UNIT – 3

NETWORK LAYER

The network layer is concerned with getting packets from the source all the way to the destination. Getting to the destination may require making many hops at intermediate routers along the way. the network layer is the lowest layer that deals with end-to-end transmission. To achieve its goals, the network layer must know about the topology of the network (i.e., the set of all routers and links) and choose appropriate paths through it, even for large networks.

3.1 NETWORK LAYER DESIGN ISSUES:

- **Store-And-Forward Switching**
- **Services Provided To Transport Layer**
- **Implementation Of Connection Less Service**
- **Implementation Of Connection-Oriented Service**
- **Comparison Of Virtual-Circuit And Datagram Subnet**

Store-And-Forward Switching :

The major components of the system are the carrier's equipment, shown inside the shaded oval, and the customers' equipment, shown outside the oval. Host *H1* is directly connected to one of the carrier's routers, *A*, by a leased line. In contrast, *H2* is on a LAN with a router, *F*, owned and operated by the customer. This router also has a leased line to the carrier's equipment. We have shown *F* as being outside the oval because it does not belong to the carrier, but in terms of construction, software, and protocols, it is probably no different from the carrier's routers.

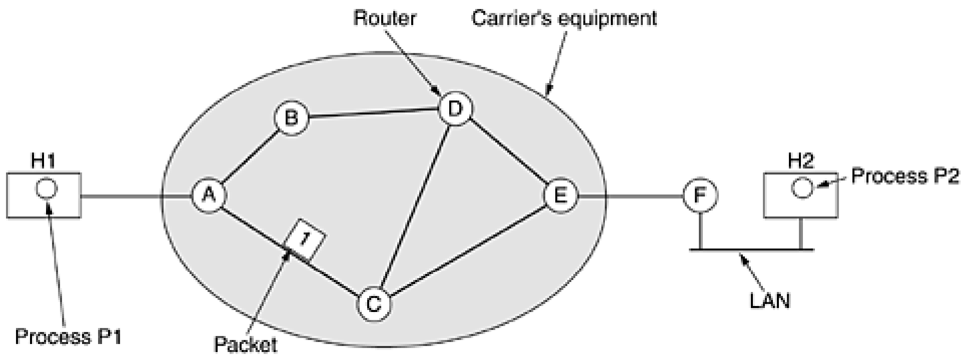


Figure: The environment of the network layer protocols

Services Provided to the Transport Layer :

The network layer provides services to the transport layer at the network layer/transport layer interface.

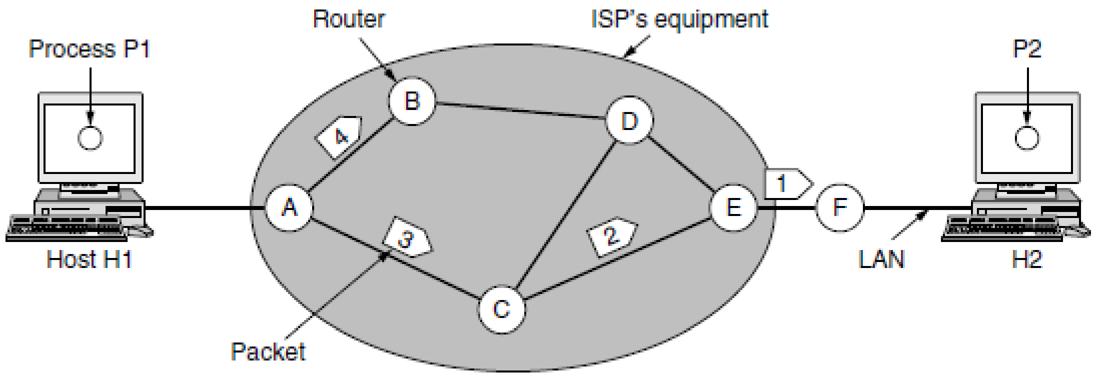
1. The services should be independent of the router technology.
2. The transport layer should be shielded from the number, type, and topology of the routers present.
3. The network addresses made available to the transport layer should use a uniform numbering plan, even across LANs and WANs.

This viewpoint leads to the conclusion that the network service should be connectionless, with primitives SEND PACKET and RECEIVE PACKET and little else. In particular, no packet ordering and flow control should be done, because the hosts are going to do that anyway and there is usually little to be gained by doing it twice.

Implementation of Connection Less Service:

If connectionless service is offered, packets are injected into the network individually and routed independently of each other. No advance setup is needed. In this context, the packets are frequently called **datagrams** (in analogy with telegrams) and the network is called a **datagram network**.

Suppose that the process *P1* in the following figure has a long message for *P2*. It hands the message to the transport layer, with instructions to deliver it to process *P2* on host *H2*. The transport layer code runs on *H1*, typically within the operating system. It prepends a transport header to the front of the message and hands the result to the network layer, probably just another procedure within the operating system.



A's table (initially)

Dest.	Line
A	–
B	B
C	C
D	B
E	C
F	C

A's table (later)

A	–
B	B
C	C
D	B
E	B
F	B

C's table

A	A
B	A
C	–
D	E
E	E
F	E

E's table

A	C
B	D
C	C
D	D
E	–
F	F

Figure. Routing within a datagram network

Let us assume for this example that the message is four times longer than the maximum packet size, so the network layer has to break it into four packets, 1, 2, 3, and 4, and send each of them in turn to router *A* using some point-to-point protocol, for example, PPP.

Implementation of Connection-Oriented Service:

If connection-oriented service is used, a path from the source router all the way to the destination router must be established before any data packets can be sent. This connection is called a **VC** (**virtual circuit**), in analogy with the physical circuits set up by the telephone system, and the network is called a **virtual-circuit network**. The idea behind virtual circuits is to avoid having to choose a new route for every packet sent.

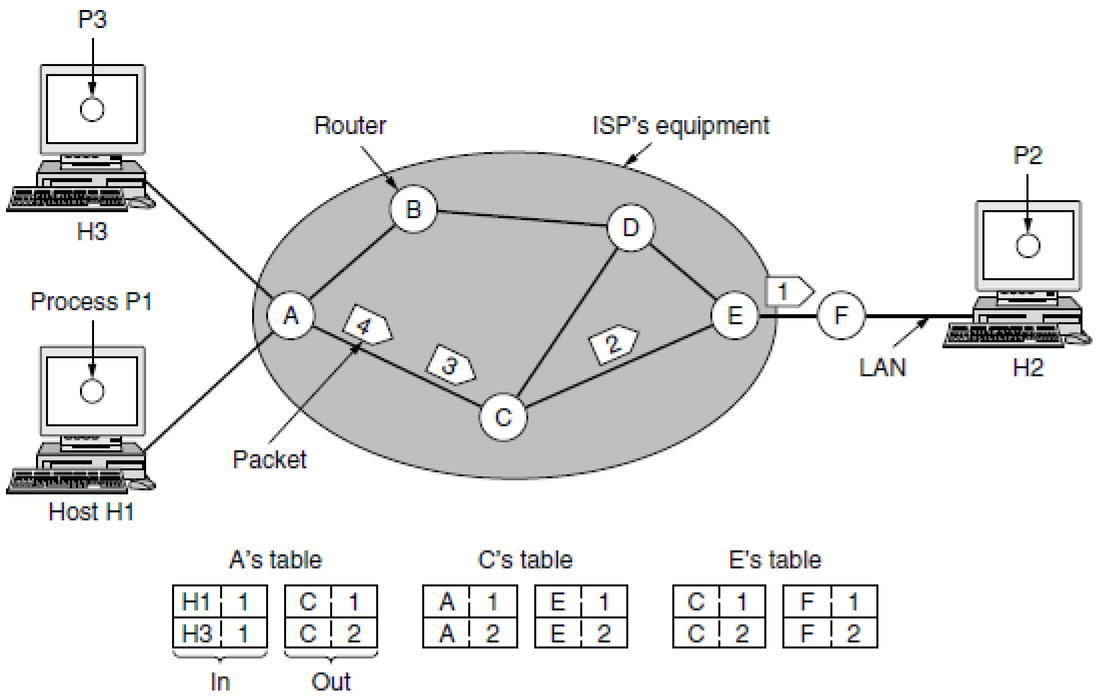


Figure. Routing within a virtual-circuit network.

Here, host *H1* has established connection 1 with host *H2*. This connection is remembered as the first entry in each of the routing tables. The first line of *A*'s table says that if a packet bearing connection identifier 1 comes in from *H1*, it is to be sent to router *C* and given connection identifier 1. Similarly, the first entry at *C* routes the packet to *E*, also with connection identifier 1.

Comparison of Virtual-Circuit and Datagram Networks:

Issue	Datagram network	Virtual-circuit network
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

3.2 ROUTING ALGORITHMS

The main function of the network layer is routing packets from the source machine to the destination machine. In most networks, packets will require multiple hops to make the journey. The algorithms that choose the routes and the data structures that they use are a major area of network layer design. The **routing algorithm** is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on. Routing algorithms can be grouped into two major classes: nonadaptive and adaptive.

Nonadaptive algorithms do not base their routing decisions on any measurements or estimates of the current topology and traffic. **Adaptive algorithms**, in contrast, change their routing decisions to reflect changes in the topology, and sometimes changes in the traffic as well. These **dynamic routing** algorithms differ in where they get their information.

3.2.1 The Optimality Principle

It may be helpful to note that one can make a general statement about optimal routes without regard to network topology or traffic. This statement is known as the **optimality principle**.

It states that if router *J* is on the optimal path from router *I* to router *K*, then the optimal path from *J* to *K* also falls along the same route. As a direct consequence of the optimality principle, we can see that the set of optimal routes from all sources to a given destination form a tree rooted at the destination. Such a tree is called a **sink tree**. Since a sink tree is indeed a tree, it does not contain any loops, so each packet will be delivered within a finite and bounded number of hops. The goal of all routing algorithms is to discover and use the sink trees for all routers.

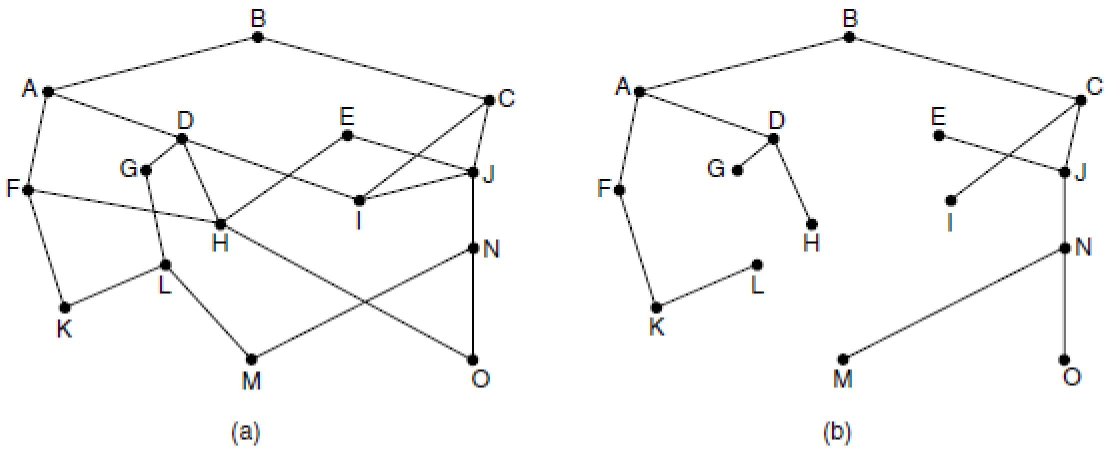


Figure: (a) A network. (b) A sink tree for router *B*.

3.2.2. Shortest Path Algorithm:

The concept of a **shortest path** deserves some explanation. One way of measuring path length is the number of hops. The idea is to build a graph of the network, with each node of the graph representing a router and each edge of the graph representing a communication line, or link. To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.

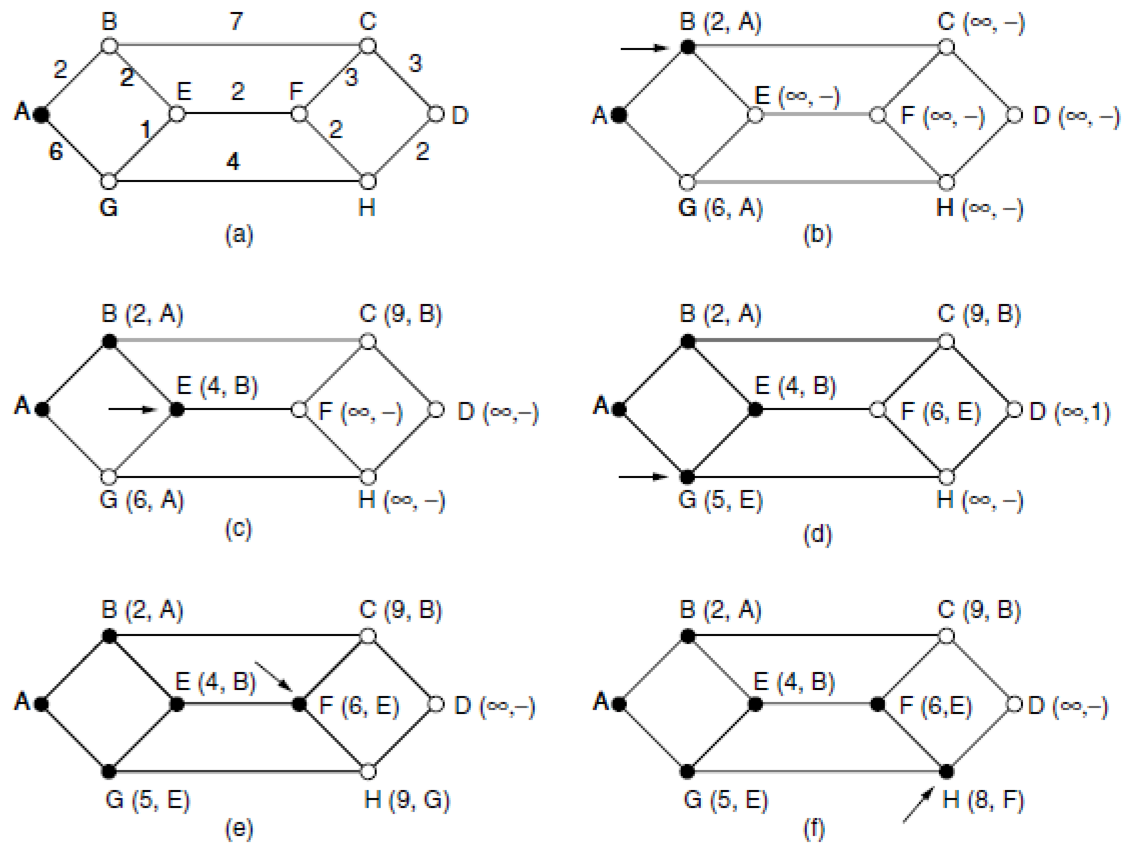
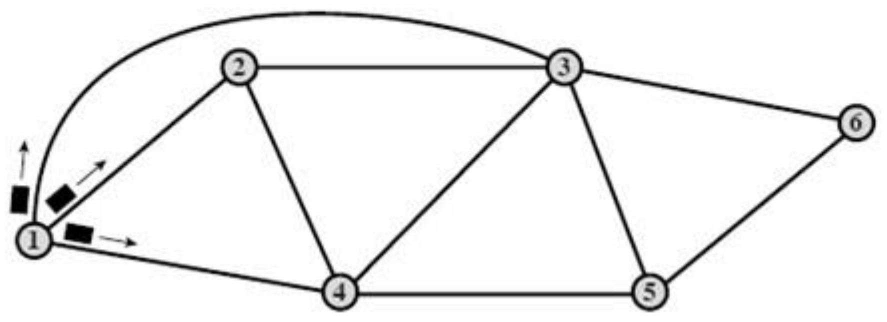


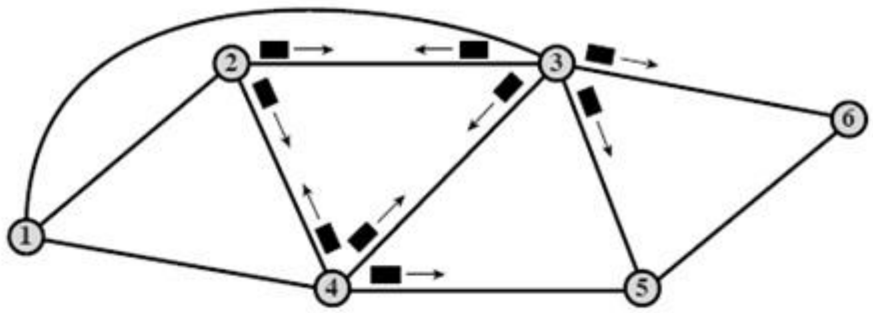
Figure: The first six steps used in computing the shortest path from *A* to *D*.
The arrows indicate the working node.

Several algorithms for computing the shortest path between two nodes of a graph are known. This one is due to **Dijkstra** (1959) and finds the shortest paths between a source and all destinations in the network. Each node is labeled (in parentheses) with its distance from the source node along the best known path.
The distances must be non-negative, as they will be if they are based on real quantities like bandwidth and delay. Initially, no paths are known, so all nodes are labeled with infinity. As the algorithm proceeds and paths are found, the labels may change, reflecting better paths. A label may be either tentative or permanent.
Initially, all labels are tentative. When it is discovered that a label represents the shortest possible path from the source to that node, it is made permanent and never changed thereafter.

3.2.3 Flooding

When a routing algorithm is implemented, each router must make decisions based on local knowledge, not the complete picture of the network. A simple local technique is **flooding**, in which every incoming packet is sent out on every outgoing line except the one it arrived on.





Flooding obviously generates vast numbers of duplicate packets, in fact, an infinite number unless some measures are taken to damp the process. One such measure is to have a hop counter contained in the header of each packet that is decremented at each hop, with the packet being discarded when the counter reaches zero.

Ideally, the hop counter should be initialized to the length of the path from source to destination. If the sender does not know how long the path is, it can initialize the counter to the worst case, namely, the full diameter of the network. Flooding with a hop count can produce an exponential number of duplicate packets as the hop count grows and routers duplicate packets they have seen before. A better technique for damming the flood is to have routers keep track of which packets have been flooded, to avoid sending them out a second time. One way to achieve this goal is to have the source router put a sequence number in each packet it receives from its hosts. Each router then needs a list per source router telling which sequence numbers originating at that source have already been seen. If an incoming packet is on the list, it is not flooded.

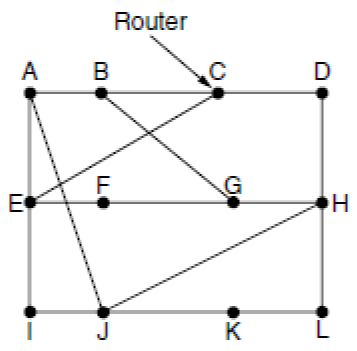
3.2.4 Distance Vector Routing (Adaptive algorithms / Dynamic routing algorithms)

Computer networks generally use dynamic routing algorithms that are more complex than flooding, but more efficient because they find shortest paths for the current topology.

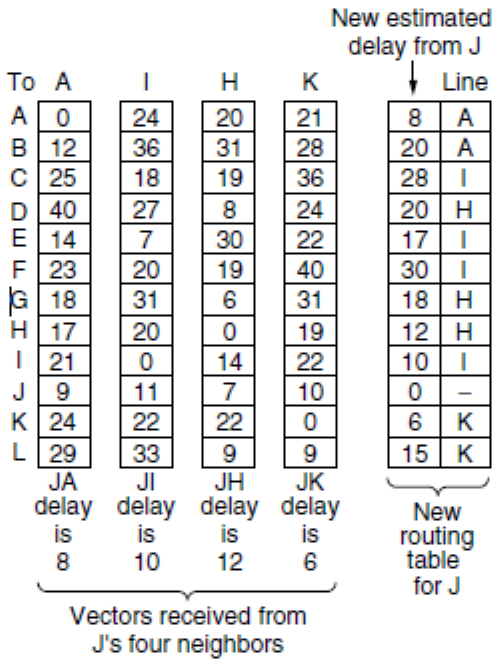
A **distance vector routing** algorithm operates by having each router maintain a table (i.e., a vector) giving the best known distance to each destination and which link to use to get there. These tables are updated by exchanging information with the neighbors. Eventually, every router knows the best link to reach each destination.

The distance vector routing algorithm is sometimes called by other names, most commonly the distributed **Bellman-Ford** routing algorithm, after the researchers who developed it . It was the original ARPANET routing algorithm.In distance vector routing, each router maintains a routing table indexed by, and containing one entry for each router in the network.

This entry has two parts: The preferred outgoing line to use for that destination and an estimate of the distance to that destination. The distance might be measured as the number of hops or using another metric. This updating process is illustrated in Figure Part (a) shows a network. The first four columns of part (b) show the delay vectors received from the neighbors of router *J*. *A* claims to have a 12-msec delay to *B*, a 25-msec delay to *C*, a 40- msec delay to *D*, etc. Suppose that *J* has measured or estimated its delay to its neighbors, *A, I, H*, and *K*, as 8, 10, 12, and 6 msec, respectively.



(a)



To	A	I	H	K	New estimated delay from J	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	–
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

Vectors received from J's four neighbors

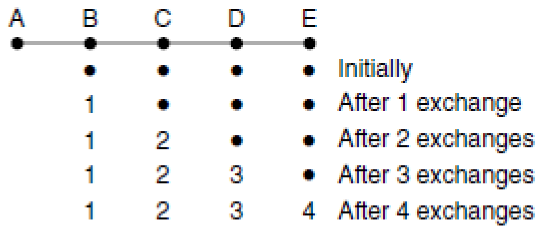
(b)

Figure: (a) A network. (b) Input from *A*, *I*, *H*, *K*, and the new routing table for *J*.

Consider how *J* computes its new route to router *G*. It knows that it can get to *A* in 8 msec, and furthermore *A* claims to be able to get to *G* in 18 msec, so *J* knows it can count on a delay of 26 msec to *G* if it forwards packets bound for *G* to *A*. Similarly, it computes the delay to *G* via *I*, *H*, and *K* as 41 (31 + 10), 18 (6 + 12), and 37 (31 + 6) msec, respectively. The best of these values is 18, so it makes an entry in its routing table that the delay to *G* is 18 msec and that the route to use is via *H*. The same calculation is performed for all the other destinations, with the new routing table shown in the last column of the figure.

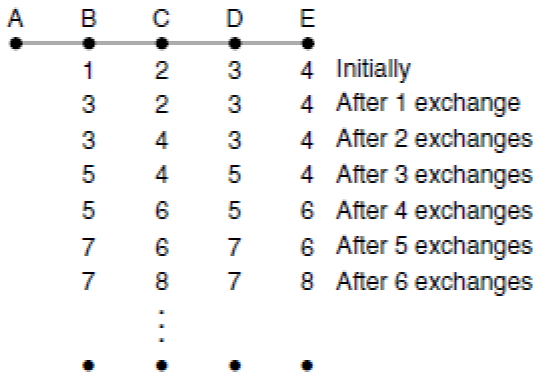
The Count-to-Infinity Problem

The settling of routes to best paths across the network is called **convergence**. Distance vector routing is useful as a simple technique by which routers can collectively compute shortest paths, but it has a serious drawback in practice: although it converges to the correct answer, it may do so slowly. In particular, it reacts rapidly to good news, but leisurely to bad news. Consider a router whose best route to destination *X* is long. If, on the next exchange, neighbor *A* suddenly reports a short delay to *X*, the router just switches over to using the line to *A* to send traffic to *X*. In one vector exchange, the good news is processed.



A	B	C	D	E	
	•	•	•	•	Initially
	1	•	•	•	After 1 exchange
	1	2	•	•	After 2 exchanges
	1	2	3	•	After 3 exchanges
	1	2	3	4	After 4 exchanges

(a)



A	B	C	D	E	
	1	2	3	4	Initially
	3	2	3	4	After 1 exchange
	3	4	3	4	After 2 exchanges
	5	4	5	4	After 3 exchanges
	5	6	5	6	After 4 exchanges
	7	6	7	6	After 5 exchanges
	7	8	7	8	After 6 exchanges
		⋮			
	•	•	•	•	

(b)

When *A* comes up, the other routers learn about it via the vector exchanges. For simplicity, we will assume that there is a gigantic gong somewhere that is struck periodically to initiate a vector exchange at all routers simultaneously. At the time of the first exchange, *B* learns that its left-hand neighbor has zero delay to *A*. *B* now makes an entry in its routing table indicating that *A* is one hop away to the left. All the other routers still think that *A* is down. At this point, the routing table entries for *A* are as shown in the second row of Fig. (a). On the next exchange, *C* learns that *B* has a path of length 1 to *A*, so it updates its routing table to indicate a path of length 2, but *D* and *E* do not hear the good news until later. Clearly, the good news is spreading at the rate of one hop per exchange. In a network whose longest path is of length *N* hops, within *N* exchanges everyone will know about newly revived links and routers.

Consider the Fig.(b), in which all the links and routers are initially up. Routers *B*, *C*, *D*, and *E* have distances to *A* of 1, 2, 3, and 4 hops, respectively. Suddenly, either *A* goes down or the link between *A* and *B* is cut (which is effectively the same thing from *B*'s point of view). At the first packet exchange, *B* does not hear anything from *A*. Fortunately, *C* says ''Do not worry; I have a path to *A* of length 2.'' Little does *B* suspect that *C*'s path runs through *B* itself. For all *B* knows, *C* might have ten links all with separate paths to *A* of length 2. As a result, *B* thinks it can reach *A* via *C*, with a path length of 3. *D* and *E* do not update their entries for *A* on the first exchange.

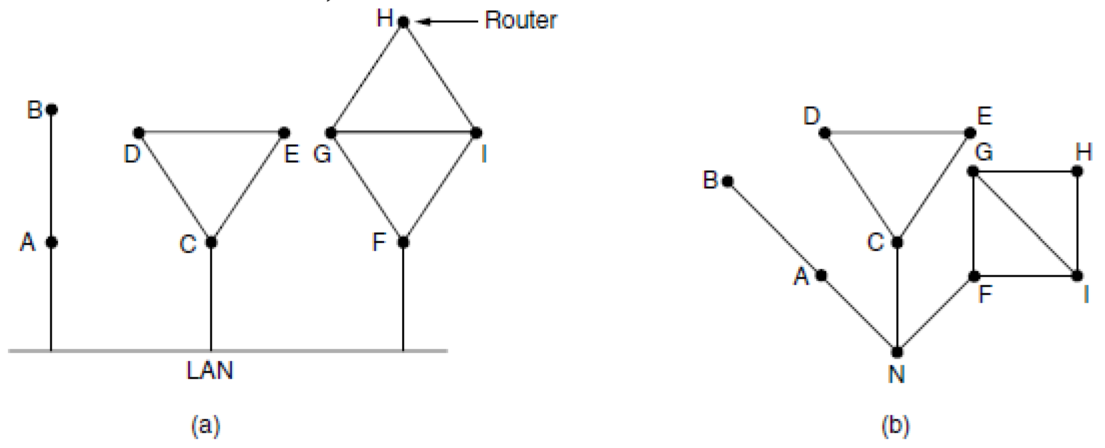
3.2.5 Link State Routing (Adaptive algorithms / Dynamic routing algorithms)

The idea behind link state routing is fairly simple and can be stated as five parts. Each router must do the following things to make it work:

1. Discover its neighbors and learn their network addresses.
2. Set the distance or cost metric to each of its neighbors.
3. Construct a packet telling all it has just learned.
4. Send this packet to and receive packets from all other routers.
5. Compute the shortest path to every other router.

Learning about the Neighbors:

When a router is booted, its first task is to learn who its neighbors are. It accomplishes this goal by sending a special HELLO packet on each point-to-point line. The router on the other end is expected to send back a reply giving its name. These names must be globally unique because when a distant router later hears that three routers are all connected to *F*, it is essential that it can determine whether all three mean the same *F*.

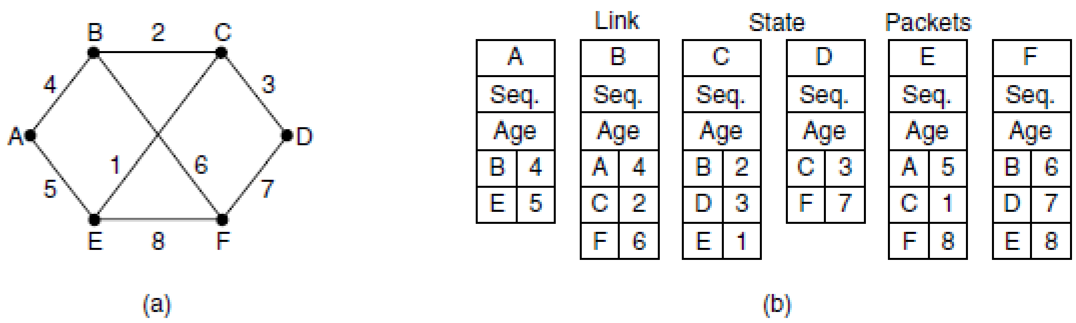


Setting Link Costs

The link state routing algorithm requires each link to have a distance or cost metric for finding shortest paths. The cost to reach neighbors can be set automatically, or configured by the network operator. The most direct way to determine this delay is to send over the line a special ECHO packet that the other side is required to send back immediately. By measuring the round-trip time and dividing it by two, the sending router can get a reasonable estimate of the delay.

Building Link State Packets

Once the information needed for the exchange has been collected, the next step is for each router to build a packet containing all the data. The packet starts with the identity of the sender, followed by a sequence number and age (to be described later) and a list of neighbors. The cost to each neighbor is also given.



A	B	C	D	E	F
Seq.	Seq.	Seq.	Seq.	Seq.	Seq.
Age	Age	Age	Age	Age	Age
B 4	A 4	B 2	C 3	A 5	B 6
E 5	C 2	D 3	F 7	C 1	D 7
	F 6	E 1		F 8	E 8

Figure: (a) A network. (b) The link state packets for this network.

Building the link state packets is easy. The hard part is determining when to build them. One possibility is to build them periodically, that is, at regular intervals.

Distributing the Link State Packets:

The trickiest part of the algorithm is distributing the link state packets. All of the routers must get all of the link state packets quickly and reliably. If different routers are using different versions of the topology, the routes they compute can have inconsistencies such as loops, unreachable machines, and other problems.

First, we will describe the basic distribution algorithm. After that we will give some refinements. The fundamental idea is to use flooding to distribute the link state packets to all routers. To keep the flood in check, each packet contains a sequence number that is incremented for each new packet sent. Routers keep track of all the (source router, sequence) pairs they see. When a new link state packet comes in, it is checked against the list of packets already seen.

Computing the New Routes:

Once a router has accumulated a full set of link state packets, it can construct the entire network graph because every link is represented. Every link is, in fact, represented twice, once for each direction. The different directions may even have different costs. The shortest-path computations may then find different paths from router *A* to *B* than from router *B* to *A*.

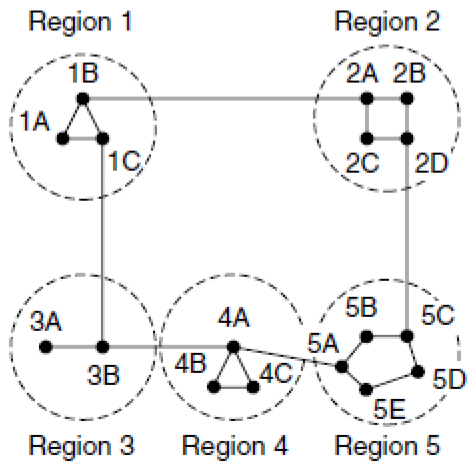
Now Dijkstra’s algorithm can be run locally to construct the shortest paths to all possible destinations. The results of this algorithm tell the router which link to

3.2.6 Hierarchical Routing

As networks grow in size, the router routing tables grow proportionally. Not only is router memory consumed by ever-increasing tables, but more CPU time is needed to scan them and more bandwidth is needed to send status reports about them. At a certain point, the network may grow to the point where it is no longerfeasible for every router to have an entry for every other router, so the routing will have to be done hierarchically, as it is in the telephone network.

When hierarchical routing is used, the routers are divided into what we will call **regions**. Each router knows all the details about how to route packets to destinations within its own region but knows nothing about the internal structure of other regions. When different networks are interconnected, it is natural to regard each one as a separate region to free the routers in one network from having to know the topological structure of the other ones.

Figure gives a quantitative example of routing in a two-level hierarchy with five regions. The full routing table for router *1A* has 17 entries, as shown in Fig. (b). When routing is done hierarchically, as in Fig.(c), there are entries for all the local routers, as before, but all other regions are condensed into a single router, so all traffic for region 2 goes via the *1B-2A* line, but the rest of the remote traffic goes via the *1C-3B* line. Hierarchical routing has reduced the table from 17 to 7 entries. As the ratio of the number of regions to the number of routers per region grows, the savings in table space increase.



(a)

Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

(b)

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

Figure. Hierarchical routing.

3.2.7 Broadcast Routing:

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

Various methods have been proposed for doing it.

One broadcasting method that requires no special features from the network is for the source to simply send a distinct packet to each destination. Not only is the method wasteful of bandwidth and slow, but it also requires the source to have a complete list of all destinations. This method is not desirable in practice, even though it is widely applicable.

An improvement is **multidestination routing**, in which each packet contains either a list of destinations or a bit map indicating the desired destinations. When a packet arrives at a router, the router checks all the destinations to determine the set of output lines that will be needed. The router generates a new copy of the packet for each output line to be used and includes in each packet only those destinations
that are to use the line. In effect, the destination set is partitioned among the output lines. After a sufficient number of hops, each packet will carry only one destination like a normal packet.
The idea for **reverse path forwarding** is elegant and remarkably simple.
When a broadcast packet arrives at a router, the router checks to see if the packet arrived on the link that is normally used for sending packets *toward* the source of the broadcast. If so, there is an excellent chance that the broadcast packet itself followed the best route from the router and is therefore the first copy to arrive at the router. This being the case, the router forwards copies of it onto all links except the one it arrived on. If, however, the broadcast packet arrived on a link other than the preferred one for
reaching the source, the packet is discarded as a likely duplicate.

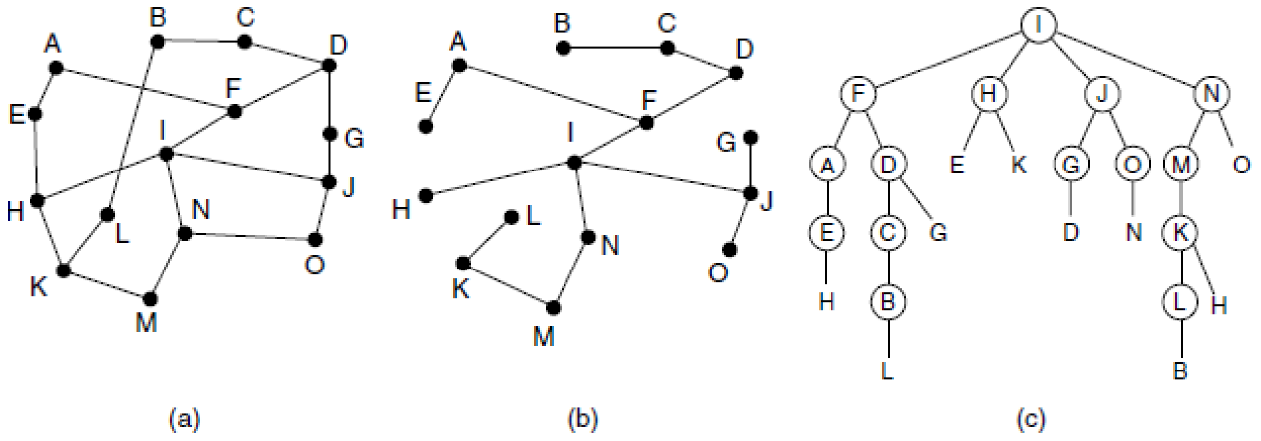


Figure . Reverse path forwarding. (a) A network. (b) A sink tree. (c) The tree built by reverse path forwarding.

An example of reverse path forwarding is shown in Fig.Part (a) shows a network, part (b) shows a sink tree for router *I* of that network, and part (c) shows how the reverse path algorithm works. On the first hop, *I* sends packets to *F*, *H*, *J*, and *N*, as indicated by the second row of the tree. Each of these packets
arrives on the preferred path to *I* (assuming that the preferred path falls along the sink tree) and is so indicated by a circle around the letter. On the second hop, eight packets are generated, two by each of the routers that received a packet on the first hop.

A **spanning tree** is a subset of the network that includes all the routers but contains no loops. Sink trees are spanning trees. If each router knows which of its lines belong to the spanning tree, it can copy an incoming broadcast packet onto all the spanning tree lines except the one it arrived on. This method makes excellent use of bandwidth, generating the absolute minimum number of packets necessary to do the job.

3.2.8 Multicast Routing

Some applications, such as a multiplayer game or live video of a sports event streamed to many viewing locations, send packets to multiple receivers. Unless the group is very small, sending a distinct packet to each receiver is expensive.

On the other hand, broadcasting a packet is wasteful if the group consists of, say, 1000 machines on a million-node network, so that most receivers are not interested in the message (or worse yet, they are definitely interested but are not supposed to see it). Thus, we need a way to send messages to well-defined groups that are numerically large in size but small compared to the network as a whole.

Sending a message to such a group is called **multicasting**, and the routing algorithm used is called **multicast routing**. All multicasting schemes require some way to create and destroy groups and to identify which routers are members of a group. How these tasks are accomplished is not of concern to the routing algorithm. For now, we will assume that each group is identified by a multicast address and that routers know the groups to which they belong.

Multicast routing schemes build on the broadcast routing schemes we have already studied, sending packets along spanning trees to deliver the packets to the members of the group while making efficient use of bandwidth. However, the best spanning tree to use depends on whether the group is dense, with receivers scattered over most of the network, or sparse, with much of the network not belonging to the group.

As an example, consider the two groups, 1 and 2, in the network shown in Fig.(a). Some routers are attached to hosts that belong to one or both of these groups, as indicated in the figure. A spanning tree for the leftmost router is shown in Fig.(b). This tree can be used for broadcast but is overkill for multicast, as can be seen from the two pruned versions that are shown next. In Fig. (c), all the links that do not lead to hosts that are members of group 1 have been removed. The result is the multicast spanning tree for the leftmost router to send to group 1. Packets are forwarded only along this spanning tree, which is more efficient than the broadcast tree because there are 7 links instead of 10. Fig.(d) shows the multicast spanning tree after pruning for group 2. It is efficient too, with only five links this time. It also shows that different multicast groups have different spanning trees.

Various ways of pruning the spanning tree are possible. The simplest one can be used if link state routing is used and each router is aware of the complete topology, including which hosts belong to which groups. Each router can then construct its own pruned spanning tree for each sender to the group in question by constructing a sink tree for the sender as usual and then removing all links that do not connect group members to the sink node. **MOSPF** (**Multicast OSPF**) is an example of a link state protocol that works in this way.

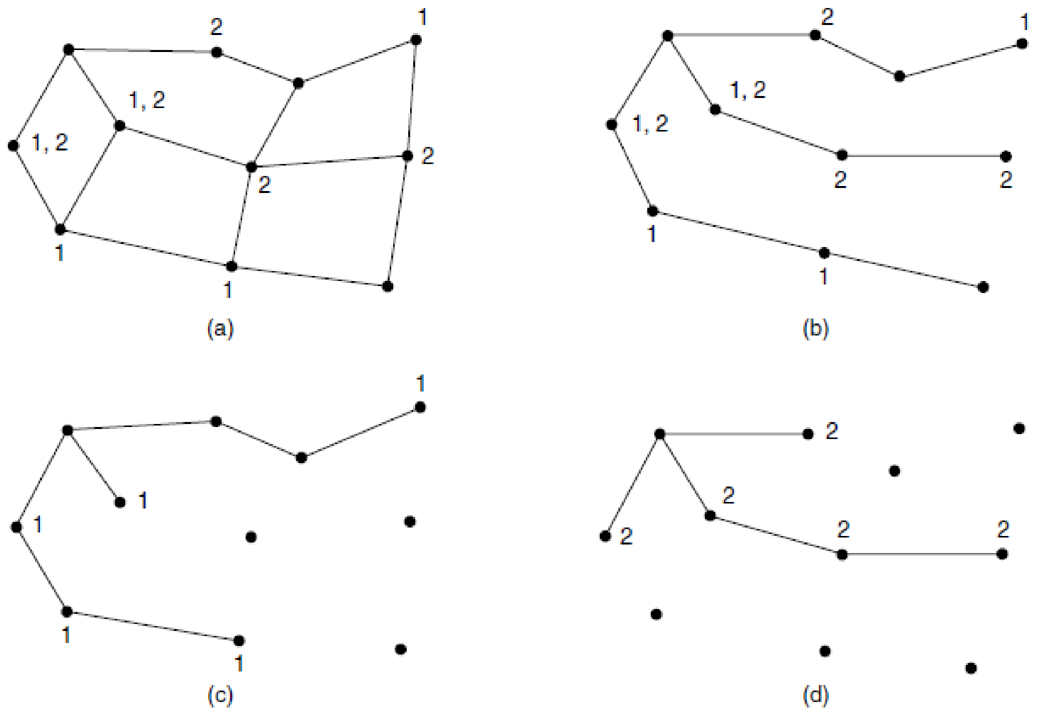


Figure. (a) A network. (b) A spanning tree for the leftmost router. (c) A multicast tree for group 1. (d) A multicast tree for group 2.

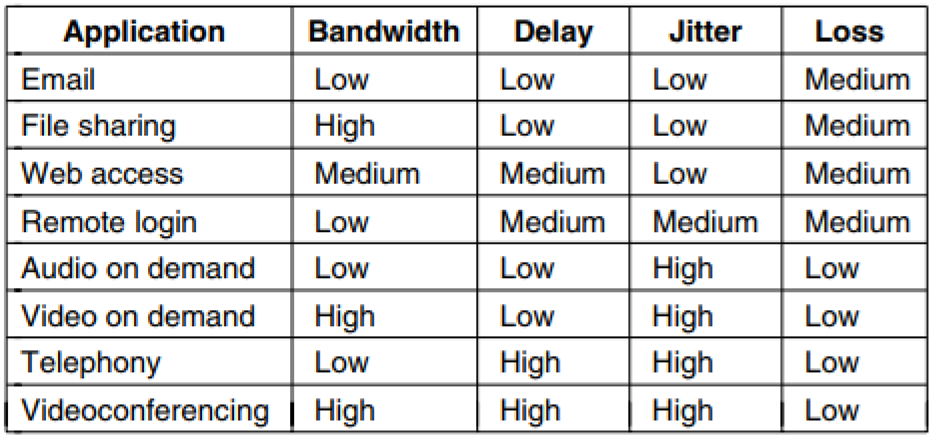
3.4 QUALITY OF SERVICE

Quality of service (**QoS**) refers to any technology that manages data traffic to reduce packet loss, latency and jitter on the **network**. **QoS** controls and manages **network** resources by setting priorities for specific types of data on the **network**. To quantitatively measure quality of service, several related aspects of the network service are often considered, such as packet loss, bit rate, throughput, transmission delay, availability, jitter, etc.
In the field of computer networking and other packet-switched telecommunication networks, quality of service refers to traffic prioritization and resource reservation control mechanisms rather than the achieved service quality. Quality of service is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow.

Four issues must be addressed to ensure quality of service:

1. What applications need from the network.
2. How to regulate the traffic that enters the network.
3. How to reserve resources at routers to guarantee performance.
4. Whether the network can safely accept more traffic.

Application Requirements
A stream of packets from a source to a destination is called a flow (Clark,
1988). A flow might be all the packets of a connection in a connection-oriented network, or all the packets sent from one process to another process in a connectionless network.



Application	Bandwidth	Delay	Jitter	Loss
Email	Low	Low	Low	Medium
File sharing	High	Low	Low	Medium
Web access	Medium	Medium	Low	Medium
Remote login	Low	Medium	Medium	Medium
Audio on demand	Low	Low	High	Low
Video on demand	High	Low	High	Low
Telephony	Low	High	High	Low
Videoconferencing	High	High	High	Low

5.4.2 Traffic Shaping

Traffic shaping is a bandwidth management technique used on computer networks which delays some or all datagrams to bring them into compliance with a desired *traffic profile*.

Traffic shaping is a technique for regulating the average rate and burstiness of a flow of data that enters the network. The goal is to allow applications to transmit a wide variety of traffic that suits their needs, including some bursts, yet have a simple and useful way to describe the possible traffic patterns to the network. When a flow is set up, the user and the network (i.e., the customer and the provider) agree on a certain traffic pattern (i.e., shape) for that flow.

Traffic shaping reduces congestion and thus helps the network live up to its promise. However, to make it work, there is also the issue of how the provider can tell if the customer is following the agreement and what to do if the customer is not. Packets in excess of the agreed pattern might be dropped by the network, or they might be marked as having lower priority. Monitoring a traffic flow is called traffic policing.

• Traffic shaping is a technique for regulating the average rate and burstiness of a flow of data that enters the network.

Two types of traffic shaping techniques.

– **Leaky Bucket algorithm**

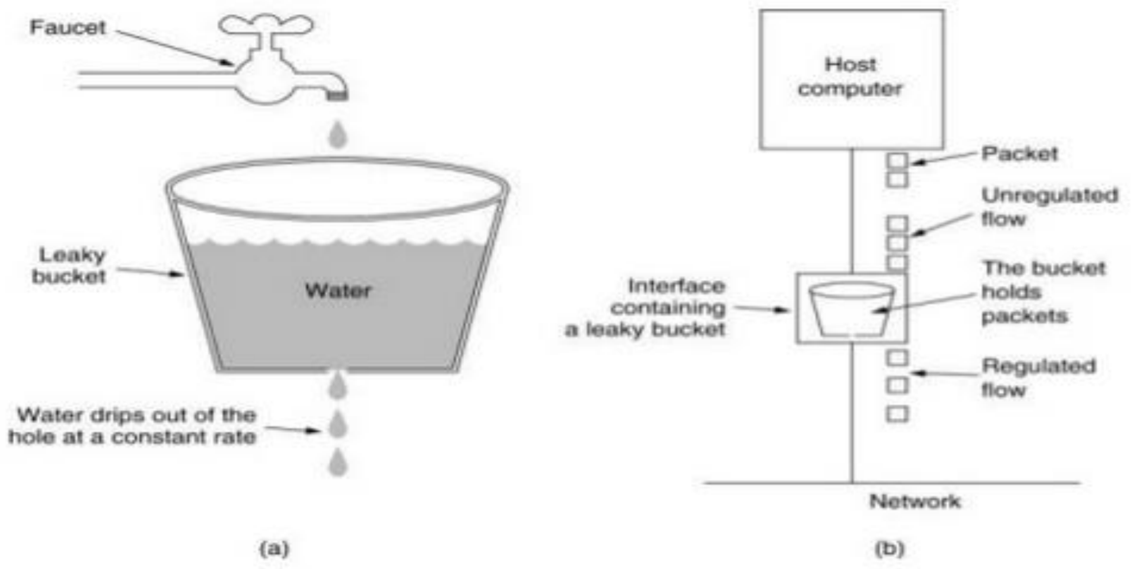
– **Token Bucket algorithm**

Leaky Bucket :

Suppose we have a bucket in which we are pouring water in a random order but we have to get water in a fixed rate, for this we will make a hole at the bottom of the bucket. It will ensure that water coming out is in a some fixed rate, and also if bucket will full we will stop pouring in it.

The input rate can vary, but the output rate remains constant. Similarly, in networking, a technique called leaky bucket can smooth out bursty traffic. Bursty chunks are stored in the bucket and sent out at an average rate.

The Leaky Bucket Algorithm



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

Leaky Bucket Algorithm

Algorithm

Step - 1 : Initialize the counter to **'n'** at every tick of clock.

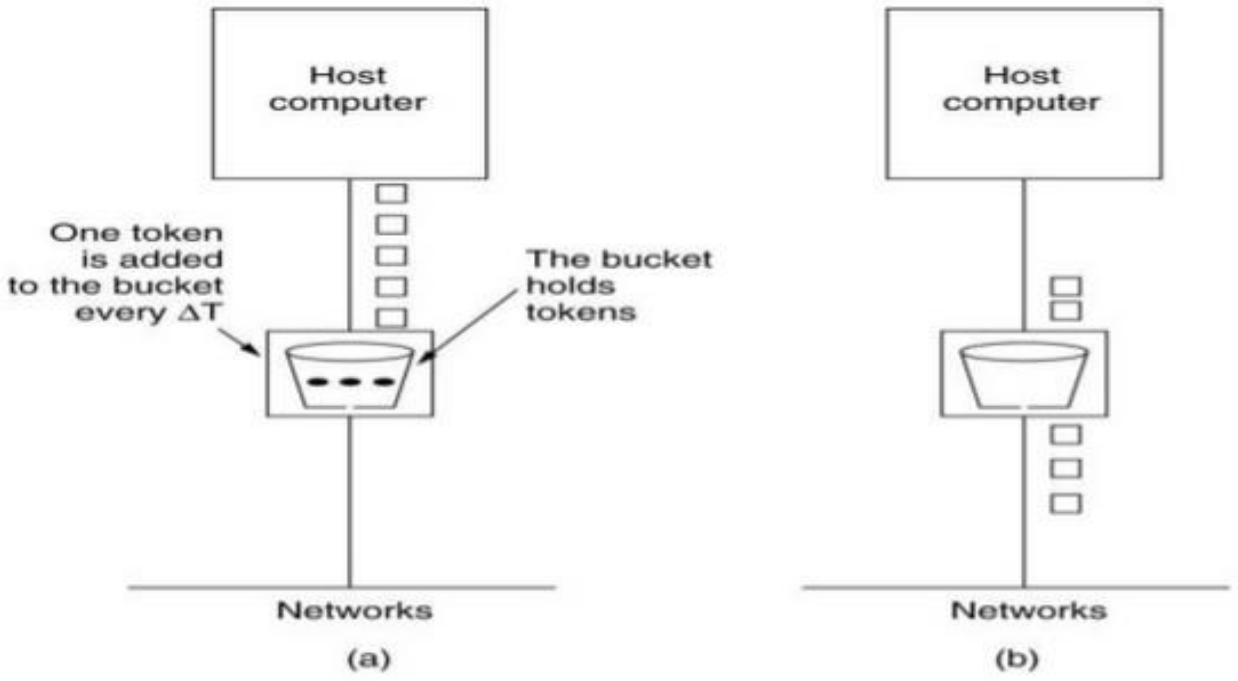
Step - 2 : If **n** is greater than the size of packet in the front of queue send the packet into the network and decrement the counter by size of packet. Repeat the step until **n** is less than the size of packet.

Step - 3 : Reset the counter and go to Step - 1.

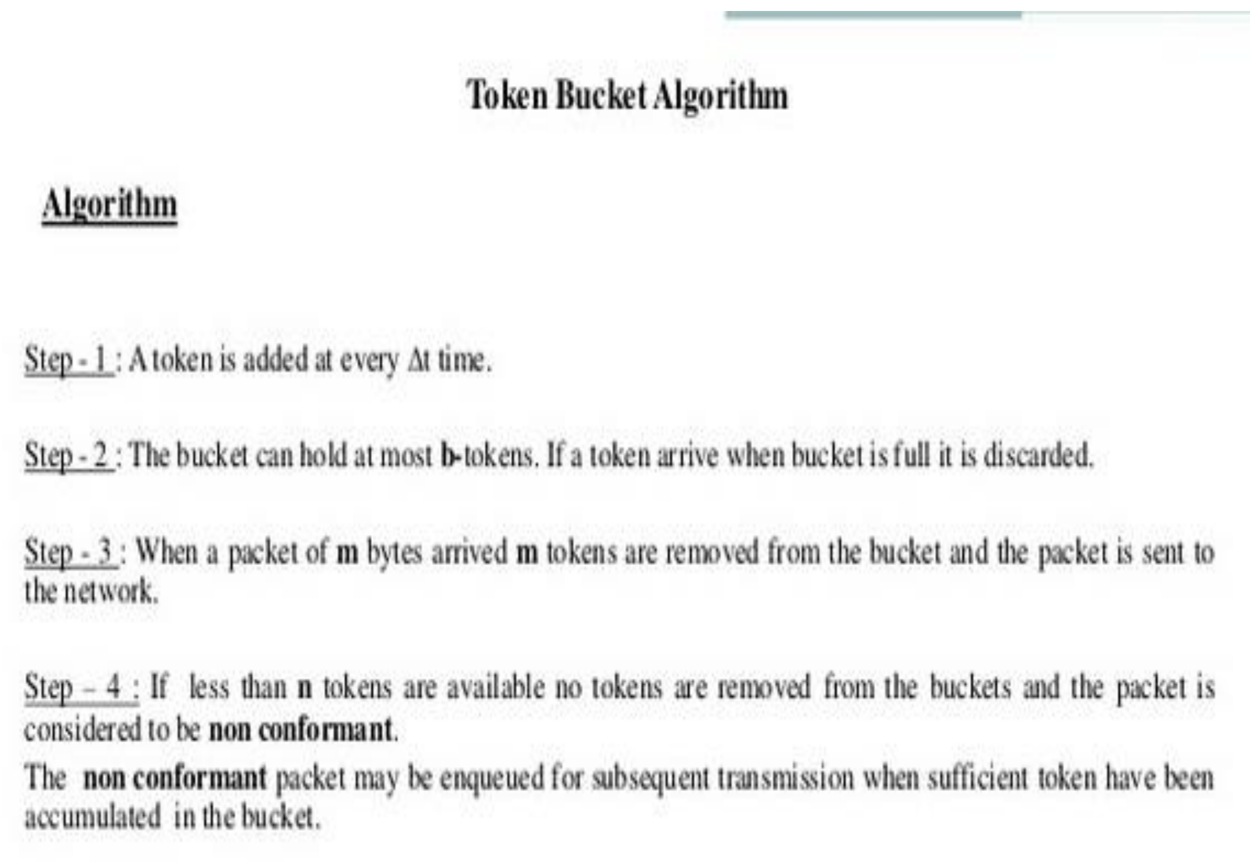
Token Bucket Algorithm

- The **Token Bucket Algorithm** compare to Leaky Bucket Algorithm allow the output rate vary depending on the size of burst.
- In this algorithm the buckets holds token to transmit a packet, the host must capture and destroy one token.
- Tokens are generated by a clock at the rate of one token every Δt sec.
- Idle hosts can capture and save up tokens (up to the max. size of the bucket) in order to send larger bursts later.

Token Bucket Algorithm



(a) Before (b) After



Token Bucket Algorithm

Algorithm

Step - 1 : A token is added at every Δt time.

Step - 2 : The bucket can hold at most **b**-tokens. If a token arrive when bucket is full it is discarded.

Step - 3 : When a packet of **m** bytes arrived **m** tokens are removed from the bucket and the packet is sent to the network.

Step – 4 : If less than **n** tokens are available no tokens are removed from the buckets and the packet is considered to be **non conformant**.

The **non conformant** packet may be enqueued for subsequent transmission when sufficient token have been accumulated in the bucket.

Difference between Leaky and Token buckets –

Leaky Bucket	Token Bucket
When the host has to send a packet , packet is thrown in bucket.	In this leaky bucket holds tokens generated at regular intervals of time.
Bucket leaks at constant rate	Bucket has maximum capacity.
Bursty traffic is converted into uniform traffic by leaky bucket.	If there is a ready packet , a token is removed from Bucket and packet is send.
In practice bucket is a finite queue outputs at finite rate	If there is a no token in bucket, packet can not be send.

Some advantage of token Bucket over leaky bucket –

- If bucket is full in token Bucket , token are discard not packets. While in leaky bucket, packets are discarded.
- Token Bucket can send Large bursts can faster rate while leaky bucket always sends packets at constant rate.

5.4.3 Packet Scheduling

Algorithms that allocate router resources among the packets of a flow and between competing flows are called packet scheduling algorithms. Three different
kinds of resources can potentially be reserved for different flows:
1. Bandwidth.
2. Buffer space.
3. CPU cycles.

The first one, bandwidth, is the most obvious. If a flow requires 1 Mbps and the outgoing line has a capacity of 2 Mbps, trying to direct three flows through that line is not going to work. Thus, reserving bandwidth means not oversubscribing any output line.

A second resource that is often in short supply is buffer space. When a packet arrives, it is buffered inside the router until it can be transmitted on the chosen outgoing line. The purpose of the buffer is to absorb small bursts of traffic as the flows contend with each other. If no buffer is available, the packet has to be discarded since there is no place to put it. For good quality of service, some buffers might be reserved for a specific flow so that flow does not have to compete for buffers with other flows.

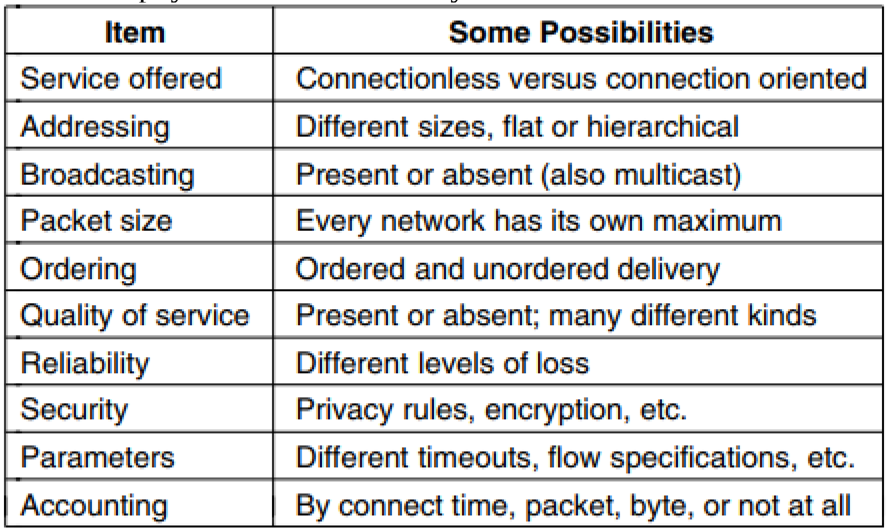
Finally, CPU cycles may also be a scarce resource. It takes router CPU time to process a packet, so a router can process only a certain number of packets per second. While modern routers are able to process most packets quickly, some kinds of packets require greater CPU processing.

5.5 INTERNETWORKING

When two or more networks are connected to form an internetwork, or more simply an internet.

5.5.1 How Networks Differ

Networks can differ in many ways. Some of the differences, such as different modulation techniques or frame formats, are internal to the physical and data link layers.



Item	Some Possibilities
Service offered	Connectionless versus connection oriented
Addressing	Different sizes, flat or hierarchical
Broadcasting	Present or absent (also multicast)
Packet size	Every network has its own maximum
Ordering	Ordered and unordered delivery
Quality of service	Present or absent; many different kinds
Reliability	Different levels of loss
Security	Privacy rules, encryption, etc.
Parameters	Different timeouts, flow specifications, etc.
Accounting	By connect time, packet, byte, or not at all

Figure: Some of the many ways networks can differ.

5.5.3 Tunneling

In computer networks, a **tunneling protocol** is a communications protocol that allows for the movement of data from one network to another. It involves allowing private network communications to be sent across a public network, such as the Internet, through a process called encapsulation. A tunneling protocol may, for example, allow a foreign protocol to run over a network that does not support that particular protocol, such as running IPv6 over IPv4. Another important use is to provide services that are impractical or unsafe to be offered using only the underlying network services, such as providing a corporate network address to a remote user whose physical network address is not part of the corporate network.

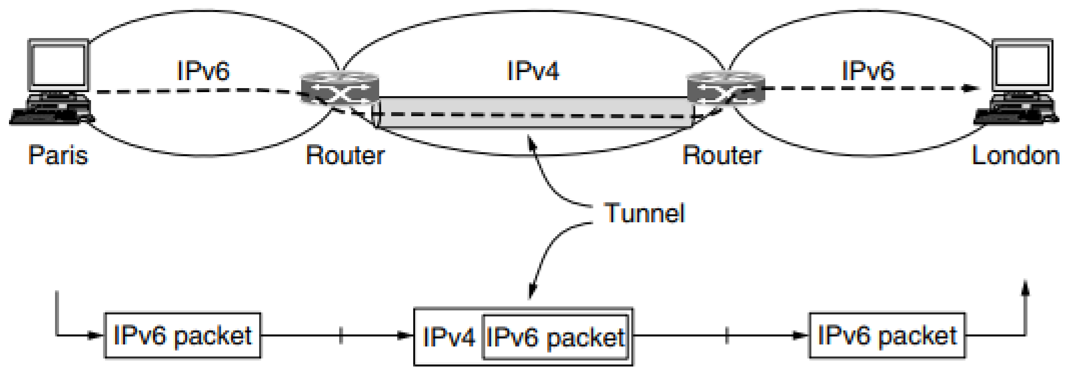


Figure: Tunneling a packet from Paris to London.

As an example, think of an international bank with an IPv6 network in Paris, an IPv6 network in London and connectivity between the offices via the IPv4 Internet. This situation is shown in Fig.The solution to this problem is a technique called tunneling. To send an IP packet to a host in the London office, a host in the Paris office constructs the packet containing an IPv6 address in London, and sends it to the multiprotocol router that connects the Paris IPv6 network to the IPv4 Internet. When this router gets the IPv6 packet, it encapsulates the packet with an IPv4 header addressed to the IPv4 side of the multiprotocol router that connects to the London IPv6 network. That is, the router puts a (IPv6) packet inside a (IPv4) packet. When this wrapped packet arrives, the London router removes the original IPv6 packet and sends it onward to the destination host.

Tunneling is widely used to connect isolated hosts and networks using other networks.

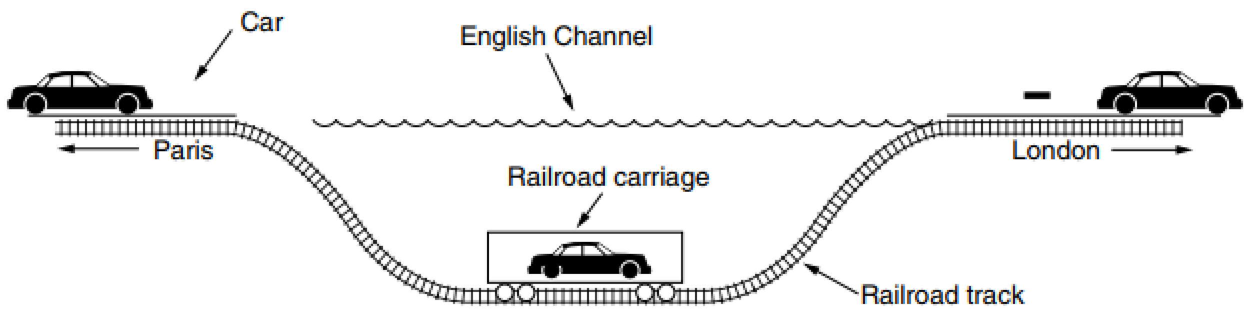


Figure: Tunneling a car from France to England.

5.5.5 Packet Fragmentation

Each network or link imposes some maximum size on its packets. These limits have various causes, among them
1. Hardware (e.g., the size of an Ethernet frame).
2. Operating system (e.g., all buffers are 512 bytes).
3. Protocols (e.g., the number of bits in the packet length field).
4. Compliance with some (inter)national standard.
5. Desire to reduce error-induced retransmissions to some level.
6. Desire to prevent one packet from occupying the channel too long.
The result of all these factors is that the network designers are not free to choose any old maximum packet size they wish.

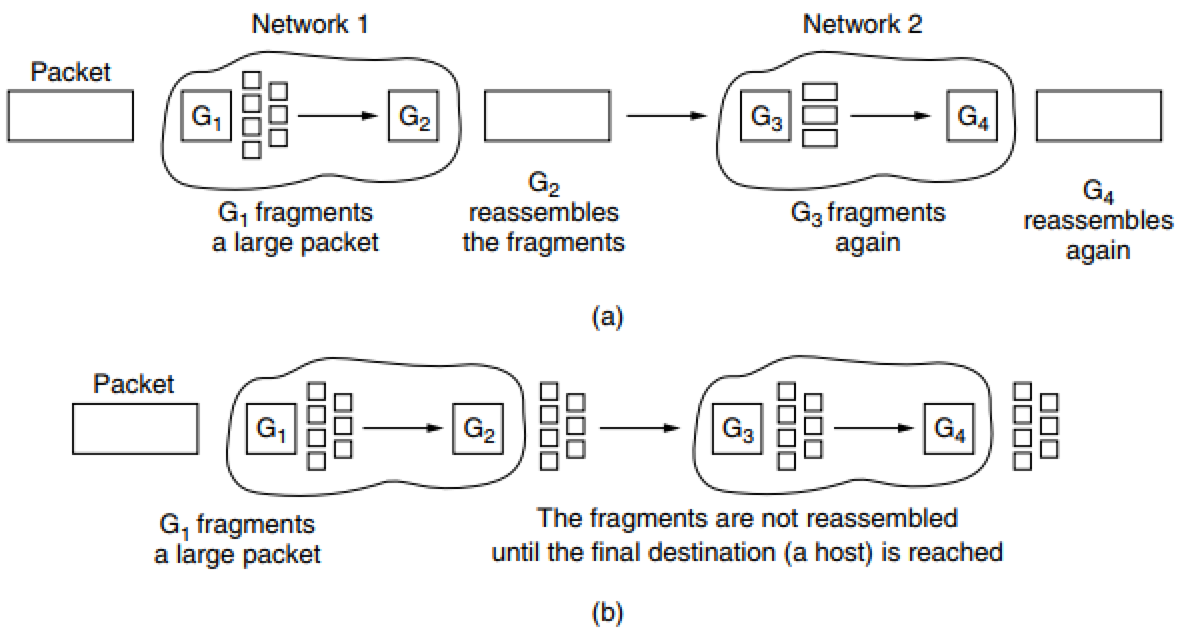

Network 1
Network 2
Packet
G1
G2
G3
G4
G1 fragments a large packet
G2 reassembles the fragments
G3 fragments again
G4 reassembles again
(a)
Packet
G1 fragments a large packet
The fragments are not reassembled until the final destination (a host) is reached
(b)

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

With *transparent fragmentation* end hosts (sender and receiver) are unaware that fragmentation has taken place. A gateway fragments a packet, and the next-hop gateway on the same network reassembles the fragments back into the original packet.

The first strategy is to make fragmentation caused by a ‘‘smallpacket’’ network transparent to any subsequent networks through which the packet must pass on its way to the ultimate destination. This option is shown in Fig.

In this approach, when an oversized packet arrives at G1, the router breaks it up into fragments. Each fragment is addressed to the same exit router, G2, where the pieces are recombined. In this way, passage through the small-packet network is made transparent. Subsequent networks are not even aware that fragmentation has occurred.
Transparent fragmentation is straightforward but has some problems. For one thing, the exit router must know when it has received all the pieces, so either a count field or an ‘‘end of packet’’ bit must be provided.

The other fragmentation strategy is to refrain from recombining fragments at any intermediate routers. Once a packet has been fragmented, each fragment is treated as though it were an original packet. The routers pass the fragments, as shown in Fig.(b), and reassembly is performed only at the destination host.

THE NETWORK LAYER IN THE INTERNET:

1. Make sure it works. Do not finalize the design or standard until multiple prototypes have successfully communicated with each other.
2. Keep it simple. When in doubt, use the simplest solution.
3. Make clear choices. If there are several ways of doing the same thing, choose one.
4. Exploit modularity. This principle leads directly to the idea of having protocol stacks, each of whose layers is independent of all the other ones.
5. Expect heterogeneity. Different types of hardware, transmission facilities, and applications will occur on any large network.
6. Avoid static options and parameters. If parameters are unavoidable (e.g., maximum packet size), it is best to have the sender and receiver negotiate a value rather than defining fixed choices.
7. Look for a good design; it need not be perfect. Often, the designers have a good design but it cannot handle some weird special case.
8. Be strict when sending and tolerant when receiving. In other words, send only packets that rigorously

comply with the standards, but expect incoming packets that may not be fully conformant and try to deal with them.

9. Think about scalability. If the system is to handle millions of hosts and billions of users effectively, no centralized databases of any kind are tolerable and load must be spread as evenly as possible over the available resources.
10. Consider performance and cost. If a network has poor performance or outrageous costs, nobody will use it.

5.6.1 The IP Version 4 Protocol

An IPv4 datagram consists of a header part and a body or payload part. The header has a 20-byte fixed part and a variable-length optional part. The header format is shown in Fig. The bits are transmitted from left to right and top to bottom, with the high-order bit of the Version field going first.

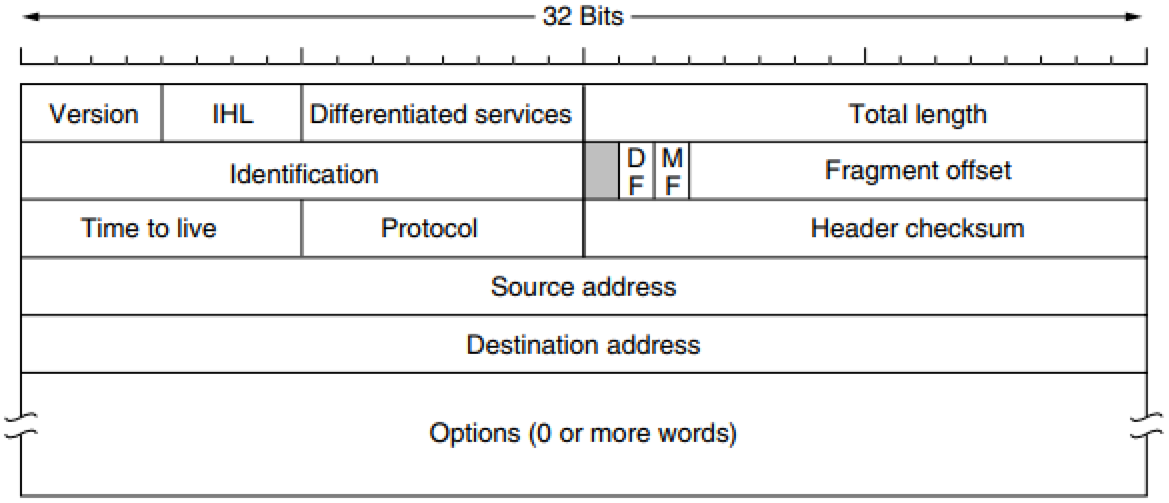


Figure: The IPv4 (Internet Protocol) header.

The Version field keeps track of which version of the protocol the datagram belongs to. Version 4 dominates the Internet today. Since the header length is not constant, a field in the header, IHL, is provided to tell how long the header is, in 32-bit words. The minimum value is 5, which applies when no options are present. The maximum value of this 4-bit field is 15, The Differentiated services field is one of the few fields that has changed its meaning (slightly) over the years. Originally, it was called the Type of service field. It was and still is intended to distinguish between different classes of service. Various combinations of reliability and speed are possible.

The Total length includes everything in the datagram—both header and data. The maximum length is 65,535 bytes. The Identification field is needed to allow the destination host to determine which packet a newly arrived fragment belongs to. All the fragments of a packet contain the same Identification value.
Next comes an unused bit, Then come two 1-bit fields related to fragmentation. DF stands for Don't Fragment. It is an order to the routers not to fragment the packet. MF stands for More Fragments. All fragments except the last one have this bit set. It is needed to know when all fragments of a datagram have arrived.
The Fragment offset tells where in the current packet this fragment belongs. All fragments except the last one in a datagram must be a multiple of 8 bytes. The TtL (Time to live) field is a counter used to limit packet lifetimes. It was originally supposed to count time in seconds, allowing a maximum lifetime of 255 sec. It must be decremented on each hop and is supposed to be decremented multiple times when a packet is queued for a long time in a router.
The Protocol field tells it which transport process to give the packet to. TCP is one possibility, but so are UDP and some others. The numbering of protocols is global across the entire Internet.
Since the header carries vital information such as addresses, it rates its own checksum for protection, the Header checksum. The algorithm is to add up all the 16-bit halfwords of the header as they arrive, using one's complement

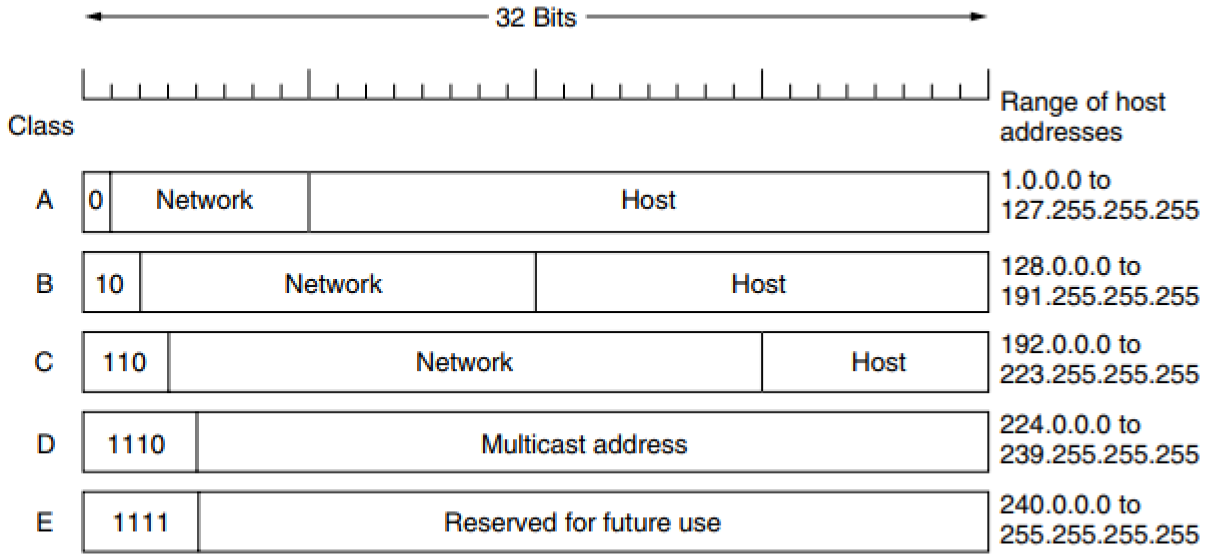
arithmetic, and then take the one's complement of the result. For purposes of this algorithm, the Header checksum is assumed to be zero upon arrival.
The Source address and Destination address indicate the IP address of the source and destination network interfaces. The Options field was designed to provide an escape to allow subsequent versions of the protocol to include information not present in the original design**.**

Option	Description
Security	Specifies how secret the datagram is
Strict source routing	Gives the complete path to be followed
Loose source routing	Gives a list of routers not to be missed
Record route	Makes each router append its IP address
Timestamp	Makes each router append its address and timestamp

Figure: Some of the IP options.

5.6.2 IP Addresses

A defining feature of IPv4 is its 32-bit addresses. Every host and router on the Internet has an IP address that can be used in the Source address and Destination address fields of IP packets. It is important to note that an IP address does not actually refer to a host. It really refers to a network interface, so if a host is on two networks, it must have two IP addresses. IP addresses are written in dotted decimal notation. In this format, each of the 4 bytes is written in decimal, from 0 to 255.


32 Bits
Class
Range of host addresses
A
0
Network
Host
1.0.0.0 to 127.255.255.255
B
10
Network
Host
128.0.0.0 to 191.255.255.255
C
110
Network
Host
192.0.0.0 to 223.255.255.255
D
1110
Multicast address
224.0.0.0 to 239.255.255.255
E
1111
Reserved for future use
240.0.0.0 to 255.255.255.255

The class A, B, and C formats allow for up to 128 networks with 16 million hosts each, 16,384 networks with up to 65,536 hosts each, and 2 million networks (e.g., LANs) with up to 256 hosts each (although a few of these are special). Also supported is multicast (the class D format), in which a datagram is directed to multiple hosts. Addresses beginning with 1111 are reserved for use in the future.

5.6.3 IP Version 6:

IP has been in heavy use for decades. It has worked extremely well, as demonstrated by the exponential growth of the Internet. IPv6 (IP version 6) is a replacement design that does just that. It uses 128-bit addresses; a shortage of these addresses is not likely any time in the foreseeable future. However, IPv6 has proved very difficult to deploy. It is a different network layer protocol that does not really interwork with IPv4, despite many similarities.

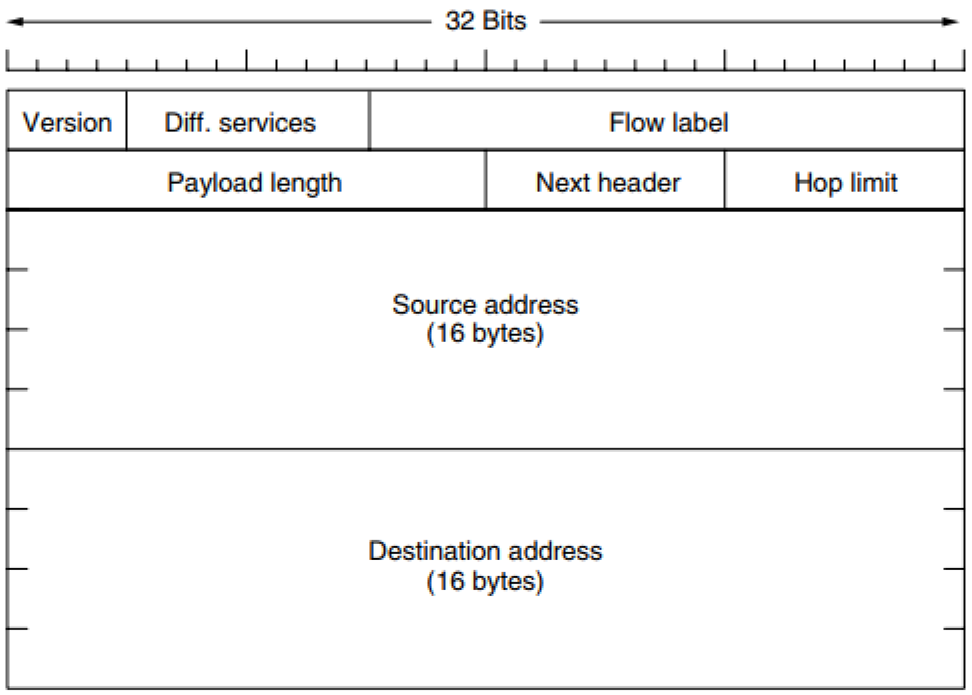


Figure: The IPv6 fixed header (required)

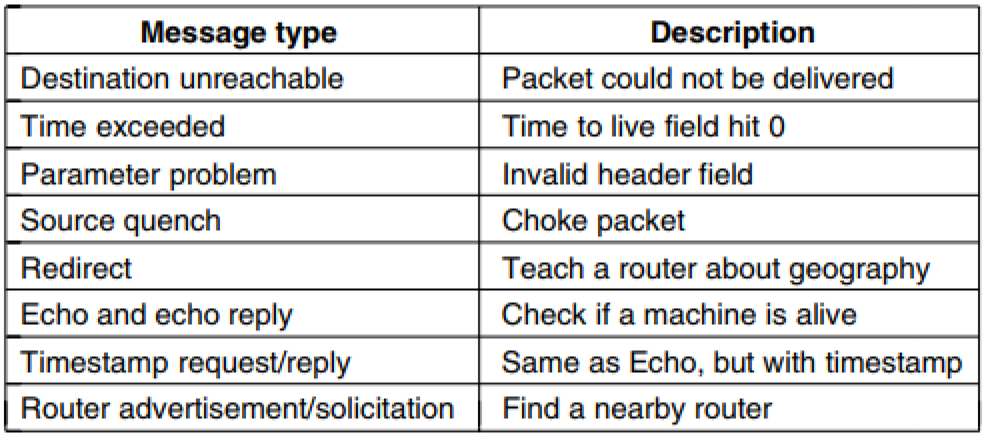
The IPv6 header is shown in Fig. The Version field is always 6 for IPv6 (and 4 for IPv4). During the transition period from IPv4, which has already taken more than a decade, routers will be able to examine this field to tell what kind of packet they have.
The Differentiated services field (originally called Traffic class) is used to
distinguish the class of service for packets with different real-time delivery requirements. It is used with the differentiated service architecture for quality of service in the same manner as the field of the same name in the IPv4 packet.
The Flow label field provides a way for a source and destination to mark groups of packets that have the same requirements and should be treated in the same way by the network, forming a pseudoconnection.
The Payload length field tells how many bytes follow the 40-byte header. The name was changed from the IPv4 Total length field because the meaning was changed slightly: the 40 header bytes are no longer counted as part of the length.
The Next header field lets the cat out of the bag. The reason the header could be simplified is that there can be additional (optional) extension headers.
The Hop limit field is used to keep packets from living forever. It is, in practice, the same as the Time to live field in IPv4, namely, a field that is decremented on each hop.
Next come the Source address and Destination address fields.

5.6.4 Internet Control Protocols:

In addition to IP, which is used for data transfer, the Internet has several companion control protocols that are used in the network layer. They include ICMP,
ARP, and DHCP.

IMCP—The Internet Control Message Protocol

The operation of the Internet is monitored closely by the routers. When something unexpected occurs during packet processing at a router, the event is reported to the sender by the ICMP (Internet Control Message Protocol). ICMP is also used to test the Internet. About a dozen types of ICMP messages are defined.
Each ICMP message type is carried encapsulated in an IP packet. The most important ones are listed in Fig.



Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo and echo reply	Check if a machine is alive
Timestamp request/reply	Same as Echo, but with timestamp
Router advertisement/solicitation	Find a nearby router

The DESTINATION UNREACHABLE message is used when the router cannot
locate the destination or when a packet with the DF bit cannot be delivered because a ‘‘small-packet’’ network stands in the way.
The TIME EXCEEDED message is sent when a packet is dropped because its TtL (Time to live) counter has reached zero.
The PARAMETER PROBLEM message indicates that an illegal value has been detected in a header field. This problem indicates a bug in the sending host’s IP software or possibly in the software of a router transited.
The SOURCE QUENCH message was long ago used to throttle hosts that were sending too many packets. When a host received this message, it was expected to slow down.
The REDIRECT message is used when a router notices that a packet seems to be routed incorrectly. It is used by the router to tell the sending host to update to a better route.
The ECHO and ECHO REPLY messages are sent by hosts to see if a given destination is reachable and currently alive. Upon receiving the ECHO message, the destination is expected to send back an ECHO REPLY message.
The TIMESTAMP REQUEST and TIMESTAMP REPLY messages are similar, except that the arrival time of the message and the departure time of the reply are recorded in the reply.
The ROUTER ADVERTISEMENT and ROUTER SOLICITATION messages are used to let hosts find nearby routers. A host needs to learn the IP address of at least one router to be able to send packets off the local network.

OSPF (Open Shortest Path First),— An Interior Gateway Routing Protocol

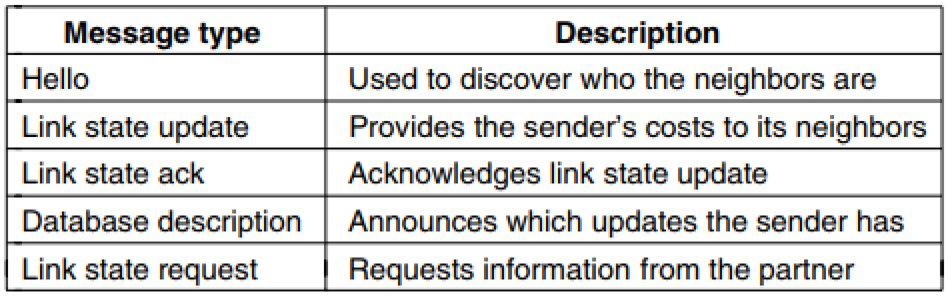
Open Shortest Path First (**OSPF**) is a routing protocol for Internet Protocol (IP) networks. It uses a link state routing (LSR) algorithm and falls into the group of interior gateway protocols (IGPs), operating within a single autonomous system (AS).
OSPF is a widely used IGP in large enterprise networks. IS-IS, another LSR-based protocol, is more common in large service provider networks.
Open Shortest Path First (OSPF) was designed as an interior gateway protocol, for use in an autonomous system such as a local area network (LAN). It implements Dijkstra's algorithm, also known as the shortest path first (SPF) algorithm. As a link-state routing protocol it was based on the link-state algorithm developed for the ARPANET in 1980 and the IS-IS (Intermediate-System to Intermediate-System), routing protocol. OSPF was first standardized in 1989.

As a link state routing protocol, OSPF maintains link state databases, which are really network topology maps, on every router on which it is implemented. The *state* of a given route in the network is the cost, and OSPF algorithm allows every router to calculate the cost of the routes to any given reachable destination
Given the long experience with other routing protocols, the group designing OSPF had a long list of requirements that had to be met. First, the algorithm had to be published in the open literature, hence the ‘‘O’’ in OSPF.
Second, the new protocol had to support a variety of distance metrics, including physical distance, delay, and so on.

Third, it had to be a dynamic algorithm, one that adapted to changes in the topology automatically and quickly. Fourth, and new for OSPF, it had to support routing based on type of service. The new protocol had to be able to route real-time traffic one way and other traffic a different way.

Fifth, and related to the above, OSPF had to do load balancing, splitting the load over multiple lines. Most previous protocols sent all packets over a single best route, even if there were two routes that were equally good.

Sixth, support for hierarchical systems was needed. OSPF supports both point-to-point links (e.g., SONET) and broadcast networks (e.g., most LANs). OSPF operates by abstracting the collection of actual networks, routers, and links into a directed graph in which each arc is assigned a weight (distance, delay, etc.).
A point-to-point connection between two routers is represented by a pair of arcs, one in each direction. Their weights may be different. A broadcast network is represented by a node for the network itself, plus a node for each router.



Message type	Description
Hello	Used to discover who the neighbors are
Link state update	Provides the sender's costs to its neighbors
Link state ack	Acknowledges link state update
Database description	Announces which updates the sender has
Link state request	Requests information from the partner

Figure: The five types of OSPF messages.

When a router boots, it sends HELLO messages on all of its point-to-point lines and multicasts them on LANs to the group consisting of all the other routers. During normal operation, each router periodically floods LINK STATE UPDATE messages to each of its adjacent routers. DATABASE DESCRIPTION messages give the sequence numbers of all the link state entries currently held by the sender. Either partner can request link state information from the other one by using LINK STATE REQUEST messages.

Border Gateway Protocol (BGP)—The Exterior Gateway Routing Protocol

Border Gateway Protocol (**BGP**) is a standardized exterior **gateway protocol** designed to exchange **routing** and reach ability information among autonomous systems (AS) on the Internet.

The Border Gateway Protocol makes routing decisions based on paths, network policies, or rule-sets configured by a network administrator and is involved in making core routing decisions.

BGP may be used for routing within an autonomous system. In this application it is referred to as Interior Border Gateway Protocol, Internal BGP, or iBGP. In contrast, the Internet application of the protocol may be referred to as Exterior Border Gateway Protocol, External BGP, or eBGP.

Within a single AS, OSPF and IS-IS are the protocols that are commonly used. Between ASes, a different protocol, called BGP (Border Gateway Protocol), is used. A different protocol is needed because the goals of an intradomain protocol and an interdomain protocol are not the same. All an intradomain protocol has to do is move packets as efficiently as possible from the source to the destination. It does not have to worry about politics. In contrast, interdomain routing protocols have to worry about politics a great deal. For example, a corporate AS might want the ability to send packets to any Internet site and receive packets from any Internet site.

BGP Messages:

- Open
  - Announces AS ID
  - Determines hold timer – interval between keep_alive or update messages, zero interval implies no keep_alive
- Keep_alive
  - Sent periodically (but before hold timer expires) to peers to ensure connectivity.
  - Sent in place of an UPDATE message
- Notification
  - Used for error notification
  - TCP connection is closed *immediately* after notification

Basic Operations:

- Learns multiple paths via internal and external BGP speakers
- Picks the best path and installs in the forwarding table
- Policies applied by influencing the best path selection.

Internet Group Management Protocol (IGMP):

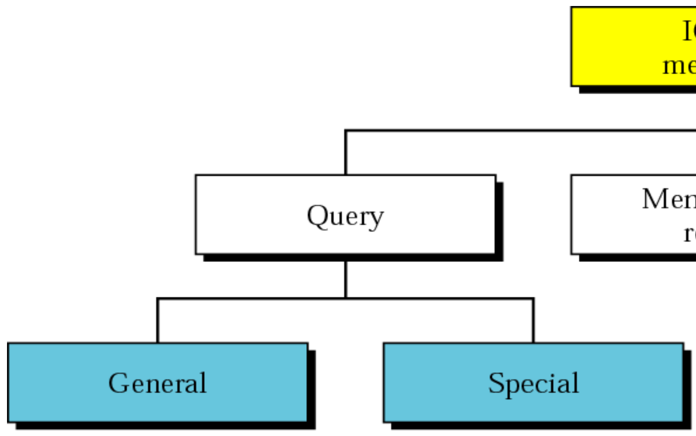
The Internet Group Management Protocol (IGMP) is a communications protocol used by hosts and adjacent routers on IPv4 networks to establish multicast group memberships. IGMP is an integral part of IP multicast.

IGMP can be used for one-to-many networking applications such as online streaming video and gaming, and allows more efficient use of resources when supporting these types of applications.

IGMP is used on IPv4 networks. Multicast management on IPv6 networks is handled by Multicast Listener Discovery (MLD) which is a part of ICMPv6 in contrast to IGMP's bare IP encapsulation.

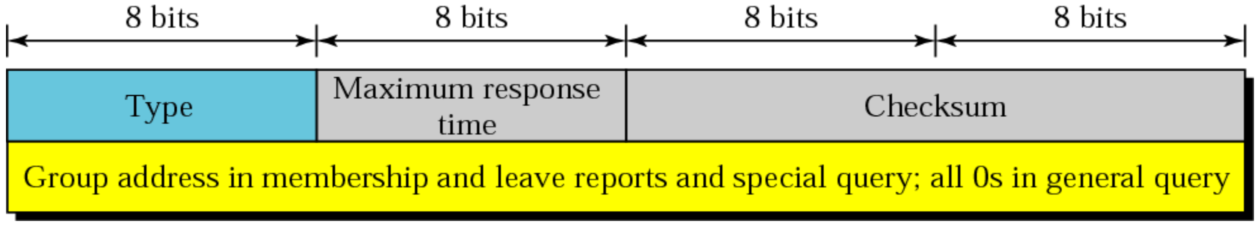
The Internet Group Management Protocol (IGMP) is an Internet protocol that provides a way for an Internet computer to report its multicast group membership to adjacent routers. Multicasting allows one computer on the Internet to send content to multiple other computers that have identified themselves as interested in receiving the originating computer's content. Multicasting can be used for such applications as updating the address books of mobile computer users in the field, sending out company newsletters to a distribution list, and "broadcasting" high-bandwidth programs of streaming media to an audience that has "tuned in" by setting up a multicast group membership.

IGMP has three types of messages: the query, the membership report, and the leave report. There are two types of query messages, general and special.



A multicast router connected to a network has a list of multicast addresses of the groups with at least one loyal member in that network. For each group, there is one router that has the duty of distributing the multicast packets destined for that group.

IGMP Message format:


8 bits
8 bits
8 bits
8 bits
Type
Maximum response time
Checksum
Group address in membership and leave reports and special query; all 0s in general query