UNIT-V

Objectives:

Familiarity with the basic routing protocols and congestion control protocols of network layer, and how they can be used to assist in network design and implementation.

Syllabus:

UNIT - I: Network layer

Network Layer design issues: store-and forward packet switching, services provided transport layers, implementation connection less services, implementation connection oriented services, comparison of virtual –circuit and datagram subnets Routing Algorithm –shortest path routing, flooding, distance vector routing, link state routing, Hierarchical routing, Broadcast routing, Multicasting routing, routing for mobiles Hosts, routing in Adhoc networks, congestion control algorithms- Load shedding, Congestion control in Data gram Subnet.

Routing algorithms- shortest path routing, distance vector, link state routing, and hierarchical routing. Congestion control algorithms-congestion control in virtual circuit subnets, datagram subnet, leaky bucket, token bucket. The network layer in the Internet: The IP protocol, IPAddresses-IPv4, IPv6.

Outcomes:

Students will be able to

- To understand the way protocols currently in use in the Internet work and the requirements for designing network protocols.
- To be able to capture and analyze network traffic.

- ➢ To have a grounding in the theory of basic network performance analysis
- ➢ Understand and building the skills of subnetting and routing mechanisms.

Learning Material

ROUTING ALGORITHMS

In order to transfer the packets from source to the destination, the network layer must determine the best route through which packets can be transmitted.

Whether the network layer provides datagram service or virtual circuit service, the main job of the network layer is to provide the best route. The routing protocol provides this job.

The routing protocol is a routing algorithm that provides the best path from the source to the destination. The best path is the path that has the "least-cost path" from source to the destination.

Routing is the process of forwarding the packets from source to the destination but the best route to send the packets is determined by the routing algorithm.

The routing algorithm is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on.

Properties of Routing Algorithms:-

- ✓ Correctness and Simplicity
- ✓ Robustness: Ability of the network to deliver packets via some route even in the face of failures.

- ✓ Stability: The algorithm should converge to equilibrium fast in the face of changing conditions in the network.
- ✓ Fairness and Optimality
- ✓ Efficiency: Minimum overhead.

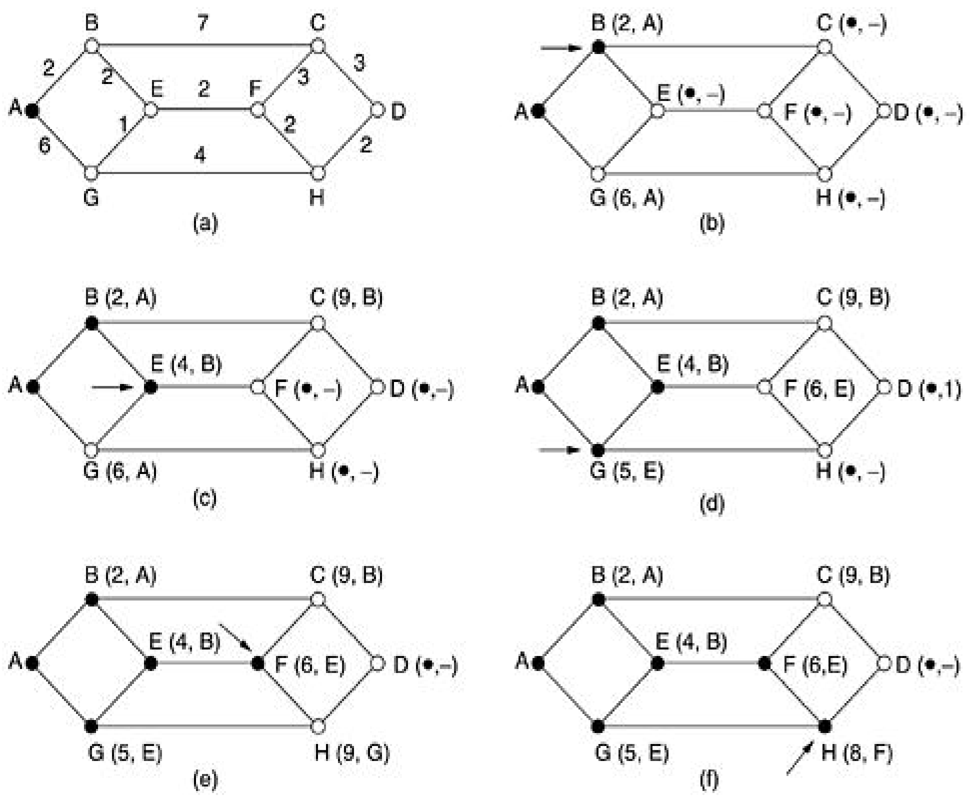
1.2.2 Shortest Path Routing

• A technique to study routing algorithms: The idea is to build a graph of the subnet, with each node of the graph representing a router and each arc of the graph representing a communication line (often called a link).

• To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.

• One way of measuring path length is the number of hops. Another metric is the geographic distance in kilometers. Many other metrics are also possible. For example, each arc could be labeled with the mean queuing and transmission delay for some standard test packet as determined by hourly test runs.

• In the general case, the labels on the arcs could be computed as a function of the distance, bandwidth, average traffic, communication cost, mean queue length, measured delay, and other factors. By changing the weighting function, the algorithm would then compute the "shortest" path measured according to any one of a number of criteria or to a combination of criteria.



The first five steps used in computing the shortest path from A to D. The arrows indicate the working node.

To illustrate how the labelling algorithm works, look at the weighted, undirected graph of Fig), where the weights represent, for example, distance.

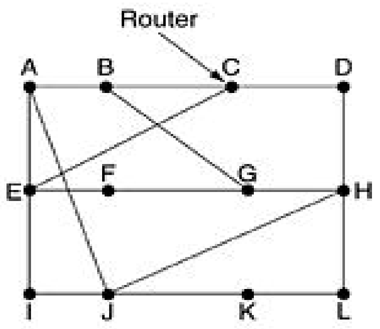
- We want to find the shortest path from A to D. We start out by marking node A as permanent, indicated by a filled-in circle.
- Then we examine, in turn, each of the nodes adjacent to A (the working node), relabeling each one with the distance to A.
- Whenever a node is relabelled, we also label it with the node from which the probe was made so that we can reconstruct the final path later.

- Having examined each of the nodes adjacent to A, we examine all the tentatively labelled nodes in the whole graph and make the one with the smallest label permanent, as shown in Fig.
- This one becomes the new working node. We now start at B and examine all nodes adjacent to it. If the sum of the label on B and the distance from B to the node being considered is less than the label on that node, we have a shorter path, so the node is relabelled.

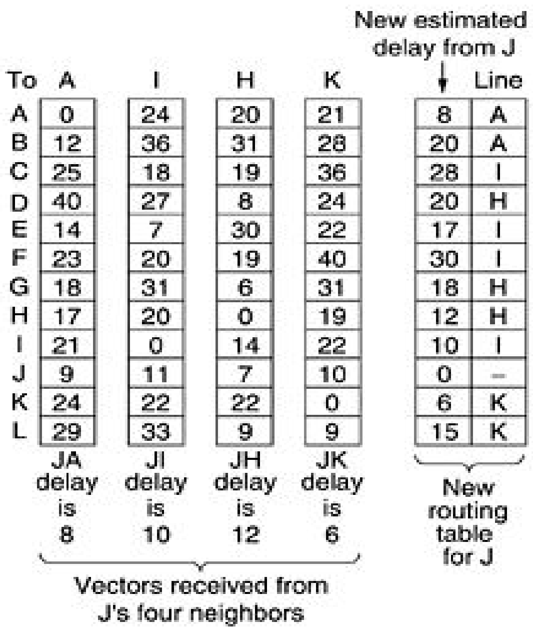
After all the nodes adjacent to the working node have been inspected and the tentative labels changed if possible, the entire graph is searched for the tentatively-labelled node with the smallest value. This node is made permanent and becomes the working node for the next round. Figure shows the first five steps of the algorithm.

1.2.4 Distance Vector Routing

- Distance vector routing algorithms operate by having each router maintain a table (i.e, a vector) giving the best known distance to each destination and which line to use to get there.
- These tables are updated by exchanging information with the neighbors.
- The distance vector routing algorithm is sometimes called by other names, most commonly the distributed Bellman-Ford routing algorithm and the Ford-Fulkerson algorithm, after the researchers who developed it (Bellman, 1957; and Ford and Fulkerson, 1962).
- It was the original ARPANET routing algorithm and was also used in the Internet under the name RIP.



(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		

Vectors received from J's four neighbors

New routing table for J

(b)

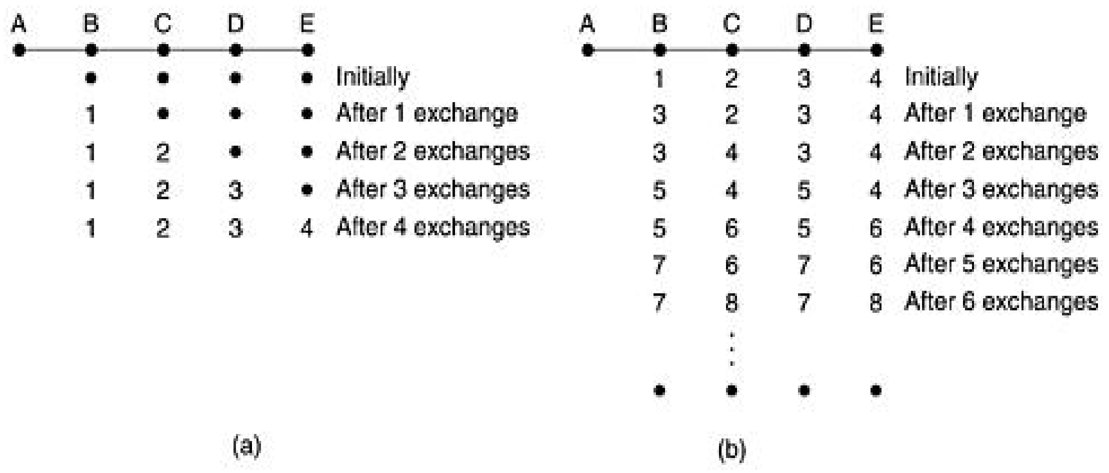
(a) A subnet. (b) Input from A, I, H, K, and the new routing table for J.

- Part (a) shows a subnet. The first four columns of part (b) show the delay vectors received from the neighbours 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 neighbours, A, I, H, and K as 8, 10, 12, and 6 msec, respectively.
- Consider how J computes its new route to router G. It knows that it can get to A in 8 msec, and 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

Distance vector routing works in theory but 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.



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)

Figure. The count-to-infinity problem.

Consider the five-node (linear) subnet of Fig. 5-10, where the delay metric is the number of hops. Suppose A is down initially and all the other routers know this. In other words, they have all recorded the delay to A as infinity.

Now let us consider the situation of Fig. 5-10(b), in which all the lines and routers are initially up. Routers B, C, D, and E have distances to A of 1, 2, 3, and 4, respectively.

Suddenly A goes down, or alternatively, the line 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 know that C's path runs through B itself. For all B knows, C might have ten lines 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.

On the second exchange, C notices that each of its neighbors claims to have a path to A of length 3. It picks one of the them at random and makes its new distance to A 4, as shown in the third row of Fig. 10(b). Subsequent exchanges produce the history shown in the rest of Fig. 10(b).

From this figure, it should be clear why bad news travels slowly: no router ever has a value more than one higher than the minimum of all its neighbors. Gradually, all routers work their way up to infinity, but the number of exchanges required depends on the numerical value used for infinity. For this reason, it is wise to set infinity to the longest path plus 1. If the metric is time delay, there is no well-defined upper bound, so a high value is needed to prevent a path with a long delay from being treated as down. Not entirely surprisingly, this problem is known as the count-to-infinity problem.

1.2.5 Link State Routing

Distance vector routing was used in the ARPANET until 1979, when it was replaced by link state routing. Two primary problems caused its demise.

First, since the delay metric was queue length, it did not take line bandwidth into account when choosing routes

The second problem also existed, namely, the algorithm often took too long to converge (the count-to-infinity problem).

For these reasons, it was replaced by an entirely new algorithm, now called link state routing. Variants of link state routing are now widely used.

The idea behind link state routing is simple and can be stated as five parts. Each router must do the following:

1. Discover its neighbors and learn their network addresses.

2. Measure the delay or cost to each of its neighbors.

3. Construct a packet telling all it has just learned.

4. Send this packet to all other routers.

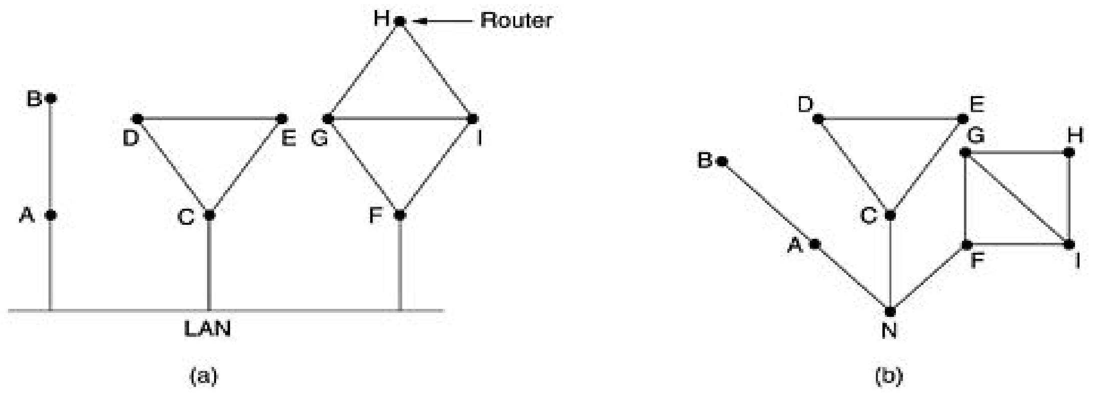
5. Compute the shortest path to every other router.

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 telling who it is.

Fig. 11(a) illustrates a LAN to which three routers, A, C, and F, are directly connected. Each of these routers is connected to one or more additional routers, as shown.

Figure 11. (a) Nine routers and a LAN. (b) A graph model of (a).



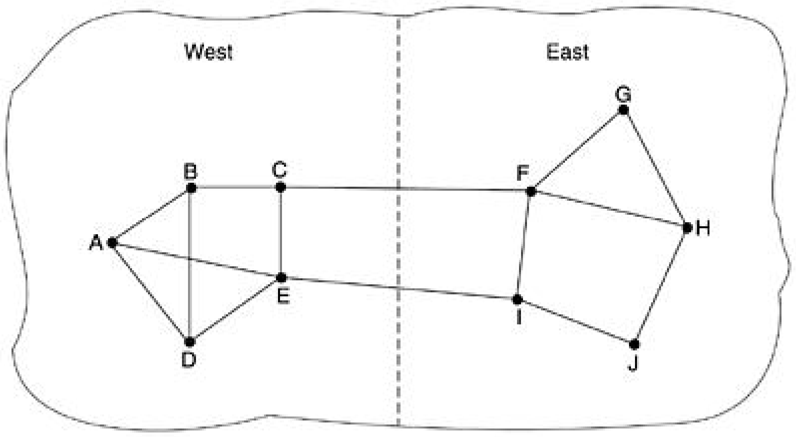
One way to model the LAN is to consider it as a node itself, as shown in Fig. 5-11(b). Here we have introduced a new, artificial node, N, to which A, C, and F are connected. The fact that it is possible to go from A to C on the LAN is represented by the path ANC here.

Measuring Line Cost

The link state routing algorithm requires each router to know, or at least have a reasonable estimate of the delay to each of its neighbors. 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. For even better results, the test can be conducted several times, and the average used. Of course, this method implicitly assumes the delays are symmetric, which may not always be the case.

Unfortunately, there is also an argument against including the load in the delay calculation. Consider the subnet of Fig. 5-12, which is divided into two parts, East and West, connected by two lines, CF and EI.

Figure 12. A subnet in which the East and West parts are connected by two lines.



Suppose that most of the traffic between East and West is using line CF, and as a result, this line is heavily loaded with long delays. Including queueing delay in the shortest path calculation will make EI more attractive. After the new routing tables have been installed, most of the East-West traffic will now go over EI, overloading this line. Consequently, in the next update, CF will appear to be the shortest path. As a result, the routing tables may oscillate wildly, leading to erratic routing and many potential problems

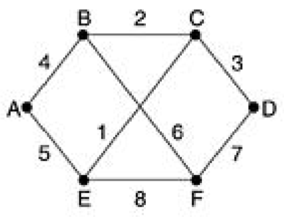
Building Link State Packets

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.

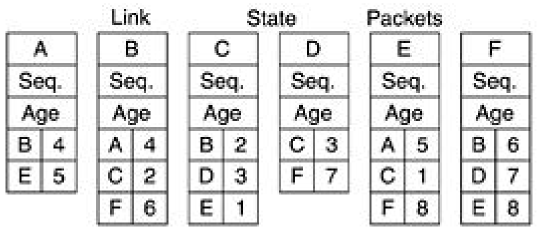
The packet starts with the identity of the sender, followed by a sequence number and age, and a list of neighbors.

For each neighbor, the delay to that neighbor is given. An example subnet is given in Fig.13(a) with delays shown as labels on the lines. The corresponding link state packets for all six routers are shown in Fig. 13(b).

Figure 13. (a) A subnet. (b) The link state packets for this subnet.



(a)



Link State Packets

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

(b)

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. Another possibility is to build them when some significant event occurs, such as a line or neighbor going down or coming back up again or changing its properties appreciably.

Distributing the Link State Packets

The trickiest part of the algorithm is distributing the link state packets reliably. As the packets are distributed and installed, the routers getting the first ones will change their routes.

Consequently, the different routers may be using different versions of the topology, which can lead to inconsistencies, loops, unreachable machines, and other problems.

First we will describe the basic distribution algorithm. Later we will give some refinements.

The fundamental idea is to use flooding to distribute the link state packets. 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. If it is new, it is forwarded on all lines except the one it arrived on. If it is a duplicate, it is discarded.

If a packet with a sequence number lower than the highest one seen so far ever arrives, it is rejected as being obsolete since the router has more recent data.

This algorithm has a few problems, but they are manageable.

First, if the sequence numbers wrap around, confusion will reign. The solution here is to use a 32-bit sequence number. With one link state packet per second, it would take 137 years to wrap around, so this possibility can be ignored.

Second, if a router ever crashes, it will lose track of its sequence number. If it starts again at 0, the next packet will be rejected as a duplicate.

Third, if a sequence number is ever corrupted and 65,540 is received instead of 4 (a 1-bit error), packets 5 through 65,540 will be rejected as obsolete, since the current sequence number is thought to be 65,540.

Computing the New Routes

Once a router has accumulated a full set of link state packets, it can construct the entire subnet graph because every link is represented. Every link is, in fact, represented twice, once for each direction. The two values can be averaged or used separately.

Now Dijkstra's algorithm can be run locally to construct the shortest path to all possible destinations. The results of this algorithm can be installed in the routing tables, and normal operation resumed.

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 and more bandwidth is also increasing.

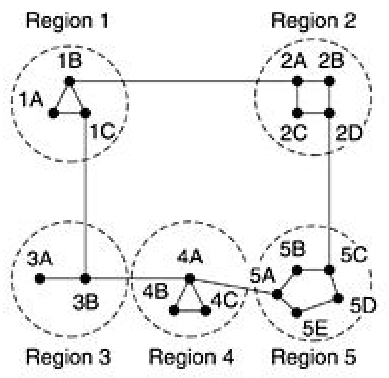
At a certain point the network may grow to the point where it is no longer feasible for every router to have an entry for every other router, so the routing will have to be done hierarchically, as it is in the telephone network.

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

For huge networks, a two-level hierarchy may be insufficient; it may be necessary to group the regions into clusters, the clusters into zones, the zones into groups, and so on.

Figure 15 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. 15(b). When routing is done hierarchically, as in Fig. 15(c), there are entries for all the local routers as before, but all other regions have been 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.

Figure 15. Hierarchical routing.



(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)

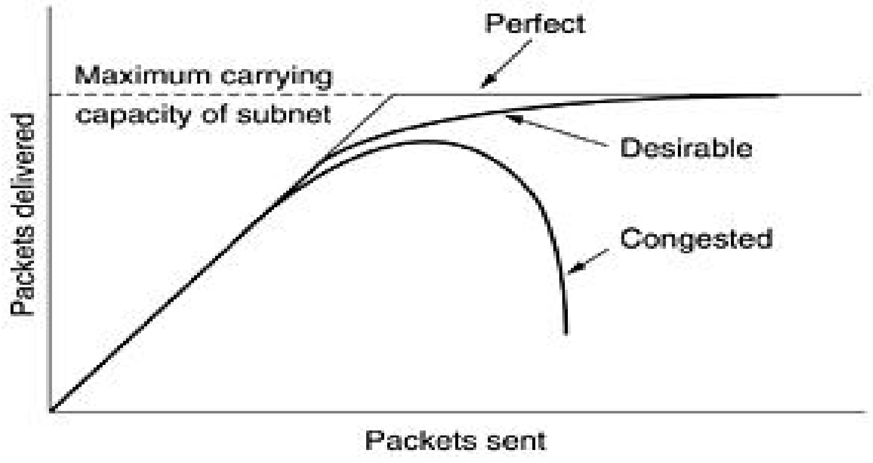
Unfortunately, these gains in space are not free. There is a penalty to be paid, and this penalty is in the form of increased path length. For example, the best route from 1A to 5C is via region 2, but with hierarchical routing all traffic to region 5 goes via region 3, because that is better for most destinations in region 5.

When a single network becomes very large, an interesting question is: How many levels should the hierarchy have? For example, consider a subnet with 720 routers. If there is no hierarchy, each router needs 720 routing table entries. If the subnet is partitioned into 24 regions of 30 routers each, each router needs 30 local entries plus 23 remote entries for a total of 53 entries. If a three-level hierarchy is chosen, with eight clusters, each containing 9 regions of 10 routers, each router needs 10 entries for local routers, 8 entries for routing to other regions within its own cluster, and 7 entries for distant clusters, for a total of 25 entries.

Congestion Control Algorithms

When too many packets are present in (a part of) the subnet, performance degrades. This situation is called congestion. Figure 25 depicts the symptom. When the number of packets dumped into the subnet by the hosts is within its carrying capacity, they are all delivered (except for a few that are afflicted with transmission errors) and the number delivered is proportional to the number sent. However, as traffic increases too far, the routers are no longer able to cope and they begin losing packets. This tends to make matters worse. At very high trafffic, performance collapses completely and almost no packets are delivered.

Figure 24. When too much traffic is offered, congestion sets in and performance degrades sharply.



Congestion can be brought on by several factors. If all of a sudden, streams of packets begin arriving on three or four input lines and all need the same output line, a queue will build up. If there is insufficient memory to hold all of them, packets will be lost. Adding more memory may help up to a point, but Nagle (1987) discovered that if routers have an infinite amount of memory, congestion gets worse, not better, because by the time packets get to the front of the queue, they have already timed out (repeatedly) and duplicates have been sent. All these packets will be dutifully forwarded to the next router, increasing the load all the way to the destination.

Slow processors can also cause congestion. If the routers' CPUs are slow at performing the bookkeeping tasks required of them (queueing buffers, updating tables, etc.), queues can build up, even though there is excess line capacity. Similarly, low-bandwidth lines can also cause congestion. Upgrading the lines but not changing the processors, or vice versa, often helps a little, but frequently just shifts the bottleneck. Also, upgrading part, but not all, of the system, often just moves the bottleneck somewhere else. The real problem is frequently a mismatch between parts of the system. This problem will persist until all the components are in balance.

It is worth explicitly pointing out the difference between congestion control and flow control, as the relationship is subtle. Congestion control has to do with making sure the subnet is able to carry the offered traffic. It is a global issue, involving the behavior of all the hosts, all the routers, the store-and-forwarding processing within the routers, and all the other factors that tend to diminish the carrying capacity of the subnet.

Flow control, in contrast, relates to the point-to-point traffic between a given sender and a given receiver. Its job is to make sure that a fast sender cannot continually transmit data faster than the receiver is able to absorb it. Flow control frequently involves some direct feedback from the receiver to the sender to tell the sender how things are doing at the other end.

To see the difference between these two concepts, consider a fiber optic network with a capacity of 1000 gigabits/sec on which a supercomputer is trying to transfer a file to a personal computer at 1 Gbps. Although there is no congestion (the network itself is not in trouble), flow control is needed to force the supercomputer to stop frequently to give the personal computer a chance to breathe.

At the other extreme, consider a store-and-forward network with 1-Mbps lines and 1000 large computers, half of which are trying to transfer files at 100 kbps to the other half. Here the problem is not that of fast senders overpowering slow receivers, but that the total offered traffic exceeds what the network can handle.

The reason congestion control and flow control are often confused is that some congestion control algorithms operate by sending messages back to the various sources telling them to slow down when the network gets into trouble. Thus, a host can get a "slow down" message either because the receiver cannot handle the load or because the network cannot handle it. We will come back to this point later.

We will start our study of congestion control by looking at a general model for dealing with it. Then we will look at broad approaches to preventing it in the first place.

Load Shedding

When none of the above methods make the congestion disappear, routers can bring out the heavy artillery: load shedding. Load shedding is a fancy way of saying that when routers are being inundated by packets that they cannot handle, they just throw them away. The term comes from the world of electrical power generation, where it refers to the practice of utilities intentionally blacking out certain areas to save the entire grid from collapsing on hot summer days when the demand for electricity greatly exceeds the supply.

A router drowning in packets can just pick packets at random to drop, but usually it can do better than that. Which packet to discard may depend on the applications running. For file transfer, an old packet is worth more than a new one because dropping packet 6 and keeping packets 7 through 10 will cause a gap at the receiver that may force packets 6 through 10 to be retransmitted (if the receiver routinely discards out-of-order packets). In a 12-packet file, dropping 6 may require 7 through 12 to be retransmitted, whereas dropping 10 may require only 10 through 12 to be retransmitted. In contrast, for multimedia, a new packet is more important than an old one. The former policy (old is better than new) is often called wine and the latter (new is better than old) is often called milk.

A step above this in intelligence requires cooperation from the senders. For many applications, some packets are more important than others. For example, certain algorithms for compressing video periodically transmit an entire frame and then send subsequent frames as differences from the last full frame. In this case, dropping a packet that is part of a difference is preferable to dropping one that is part of a full frame. As another example, consider transmitting a document containing ASCII text and pictures. Losing a line of pixels in some image is far less damaging than losing a line of readable text.

To implement an intelligent discard policy, applications must mark their packets in priority classes to indicate how important they are. If they do this, then when packets have to be discarded, routers can first drop packets from the lowest class, then the next lowest class, and so on. Of course, unless there is some significant incentive to mark packets as anything other than VERY IMPORTANT— NEVER, EVER DISCARD, nobody will do it.

The incentive might be in the form of money, with the low-priority packets being cheaper to send than the high-priority ones. Alternatively, senders might be allowed to send high-priority packets under conditions of light load, but as the load increased they would be discarded, thus encouraging the users to stop sending them.

Another option is to allow hosts to exceed the limits specified in the agreement negotiated when the virtual circuit was set up (e.g., use a higher bandwidth than allowed), but subject to the condition that all excess traffic be marked as low priority. Such a strategy is actually not a bad idea, because it makes more efficient use of idle resources, allowing hosts to use them as long as nobody else is interested, but without establishing a right to them when times get tough.

Random Early Detection

It is well known that dealing with congestion after it is first detected is more effective than letting it gum up the works and then trying to deal with it. This observation leads to the idea of discarding packets before all the buffer space is really exhausted. A popular algorithm for doing this is called RED (Random Early Detection) (Floyd and Jacobson, 1993). In some transport protocols (including TCP), the response to lost packets is for the source to slow down. The reasoning behind this logic is that TCP was designed for wired networks and wired networks are very reliable, so lost packets are mostly due to buffer overruns rather than transmission errors. This fact can be exploited to help reduce congestion.

By having routers drop packets before the situation has become hopeless (hence the ''early'' in the name), the idea is that there is time for action to be taken before it is too late. To determine when to start discarding, routers maintain a running average of their queue lengths. When the average queue length on some line exceeds a threshold, the line is said to be congested and action is taken.

Since the router probably cannot tell which source is causing most of the trouble, picking a packet at random from the queue that triggered the action is probably as good as it can do.

How should the router tell the source about the problem? One way is to send it a choke packet, as we have described. A problem with that approach is that it puts even more load on the already congested network. A different strategy is to just discard the selected packet and not report it. The source will eventually notice the lack of acknowledgement and take action. Since it knows that lost packets are generally caused by congestion and discards, it will respond by slowing down instead of trying harder. This implicit form of feedback only works when sources respond to lost packets by slowing down their transmission rate. In wireless networks, where most losses are due to noise on the air link, this approach cannot be used.

Congestion Control in Datagram Subnets

Let us now turn to some approaches that can be used in datagram subnets (and also in virtual-circuit subnets). Each router can easily monitor the utilization of its output lines and other resources. For example, it can associate with each line a real variable, u, whose value, between 0.0 and 1.0, reflects the recent utilization of that line. To maintain a good estimate of u, a sample of the instantaneous line utilization, f (either 0 or 1), can be made periodically and u updated according to

where the constant a determines how fast the router forgets recent history.

Whenever u moves above the threshold, the output line enters a "warning" state. Each newly-arriving packet is checked to see if its output line is in warning state. If it is, some action is taken. The action taken can be one of several alternatives, which we will now discuss.

The Warning Bit

The old DECNET architecture signaled the warning state by setting a special bit in the packet's header. So does frame relay. When the packet arrived at its destination, the transport entity copied the bit into the next acknowledgement sent back to the source. The source then cut back on traffic.

As long as the router was in the warning state, it continued to set the warning bit, which meant that the source continued to get acknowledgements with it set. The source monitored the fraction of acknowledgements with the bit set and adjusted its transmission rate accordingly. As long as the warning bits continued to flow in, the source continued to decrease its transmission rate. When they slowed to a trickle, it increased its transmission rate. Note that since every router along the path could set the warning bit, traffic increased only when no router was in trouble.

Choke Packets

The previous congestion control algorithm is fairly subtle. It uses a roundabout means to tell the source to slow down. Why not just tell it directly? In this approach, the router sends a choke packet back to the source host, giving it the destination found in the packet. The original packet is tagged (a header bit is turned on) so that it will not generate any more choke packets farther along the path and is then forwarded in the usual way.

When the source host gets the choke packet, it is required to reduce the traffic sent to the specified destination by X percent. Since other packets aimed at the same destination are probably already under way and will generate yet more choke packets, the host should ignore choke packets referring to that destination for a fixed time interval. After that period has expired, the host listens for more choke packets for another interval. If one arrives, the line is still congested, so the host reduces the flow still more and begins ignoring choke packets again. If no choke packets arrive during the listening period, the host may increase the flow again. The feedback implicit in this protocol can help prevent congestion yet not throttle any flow unless trouble occurs.

Hosts can reduce traffic by adjusting their policy parameters, for example, their window size. Typically, the first choke packet causes the data rate to be reduced to 0.50 of its previous rate, the next one causes a reduction to 0.25, and so on. Increases are done in smaller increments to prevent congestion from reoccurring quickly.

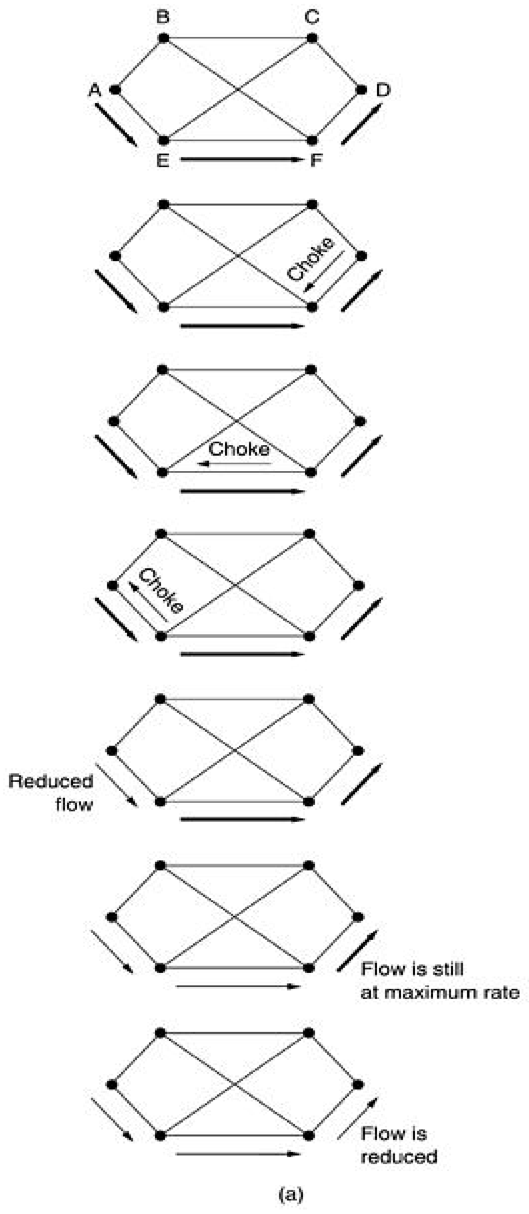
Several variations on this congestion control algorithm have been proposed. For one, the routers can maintain several thresholds. Depending on which threshold has been crossed, the choke packet can contain a mild warning, a stern warning, or an ultimatum.

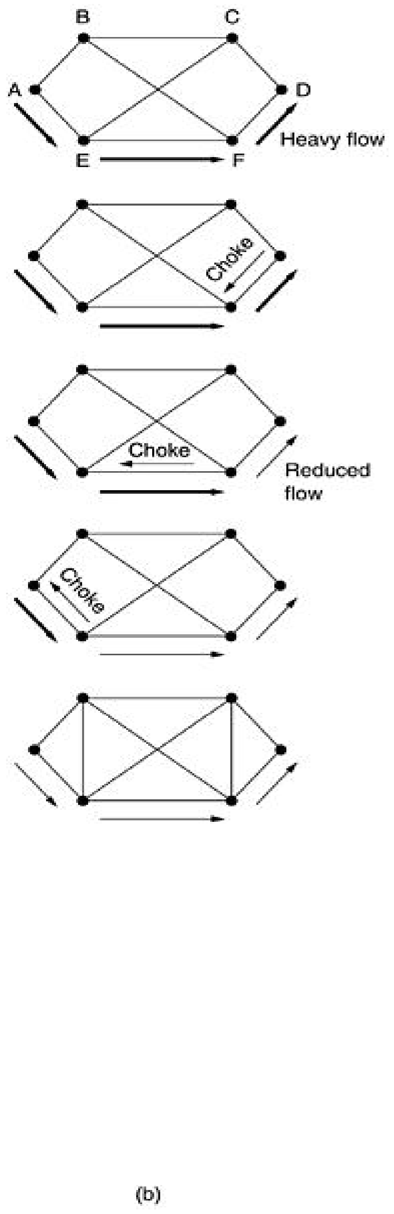
Another variation is to use queue lengths or buffer utilization instead of line utilization as the trigger signal. The same exponential weighting can be used with this metric as with u, of course.

Hop-by-Hop Choke Packets

At high speeds or over long distances, sending a choke packet to the source hosts does not work well because the reaction is so slow. Consider, for example, a host in San Francisco (router A in Fig. 5-28) that is sending traffic to a host in New York (router D in Fig. 5-28) at 155 Mbps. If the New York host begins to run out of buffers, it will take about 30 msec for a choke packet to get back to San Francisco to tell it to slow down. The choke packet propagation is shown as the second, third, and fourth steps in Fig. 5-28(a). In those 30 msec, another 4.6 megabits will have been sent. Even if the host in San Francisco completely shuts down immediately, the 4.6 megabits in the pipe will continue to pour in and have to be dealt with. Only in the seventh diagram in Fig. 5-28(a) will the New York router notice a slower flow.

Figure 25. (a) A choke packet that affects only the source. (b) A choke packet that affects each hop it passes through.





An alternative approach is to have the choke packet take effect at every hop it passes through, as shown in the sequence of Fig. 25(b). Here, as soon as the choke packet reaches F, F is required to reduce the flow to D. Doing so will require F to devote more buffers to the flow, since the source is still sending away at full blast, but it gives D immediate relief, like a headache remedy in a television commercial. In the next step, the choke packet reaches E, which tells E to reduce the flow to F. This action puts a greater demand on E's buffers but gives F immediate relief. Finally, the choke packet reaches A and the flow genuinely slows down.

In the network layer, before the network can make Quality of service guarantees, it must know what traffic is being guaranteed. One of the main causes of congestion is that traffic is often bursty.

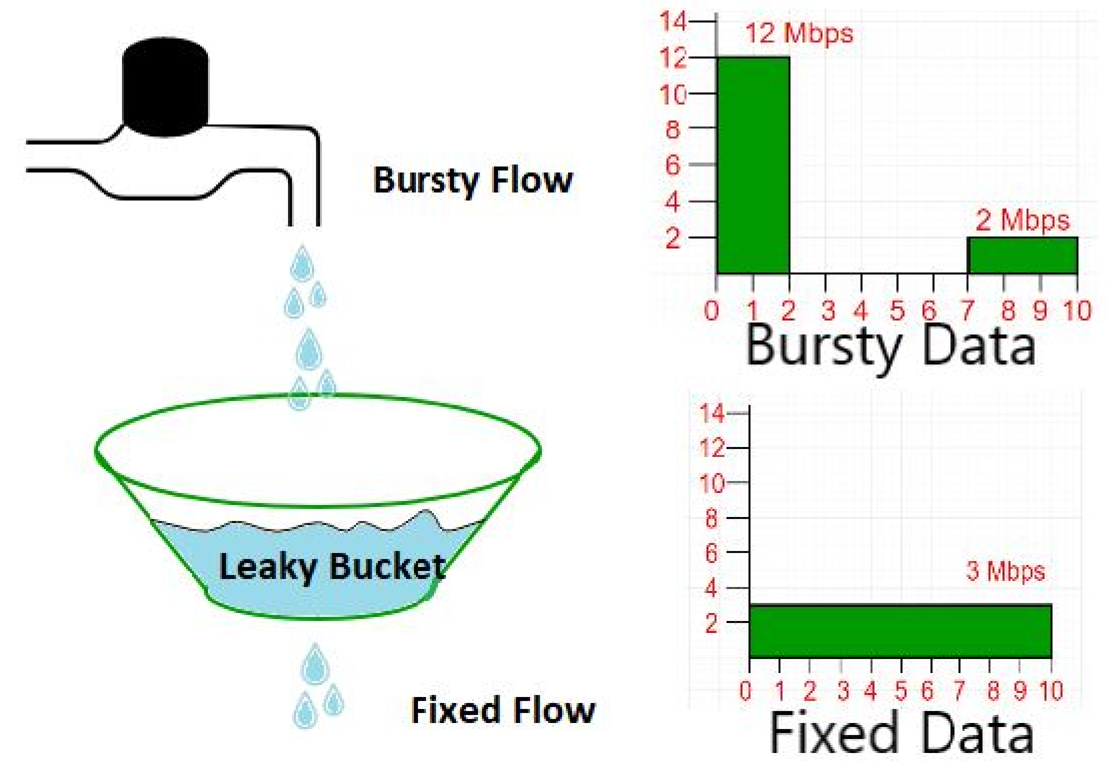
To understand this concept first we have to know little about traffic shaping. Traffic Shaping is a mechanism to control the amount and the rate of the traffic sent to the network. Approach of congestion management is called Traffic shaping. Traffic shaping helps to regulate rate of data transmission and reduces congestion.

There are 2 types of traffic shaping algorithms:

1. Leaky Bucket
2. Token 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.



In the figure, we assume that the network has committed a bandwidth of 3 Mbps for a host. The use of the leaky bucket shapes the input traffic to make it conform to this commitment. In Figure the host sends a burst of data at a rate of 12 Mbps for 2 s, for a total of 24 Mbits of data. The host is silent for 5 s and then sends data at a rate of 2 Mbps for 3 s, for a total of 6 Mbits of data. In all, the host has sent 30 Mbits of data in 10 s. The leaky bucket smooths the traffic by sending out data at a rate of 3 Mbps during the same 10 s.

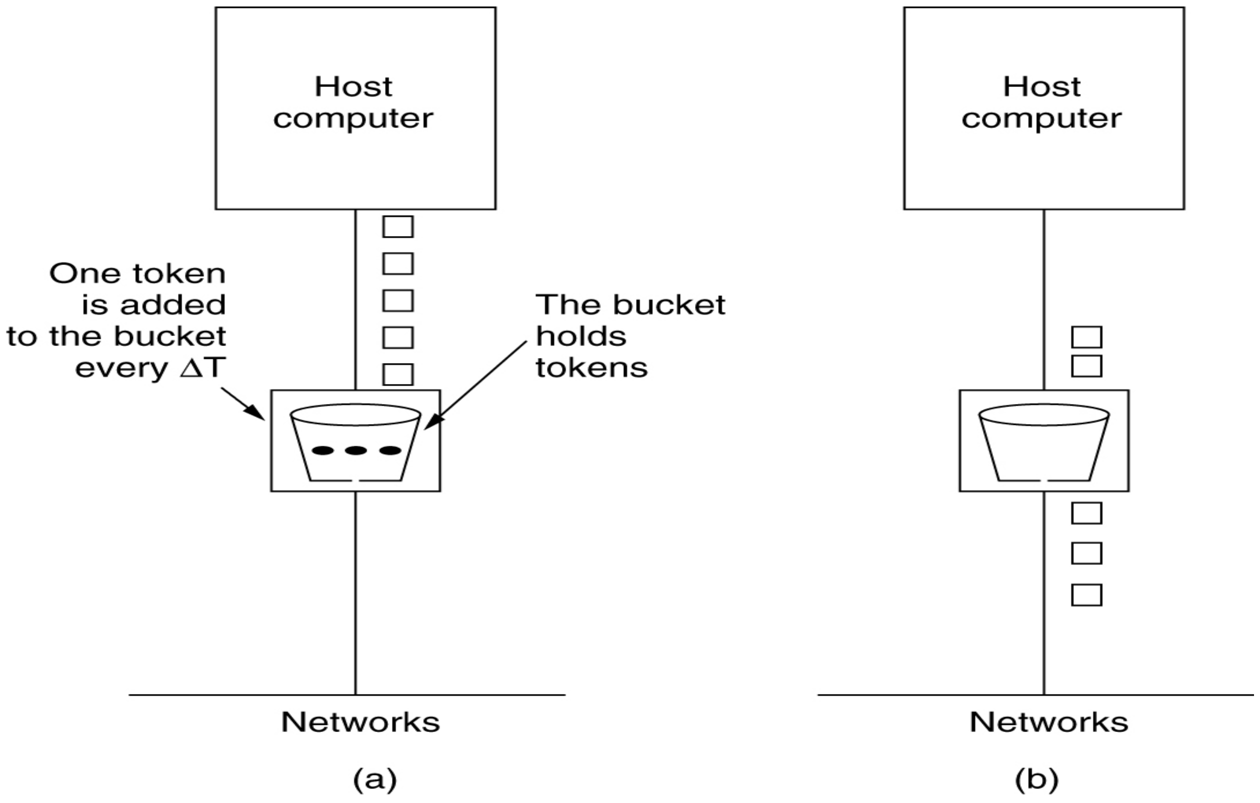
Without the leaky bucket, the beginning burst may have hurt the network by consuming more bandwidth than is set aside for this host. We can also see that the leaky bucket may prevent congestion.

Token Bucket Algorithm

The leaky bucket algorithm enforces output pattern at the average rate, no matter how bursty the traffic is. So in order to deal with the bursty traffic we need a flexible algorithm so that the data is not lost. One such algorithm is token bucket algorithm.

Steps of this algorithm can be described as follows:

1. In regular intervals tokens are thrown into the bucket.
2. The bucket has a maximum capacity.
3. If there is a ready packet, a token is removed from the bucket, and the packet is sent.
4. If there is no token in the bucket, the packet cannot be sent.



Let’s understand with an example,

In figure (A) we see a bucket holding three tokens, with five packets waiting to be transmitted. For a packet to be transmitted, it must capture and destroy one token. In figure (B) We see that three of the five packets have gotten through, but the other two are stuck waiting for more tokens to be generated.

Network Layer

IPV4

- ✓ Address space
- ✓ A protocol such as IPV4 that defines addresses has an address space.
- ✓ An address space is the total number of address used by the protocol.
- ✓ If a protocol uses N bits to define an address, the address space is 2^N.
- ✓ IPV4 uses 32-bit addresses, which means the address space is 2^32.
- ✓ Notations
- ✓ There are two common notations to show an IPV4 address.
    - o 1. Binary Notation
    - o 2. Dotted-Decimal notation
- ✓ In binary notation, the IPV4 address is displayed as 32-bits or 4-bytes.
- ✓ The following is an example of an IPV4 address in binary notation
    - o 01110101 10010101 00011101 00000010
- ✓ To make easier to read IPV4 address are written in decimal form with a decimal point(dot) separating the bytes.
    - o 117.149.29.
- ✓ The value ranging from 0 to 255
- ✓ For Example:::

- ✓ Classful Addressing
- ✓ IPV4 addressing, at its inception, used the concept of classes.

- ✓ In classful addressing, the address space is divided into five classes : A,B,C,D and E.
- ✓ One problem with classful addressing is that each class is divided into a fixed number of blocks with each block having a fixed size as shown in figure
- ✓ Classless Addressing
- ✓ To overcome address depletion and give more organizations access the internet classless addressing was designed and implemented.
- ✓ In this scheme, there are no classes, but the addresses are still granted in blocks.
- ✓ Address Blocks
- ✓ In classless addressing, any host wanted to connect internet, it is granted a block(range) of address.
- ✓ The block ovaries based on nature and size.
- ✓ Restrictions
- ✓ The address in a block must be contiguous, one after another.
- ✓ The no.of address in a block must be a power of 2 (1,2,4,8,....)
- ✓ The first address must be evenly divisible by the number of addresses.
- ✓ Network Address Translation (NAT)
- ✓ To access Internet, one public IP address is needed, but we can use private IP address in our private network.
- ✓ The idea of NAT is to allow multiple devices to access Internet through a single public address.

- ✓ IPV6:
- ✓ Advantages
- ✓ Larger address space.
- ✓ Better header format.
- ✓ New options.—additional functionalities

- ✓ Provide for extensions.--- suport new technologies
- ✓ Support for resource allocation.--- avoid traffic
- ✓ Support for more security.--confidentiality
- ✓ Structure
- ✓ An IPv6 address consists of
    - o 16 bytes (octets); it is 128 bits long.
- ✓ Hexadecimal colon notation:
- ✓ To make addresses more readable, IPv6 specifies hexadecimal colon notation.
- ✓ In this notation, 128 bits is divided into eight sections, each 2 bytes in length.
- ✓ Two bytes in hexadecimal notation requires four hexadecimal digits.
- ✓ Therefore, the address consists of 32 hexadecimal digits, with every four digits separated by a colon.
- ✓ Abbreviation:
- ✓ Although the IP address, even in hexadecimal format, is very long, many of the digits are zeros.
- ✓ In this case, we can abbreviate the address. The leading zeros of a section (four digits between two colons) can be omitted.
- ✓ Only the leading zeros can be dropped, not the trailing zeros.
- ✓ Using this form of abbreviation, 0074 can be written as 74, OOOF as F, and 0000 as 0.
- ✓ Note that 3210 cannot be abbreviated.
- ✓ Address Space
- ✓ IPV6 has a much larger address space; 2^128 addresses are available.
- ✓ The designers of IPv6 divided the address into several categories.
- ✓ A few leftmost bits, called the type prefix, in each address define its category.
- ✓ The type prefix is variable in length.

- ✓ In this way, there is no ambiguity; when an address is given, the type prefix can easily be determined.
- ✓ The third column shows the fraction of each type of address relative to the whole address space.
- ✓ Unicast Addresses:
- ✓ A unicast address defines a single computer. The packet sent to a unicast address must be delivered to that specific computer.
- ✓ IPv6 defines two types of unicast addresses: geographically based and provider-based.
- ✓ We discuss the second type here; the first type is left for future definition.
- ✓ The provider-based address is generally used by a normal host as a unicast address.
- ✓ Fields for the provider-based address are as follows:
- ✓ Type identifier. This 3-bit field defines the address as a provider-based address.
- ✓ Registry identifier. This 5-bit field indicates the agency that has registered the address.
- ✓ Currently three registry centers have been defined.
- ✓ 1. INTERNIC (code11000) is the center for North America;
- ✓ 2.RIPNIC (code 01000) is the center for European registration;
- ✓ 3. APNIC (code 10100) is for Asian and Pacific countries.
- ✓ Provider identifier: This variable-length field identifies the provider for Internet access (such as an ISP). A 16-bit length is recommended for this field.
- ✓ Subscriber identifier: When an organization subscribes to the Internet through a provider, it is assigned a subscriber identification. A 24-bit length is recommended for this field.

- ✓ Subnet identifier: Each subscriber can have many different sub networks, and each sub network can have an identifier. A 32 bit length is recommended for this field.
- ✓ Node identifier: The last field defines the identity of the node connected to a subnet. A length of 48 bits is recommended for this field.
- ✓ 3+8+16+32+48=128
- ✓ Multicast Addresses
- ✓ Multicast addresses are used to define a group of hosts instead of just one. A packet sent to a multicast address must be delivered to each member of the group.
- ✓ The second field is a flag that defines the group address as either permanent or transient.
- ✓ A permanent group address is defined by the Internet authorities and can be accessed at all times.
- ✓ A transient group address, on the other hand, is used only temporarily. Systems engaged in a teleconference, for example, can use a transient group address.
- ✓ The third field defines the scope of the group address. Many different scopes have been defined.
- ✓ Anycast Addresses
- ✓ IPv6 also defines anycast addresses. An anycast address, like a multicast address, also defines a group of nodes.
- ✓ However, a packet destined for an anycast address is delivered to only one of the members of the anycast group.
- ✓ Reserved Addresses
- ✓ Another category in the address space is the reserved address. These addresses start with eight O's (type prefix is 00000000).
- ✓ A few subcategories are defined in this category represent in diagram.

- ✓ An unspecified address is used when a host does not know its own address and sends an inquiry to find its address.
- ✓ A loopback address is used by a host to test itself without going into the network.
- ✓ A compatible address is used during the transition from IPv4 to IPv6.
    - ▪ It is used when a computer using IPv6 wants to send a message to another computer using IPv6, but the message needs to pass through a part of the network that still operates in IPv4.
- ✓ Mapped address is also used during transition. However, it is used when a computer that has migrated to IPv6 wants to send a packet to a computer still using IPv4.
- ✓ Local Addresses
- ✓ These addresses are used when an organization wants to use IPv6 protocol without being connected to the global Internet.
- ✓ In other words, they provide addressing for private networks.
- ✓ A link local address is used in an isolated subnet; a site local address is used in an isolated site with several subnets.

SECTION-A

Assignment-Cum-Tutorial Questions

Objective Questions

1. The router algorithm takes the decision to changes the route when [ ]

a) router changes b) topology changes

c) user changes d) transmission time does not change

2. The network layer protocol of internet is [ ]

a) Ethernet b) internet protocol

c) hypertext transfer protocol d) none of the mentioned

3. The network layer concerns with [ ]

a) bits b) frames c) packets d) none

4.In link state routing, after the construction of link state packets new routes are computed using [ ]

a) Bellman Ford algorithm b) DES algorithm

c) Dijkstra's algorithm d) Leaky bucket algorithm

5.Count-to-Infinity problem occurs in [ ]

a) distance vector routing b) short path first

c) link state routing d) hierarchical routing

6. In distance vector routing algorithm, the routing tables are updated[ ]

a) by exchanging information with the neighbours

b) automatically

c) using the backup database

d) by the server

7.In Congestion Control, a bit can be set in a packet moving in direction opposite to congestion in [ ]

a)Implicit Signaling b)Backward Signaling

c) Source Signaling d)Data Signaling

8. A packet which is sent by a node to source to inform it of congestion is called [ ]

a)Congestion Packet b)Change Packet

c)Choke Packet d)Control Packet

9. In _______ congestion control, policies are applied to prevent congestion before it happens. [ ]

a) open-loop b) closed-loop c) either (a) or (b) d) neither (a) nor (b)

10. ________ is a characteristic that a flow needs. Lack of it means losing a packet or acknowledgment, which entails retransmission[ ]

(a)Reliability b) Delay c) Jitter d) Bandwidth

SECTION-B

Subjective Questions

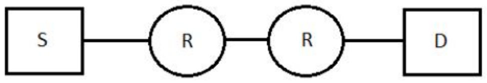
1. With an example explain the distance vector routing algorithms used in computer networks.

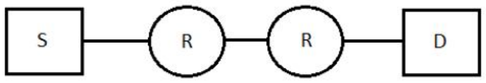
2. State and Justify “Link state routing shows better performance than the Distance vector routing”, and explain it comprehensively.
3. With an example explain the Hierarchical routing algorithms used in computer networks.
4. What is Count to infinity problem? Explain with suitable example.
5. Explain congestion control in Datagram subnets.
6. Explain congestion control in virtual circuit subnets
7. Explain leaky bucket algorithm.
8. Explain token bucket algorithm
9. Discuss the Internet Protocol addressing methods.
10. Explain with an example of subnet mask in internet protocol.

Problems

1. Assume that source S and destination D are connected through two intermediate routers labeled R. Determine how many times each packet has to

visit the network layer and the data link layer during a transmission from S to D.





2. Change the following IP addresses from binary notation to dotted-decimal notation.

i. 01111111 11110000 01100111 01111101

ii. 10101111 11000000 11111000 00011101

3. Change the following IP addresses from dotted-decimal notation to binary notation.

i. 114.34.2.8

ii. 129.14.6.8