to y (through x) is computed to be 2, and the table is updated accordingly.

3.) In the second iteration, nodes v and y are found to have the least-cost paths (2), and we break the tie arbitrarily and add y to the set N' so that N' now contains u, x, and y. The cost to the remaining nodes not yet in N' , that is, nodes v, w, and z, are updated via line 12 of the LS algorithm, yielding the results shown in the third row in the Table 4.3.

4.) And so on. . . .

- When the LS algorithm terminates, we have, for each node, its predecessor along the least-cost path from the source node.



Destination	Link
v	(u, v)
w	(u, x)
x	(u, x)
y	(u, x)
z	(u, x)

Figure 16: Least cost path and forwarding table for nodule u

Q.18 Explain the basic idea behind Distance-Vector (DV) Routing Algorithm and also explain Dijskstra's equation.

Ans

- Whereas the LS algorithm is an algorithm using global information, the distance vector (DV) algorithm is ***iterative, asynchronous, and distributed***.
- ***It is distributed*** in that each node receives some information from one or more of its directly attached neighbors, performs a calculation, and then distributes the results of its calculation back to its neighbors.
- ***It is iterative*** in that this process continues on until no more information is exchanged between neighbors.
- ***The algorithm is asynchronous*** in that it does not require all of the nodes to operate in lockstep with each other.
- Let $d_x(y)$ be the cost of the least-cost path from node x to node y. Then the least costs are related by the Bellman-Ford equation, namely,

$$d_x(y) = min_v\{c(x,v) + d_v(y)\}, \quad \text{.....eq.[1]}$$

Where, min_v in the equation is taken over all of x's neighbors.

- Let's validate above equation, let's check it for source node u and destination node z in Figure 17. The source node u has three neighbors: nodes v, x, and w.
- By walking along various paths in the graph, it is easy to see that $d_v(z) = 5$, $d_x(z) = 3$, and $d_w(z) = 3$.
- Plugging these values into Equation[1], along with the costs $c(u,v) = 2$, $c(u,x) = 1$, and $c(u,w) = 5$, gives $d_u(z) = min\{2 + 5, 5 + 3, 1 + 3\} = 4$, which is obviously true and which is exactly what the Dijskstra algorithm gave us for the same network.

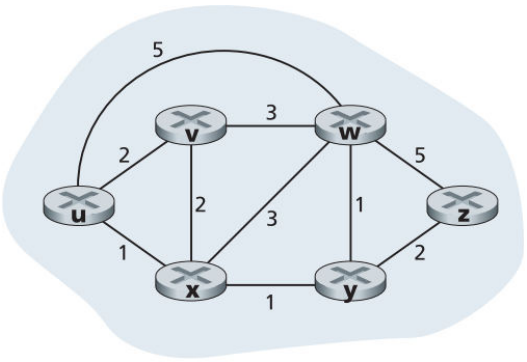


Figure 17: Abstract graph model of a computer network

- **<u>The basic idea is as follows</u>**. Each node x begins with $D_x(y)$, an estimate of the cost of the least-cost path from itself to node y, for all nodes in N. Let $D_x = [D_x(y): y \text{ in } N]$ be node x's distance vector, which is the vector of cost estimates from x to all other nodes, y, in N. With the DV algorithm, each node x maintains the following routing information:
  - ✓ For each neighbor v, the cost $c(x,v)$ from x to directly attached neighbor, v
  - ✓ Node x's distance vector, that is, $D_x = [D_x(y): y \text{ in } N]$, containing x's estimate of its cost to all destinations, y, in N
  - ✓ The distance vectors of each of its neighbors, that is, $D_v = [D_v(y): y \text{ in } N]$ for each neighbor v of x
- In the distributed, asynchronous algorithm, from time to time, each node sends a copy of its distance vector to each of its neighbors. When a node x receives a new distance vector from any of its neighbors v, it saves v's distance vector, and then uses the Bellman-Ford equation to update its own distance vector as follows:

$$D_x(y) = \min_v\{c(x,v) + D_v(y)\} \text{ for each node } y \text{ in } N$$

- If node x's distance vector has changed as a result of this update step, node x will then send its updated distance vector to each of its neighbors, which can in turn update their own distance vectors. As long as all the nodes continue to exchange their distance vectors in an asynchronous fashion, each cost estimate $D_x(y)$ converges to $d_x(y)$, the actual cost of the least-cost path from node x to node y.

Q.19 Write and explain Distance-Vector (DV) Algorithm.

Ans ***<u>Algorithm:</u>***

At each node, x:

```
1  Initialization:
2      for all destinations y in N:
3          Dx(y)=c(x,y) /*if y is not a neighbor then c(x,y)=∞*/
4      for each neighbor w
5          Dw(y)=? for all destinations y in N
```

```
6       for each neighbor w
7          send distance vector  Dx=[Dx(y):y in N] to w
8
9   loop
10      wait(until I see a link cost change to some neighbor w
11           or  until I receive a distance vector from some  neighbor w)
12      for each y in N:
13         Dx(y) = minv{c(x,v) + Dv(y)}
14
15      if Dx(y) changed for any destination y
16           send distance vector  D x= [Dx(y): y in N] to all neighbors
17
18   Forever
```

Explanation:

- In the DV algorithm, a node x updates its distance-vector estimate when it either sees a cost change in one of its directly attached links or receives a distance-vector update from some neighbor.
- But to update its own forwarding table for a given destination y, what node x really needs to know is not the shortest-path distance to y but instead the neighboring node v*(y) that is the next-hop router along the shortest path to y.
- As you might expect, the next-hop router v*(y) is the neighbor v that achieves the minimum in Line 14 of the DV algorithm. (If there are multiple neighbors v that achieve the minimum, then v*(y) can be any of the minimizing neighbors.) Thus, in Lines 13–14, for each destination y, node x also determines v*(y) and updates its forwarding table for destination y. vector to its neighbors (Lines 16–17).
- Figure 18 illustrates the operation of the DV algorithm for the simple three-node network shown at the top of the figure.
- The operation of the algorithm is illustrated in a synchronous manner, where all nodes simultaneously receive distance vectors from their neighbors, compute their new distance vectors, and inform their neighbors if their distance vectors have changed.
- The leftmost column of the figure displays three initial routing tables for each of the three nodes. For example, the table in the upper-left corner is node x's initial routing table.
- Within a specific routing table, each row is a distance vector—specifically, each node's routing table includes its own distance vector and that of each of its neighbors. Thus, the first row in node x's initial routing table is $D_x = [D_x(x), D_x(y), D_x(z)] = [0, 2, 7]$.
- The second and third rows in this table are the most recently received distance vectors from nodes y and z, respectively. Because at initialization node x has not received anything from node y or z, the entries in the second and third rows are initialized to infinity.
- After initialization, each node sends its distance vector to each of its two neighbors. This is illustrated in Figure 18 by the arrows from the first column of tables to the

second column of tables. For example, node x sends its distance vector $D_x=[0, 2, 7]$ to both nodes y and z. After receiving the updates, each node recomputes its own distance vector. For example, node x computes

$D_x(x) = 0$

$D_x(y) = min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} = min\{2 + 0, 7 + 1\} = 2$

$D_x(z) = min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\} = min\{2 + 1, 7 + 0\} = 3$

- The second column therefore displays, for each node, the node's new distance vector along with distance vectors just received from its neighbors. Note, for example,that node x's estimate for the least cost to node z, $D_x(z)$, has changed from 7 to 3.
- Also note that for node x, neighboring node y achieves the minimum in line 14 of the DV algorithm; thus at this stage of the algorithm, we have at node x that v*(y) = y and v*(z) = y.
- After the nodes recompute their distance vectors, they again send their updated distance vectors to their neighbors (if there has been a change). This is illustrated in Figure 18 by the arrows from the second column of tables to the third column of tables.
- Note that only nodes x and z send updates: node y's distance vector didn't change so node y doesn't send an update. After receiving the updates, the nodes then recompute their distance vectors and update their routing tables, which are shown in the third column.
- The process of receiving updated distance vectors from neighbors, recomputing routing table entries, and informing neighbors of changed costs of the least-cost path to a destination continues until no update messages are sent.

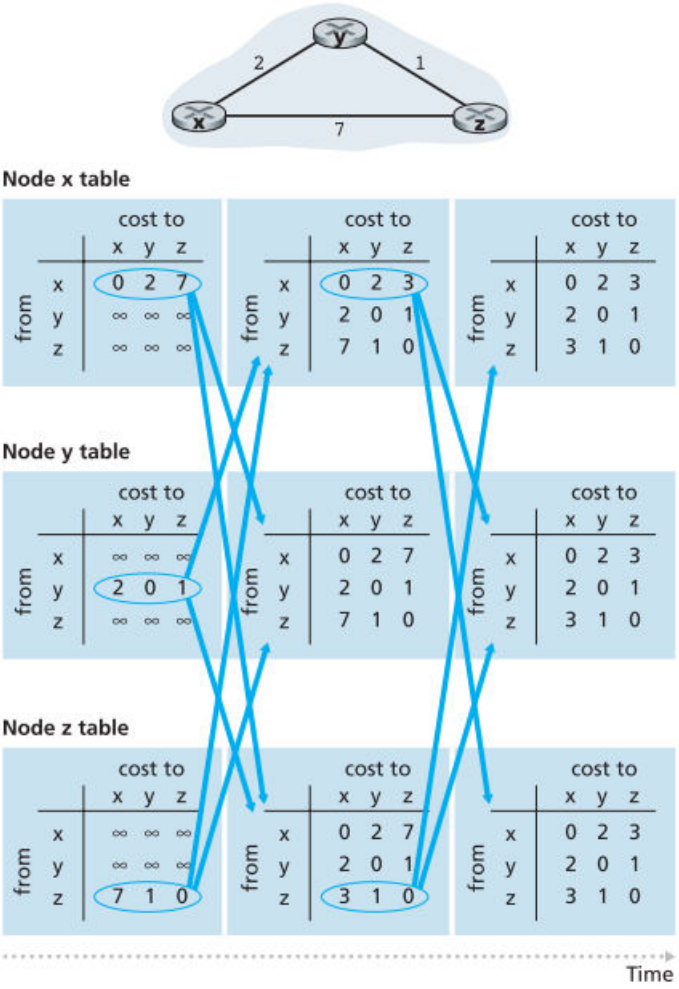


Figure 18: Distance-vector (DV) algorithm

- At this point, since no update messages are sent, no further routing table calculations will occur and the algorithm will enter a quiescent state; that is, all nodes will be performing the wait in Lines 10–11 of the DV algorithm. The algorithm remains in the quiescent state until a link cost changes.
-

Q.20 Explain Poisoned Reverse in distance vector algorithm.

Ans

- The specific looping scenario just described can be avoided using a technique known as poisoned reverse. The idea is simple—if z routes through y to get to destination x, then z will advertise to y that its distance to x is infinity, that is, z will advertise to y that $D_z(x) = \infty$ (even though z knows $D_z(x) = 5$ in truth).
- z will continue telling this little white lie to y as long as it routes to x via y. Since y believes that z has no path to x, y will never attempt to route to x via z, as long as z continues to route to x via y (and lies about doing so).
- Let's now see how poisoned reverse solves the particular looping problem we encountered before in Figure 19. As a result of the poisoned reverse, y's distance table indicates $D_z(x) = \infty$.

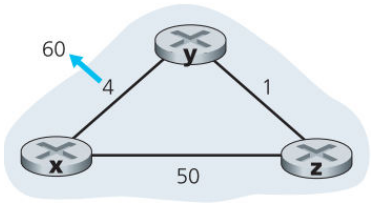


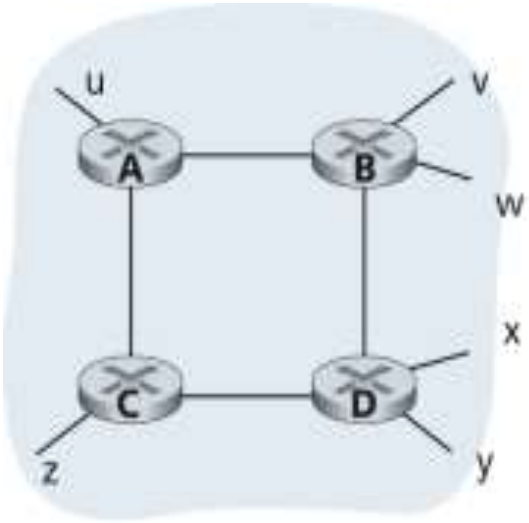
Figure 19: Changes in link cost

- When the cost of the (x, y) link changes from 4 to 60 at time t_0, y updates its table and continues to route directly to x, albeit at a higher cost of 60, and informs z of its new cost to x, that is, $D_y(x) = 60$. After receiving the update at t_1, z immediately shifts its route to x to be via the direct (z, x) link at a cost of 50. Since this is a new least-cost path to x, and since the path no longer passes through y, z now informs y that $D_z(x) = 50$ at t_2.
- After receiving the update from z, y updates its distance table with $D_y(x) = 51$. Also, since z is now on y's least-cost path to x, y poisons the reverse path from z to x by informing z at time t_3 that $D_y(x) = \infty$ (even though y knows that $D_y(x) = 51$ in truth).

Q.21 Describe an Intra-AS Routing in the Internet using **RIP protocol**.

Ans

- An intra-AS routing protocol is used to determine how routing is performed within an **autonomous system** (AS). Intra-AS routing protocols are also known as *interior gateway protocols.*
- Historically , two routing protocols have been used extensively for routin g within an autonomous system in the Internet : the **Routing Information Protocol** (RIP) and**Open Shortest Path First** (OSPF). A routing protocol closely related to OSPF is theIS - IS **(Intermediate system - intermediate system)** protocol
- RIP was one of the earliest intra-AS Internet routing protocols and is still in **widespread** use today
- 'RIP is a distance-vector protocol that operates in a manner very close to the idealized **Distance Vector** (DV) protocol . The version of RIP specified in uses hop count as a cost metric ; that is, each link has a cost of 1. In the DV algorithm in Section 4.5.2, for simplicity, costs were defined between pairs of routers.
- In RIP (and also in OSPF), costs are actually from source router to a destination subnet. RIP uses the term hop, which is the number of subnets traversed along the shortest path from source router to destination subnet, including the destination subnet.
- Figure 4.34 illustrates an AS with six leaf subnets. The table in the figure indicates the number of hops from the source A to each of the leaf subnets. The maximum cost of a path is limited to 15, thus limiting the use of RIP to autonomous systems that are fewer than 15 hops in diameter.



Destination	Hops
u	1
v	2
w	2
x	3
y	3
z	2

Figure 20: Number of hops from source router A to various subnets

- In RIP, routing updates are exchanged between neighbors approximately every 30 seconds using a RIP response message. The response message sent by a router or host contains a list of up to 25 destination subnets within the AS, as well as the sender's distance to each of those subnets.
- Response messages are also known as RIP advertisements. Let's take a look at a simple example of how RIP advertisements work. Consider the portion of an AS shown in Figure 4.35. In this figure, lines connecting the routers denote subnets. Only selected routers (A, B, C, and D) and subnets (w, x, y, and z) are labeled. Dotted lines indicate that the AS continues on; thus this autonomous system has many more routers and links than are shown.
- Each router maintains a RIP table known as a **routing table**. A router's routing table includes both the router's distance vector and the router's forwarding table.
- Figure 4.36 shows the routing table for router D. Note that the routing table has three columns.
- The first column is for the destination subnet, the second column indicates the identity of the next router along the shortest path to the destination subnet, and the third column indicates the number of hops (that is, the number of subnets that have to be traversed, including the destination subnet) to get to the destination subnet along the shortest path.
- For this example, the table indicates that to send a datagram from router D to destination subnet w, the datagram should first be forwarded to neighboring router A; the table also indicates that destination subnet w is two hops away along the shortest path.
- Similarly, the table indicates that subnet z is seven hops away via router B. The table in Figure 4.36, and the subsequent tables to come, are only partially complete.
- Now suppose that 30 seconds later, router D receives from router A the advertisement shown in Figure 4.37. Note that this advertisement is nothing other than the routing table information from router A! This information indicates, in particular, that subnet z is only four hops away from router A.
- Router D, upon receiving this advertisement, merges the advertisement (Figure 4.37) with the old routing table (Figure 4.36). In particular, router D learns that there is now a path through router A to subnet z that is shorter than the path through router B. Thus, router D updates its routing table to account for the shorter shortest path, as shown in Figure 4.38.

- Let's next consider a few of the implementation aspects of RIP. Recall that RIP routers exchange advertisements approximately every 30 seconds. If a router does not hear from its neighbor at least once every 180 seconds, that neighbor is considered to be no longer reachable; that is, either the neighbor has died or the connecting link has gone down. When this happens, RIP modifies the local routing table and then propagates this information by sending advertisements to its neighboring routers (the ones that are still reachable).

- A router can also request information about its neighbor's cost to a given destination using RIP's request message. Routers send RIP request and response messages to each other over UDP using port number 520. The UDP segment is carried between routers in a standard IP datagram.

Q.22 Describe an Intra-AS Routing in the Internet using **OSPF protocol**. OR

Discus basics of OSPF and write advancement included in OSPF protocol.

Ans

- Like RIP, OSPF routing is widely used for intra-AS routing in the Internet. OSPF and its closely related cousin, IS-IS, are typically deployed in upper-tier ISPs whereas RIP is deployed in lower-tier ISPs and enterprise networks.
- OSPF was conceived as the successor to RIP and as such has a numb er of advanced features. At its heart, however, OSPF is a **link-state protocol** that uses flooding of link -state information and a Dijkstra least -cost path algorithm.
- With OSPF, a router constructs a complete topological map (that is, a graph) of the entire autonomous system. The router then locally runs **Dijkstra 's** shortest -path algorithm to determine a shortest -path tree to all subnets, with itself as the root node.
- Individual link costs are configured by the **network administrator**. The administrator might choose to set all link costs to 1, thus achieving minimum -hop routing, or might choose to set the link weights to be inversely proportional to link capacity in order to discourage traffic from using low-bandwidth links. OSPF does not mandate a policy for how link weights are set **(that is the job of the network administrator**), but instead provides the mechanisms (protocol) for determining least -cost path routing for the given set of link weights.
- With OSPF, a router broadcasts routing information to all other routers in the autonomous system, not just to its neighboring routers. A router broadcasts link-state information whenever there is a change in a link's state (for example, a change in cost or a change in up/down status). It also broadcasts a link's state periodically (at least once every 30 minutes), even if the link's state has not changed. This periodic updating of link state advertisements adds robustness to the link state algorithm.

- OSPF advertisements are contained in OSPF messages that are carried directly by IP, with an upper -layer protocol of 89 for OSPF . Thus , the OSPF protocol must itself implement functionality such as reliable message transfer and link-state broadcast .
- The OSPF protocol also checks that links are operational (via a HELLO message that is sent to an attached neighbour and allows an OSPF router to obtain a neighboring router's database of network-wide link state.

- Some of the advances embodied in OSPF include the following:

- ***Security***. Exchanges between OSPF routers (for example, link-state updates) can be authenticated. With authentication, only trusted routers can participate in the OSPF protocol within an AS, thus preventing malicious intruders (or networking students taking their newfound knowledge out for a joyride) from injecting incorrect information into router tables. By default, OSPF packets between routers are not authenticated and could be forged.

- Two types of authentication can be configured —simple and **MD5 (Message Digest algorithm 5**) With simple authentication , the same password is configured on each router . When a router sends an OSPF packet , it includes the password in plaintext . Clearly , simple authentication is not very secure . MD5 authentication is based on shared secret keys that are configured in all the routers.
- For each OSPF packet that it sends, the router computes the MD5 hash of the content of the OSPF packet appended with the secret key.
- Then the router includes the resulting hash value in the OSPF packet. The receiving router, using the preconfigured secret key, will compute an MD5 hash of the packet and compare it with the hash value that the packet carries, thus verifying the packet's authenticity. Sequence numbers are also used with MD5 authentication to protect against replay attacks.

- ***Multiple same-cost paths.*** When multiple paths to a destination have the same cost, OSPF allows multiple paths to be used (that is, a single path need not be chosen for carrying all traffic when multiple equal-cost paths exist).
- ***Integrated support for unicast and multicast routing.*** Multicast OSPF (MOSPF) provides simple extensions to OSPF to provide for multicast routing MOSPF uses the existing OSPF link database and adds a new type of link-state advertisement to the existing OSPF link-state broadcast mechanism.
- ***Support for hierarchy within a single routing domain.*** Perhaps the most significant advance in OSPF is the ability to structure an autonomous system hierarchically.

- An OSPF autonomous system can be configured hierarchically into areas. Each area runs its own OSPF link-state routing algorithm, with each router in an area broadcasting its link state to all other routers in that area. Within each area, one or more area border routers are responsible for routing packets outside the area.
- Lastly, exactly one OSPF area in the AS is configured to be the backbone area. The *primary role of the backbone area is to route traffic between the other areas in the AS.*

- The backbone always contains all area border routers in the AS and may contain *nonborder* routers as well. Inter-area routing within the AS requires that the packet be first routed to *an area border router* (intra -area routing), then routed through the back-bone to the area border router that is in the destination area, and then routed to the final destination.

Q.23 Explain Inter-AS Routing with **BGP protocol. OR**
Explain the Basics of BGP.

Ans

- How paths are determined for source-destination pairs that span multiple ASs. The ***Border Gateway Protocol version 4***, is the standard inter -AS routing protocol in today 's Internet. It is commonly referred to as ***BGP4*** or simply as ***BGP***.
- As an inter-AS routing protocol, BGP provides each AS a means to
  1 Obtain subnet reachability information from neighboring ASs.
  2 Propagate the reachability information to all routers internal to the AS.
  3 Determine "good" routes to subnets based on the reachability information and on AS policy.
- Most importantly, BGP allows each subnet to advertise its existence to the rest of the Internet. A subnet screams "I exist and I am here," and BGP makes sure that all the ASs in the Internet know about the subnet and how to ge t there. If it weren't forBGP, each subnet would be isolated —alone and unknown by the rest of the Internet .▪ In BGP, pairs of routers exchange routing information over **semipermanent TCP connections using port 179**. The semi -permanent TCP connections for the network are shown in Figure 21.

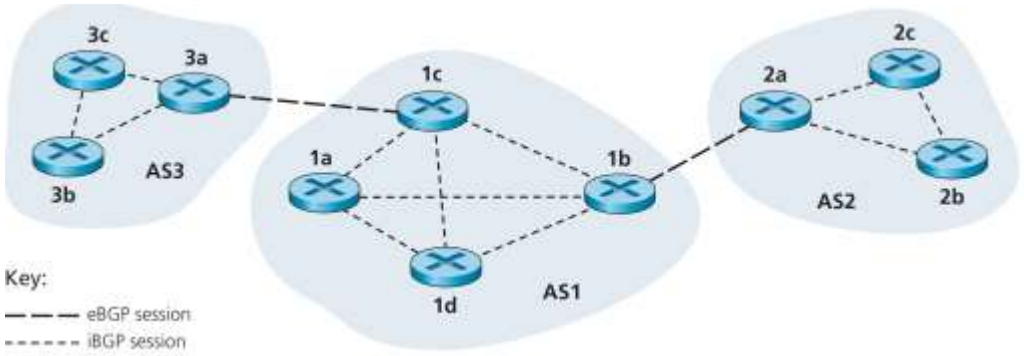


Figure 21: eBGP and iBGP sessions

- There is typically one such BGP TCP connection for each link that directly connects two routers in two different ASs; thus, in Figure 12, there is a TCP connection between gateway routers 3a and 1c and another TCP connection between gateway routers 1b and 2a.
- There are also semipermanent BGP TCP connections between routers within an AS. In particular, Figure 21 displays a common configuration of one TCP connection for each pair of routers internal to an AS, creating a mesh of TCP connections within

each AS.

- For each TCP connection, the two routers at the end of the connection are called **BGP peers**, and the TCP connection along with all the BGP messages sent over the connection is called a **BGP session**. Furthermore, a BGP session that spans two Ass is called an **external BGP (eBGP) session**, and a BGP session between routers in the same AS is called an **internal BGP (iBGP) session**. BGP allows each AS to learn which destinations are reachable via its neighboring ASs.
- In BGP, destinations are not hosts but instead are ***CIDRized prefixes***, with each prefix representing a subnet or a collection of subnets. Thus, for example, suppose there are four subnets attached to AS2: 138.16.64/24, 138.16.65/24, 138.16.66/24, and 138.16.67/24. Then AS2 could aggregate the prefixes for these four subnets and use BGP to advertise the single prefix to 138.16.64/22 to AS1.
- As another example, suppose that only the first three of those four subnets are in AS2 and the fourth subnet, 138.16.67/24, is in AS3. Because routers use longest-prefix matching for forwarding datagrams, AS3 could advertise to AS1 the more specific prefix 138.16.67/24 and AS2 could still advertise to AS1 the aggregated prefix 138.16.64/22.
- Let's now examine how BGP would distribute prefix reachability information over the BGP sessions shown in Figure 4.40. As you might expect, using the eBGP session between the gateway routers 3a and 1c, AS3 sends AS1 the list of prefixes that are reachable from AS3; and AS1 sends AS3 the list of prefixes that are reachable from AS1.
- Similarly, AS1 and AS2 exchange prefix reachability information through their gateway routers 1b and 2a. Also as you may expect, when a gateway router (in any AS) receives eBGP-learned prefixes, the gateway router uses its iBGP sessions to distribute the prefixes to the other routers in the AS.
- Thus, all the routers in AS1 learn about AS3 prefixes, including the gateway router 1b. The gateway router 1b (in AS1) can therefore re-advertise AS3's prefixes to AS2. When a router (gateway or not) learns about a new prefix, it creates an entry for the prefix in its forwarding table.

Q.24 Explain Broadcast Routing Algorithms techniques

I. **Uncontrolled Flooding**
II. **Controlled Flooding in**

Uncontrolled Flooding

- The most obvious technique for achieving broadcast is a flooding approach in which the source node sends a copy of the packet to all of its neighbors. When a node receives a broadcast packet, it duplicates the packet and forwards it to all of its neighbors.
- Clearly, if the graph is connected, this scheme will eventually deliver a copy of the broadcast packet to all nodes in the graph. Although this scheme is simple and elegant, it has a fatal flaw.

- If the graph has cycles, then one or more copies of each broadcast packet will cycle indefinitely.

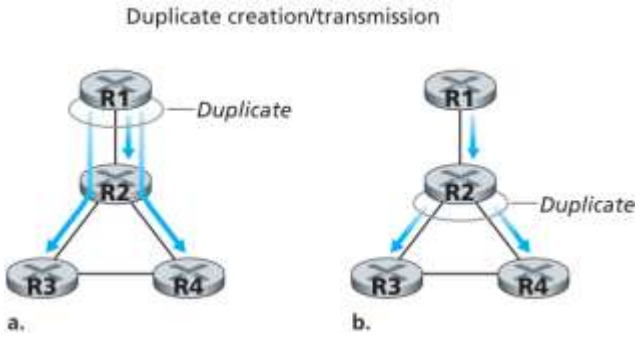


Figure 22: Source-duplication versus in-network duplication

- For example, in Figure 22, R2 will flood to R3, R3 will flood to R4, R4 will flood to R2, and R2 will flood (again!) to R3, and so on. This simple scenario results in the endless cycling of two broadcast packets, one clockwise, and one counterclockwise.
- But there can be an even more calamitous fatal flaw: When a node is connected to more than two other nodes, it will create and forward multiple copies of the broadcast packet, each of which will create multiple copies of itself (at other nodes with more than two neighbours), and so on.
- This broadcast storm, resulting from the endless multiplication of broadcast packets, would eventually result in so many broadcast packets being created that the network would be rendered useless.

Controlled Flooding

- The key to avoiding a broadcast storm is for a node to judiciously choose when to flood a packet and (e.g., if it has already received and flooded an earlier copy of a packet) when not to flood a packet.
- In practice, this can be done in one of several way. In sequence-number-controlled flooding, a source node puts its address (or other unique identifier) as well as a broadcast sequence number into a broadcast packet, then sends the packet to all of its neighbours.
- Each node maintains a list of the source address and sequence number of each broadcast packet it has already received, duplicated, and forwarded. When a node receives a broadcast packet, it first checks whether the packet is in this list. If so, the packet is dropped; if not, the packet is duplicated and forwarded to all the node's neighbors (except the node from which the packet has just been received).
- A second approach to controlled flooding is known as reverse path forwarding (RPF), also sometimes referred to as reverse path broadcast (RPB).
- The idea behind RPF is simple, yet elegant. When a router receives a broadcast packet with a given source address, it transmits the packet on all of its outgoing links (except the one on which it was received) only if the packet arrived on the link that is on its own shortest unicast path back to the source. Otherwise, the router simply

discards the incoming packet without forwarding it on any of its outgoing links.

- Such a packet can be dropped because the router knows it either will receive or has already received a copy of this packet on the link that is on its own shortest path back to the sender.)
- Note that RPF does not use unicast routing to actually deliver a packet to a destination, nor does it require that a router know the complete shortest path from itself to the source.
- RPF need only know the next neighbor on its unicast shortest path to the sender; it uses this neighbor's identity only to determine whether or not to flood a received broadcast packet. Figure 4.44 illustrates RPF. Suppose that the links drawn with thick lines represent the least-cost paths from the receivers to the source (A). Node A initially broadcasts a source-A packet to nodes C and B. Node B will forward the source-A packet it has received from A (since A is on its least-cost path to A) to both C and D.

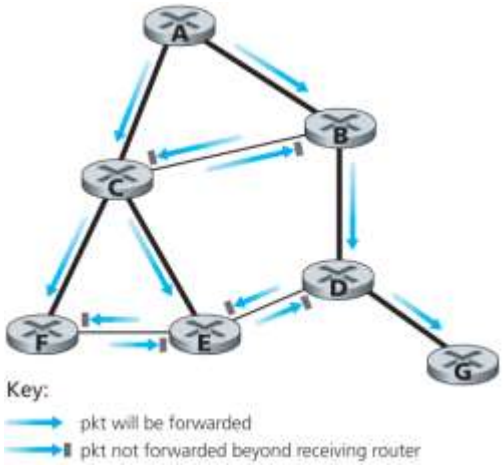


Figure 23: Reverse path forwarding

- B will ignore (drop, without forwarding) any source-A packets it receives from any other nodes (for example, from routers C or D). Let us now consider node C, which will receive a source-A packet directly from A as well as from B. Since B is not on C's own shortest path back to A, C will ignore any source-A packets it receives from B.
- On the other hand, when C receives a source-A packet directly from A, it will forward the packet to nodes B, E, and F.

Q.25 Explain method of broadcasting using "***Spanning-Tree Broadcast"***.

Ans

- Ideally, every node should receive only one copy of the broadcast packet. Examining the tree consisting of the nodes connected by thick lines in Figure 24(a), you can see that if broadcast packets were forwarded only along links within this tree, each and every network node would receive exactly one copy of the broadcast packet—exactly the solution we were looking for!
- This tree is an example of a ***spanning tree***—a tree that contains each and every node

in a graph. More formally, a spanning tree of a graph G = (N,E) is a graph G'= (N,E') such that E' is a subset of E, G' is connected, G' contains no cycles, and G' contains all the original nodes in G.

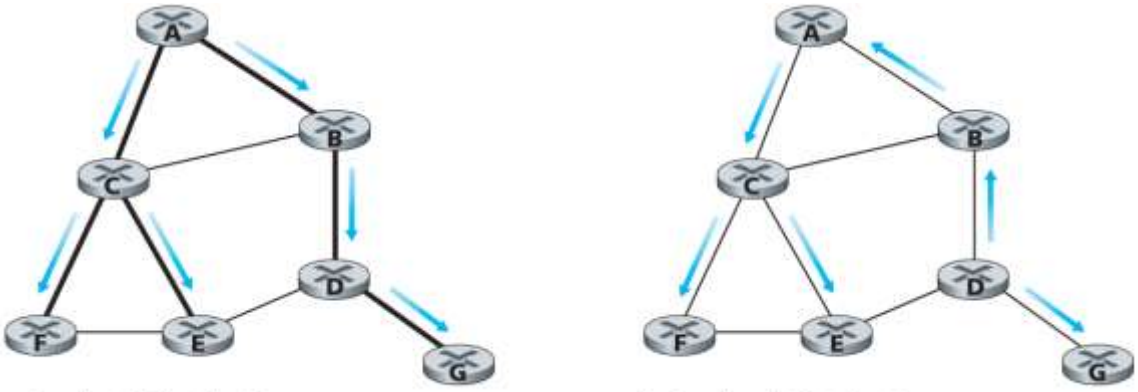
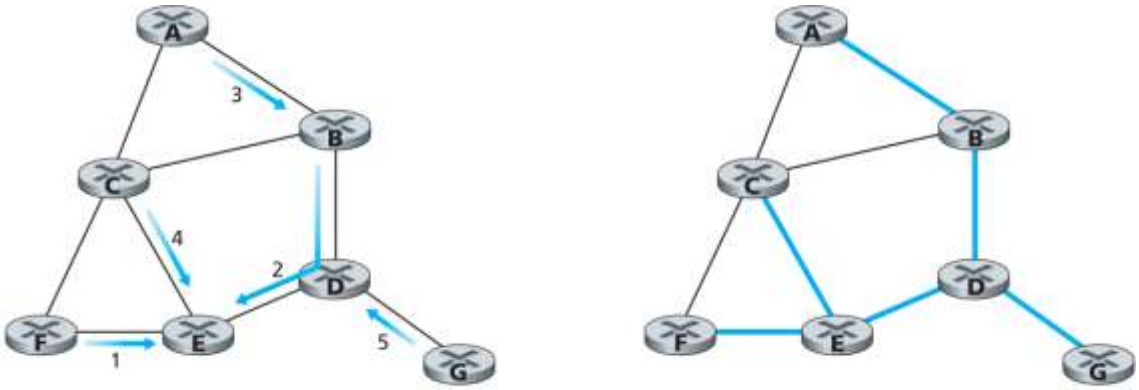


Figure 24: Broadcast along a spanning tree

- If each link has an associated cost and the cost of a tree is the sum of the link costs, then a spanning tree whose cost is the minimum of all of the graph's spanning trees is called ***a minimum spanning tree***.
- Thus, another approach to providing broadcast is for the network nodes to first construct a spanning tree. When a source node wants to send a broadcast packet, it sends the packet out on all of the incident links that belong to the spanning tree.
- A node receiving a broadcast packet then forwards the packet to all its neighbors in the spanning tree (except the neighbor from which it received the packet). Not only does spanning tree eliminate redundant broadcast packets, but once in place, the spanning tree can be used by any node to begin a broadcast, as shown in Figures 24(a) and 24(b). Note that a node need not be aware of the entire tree; it simply needs to know which of its neighbors in G spanning-tree neighbors are.
- The main complexity associated with the spanning-tree approach is the creation and maintenance of the spanning tree. Numerous distributed spanning-tree algorithms have been developed.
- In the center-based approach to building a spanning tree, a center node (also known as a ***rendezvous point*** or a ***core***) is defined. Nodes then unicast tree-join messages addressed to the center node. A tree-join message is forwarded using unicast routing toward the center until it either arrives at a node that already belongs to the spanning tree or arrives at the center.
- In either case, the path that the tree-join message has followed defines the branch of the spanning tree between the edge node that initiated the tree-join message and the center.
- One can think of this new path as being grafted onto the existing spanning tree. Figure 4.46 illustrates the construction of a center-based spanning tree. Suppose that node E is selected as the center of the tree.

- Suppose that node F first joins the tree and forwards a tree-join message to E. The single link EF becomes the initial spanning tree. Node B then joins the spanning tree by sending its tree-join message to E.



a. Stepwise construction of spanning tree

b. Constructed spanning tree

Figure 25: Center-based construction of a spanning tree

- Suppose that the unicast path route to E from B is via D. In this case, the tree-join message results in the path BDE being grafted onto the spanning tree. Node A next joins the spanning group by forwarding its tree-join message towards E.
- If A's unicast path to E is through B, then since B has already joined the spanning tree, the arrival of A's tree-join message at B will result in the AB link being immediately grafted onto the spanning tree. Node C joins the spanning tree next by forwarding its tree-join message directly to E. Finally, because the unicast routing from G to E must be via node D, when G sends its tree-join message to E, the GD link is grafted onto the spanning tree at node D.

Q.26 Write short note on **Internet Group Management Protocol** *(IGMP)*.

Ans

- The IGMP protocol version 3 operates between a host and its directly attached router (informally, we can think of the directly attached router as the first -hop router that a host would see on a path to any other host outside its own local network, or the last-hop router on any path to that host), as shown in Figure 26.
- Figure 26 shows three first-hop multicast routers, each connected to its attached hosts via one outgoing local interface. This local interface is attached to a LAN in this example, and while each LAN has multiple attached hosts, at most a few of these hosts will typically belong to a given multicast group at any given time.
- IGMP provides the means for a host to inform its attached router that an application running on the host wants to join a specific multicast group. Given that the scope of IGMP interaction is limited to a host and its attached router, another protocol is clearly required to coordinate the multicast routers (including the attached routers) throughout the Internet, so that multicast datagrams are routed to their final destinations.
- This latter functionality is accomplished by network-layer multicast routing

algorithms, such as those we will consider shortly. Network-layer multicast in the Internet thus consists of two complementary components: IGMP and multicast routing protocols.

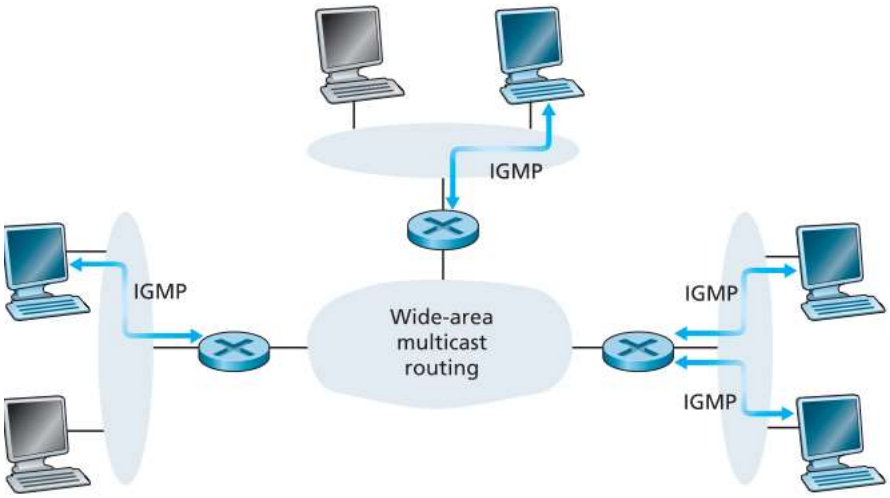


Figure 26 : The two components of network-layer multicast in the Internet: IGMP and multicast routing protocols

- IGMP has only three message types. Like ICMP, IGMP messages are carried (encapsulated) within an IP datagram, with an IP protocol number of 2. The `membership_query` message is sent by a router to all hosts on an attached interface (for example, to all hosts on a local area network) to determine the set of all multicast groups that have been joined by the hosts on that interface. Hosts respond to a `membership_query` message with an IGMP `membership_report` message.
- `membership_report` messages can also be generated by a host when an application first joins a multicast group without waiting for a `membership_query` message from the router.
- The final type of IGMP message is the `leave_group` message. Interestingly, this message is optional. But if it is optional, how does a router detect when a host leaves the multicast group? The answer to this question is that the router infers that a host is no longer in the multicast group if it no longer responds to a `membership_query` message with the given group address.
- This is an example of what is sometimes called soft state in an Internet protocol. In a ***soft state protocol,*** the state (in this case of IGMP, the fact that there are hosts joined to a given multicast group) is removed via a timeout event (in this case, via a periodic `membership_query` message from the router) if it is not explicitly refreshed (in this case, by a `membership_report` message from an attached host).
  [NOTE: It has been argued that soft-state protocols result in simpler control than ***hard-state protocols***, which not only require state to be explicitly added and removed, but also require mechanisms to recover from the situation where the entity responsible for removing state has terminated prematurely or failed.]

Q.27 Discus-Multicast Routing problem.

Ans

- The multicast routing problem is illustrated in Figure 4.49. Hosts joined to the multicast group are shaded in color; their immediately attached router is also shaded in color.
- As shown in Figure 27, only a subset of routers (those with attached hosts that are joined to the multicast group) actually needs to receive the multicast traffic. In Figure 27, only routers A, B, E, and F need to receive the multicast traffic. Since none of the hosts attached to router D are joined to the multicast group and since router C has no attached hosts, neither C nor D needs to receive the multicast group traffic.
- The goal of multicast routing, then, is to find a tree of links that connects all of the routers that have attached hosts belonging to the multicast group. Multicast packets will then be routed along this tree from the sender to all of the hosts belonging to the multicast tree.

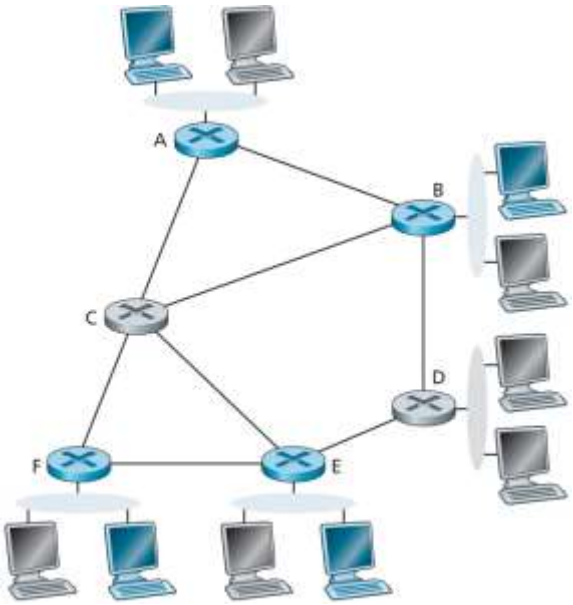


Figure 27: Multicast hosts, their attached routers, and other routers

- Of course, the tree may contain routers that do not have attached hosts belonging to the multicast group (for example, in Figure 27, it is impossible to connect routers A, B, E, and F in a tree without involving either router C or D).
- In practice, two approaches have been adopted for determining the multicast routing tree,

Multicast routing using a group-shared tree. As in the case of spanning-tree broadcast, multicast routing over a group-shared tree is based on building a tree that includes all edge routers with attached hosts belonging to the multicast group.

- In practice, a center-based approach is used to construct the multicast routing tree, with edge routers with attached hosts belonging to the multicast group sending (via unicast) join messages addressed to the center node. As in the broadcast case, a join

message is forwarded using unicast routing toward the center until it either arrives at a router that already belongs to the multicast tree or arrives at the center.

- All routers along the path that the join message follows will then forward received multicast packets to the edge router that initiated the multicast join. A critical question for center-based tree multicast routing is the process used to select the center.

- ***Multicast routing using a source-based tree.*** While group-shared tree multicast routing constructs a single, shared routing tree to route packets from all senders, the second approach constructs a multicast routing tree for each source in the multicast group. In practice, an RPF algorithm (with source node x) is used to construct a multicast forwarding tree for multicast datagrams originating at source x. The RPF broadcast algorithm we studied earlier requires a bit of tweaking for use in multicast.
- To see why, consider router D in Figure 28. Under broadcast RPF, it would forward packets to router G, even though router G has no attached hosts that are joined to the multicast group.

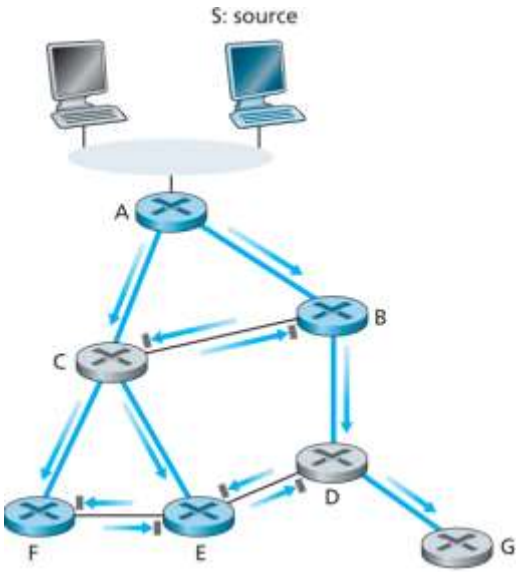


Figure 28: Reverse path forwarding, the multicast case

- While this is not so bad for this case where D has only a single downstream router, G, imagine what would happen if there were thousands of routers downstream from D! Each of these thousands of routers would receive unwanted multicast packets.
- The solution to the problem of receiving unwanted multicast packets under RPF is known as ***pruning***. A multicast router that receives multicast packets and has no attached hosts joined to that group will send a prune message to its upstream router. If a router receives prune messages from each of its downstream routers, then it can forward a prune message upstream.