Unit-3

Network Layer:

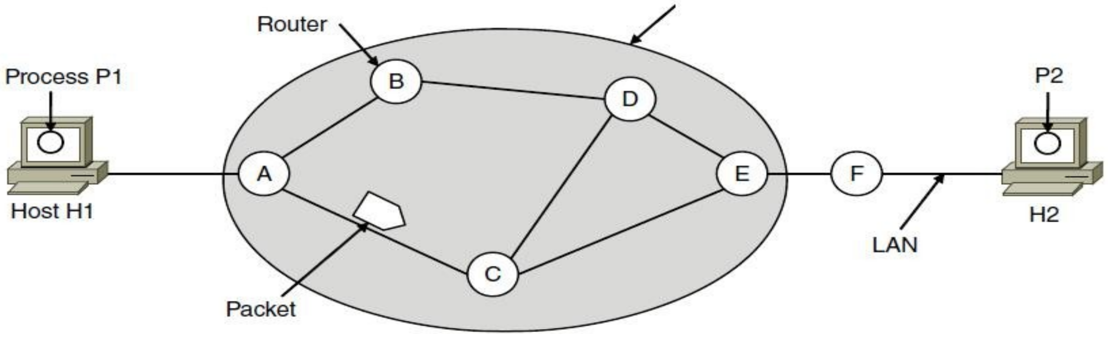
The network layer is concerned with getting packets from the source all the way to the destination. Getting to the destination may require making many hops at intermediate routers along the way. This function clearly contrasts with that of the data link layer, which has the more modest goal of just moving frames from one end of a wire to the other. Thus, 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. It must also take care when choosing routes to avoid overloading some of the communication lines and routers while leaving others idle. Finally, when the source and destination are in different networks, new problems occur. It is up to the network layer to deal with them.

Network Layer: Design Issues

1. Store-and-forward packet switching
2. Services provided to transport layer
3. Implementation of connectionless service
4. Implementation of connection-oriented service
5. Comparison of virtual-circuit and datagram networks

1 Store-and-forward packet switching



A host with a packet to send transmits it to the nearest router, either on its own LAN or over a point-to-point link to the ISP. The packet is stored there until it has fully arrived and the link has finished its processing by verifying the checksum. Then it is forwarded to the next router along the path until it reaches the destination host, where it is delivered. This mechanism is store-and-forward packet switching.

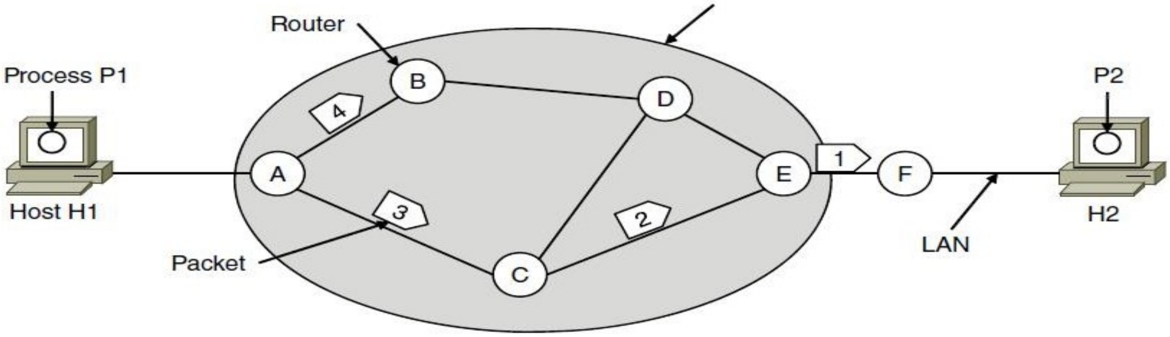
2 Services provided to transport layer

The network layer provides services to the transport layer at the network layer/transport layer interface. The services need to be carefully designed with the following goals in mind:

1. Services independent of router technology.
2. Transport layer shielded from number, type, topology of routers.
3. Network addresses available to transport layer use uniform numbering plan, even across LANs and WANs

3 Implementation of connectionless 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**.



A’s table (initially)

A	☒
B	B
C	C
D	B
E	C
F	C

Dest. Line

A’s table (later)

A	☒
B	B
C	C
D	B
E	D
F	D

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

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 eachof them in turn to router *A*.

Every router has an internal table telling it where to send packets for each of the possible destinations. Each table entry is a pair(destination and the outgoing line). Only directly connected lines can be used.

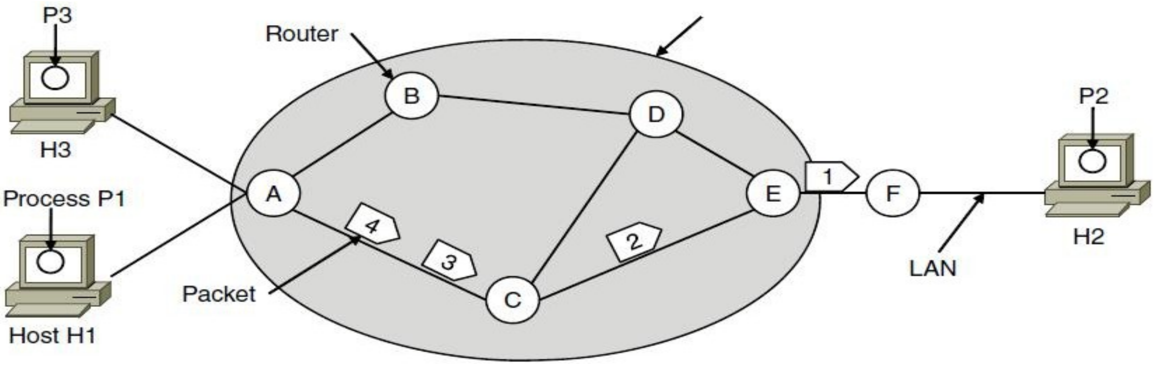
A’s initial routing table is shown in the figure under the label ‘‘initially.’’

At *A*, packets 1, 2, and 3 are stored briefly, having arrived on the incoming link. Then each packet is forwarded according to *A*’s table, onto the outgoing link to *C* within a new frame. Packet 1 is then forwarded to *E* and then to *F*.

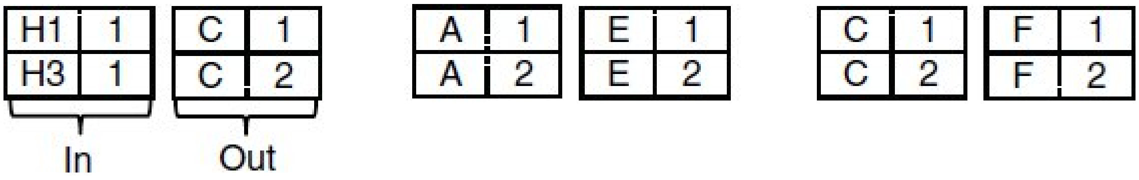
However, something different happens to packet 4. When it gets to *A* it is sent to router *B*, even though it is also destined for *F*. For some reason (traffic jam along ACE path), *A* decided to send packet 4 via a different route than that of the first three packets. Router A updated its routing table, as shown under the label ‘‘later.’’

The algorithm that manages the tables and makes the routing decisions is called the **routing algorithm**.

4 Implementation of connection-oriented service



A’s table C’s Table E’s Table



H1	1	C	1
H3	1	C	2

A	1	E	1
A	2	E	2

C	1	F	1
C	2	F	2

If connection-oriented service is used, a path from the source router all the way to thedestination router must be established before any data packets can be sent. This connection is called a **VC** (**virtual circuit**), and the network is called a **virtual-circuit network**

When a connection is established, a route from the source machine to the destination machine is chosen as part of the connection setup and stored in tables inside the routers. That route is used for all traffic flowing over the connection, exactly the same way that the telephone system works. When the connection is released, the virtual circuit is also terminated. With connection-oriented service, each packet carries an identifier telling which virtual circuit it belongs to.

As an example, consider the situation shown in Figure. 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.

Now let us consider what happens if *H3* also wants to establish a connection to *H2*. It chooses connection identifier 1 (because it is initiating the connection and this is its only connection) and tells the network to establish the virtual circuit.

This leads to the second row in the tables. Note that we have a conflict here because although *A* can easily distinguish connection 1 packets from *H1* from connection 1 packets from *H3*, *C* cannot do this. For this reason, *A* assigns a different connection identifier to the outgoing traffic for the second connection. Avoiding conflicts of this kind is why routers need the ability to replace connection identifiers in outgoing packets.In some contexts, this process is called **label switching**. An example of a connection-oriented network service is **MPLS** (**Multi Protocol Label Switching**).

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

Routing Algorithms

The main function of NL (Network Layer) is routing packets from the source machine to the destination machine. There are two processes inside router:

a) One of them handles each packet as it arrives, looking up the outgoing line to use for it in the routing table. This process is forwarding.

b) The other process is responsible for filling in and updating the routing tables. That is where the routing algorithm comes into play. This process is routing.

Regardless of whether routes are chosen independently for each packet or only when new connections are established, certain properties are desirable in routing algorithm **correctness, simplicity, robustness, stability, fairness, optimality**

Routing algorithms can be grouped into two major classes:

1) Non adaptive (Static Routing)
2) adaptive. (Dynamic Routing)

Non adaptive algorithm do not base their routing decisions on measurements or estimates of the current traffic and topology. Instead, the choice of the route to use to get from I to J is computed in advance, off line, and downloaded to the routers when the network is booted. This procedure is sometimes called static routing.

Adaptive algorithm, in contrast, change their routing decisions to reflect changes in thetopology, and usually the traffic as well.
Adaptive algorithms differ in

1) Where they get their information (e.g., locally, from adjacent routers, or from all routers),
2) When they change the routes (e.g., every ΔT sec, when the load changes or when the topology changes), and
3) What metric is used for optimization (e.g., distance, number of hops, or estimated transittime).

This procedure is called dynamic routing

Different Routing Algorithms

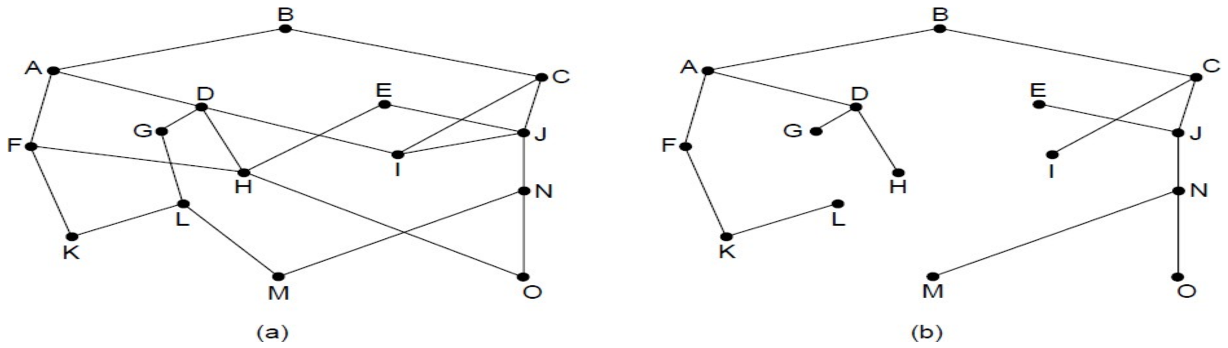
- Optimality principle
- Shortest path algorithm
- Flooding
- Distance vector routing
- Link state routing
- Hierarchical Routing

The Optimality Principle

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

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**. The goal of all routing algorithms is to discover and use the sink trees for all routers



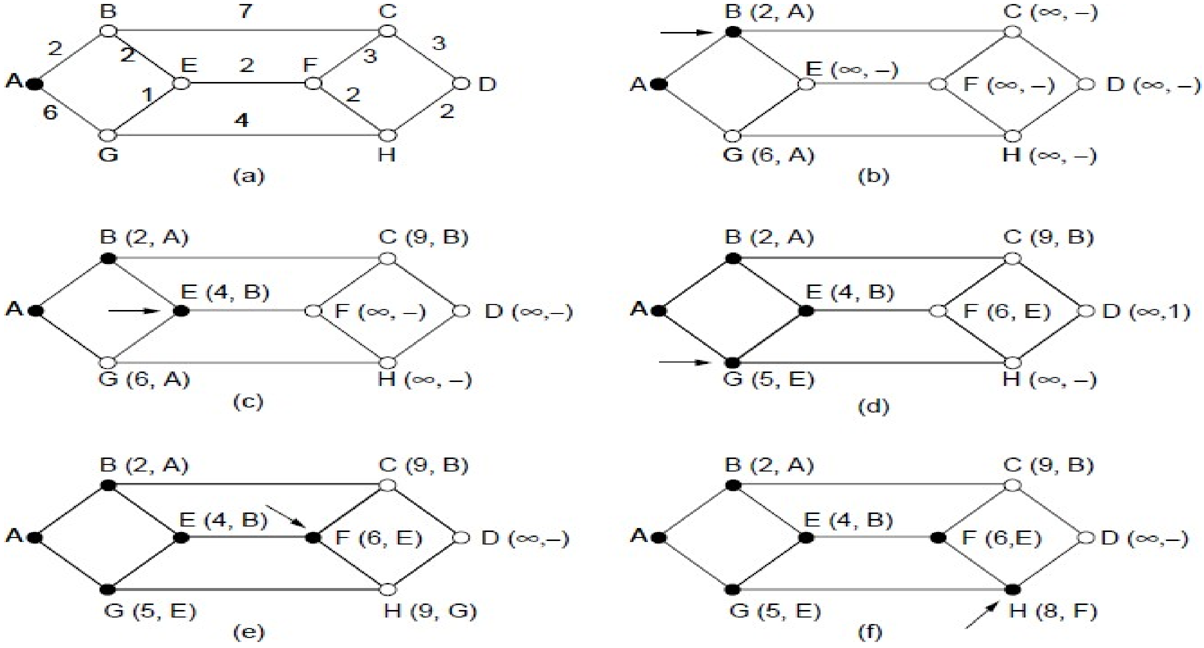
(a) A network. (b) A sink tree for router *B*.

Shortest Path Routing

The idea is to build a graph of the subnet, with each node of the graph representing a routerand each arc 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 pathbetween them on the graph

1. Start with the local node (router) as the root of the tree. Assign a cost of 0 to this node andmake it the first permanent node.
2. Examine each neighbor of the node that was the last permanent node.
3. Assign a cumulative cost to each node and make it tentative
4. Among the list of tentative nodes
    a. Find the node with the smallest cost and make it Permanent
    b. If a node can be reached from more than one route then select the route with theshortest cumulative cost.
5. Repeat steps 2 to 4 until every node becomes permanent



Flooding

- Another static algorithm 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, which 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.
- A variation of flooding that is slightly more practical is selective flooding. In this algorithm the routers do not send every incoming packet out on every line, only on those lines that are going approximately in the right direction.
- Flooding is not practical in most applications.

Distance Vector Routing

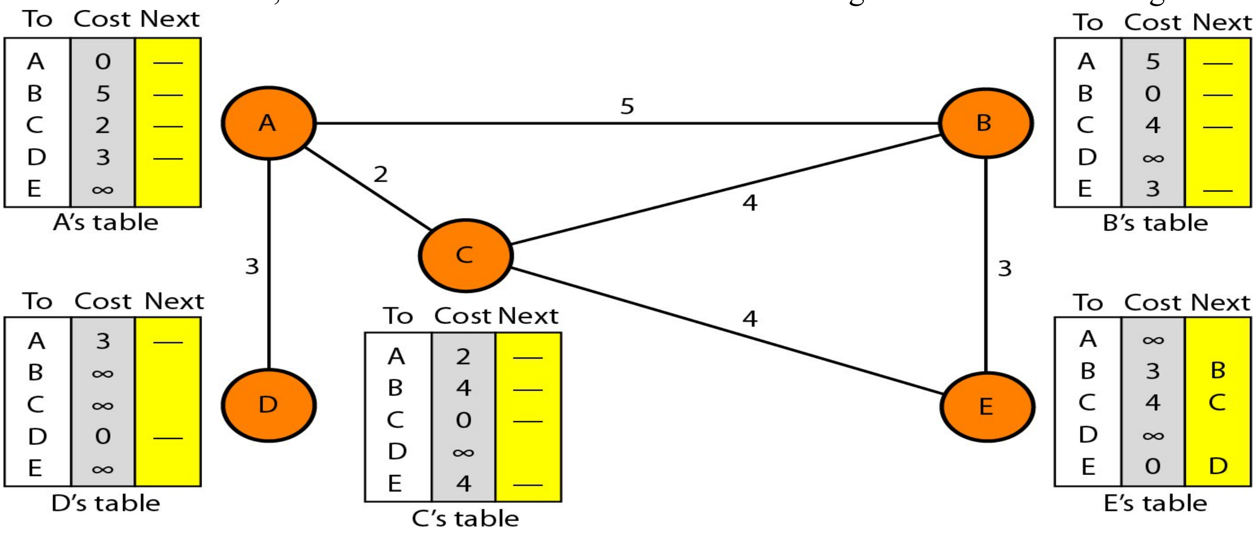
In distance vector routing, the least-cost route between any two nodes is the route with minimum distance. In this protocol, as the name implies, each node maintains a vector (table) of minimum distances to every node.

Mainly 3 things in this

Initialization
Sharing
Updating

Initialization

Each node can know only the distance between itself and its immediate neighbors, those directly connected to it. So for the moment, we assume that each node can send a message to the immediate neighbors and find the distance between itself and these neighbors. Below fig shows the initial tables for each node. The distance for any entry that is not a neighbor is marked as infinite (unreachable).



To	Cost	Next
A	0	—
B	5	—
C	2	—
D	3	—
E	∞	

A's table

To	Cost	Next
A	5	—
B	0	—
C	4	—
D	∞	
E	3	—

B's table

To	Cost	Next
A	3	—
B	∞	
C	∞	
D	0	—
E	∞	

D's table

To	Cost	Next
A	2	—
B	4	—
C	0	—
D	∞	
E	4	—

C's table

To	Cost	Next
A	∞	
B	3	B
C	4	C
D	∞	
E	0	D

E's table

Initialization of tables in distance vector routing

Sharing

The whole idea of distance vector routing is the sharing of information between neighbors. Although node A does not know about node E, node C does. So if node C shares its routing table with A, node A can also know how to reach node E. On the other hand, node C does not know how to reach node D, but node A does. If node A shares its routing table with node C, node C also knows how to reach node D. In other words, nodes A and C, as immediate neighbors, can improve their routing tables if they help each other.

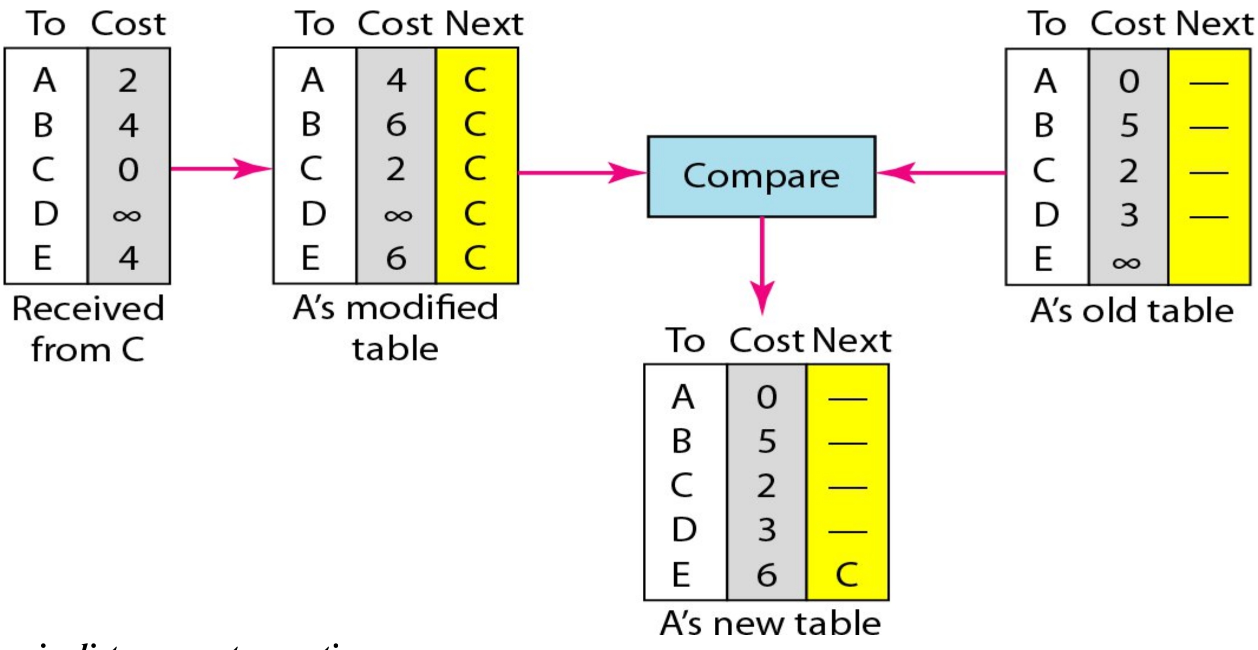
NOTE: In distance vector routing, each node shares its routing table with its immediate neighbors periodically and when there is a change

Updating

When a node receives a two-column table from a neighbor, it needs to update its routingtable. Updating takes three steps:

1. The receiving node needs to add the cost between itself and the sending node to each valuein the second column. (x+y)
2. If the receiving node uses information from any row. The sending node is the next node inthe route.
3. The receiving node needs to compare each row of its old table with the corresponding rowof the modified version of the received table.
    a. If the next-node entry is different, the receiving node chooses the row with thesmaller cost. If there is a tie, the old one is kept.
    b. If the next-node entry is the same, the receiving node chooses the new row.

For example, suppose node C has previously advertised a route to node X with distance 3. Suppose that now there is no path between C and X; node C now advertises this route with a distance of infinity. Node A must not ignore this value even though its old entry is smaller. The old route does not exist anymore. The new route has a distance of infinity.



To	Cost
A	2
B	4
C	0
D	∞
E	4

Received from C

To	Cost	Next
A	4	C
B	6	C
C	2	C
D	∞	C
E	6	C

A's modified table

To	Cost	Next
A	0	—
B	5	—
C	2	—
D	3	—
E	∞	

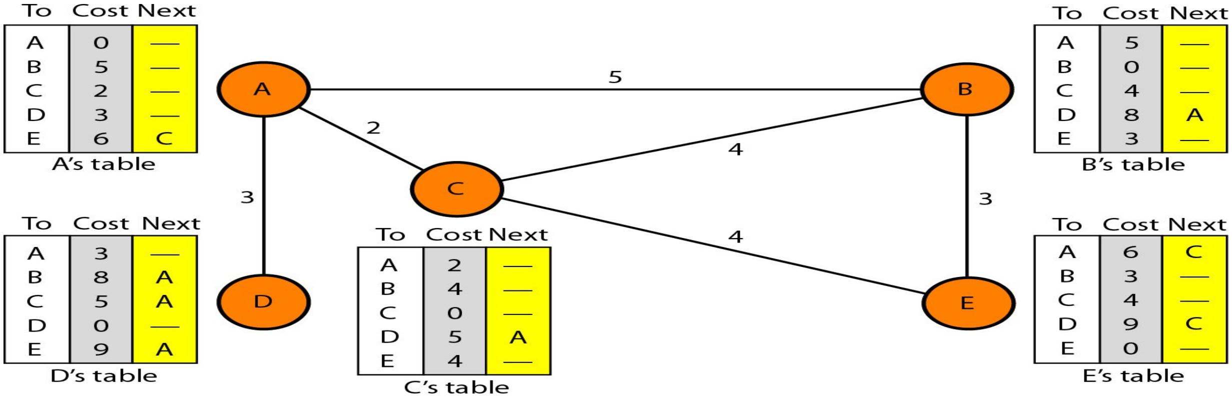
A's old table

To	Cost	Next
A	0	—
B	5	—
C	2	—
D	3	—
E	6	C

A's new table

Updating in distance vector routing

Final Diagram



A's table

To	Cost	Next
A	0	—
B	5	—
C	2	—
D	3	—
E	6	C

B's table

To	Cost	Next
A	5	—
B	0	—
C	4	—
D	8	A
E	3	—

D's table

To	Cost	Next
A	3	—
B	8	A
C	5	A
D	0	—
E	9	A

C's table

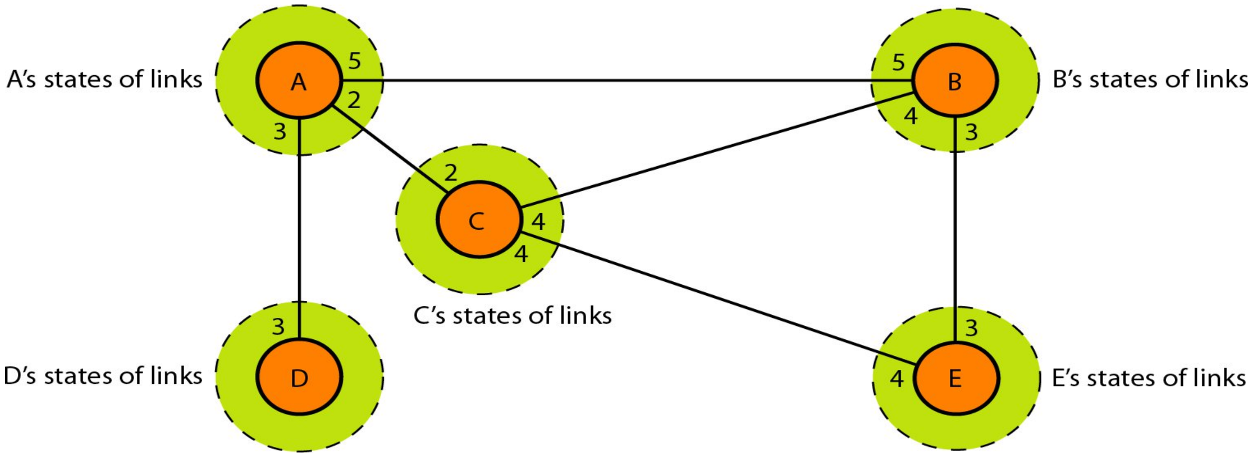
To	Cost	Next
A	2	—
B	4	—
C	0	—
D	5	A
E	4	—

E's table

To	Cost	Next
A	6	C
B	3	—
C	4	—
D	9	C
E	0	—

Link State Routing (Dijkstra algorithm)

Link state routing is based on the assumption that, although the global knowledge about the topology is not clear, each node has partial knowledge: it knows the state (type, condition, and cost) of its links. **In other words, the whole topology can be compiled from the** partial knowledgeof each node



Building Routing Tables

1. Creation of the states of the links by each node, called the link state packet (LSP).
2. Dissemination of LSPs to every other router, called **flooding, in an efficient and** reliable way.
3. Formation of a shortest path tree for each node.
4. Calculation of a routing table based on the shortest path tree

I. **Creation of Link State Packet (LSP)** A link state packet can carry a large amount of information. For the moment, we assume that it carries a minimum amount of data: the node identity, the list of links, a sequence number, and age. The first two, node identity and the list of links, are needed to make the topology. The third, sequence number, facilitates flooding and distinguishes new LSPs from old ones. The fourth, age, prevents old LSPs from remaining in the domain for a long time.

LSPs are generated on two occasions:

1. When there is a change in the topology of the domain

a. on a periodic

2. It sends a copy of it out of each interface except the one from which the packet arrived. This guarantees that flooding stops somewhere in the domain (where a node has only one interface basis: The period in this case is much longer compared to distance vector. The timer set for periodic dissemination is normally in the range of **60 min or 2 h** based onthe implementation. A longer period ensures that flooding does not create too much trafficon the network.

II. **Flooding of LSPs:** After a node has prepared an LSP, it must be disseminated to all othernodes, not only to its neighbors. The process is called flooding and based on the following

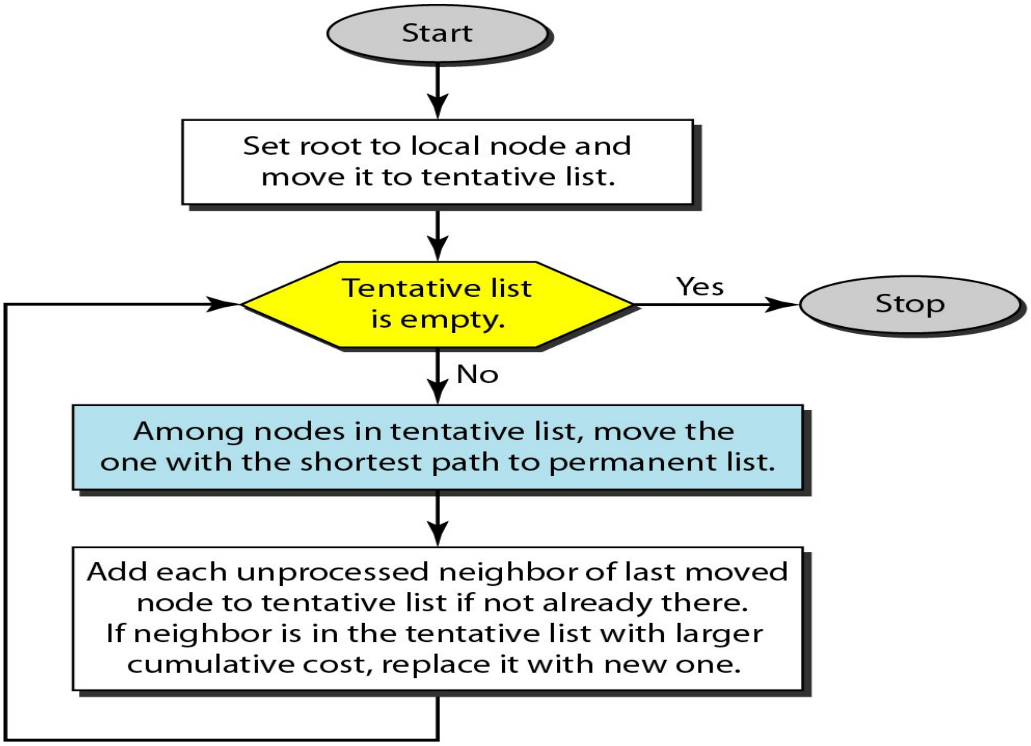
1. The creating node sends a copy of the LSP out of each interface
2. A node that receives an LSP compares it with the copy it may already have. If the newly arrived LSP is older than the one it has (found by checking the sequence number),it discards the LSP. If it is newer, the node does the following:

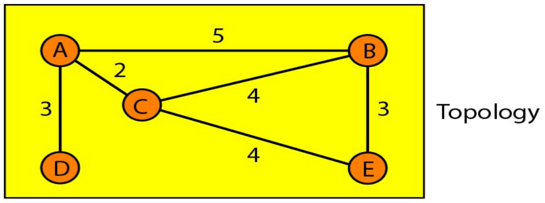
b. It discards the old LSP and keeps the new one).

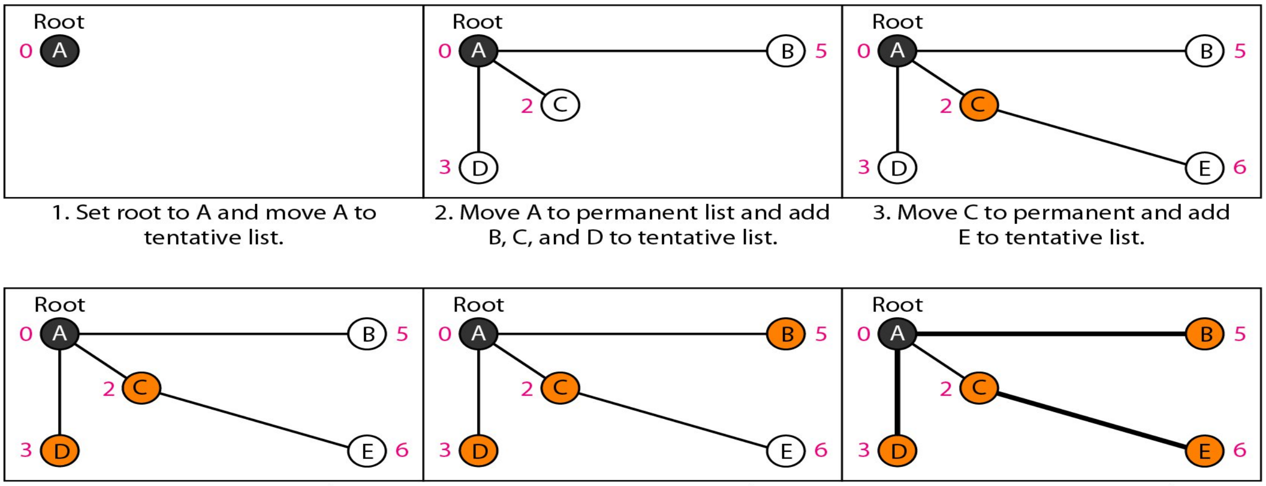
III. Formation of Shortest Path Tree: Dijkstra Algorithm

A shortest path tree is a tree in which the path between the root and every other node is the shortest.

The Dijkstra algorithm creates a shortest path tree from a graph. The algorithm divides the nodes into two sets: **tentative and permanent.** It finds the neighbors of a current node, makesthem tentative, examines them, and if they pass the criteria, makes them permanent.


Start
Set root to local node and move it to tentative list.
Tentative list is empty.
Yes
Stop
No
Among nodes in tentative list, move the one with the shortest path to permanent list.
Add each unprocessed neighbor of last moved node to tentative list if not already there.
If neighbor is in the tentative list with larger cumulative cost, replace it with new one.





1. Set root to A and move A to tentative list.

2. Move A to permanent list and add B, C, and D to tentative list.

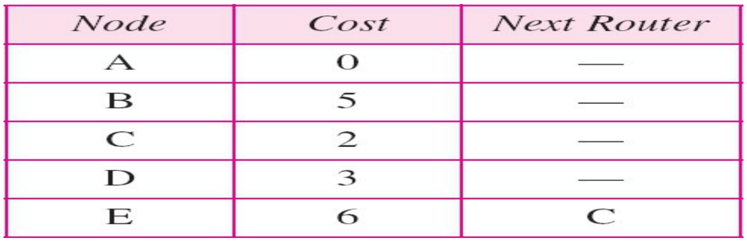
3. Move C to permanent and add E to tentative list.

4. Move D to permanent list.

5. Move B to permanent list.

6. Move E to permanent list (tentative list is empty).

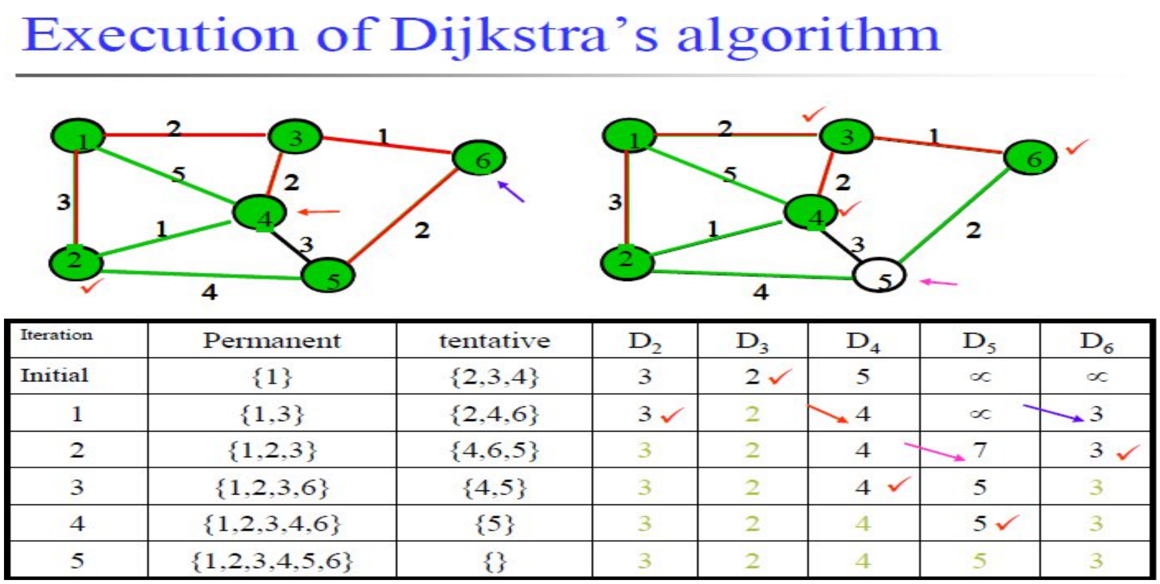
IV. Calculation of a routing table



Node	Cost	Next Router
A	0	—
B	5	—
C	2	—
D	3	—
E	6	C

Fig: Routing table for node A

Example 2:-



Execution of Dijkstra's algorithm

Iteration	Permanent	tentative	D_2	D_3	D_4	D_5	D_6
Initial	{1}	{2,3,4}	3	2 ✓	5	∞	∞
1	{1,3}	{2,4,6}	3 ✓	2	4	∞	3
2	{1,2,3}	{4,6,5}	3	2	4	7	3 ✓
3	{1,2,3,6}	{4,5}	3	2	4 ✓	5	3
4	{1,2,3,4,6}	{5}	3	2	4	5 ✓	3
5	{1,2,3,4,5,6}	{}	3	2	4	5	3

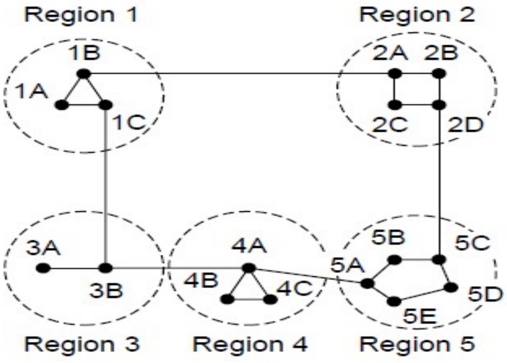
Hierarchical Routing

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 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. 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.

For huge networks, a two-level hierarchy may be insufficient; it may be necessary to group theregions into clusters, the clusters into zones, the zones into groups, and so on, until we run outof names for aggregations



(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)

When a single network becomes very large, an interesting question is ‘‘how many levels should the hierarchy have?’’

For example, consider a network with 720 routers. If there is no hierarchy, each router needs720 routing table entries. If the network 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 8 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

Internetworking:

Two or more networks are connected to form an **internetwork**, or more simply an **internet.** Internetworking is the practice of interconnecting multiple computer networks, such that any pair of hosts in the connected networks can exchange messages irrespective of their hardware-level networking technology. The resulting system of interconnected networks are called an internetwork, or simply an internet.

Tunneling

Handling the general case of making two different networks interwork is exceedingly difficult. However, there is a common special case that is manageable even for different network protocols. This case is where the source and destination hosts are on the same type of network, but there is a different network in between. As an example, think of an international bank with an IPv6 network in Paris, an IPv6 network in London and connectivity between the offices via the IPv4 Internet. This situation is shown in Fig. 5-40.

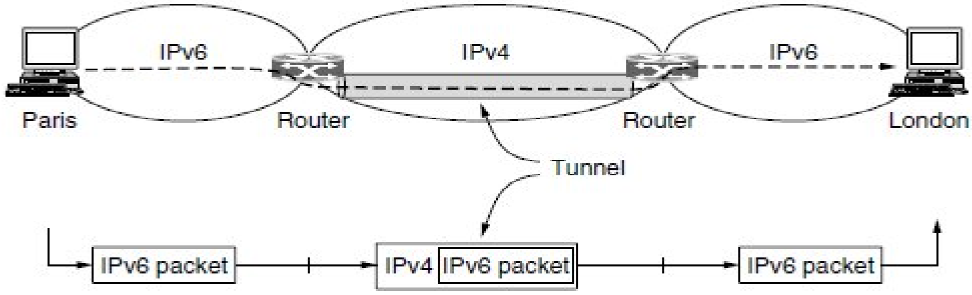


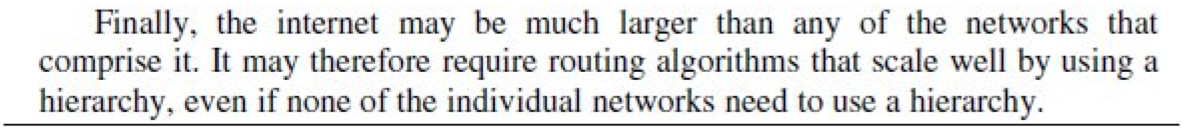
Figure 5-40. Tunneling a packet from Paris to London.

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

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

Internetwork Routing:

Routing through an internet poses the same basic problem as routing within a single network, but with some added complications. To start, the networks may internally use different routing algorithms. For example, one network may use link state routing and another distance vector routing. Since link state algorithms need to know the topology but distance vector algorithms do not, this difference alone would make it unclear how to find the shortest paths across the internet.



Finally, the internet may be much larger than any of the networks that comprise it. It may therefore require routing algorithms that scale well by using a hierarchy, even if none of the individual networks need to use a hierarchy.

Packet fragmentation:

Fragmentation is an Internet Protocol (IP) process that breaks packets into smaller pieces (fragments), so that the resulting pieces can pass through a link with a smaller maximum transmission unit (MTU) than the original packet size.ie,

Fragmentation is an important function of network layer. It is technique in which gateways break up or divide larger packets into smaller ones called fragments. Each fragment is then sent as a separate internal packet. Each fragment has its separate header and trailer.

Sometimes, a fragmented datagram also get fragmented when it encounter a network that handle smaller fragments. Thus, a datagram can be fragmented several times before it reaches final destination. Reverse process of the fragmentation is difficult. Reassembly of fragments is usually done by the destination host because each fragment has become an independent datagram.

There are two different strategies for the recombination or we can say reassembly of fragments : Transparent Fragmentation, and Non-Transparent Fragmentation.

1. **Transparent Fragmentation :**

   This fragmentation is done by one network is made transparent to all other subsequent networks through which packet will pass. Whenever a large packet arrives at a gateway, it breaks packet into smaller fragments as shown in the following figure gateway G1 breaks a packet into smaller fragments.

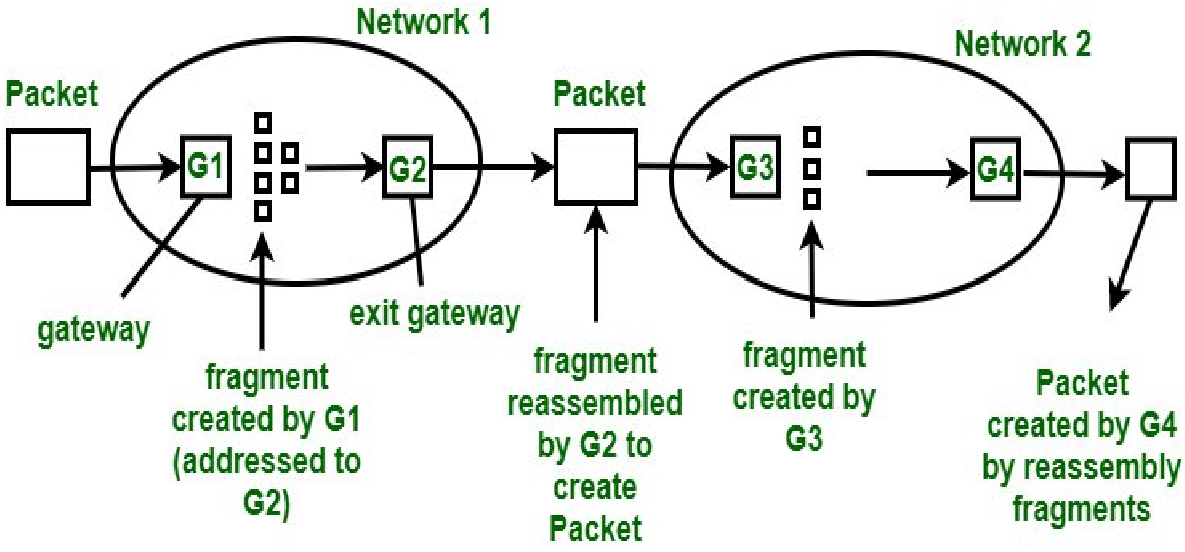

Network 1
Network 2
Packet
G1
G2
Packet
G3
G4
gateway
fragment created by G1 (addressed to G2)
exit gateway
fragment reassembled by G2 to create Packet
fragment created by G3
Packet created by G4 by reassembly fragments

Figure – Transparent Fragmentation

After this, each fragment is going to address to same exit gateway. Exist gateway of a network reassembles or recombines all fragments example is shown in the above figure as exit gateway, G2 of network 1 recombines all fragments created by G1 before passing them to network 2. Thus, subsequent network is not aware that fragmentation has occurred. This type of strategy is used by ATM networks . These networks use special hardware that provides transparent fragmentation of packets.

There are some disadvantages of transparency strategy which are as follows :

- Exit fragment that recombines fragments in a network must known when it has received all fragments.
- Some fragments chooses different gateways for exit that results in poor performance.
- It adds considerable overhead in repeatedly fragmenting and reassembling large packet.

2. **Non-Transparent Fragmentation :** This fragmentation is done by one network is non-transparent to the subsequent networks through which a packet passes. Packet fragmented by a gateway of a network is not recombined by exit gateway of same network as shown in the below figure.

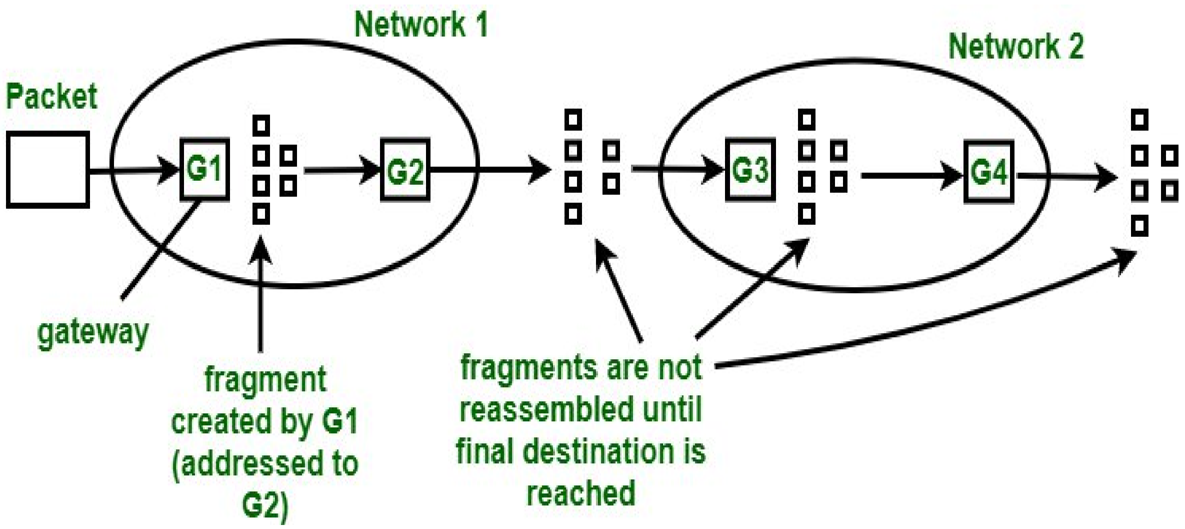

Network 1
Network 2
Packet
G1
G2
G3
G4
gateway
fragment created by G1 (addressed to G2)
fragments are not reassembled until final destination is reached

Figure – Non-transparent Fragmentation

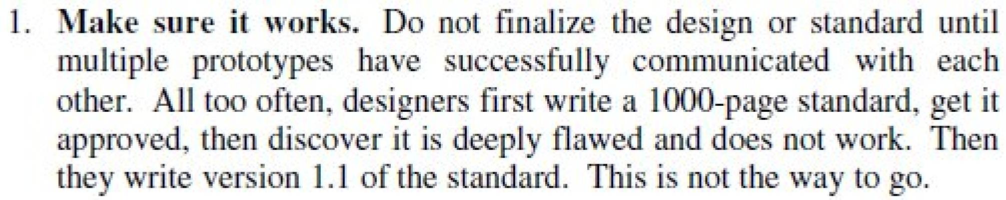
Once a packet is fragmented, each fragment is treated as original packet. All fragments of a packet are passed through exit gateway and recombination of these fragments is done at the destination host.

Disadvantages of Non-Transparent Fragmentation is as follows :

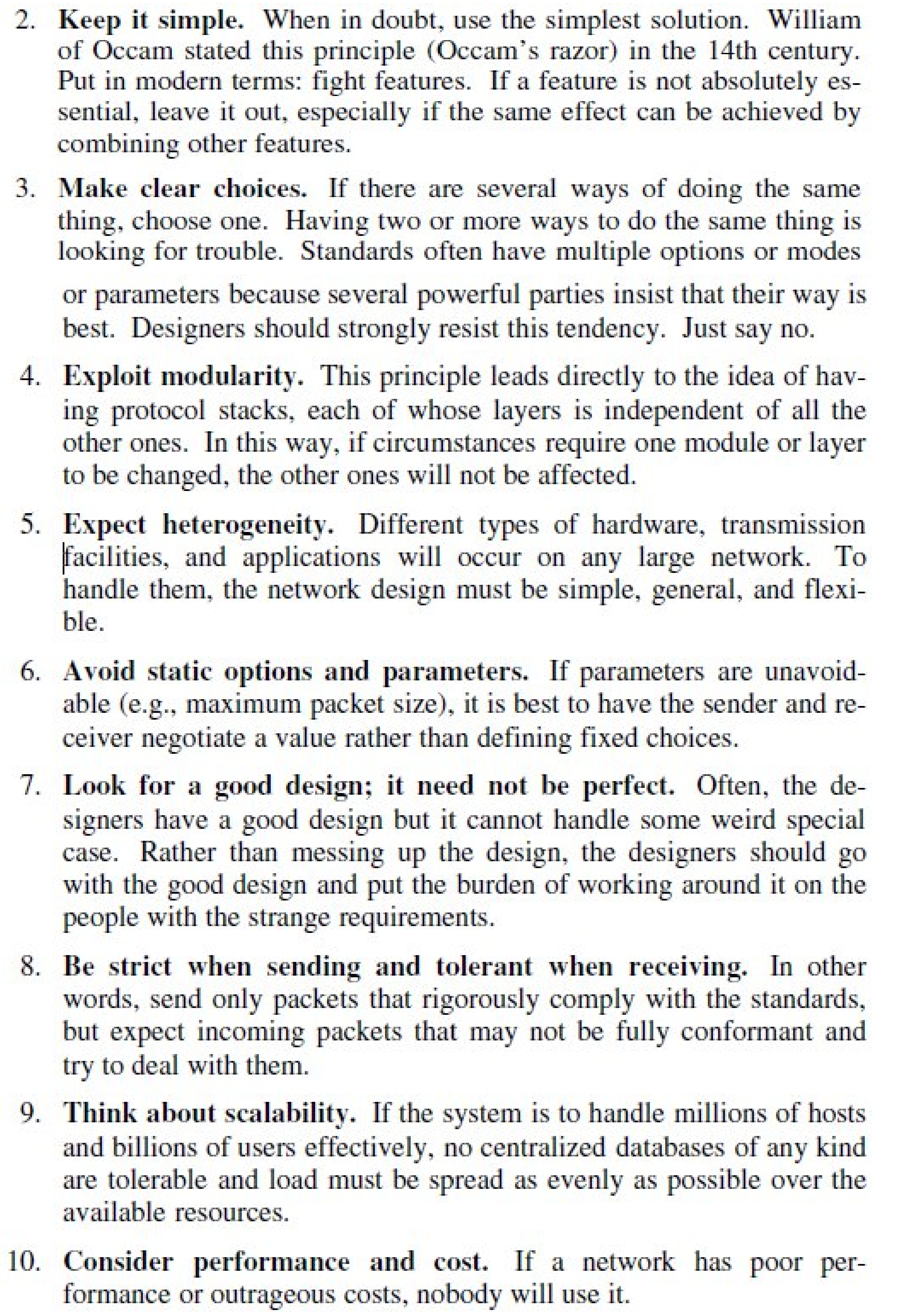
- Every host has capability of reassembling fragments.
- When a packet is fragmented, fragments should be numbered in such a way that the original data stream can be reconstructed.
- Total overhead increases due to fragmentation as each fragment must have its own header.

<u>Network layer in the Internet:</u>

<u>Top 10 principles:</u>



1. **Make sure it works.** Do not finalize the design or standard until multiple prototypes have successfully communicated with each other. All too often, designers first write a 1000-page standard, get it approved, then discover it is deeply flawed and does not work. Then they write version 1.1 of the standard. This is not the way to go.



2. **Keep it simple.** When in doubt, use the simplest solution. William of Occam stated this principle (Occam's razor) in the 14th century. Put in modern terms: fight features. If a feature is not absolutely essential, leave it out, especially if the same effect can be achieved by combining other features.
3. **Make clear choices.** If there are several ways of doing the same thing, choose one. Having two or more ways to do the same thing is looking for trouble. Standards often have multiple options or modes or parameters because several powerful parties insist that their way is best. Designers should strongly resist this tendency. Just say no.
4. **Exploit modularity.** This principle leads directly to the idea of having protocol stacks, each of whose layers is independent of all the other ones. In this way, if circumstances require one module or layer to be changed, the other ones will not be affected.
5. **Expect heterogeneity.** Different types of hardware, transmission facilities, and applications will occur on any large network. To handle them, the network design must be simple, general, and flexible.
6. **Avoid static options and parameters.** If parameters are unavoidable (e.g., maximum packet size), it is best to have the sender and receiver negotiate a value 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. Rather than messing up the design, the designers should go with the good design and put the burden of working around it on the people with the strange requirements.
8. **Be strict when sending and tolerant when receiving.** In other words, 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.

At the network layer, the Internet can be viewed as a collection of subnetworks or **Autonomous Systems** (**ASes**) that are interconnected.

There is no real structure, but several major backbones exist. These are constructed from high-bandwidth lines and fast routers. Attached to the backbones are regional (midlevel) networks, and attached to these regional networks are the LANs at many universities, companies, and Internet service providers.

A sketch of this quasi-hierarchical organization is given in Fig.

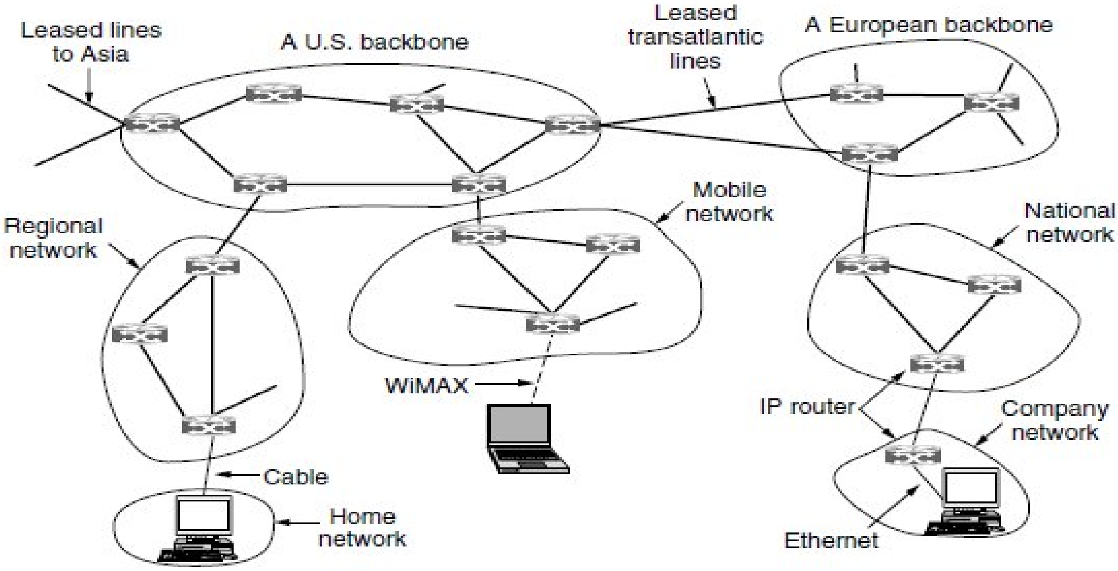
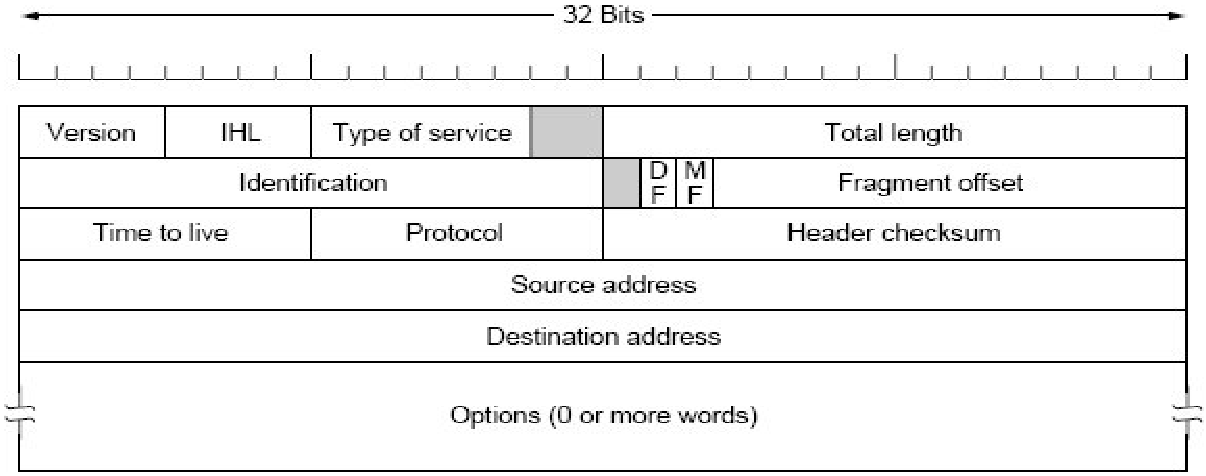


Figure 5-45. The Internet is an interconnected collection of many networks.

THE IP PROTOCOL

An IP datagram consists of a header part and a text part. The header has a 20-byte fixed part and a variable length optional part.



<table>
<tr><td colspan="4">Version</td><td colspan="4">IHL</td><td colspan="6">Type of service</td><td colspan="2"></td><td colspan="16">Total length</td></tr>
<tr><td colspan="16">Identification</td><td colspan="1"></td><td colspan="1">DF</td><td colspan="1">MF</td><td colspan="13">Fragment offset</td></tr>
<tr><td colspan="8">Time to live</td><td colspan="8">Protocol</td><td colspan="16">Header checksum</td></tr>
<tr><td colspan="32">Source address</td></tr>
<tr><td colspan="32">Destination address</td></tr>
<tr><td colspan="32">Options (0 or more words)</td></tr>
</table>

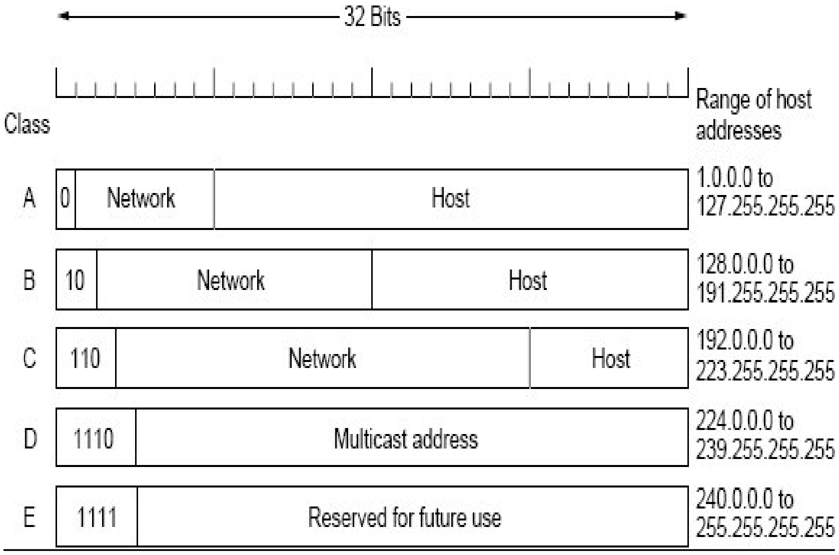
- The ***Version*** field keeps track of which version of the protocol the datagram belongs to.
- The header length is not constant, a field in the header, ***IHL***, is provided to tell how long the header is, in 32-bit words.
- The ***Type of service*** field is one of the few fields that have changed its meaning

(slightly) over the years. It was and is still intended to distinguish between different classes of service.

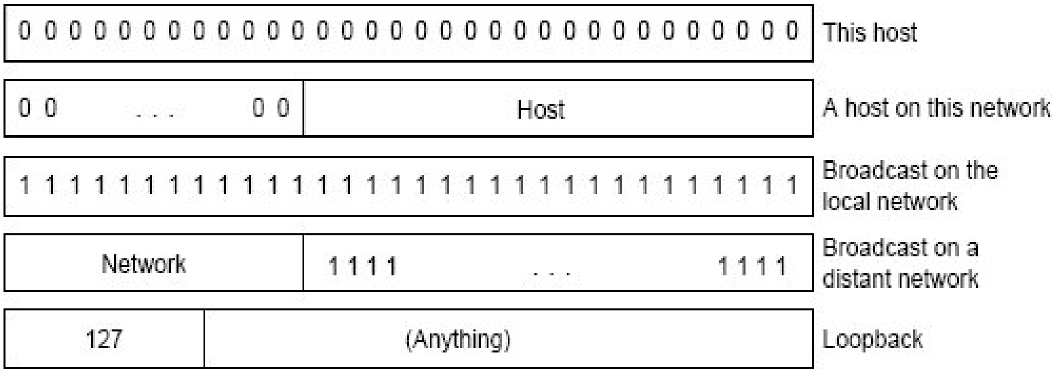
- The ***Total length*** includes everything in the datagram—both header and data. The maximum length is 65,535 bytes.
- The ***Identification*** field is needed to allow the destination host to determine which datagram a newly arrived fragment belongs to. All the fragments of a datagram contain the same *Identification* value. Next comes an unused bit and then two 1-bit fields.
- ***DF*** stands for Don't Fragment. It is an order to the routers not to fragment the datagram because the destination is incapable of putting the pieces back together again.
- ***MF*** stands for More Fragments. All fragments except the last one have this bit set. It is needed to know when all fragments of a datagram have arrived.
- The ***Fragment offset*** tells where in the current datagram this fragment belongs. All fragments except the last one in a datagram must be a multiple of 8 bytes, the elementary fragment unit. Since 13 bits are provided, there is a maximum of 8192 fragments per datagram, giving a maximum datagram length of 65,536 bytes, one more than the ***Total length* field.**
- The ***Time to live*** field is a counter used to limit packet lifetimes. It is supposed to count time in seconds, allowing a maximum lifetime of 255 sec.
- When the network layer has assembled a complete datagram, it needs to know what to do with it. The ***Protocol*** field tells it which transport process to give it to. TCP is one possibility, but so are UDP and some others. The numbering of protocols is global across the entire Internet.
- The ***Header checksum*** verifies the header only.
- The ***Source address* and *Destination address*** indicate the network number and host number.
- The ***Options*** field was designed to provide an escape to allow subsequent versions of the protocol to include information not present in the original design, to permit experimenters to try out new ideas, and to avoid allocating header bits to information that is rarely needed.
- The options are variable length.

IP ADDRESS

- Every host and router on the Internet has an IP address, which encodes its network number and host number.
- All IP addresses are 32 bits long. 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 were divided into the five categories

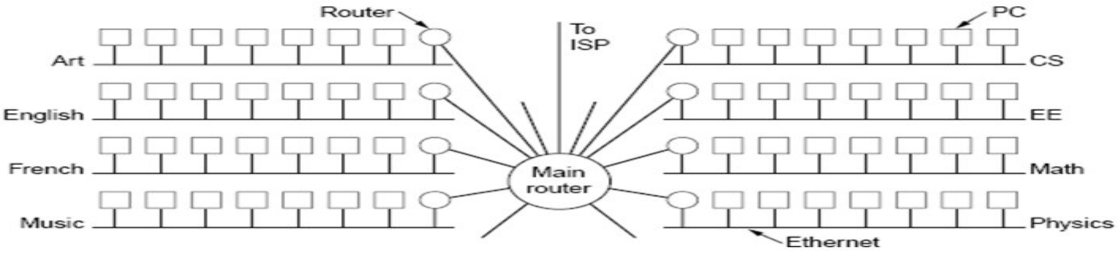

32 Bits
Class
Range of host addresses
A
0
Network
Host
1.0.0.0 to 127.255.255.255
B
10
Network
Host
128.0.0.0 to 191.255.255.255
C
110
Network
Host
192.0.0.0 to 223.255.255.255
D
1110
Multicast address
224.0.0.0 to 239.255.255.255
E
1111
Reserved for future use
240.0.0.0 to 255.255.255.255

- The values 0 and -1 (all 1s) have special meanings. The value 0 means this network or this host. The value of -1 is used as a broadcast address to mean all hosts on the indicated network.

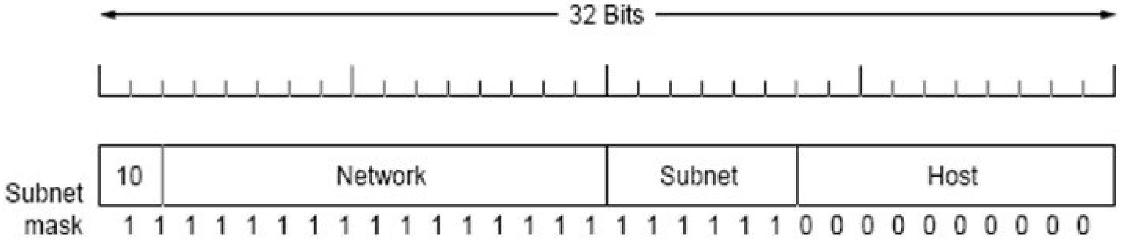


SUBNET

- All the hosts in a network must have the same network number. This property of IP addressing can cause problems as networks grow. For example
- The problem is the rule that a single class A, B, or C address refers to one network, not to a collection of LANs.
- The solution is to allow a network to be split into several parts for internal use but still act like a single network to the outside world.



- To implement subnetting, the main router needs a subnet mask that indicates the split between network + subnet number and host.
- For example, if the university has a B address(130.50.0.0) and 35 departments, it could use a 6-bit subnet number and a 10-bit host number, allowing for up to 64 Ethernets, each with a maximum of 1022 hosts.
- The subnet mask can be written as 255.255.252.0. An alternative notation is /22 to indicate that the subnet mask is 22 bits long.

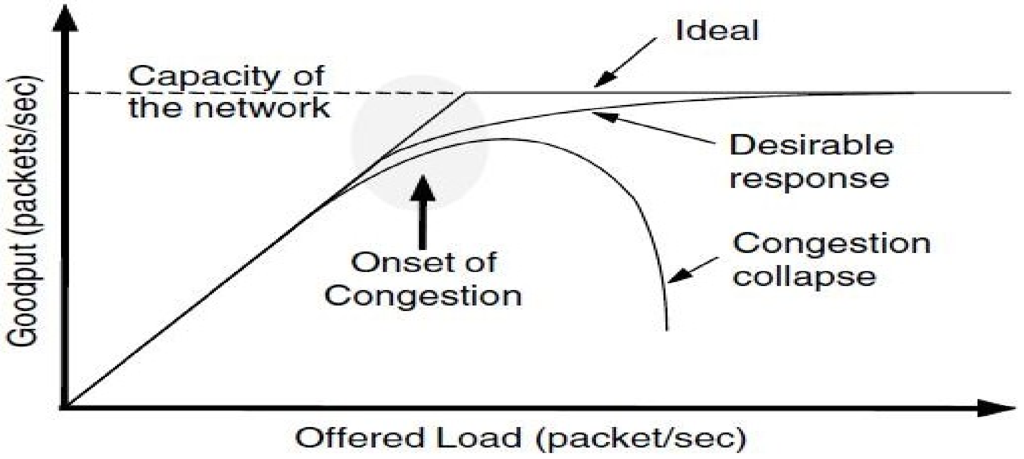


Congestion Control:

Too many packets present in (a part of) the network causes packet delay and loss that degrades performance. This situation is called congestion.

The network and transport layers share the responsibility for handling congestion. Since congestion occurs within the network, it is the network layer that directly experiences it and must ultimately determine what to do with the excess packets.

However, the most effective way to control congestion is to reduce the load that the transport layer is placing on the network. This requires the network and transport layers to work together. In this chapter we will look at the network aspects of congestion.



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

Above Figure depicts the onset of congestion. When the number of packets hosts send into the network is well within its carrying capacity, the number delivered is proportional to the number sent. If twice as many are sent, twice as many are delivered. However, as the offered load approaches the carrying capacity, bursts of traffic occasionally fill up the buffers inside routers and some packets are lost. These lost packets consume some of the capacity, so the number of delivered packets falls below the ideal curve. The network is now congested. Unlessthe network is well designed, it may experience a congestion collapse, in which performance plummets as the offered load increases beyond the capacity.This can happen because packets can be sufficiently delayed inside the network that they are no longer useful when they leave the network. the y-axis of Figure is given as goodput, which is the rate at which *useful* packets are delivered by the network.

Approaches to Congestion Control :

The presence of congestion means that the load is (temporarily) greater than the resources (in a part of the network) can handle. Two solutions come to mind: increase the resources or decrease the load. As shown in Fig. 5-22, these solutions are usually applied on different time scales to either prevent congestion or react to it once it has occurred.

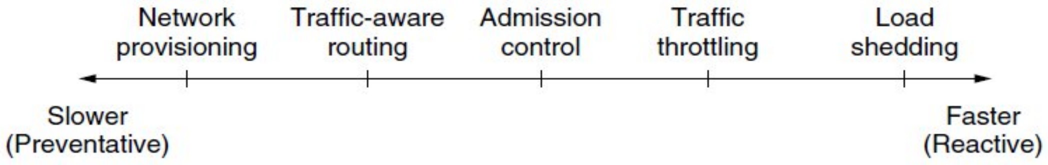


Figure 5-22. Timescales of approaches to congestion control.

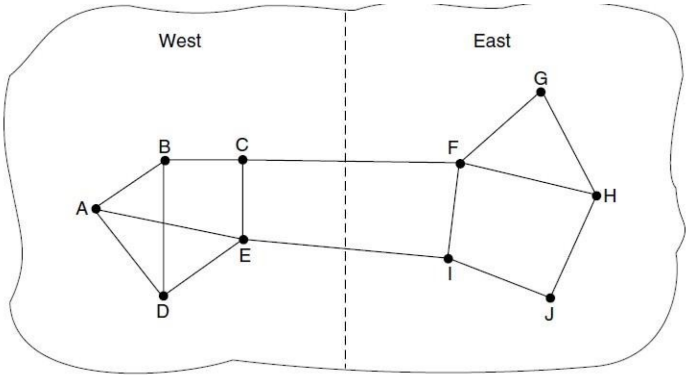
The most basic way to avoid congestion is to build a network that is well matched to the traffic that it carries. If there is a low-bandwidth link on the path along which most traffic is directed, congestion is likely. Sometimes resources can be added dynamically when there is serious congestion, for example, turning on spare outers or enabling lines that are normally used only as backups (to make the system fault tolerant).

Traffic Aware routing

These schemes adapted to changes in topology, but not to changes in load. The goal in taking load into account when computing routes is to shift traffic away from hotspots that will be the first places in the network to experience congestion.

The most direct way to do this is to set the link weight to be a function of the (fixed) link bandwidth and propagation delay plus the (variable) measured load or average queuing delay. Least-weight paths will then favor paths that are more lightly loaded, all else being equal.

Consider the network of Fig., which is divided into two parts, East and West, connected by two links, *CF* and *EI*. Suppose that most of the traffic between East and West is using link *CF*, and, as a result, this link is heavily loaded with long delays. Including queueing delay in the weight used for 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*, loading this link. 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.



If load is ignored and only bandwidth and propagation delay are considered, this problem does not occur. Attempts to include load but change weights within a narrow range only slow down routing oscillations. Two techniques can contribute to a successful solution. The first is multipath routing, in which there can be multiple paths from a source to a destination. In our example this means that the traffic can be spread across both of the East to West links. The second one is for the routing scheme to shift traffic across routes slowly enough that it is able to converge.

Traffic Throttling

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

$$u_{new} = au_{old} + (1 - a)f$$

where the constant *a* determines how fast the router forgets recent history. This is called an EWMA (Exponentially Weighted Moving Average).

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; they are warning bits, choke packets, Hop-by-Hop choke packet. These are used for congestion detection and recovery.

Warning Bit

1. A special bit in the packet header is set by the router to warn the source when congestionis detected.
2. The bit is copied and piggy-backed on the ACK and sent to the sender.
3. The sender monitors the number of ACK packets it receives with the warning bit set andadjusts its transmission rate accordingly.

Choke Packets

1. A more direct way of telling the source to slow down.
2. A choke packet is a control packet generated at a congested node and transmitted torestrict traffic flow.
3. The source, on receiving the choke packet must reduce its transmission rate by a certainpercentage.

Hop-by-Hop Choke Packets

1. Over long distances or at high speeds choke packets are not very effective.
2. A more efficient method is to send to choke packets hop-by-hop.
3. This requires each hop to reduce its transmission even before the choke packet arrive atthe source.

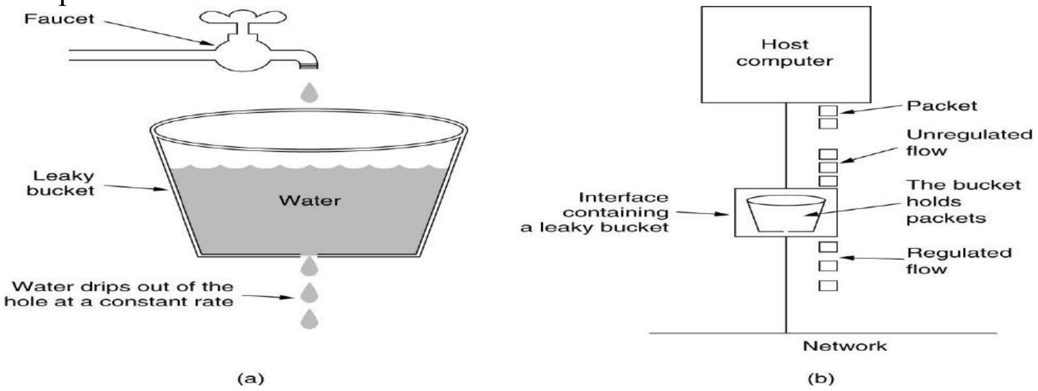
Load Shedding

1. When buffers become full, routers simply discard packets.
2. Which packet is chosen to be the victim depends on the application and on the errorstrategy used in the data link layer.
3. For a file transfer, for, e.g. cannot discard older packets since this will cause a gap in thereceived data.
4. For real-time voice or video it is probably better to throw away old data and keep new packets.Get the application to mark packets with discard priority.

Traffic Shaping

1. Another method of congestion control is to “shape” the traffic before it enters the network.
2. Traffic shaping controls the *rate* at which packets are sent (not just how many). Used inATM and Integrated Services networks.
3. At connection set-up time, the sender and carrier negotiate a traffic pattern (shape).

Traffic shaping algorithms are classified into 2 types : Leaky Bucket , Token Bucket.
Leaky Bucket Algorithm : It used to control rate in a network. It is implemented as a single- server queue with constant service time. If the bucket (buffer) overflows then packets are discarded.

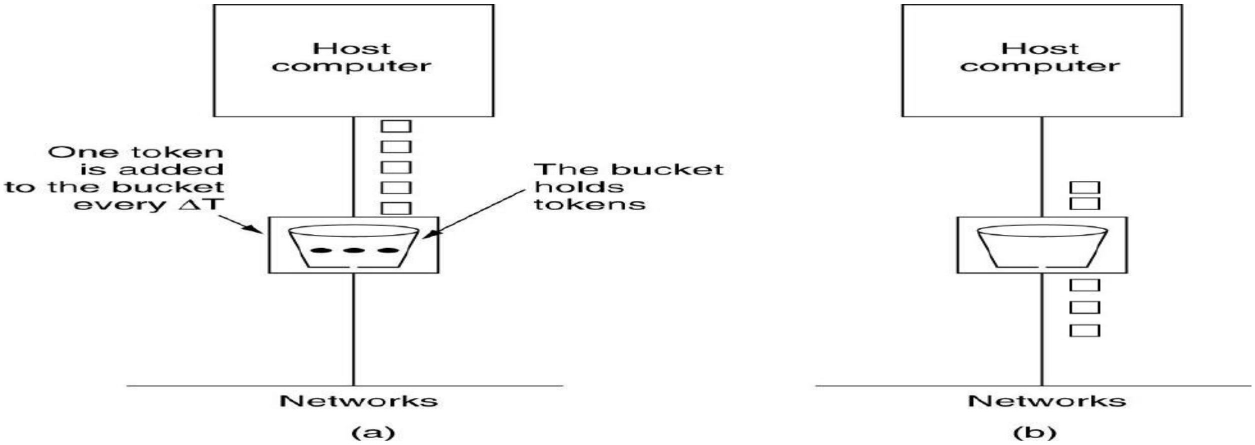


(a) A leaky bucket with water. (b) A leaky bucket with packets.

1. The leaky bucket enforces a constant output rate (average rate) regardless of the burstiness of the input. Does nothing when input is idle.
2. The host injects one packet per clock tick onto the network. This results in a uniform flowof packets, smoothing out bursts and reducing congestion.
3. When packets are the same size (as in ATM cells), the one packet per tick is okay. For variable length packets though, it is better to allow a fixed number of bytes per tick. E.g. 1024 bytes per tick will allow one 1024-byte packet or two 512-byte packets or four 256- byte packets on 1 tick

Token Bucket Algorithm

1. In contrast to the LB, the Token Bucket Algorithm, allows the output rate to vary,depending on the size of the burst.
2. In the TB algorithm, the bucket holds tokens. To transmit a packet, the host must captureand destroy one token.
3. Tokens are generated by a clock at the rate of one token every Δt sec.
4. Idle hosts can capture and save up tokens (up to the max. size of the bucket) in order tosend larger bursts later.



(a) Before. (b) After.

Leaky Bucket vs. Token Bucket

1. LB discards packets; TB does not. TB discards tokens.
2. With TB, a packet can only be transmitted if there are enough tokens to cover its length inbytes.
3. LB sends packets at an average rate. TB allows for large bursts to be sent faster by speedingup the output.
4. TB allows saving up tokens (permissions) to send large bursts. LB does not allow saving.