Module 3

Network layer design issues. Routing algorithms - The Optimality Principle, Shortest path routing, Flooding, Distance Vector Routing, Link State Routing, Multicast routing, Routing for mobile hosts. Congestion control algorithms.Quality of Service (QoS)- requirements, Techniques for achieving good QoS.

Network Layer

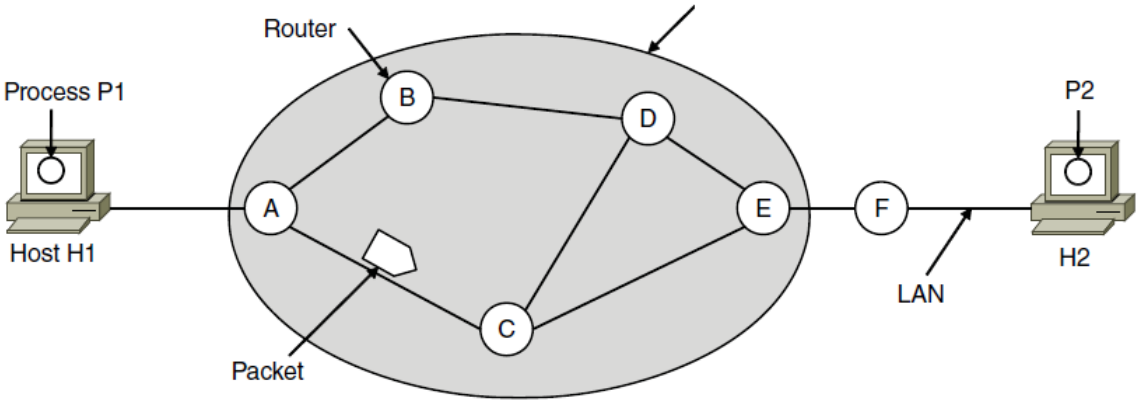
The network layer is concerned with getting packets from the source all the way to the destination. Getting to the destination may require making many hops at intermediate routers along the way. This function clearly contrasts with that of the data link layer, which has the more modest goal of just moving 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 communication subnet (i.e., the set of all routers) and choose appropriate paths through it. It must also take care to choose 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

- The major components of the network are the **ISP**'s equipment (routers connected by transmission lines), shown inside the shaded oval, and the customers' equipment, shown outside the oval.
- Host H1 is directly connected to one of the ISP's routers, A, perhaps as a home computer that is plugged into a DSL modem.
- In contrast, H2 is on a LAN, which might be an office Ethernet, with a router, F, owned and operated by the customer. This router has a leased line to the ISP's equipment.
- It is shown F as being outside the oval because it does not belong to the ISP.



This equipment is used as follows:

- 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. An important question is precisely what kind of services the network layer provides to the transport layer. The services need to be carefully designed with the following goals in mind:
  1. The services should be independent of the router technology.
  2. The transport layer should be shielded from the number, type, and topology of the routers present.
  3. The network addresses made available to the transport layer should use a uniform numbering plan, even across LANs and WANs.

The two classes of service the network layer can provide to its users:

- The network service should be connectionless, with primitives SEND PACKET and RECEIVE PACKET. No packet ordering and flow control should be done, because the hosts are going to do that. Each packet must carry the full destination address, because each packet sent is carried independently of its predecessors.

- The subnet should provide a reliable, connection-oriented service. Quality of service is the dominant factor, and without connections in the subnet, quality of service is very difficult to achieve, especially for real-time traffic such as voice and video.

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** and the network is called a **datagram network.**

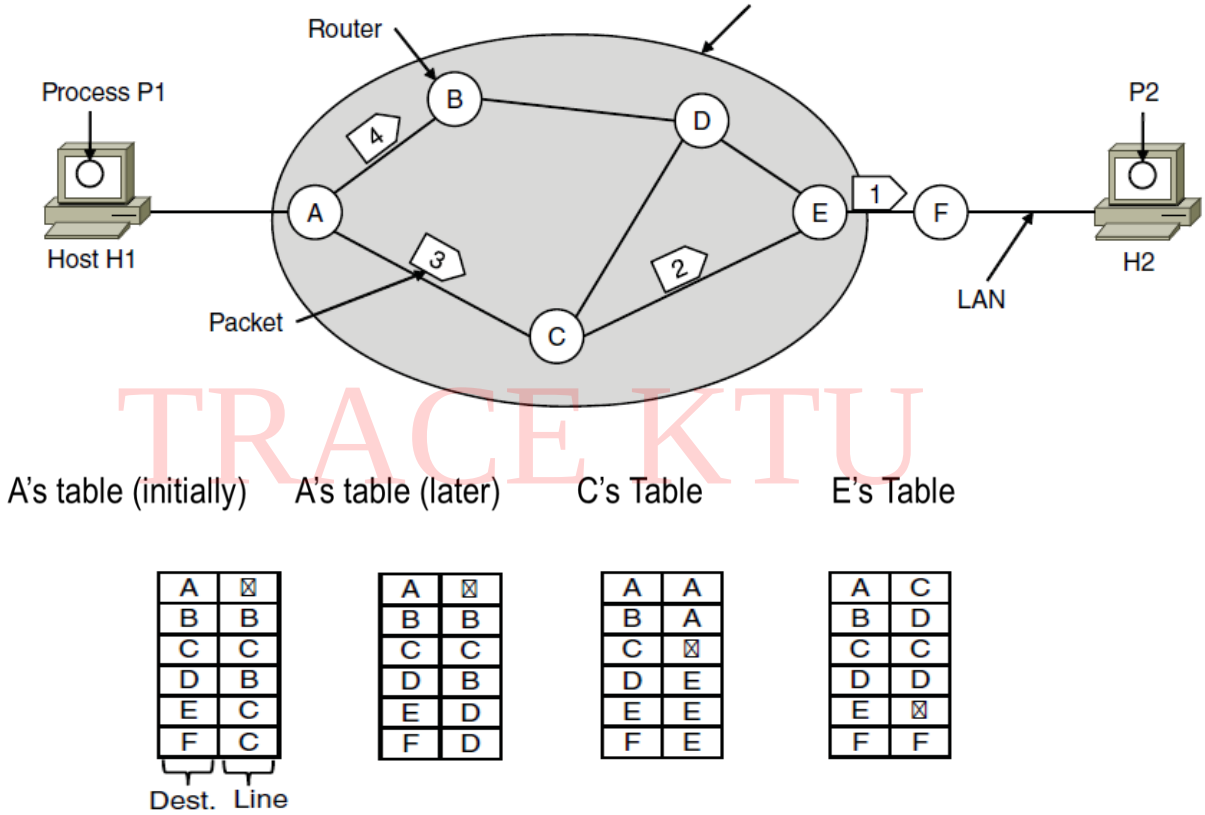


Fig.Routing within a datagram network

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

4.Implementation of connection oriented service

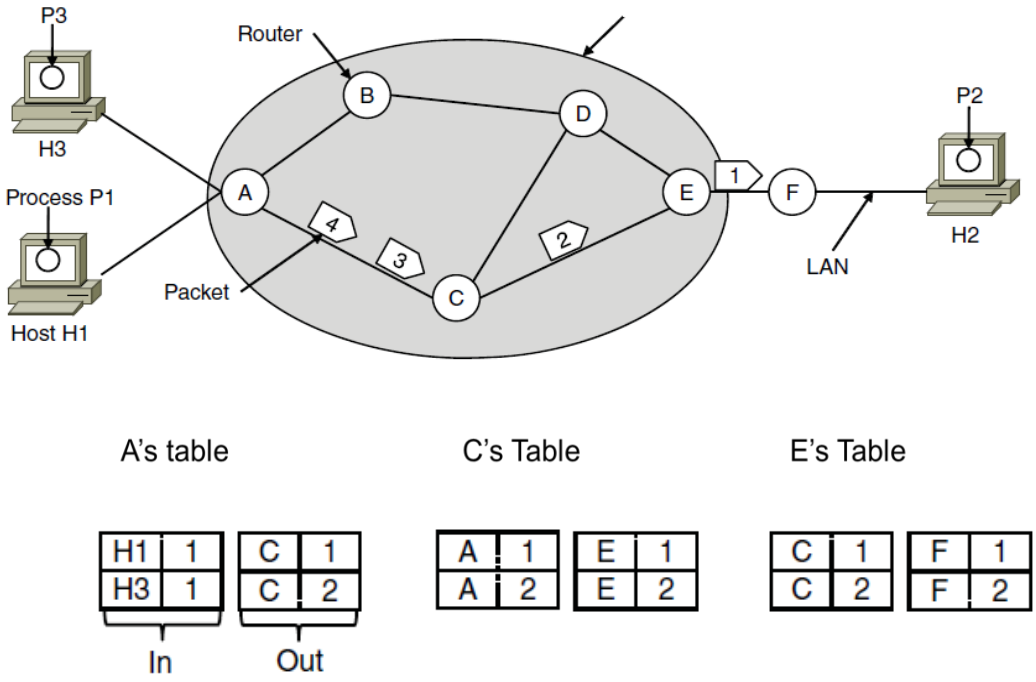


Fig.Routing within a virtual-circuit network

5.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 routing algorithm is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on. If the network uses datagrams internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time. If the network uses virtual circuits internally, routing decisions are made only when a new virtual circuit is being set up. In order to achieve its goals, the network layer must know about the topology of the communication network. It must also take care to choose routes to avoid overloading of some of the communication lines while leaving others idle. The main functions performed by the network layer are as follows:

- Routing
- Congestion Control
- Internetworking

Routing

Routing is the process of forwarding of a packet in a network so that it reaches its intended destination. The main goals of routing are:

1. **Correctness:** The routing should be done properly and correctly so that the packets may reach their proper destination.
2. **Simplicity:** The routing should be done in a simple manner so that the overhead is as low as possible. With increasing complexity of the routing algorithms the overhead also increases.
3. **Robustness:** Once a major network becomes operative, it may be expected to run continuously for years without any failures. The algorithms designed for routing should be robust enough to handle hardware and software failures and should be able to cope with changes in the topology and traffic without requiring all jobs in all hosts to be aborted and the network rebooted every time some router goes down.
4. **Stability:** The routing algorithms should be stable under all possible circumstances.
5. **Fairness:** Every node connected to the network should get a fair chance of transmitting their packets. This is generally done on a first come first serve basis.
6. **Optimality:** The routing algorithms should be optimal in terms of throughput and minimizing mean packet delays. Here there is a trade-off and one has to choose depending on his suitability.

Classification of Routing Algorithms

The routing algorithms can be classified as follows:

1. **Adaptive Routing Algorithm:** These algorithms change their routing decisions to reflect changes in the topology and in traffic as well. These get their routing information from adjacent routers or from all routers. The optimization parameters are the distance, number of hops and estimated transit time.
2. **Non-Adaptive Routing Algorithm:** These algorithms do not base their routing decisions on measurements and estimates of the current traffic and topology. Instead the route to be taken in going from one node to the other is computed in advance, off-line, and

downloaded to the routers when the network is booted. This is also known as static routing.

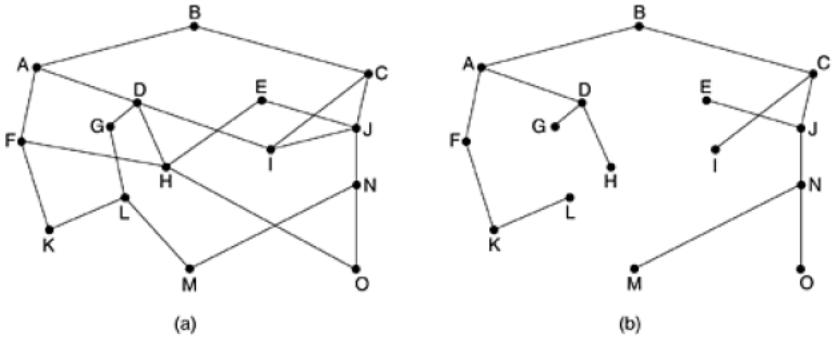
Different Routing Algorithms

- **The Optimality Principle**
- **Shortest Path Routing**
- **Flooding**
- **Distance Vector Routing**
- **Link State Routing**
- Hierarchical Routing
- Broadcast Routing
- **Multicast Routing**
- **Routing for Mobile Hosts**
- Routing in Ad Hoc Networks

The optimality Principle

Before getting into specific algorithms, one can make a general statement about optimal routes without regard to network topology or traffic. It states that if router *J* is on the optimal path from router *I* to router *K*, then the optimal path from *J* to *K* also falls along the same route. To see this,call the part of the route from *I* to *J* *r*1 and the rest of the route *r*2. If a route better than *r*2existed from *J* to *K*, it could be concatenated with *r*1 to improve the route from *I* to *K*,contradicting our statement that *r*1*r*2 is optimal.

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** and is illustrated in following Fig. , where the distance metric is the number of hops. Note that a sink tree is not necessarily unique; other trees with the same path lengths may exist. The goal of all routing algorithms is to discover and use the sink trees for all routers.



a) subnet **b) a sink tree for router B**

Shortest path Routing Algorithms

As mentioned above, the shortest paths are calculated using suitable algorithms on the graph representations of the networks. Let the network be represented by graph G (V, E) and let the number of nodes be 'N'. For all the algorithms discussed below, the costs associated with the links are assumed to be positive. A node has zero cost w.r.t itself. Further, all the links are assumed to be symmetric, i.e. if $d_{i,j}$ = cost of link from node i to node j, then $d_{i,j} = d_{j,i}$. The graph is assumed to be complete. If there exists no edge between two nodes, then a link of infinite cost is assumed. The algorithms given below find costs of the paths from all nodes to a particular node; the problem is equivalent to finding the cost of paths from a source to all destinations.

Dijkstra's Algorithm

Notation:

D_i = Length of shortest path from node 'i' to node 1.

$d_{i,j}$ = Length of path between nodes i and j .

Algorithm

Each node j is labeled with Dj, which is an estimate of cost of path from node j to node 1. Initially, let the estimates be infinity, indicating that nothing is known about the paths. We now iterate on the length of paths, each time revising our estimate to lower values, as we obtain them. Actually, we divide the nodes into two groups ; the first one, called set P contains the nodes whose shortest distances have been found, and the other Q containing all the remaining nodes. Initially P contains only the node 1. At each step, we select the node that has minimum cost path to node 1. This node is transferred to set P. At the first step, this corresponds to shifting the node closest to 1 in P. Its minimum cost to node 1 is now known. At the next step, select the next closest node from set Q and update the labels corresponding to each node using :

$$\mathbf{D_j = min [D_j , D_i + d_{j,i}]}$$

Finally, after N-1 iterations, the shortest paths for all nodes are known, and the algorithm terminates.

Principle

Let the closest node to 1 at some step be i. Then i is shifted to P. Now, for each node j , the closest path to 1 either passes through i or it doesn't. In the first case Dj remains the same. In the second case, the revised estimate of D_j is the sum $D_i + d_{i,j}$. So we take the minimum of these two cases and update D_j accordingly. As each of the nodes get transferred to set P, the estimates get closer to the lowest possible value. When a node is transferred, its shortest path length is known. So finally all the nodes are in P and the D_j's represent the minimum costs. The algorithm is guaranteed to terminate in N-1 iterations and its complexity is $O(N^2)$.

At the end each node will be labeled (see *Figure.1*) with its distance from source node along the best known path. Initially, no paths are known, so all nodes are labeled with infinity. As the

algorithm proceeds and paths are found, the labels may change reflecting better paths. Initially, all labels are tentative. When it is discovered that a label represents the shortest possible path from the source to that node, it is made permanent and never changed thereafter.

Look at the weighted undirected graph of *Figure.1(a)*, where the weights represent, for example, distance. We want to find shortest path from A to D. We start by making node A as permanent, indicated by a filled in circle. Then we examine each of the nodes adjacent to A (the working node), relabeling each one with the distance to A. Whenever a node is relabeled, we also label it with the node from which the probe was made so that we can construct the final path later. Having examined each of the nodes adjacent to A, we examine all the tentatively labeled nodes in the whole graph and make the one with the smallest label permanent, as shown in *Figure.1(b)*. This one becomes 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 the node, we have a shorter path, so the node is relabeled. 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 labeled node with the smallest value. This node is made permanent and becomes the working node for the next round. The *Figure. 1* shows the first five steps of the algorithm

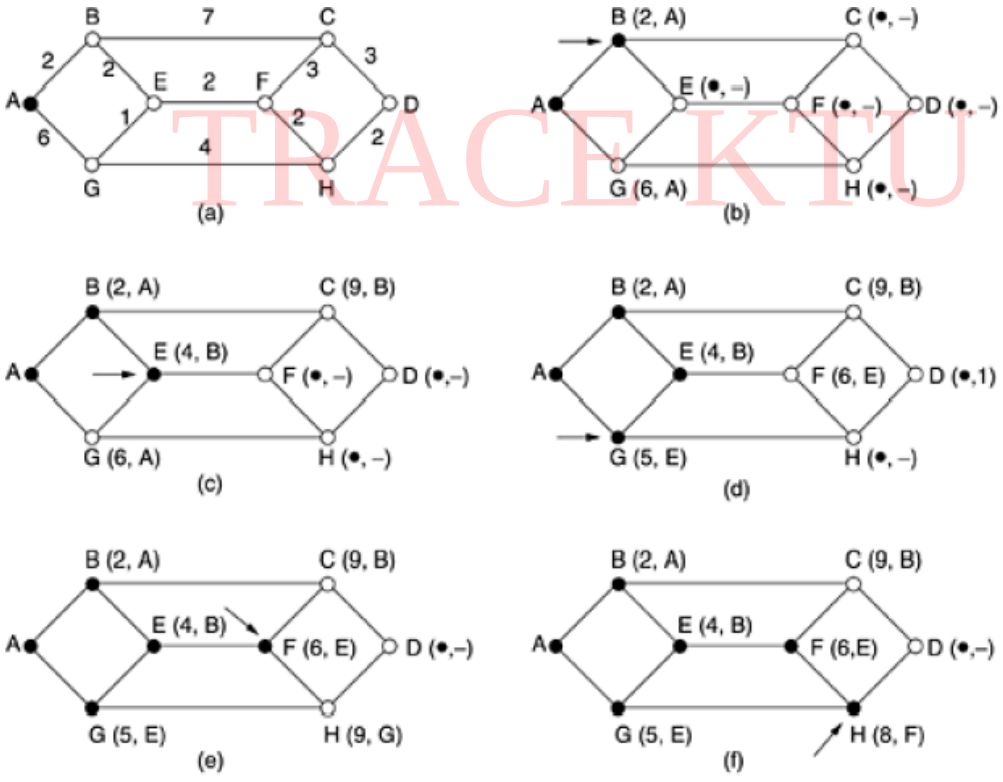
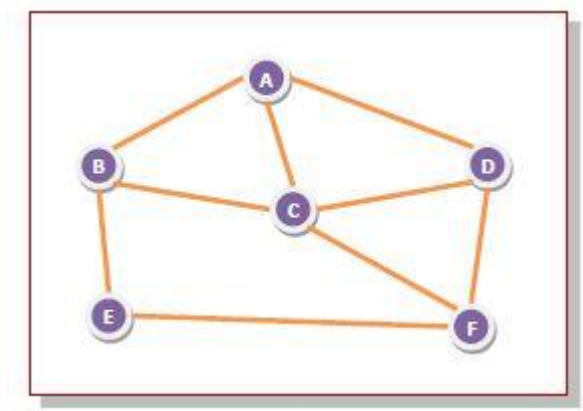


Figure 1.The first five steps used in computing the shortest path from* A *to* D. *The arrows indicate the working node.

Note: Dijkstra's Algorithm is applicable only when cost of all the nodes is non-negative.

Flooding

- Another static algorithm is **flooding**, in which every incoming packet is sent out on every outgoing line except the one it arrived on.
- For example, let us consider the network in the figure, having six routers that are connected through transmission lines.



Using flooding technique −

- An incoming packet to A, will be sent to B, C and D.
- B will send the packet to C and E.
- C will send the packet to B, D and F.
- D will send the packet to C and F.
- E will send the packet to F.
- F will send the packet to C and E.

- Flooding obviously generates vast numbers of duplicate packets. One such measure to control the duplicate packets 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. If the sender does not know how long the path is, it can initialize the counter to the worst case, namely, the full diameter of the subnet.
- Another method is to have the source router put **a sequence number** in each packet it receives from its hosts. Each router then needs a list per source router telling which sequence numbers originating at that source have already been seen. If an incoming packet is on the list, it is not flooded.
- To prevent the list from growing without bound, each list should be augmented by a counter, *k*, meaning that all sequence numbers must be less than *k* . When a packet comes in, it is easy to check if the packet is a duplicate; if so, it is discarded.
- 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
- **Uncontrolled flooding** − Here, each router unconditionally transmits the incoming data packets to all its neighbours.

- **Controlled flooding** − They use some methods to control the transmission of packets to the neighbouring nodes. The two popular algorithms for controlled flooding are Sequence Number Controlled Flooding (SNCF) and Reverse Path Forwarding (RPF).

APPLICATIONS

- Flooding is not practical in most applications, but it does have some uses. For example, in military applications, the tremendous robustness of flooding is highly desirable.
- In distributed database applications, it is sometimes necessary to update all the databases concurrently, in which case flooding can be useful.
- In wireless networks, all messages transmitted by a station can be received by all other stations within its radio range, flooding is useful.
- A fourth possible use of flooding is as a metric against which other routing algorithms can be compared. Flooding always chooses the shortest path because it chooses every possible path in parallel.

ADVANTAGES

- Highly Robust, emergency or immediate messages can be sent (eg military applications)
- Set up the route in virtual circuit
- Flooding always chooses the shortest path
- Broadcast messages to all the nodes

DISADVANTAGES:

- Flooding can be costly in terms of wasted bandwidth
- **Messages** can become duplicated in the network.

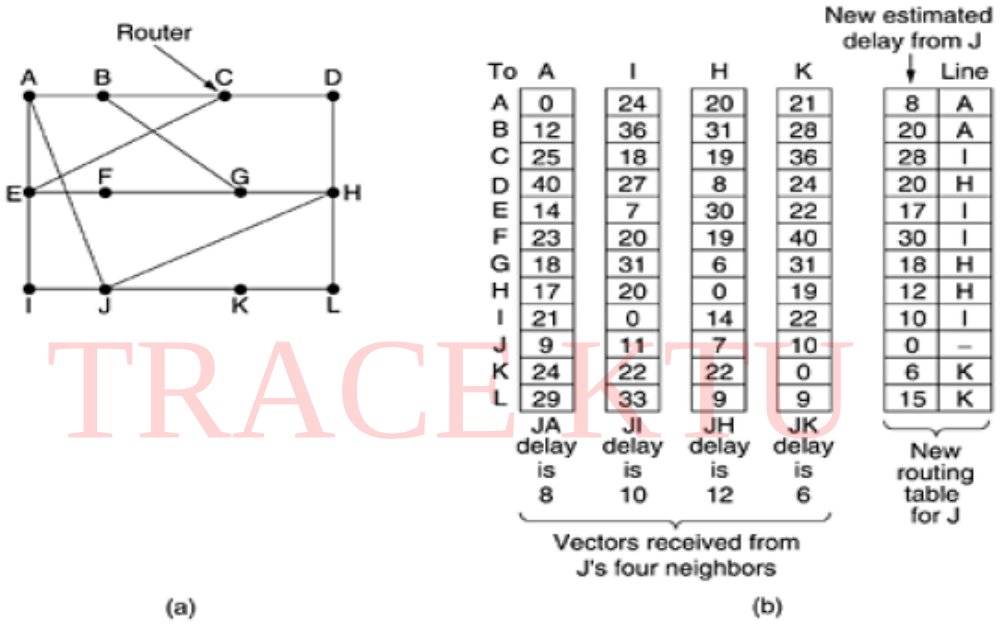
Distance vector routing

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 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.
- In distance vector routing, each router maintains a routing table indexed by, and containing one entry for, each router in the subnet.
- This entry contains two parts: the preferred outgoing line to use for that destination and an estimate of the time or distance to that destination.
- The metric used might be number of hops, time delay in milliseconds, total number of packets queued along the path, or something similar.
- The router is assumed to know the "distance" to each of its neighbors. If the metric is hops, the distance is just one hop. If the metric is queue length, the router simply examines each queue. If the metric is delay, the router can measure it directly with special ECHO packets that the receiver just timestamps and sends back as fast as it can.

- Imagine that one of these tables has just come in from neighbor X, with Xi being X's estimate of how long it takes to get to router i. If the router knows that the delay to X is m msec, it also knows that it can reach router i via X in Xi + m msec. By performing this calculation for each neighbor, a router can find out which estimate seems the best and use that estimate and the corresponding line in its new routing table. Note that the old routing table is not used in the calculation.
- This updating process is illustrated in Fig.below Part (a) shows a **subnet.** The first four columns of part (b) show the delay vectors received from the neighbors of router J. A claims to have a 12-msec delay to B, a 25-msec delay to C, a 40-msec delay to D, etc. Suppose that J has measured or estimated its delay to its neighbors, A, I, H, and K as 8, 10, 12, and 6 msec, respectively.

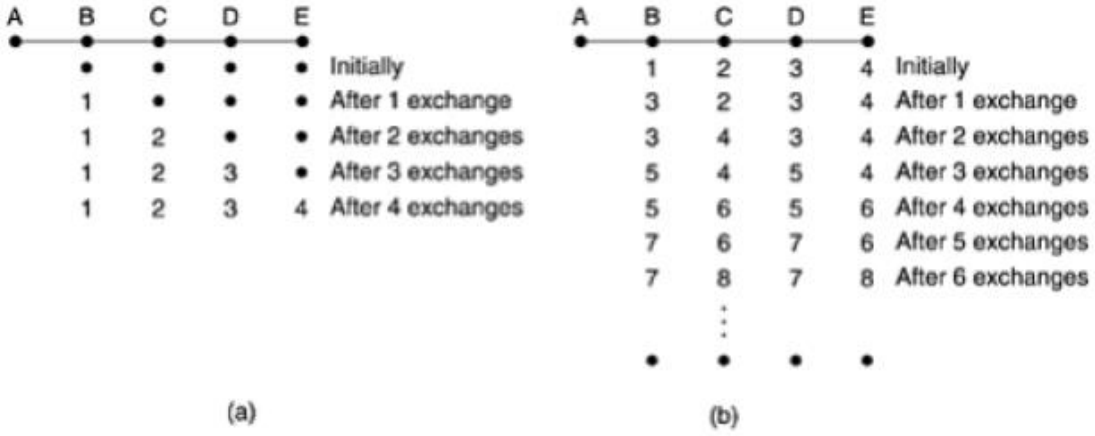
Fig(a) Subnet **Fig(b) Input from A,I,H,K and the new routing table for J**



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.

Drawback : The Count-to-Infinity Problem:

It converges to the correct answer, it may do so slowly. Consider the five-node (linear) subnet of Fig. below, 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.



- When A comes up, the other routers learn about it via the vector exchanges. At the time of the first exchange, B learns that its left neighbor has zero delay to A. B now makes an entry in its routing table that A is one hop away to the left. All the other routers still think that A is down. At this point, the routing table entries for A are as shown in the second row of Fig. above (a). On the next exchange, C learns that B has a path of length 1 to A, so it updates its routing table to indicate a path of length 2, but D and E do not hear the good news until later. Clearly, the good news is spreading at the rate of one hop per exchange. In a subnet whose longest path is of length N hops, within N exchanges everyone will know about newly-revived lines and routers.
- Fig. above (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. (b). Subsequent exchanges produce the history shown in the rest of Fig. (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 pathwith a long delay from being treated as down. This problem is known as the **count-to-infinity problem**. The core of the problem is that when X tells Y that it has a path somewhere, Y has no way of knowing whether it itself is on the path

- The delay metric was queue length, it did not take line bandwidth into account when choosing routes. The second problem the algorithm often took too long to converge (the count-to-infinity problem)

Advantages of Distance Vector routing

• It is simpler to configure and maintain than link state routing.

Drawbacks

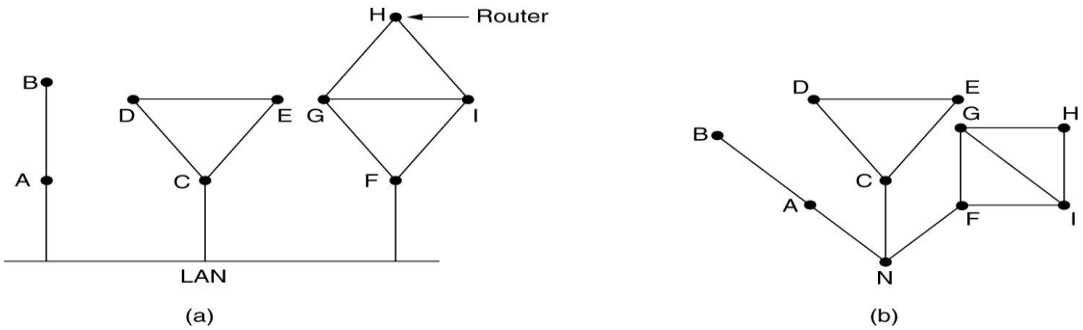
- React rapidly to good news when a router comes up.
- Though it finally converges to correct result,it takes long time when where is a bad news.
- There are several attempts to solve the problem, but none is perfect.
- It does not take line bandwidth into account.
- It took too long to converge.
- It is slower to converge than link state.
- It is at **risk** from the count-to-infinity problem.
- It creates more traffic . Hop count updates take place on a periodic basis, even if there are no changes in the network topology, so bandwidth-wasting broadcasts still occur.
- For **larger networks**, distance vector routing results in larger routing tables since each router must know about all other routers. This can lead to congestion on WAN links.

Link State Routing

- Two primary problems caused the demise of distance vector routing.
- First, since the delay metric was queue length, it did not take line bandwidth into account when choosing routes.
- The second problem was the count-to-infinity problem.
- Idea behind **LSR** is simple and can be stated as 5 parts. Each router must do the following
    - Discover its neighbors & learn their network addresses.
    - Measure the delay or cost to each of its neighbors.
    - Construct a packet telling all it has just learned.
    - Send this packet to all other routers.
    - Compute the shortest path to every other router.

1. Learning about the Neighbors

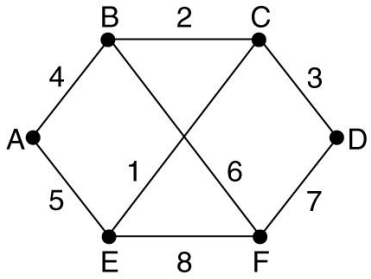
- When a router is booted, its first task is to learn who its neighbors are.
- It accomplishes this goal by sending a special **HELLO** packet on each point-to-point line.
- The router on the other end is expected to send back a reply telling who it is.
- These names must be globally unique
- When two or more routers are connected by a LAN, the situation is slightly more complicated
- One way to model the LAN is to consider it as a node itself



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

2. ***Measuring Line Cost***
   - ❖ Each router must know 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.
   - ❖ Issue is whether to take the load into account when measuring the delay.
     - ❖ To factor the load in, the round-trip timer must be started when the ECHO packet is queued.
     - ❖ To ignore the load, the timer should be started when the ECHO packet reaches the front of the queue.

3. ***Building Link State Packets (LSP)***
   - ❖ 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.



(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)

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

- ❖ 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.

4. ***Distributing the Link State Packets***

- Link state packets are to be distributed 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,
- This can lead to inconsistencies, loops, unreachable machines, and other problems.
- Basic distribution algorithm
  - 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
  - **Problem 1:** if the sequence numbers wrap around, confusion will happen.
  - solution 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.
- **Problem 2:** 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
- **Problem 3:** if a sequence number is ever corrupted and 65,540 (1000 0000 0000 0100) is received instead of 4 (0000 0000 0000 0100) (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.
- **Solution to all these problems** is to include the age of each packet after the sequence number and decrement it once per second.
- When the age hits zero, the information from that router is discarded.

5. ***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 represented twice, once for each direction.
- The two values can be averaged or used separately.

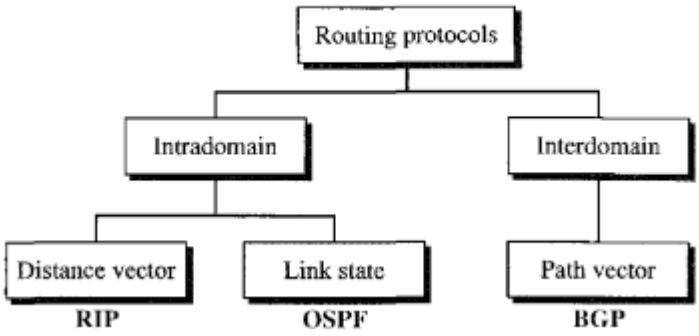
- ❖ Dijkstra's algorithm can be run locally to construct the shortest path to all possible destinations.
- ❖ The results of this algorithm can be installed in the routing tables, and normal operation resumed.

- For a subnet with n routers, each of which has k neighbors, the memory required to store the input data is proportional to kn.
- For large subnets, this can be a problem.
- Also, the computation time can be an issue.
- but, in many practical situations, link state routing works well.

Difference between link state routing &distance vector routing

Distance vector routing	Link state routing
Used in 1980 'S	Used in 1990"s
Band width is less	Band width is high
Traffic is les	Traffic is high
Count to infinity problem exist	Count to infinity problem not exists
Persistent loop	Transient loop
Protocol used is RIP	Protocol used is OSPF
Convergence is slow	Convergence is fast

Routing Information Protocol



Routing Information Protocol (RIP) is an implementation of the distance vector protocol. Open Shortest Path First (OSPF) is an implementation of the link state protocol. Border Gateway Protocol (BGP) is an implementation of the path vector protocol.

-The Routing Information Protocol (RIP) is an intradomain routing protocol used inside an autonomous system.

-Routing Information Protocol (RIP) is a dynamic routing protocol that uses hop count as a routing metric to find the best path between the source and the destination network. It is a distance-vector routing protocol that has an AD value of 120 and works on the Network layer of the OSI model. RIP uses port number 520.

Hope count:

Hop count is the number of routers occurring in between the source and destination network. The path with the lowest hop count is considered as the best route to reach a network and therefore placed in the routing table. RIP prevents routing loops by limiting the number of hops allowed in a path from source and destination. The maximum hop count allowed for RIP is 15 and a hop count of 16 is considered as network unreachable.

-It is a very simple protocol based on distance vector routing. RIP implements distance vector routing directly with some considerations:

1. In an autonomous system, we are dealing with routers and networks (links). The routers have routing tables; networks do not.
2. The destination in a routing table is a network, which means the first column defines a network address.
3. The metric used by RIP is very simple; the distance is defined as the number of links (networks) to reach the destination. For this reason, the metric in RIP is called a hop count.
4. Infinity is defined as 16, which means that any route in an autonomous system using RIP cannot have more than 15 hops.
5. The next-node column defines the address of the router to which the packet is to be sent to reach its destination

Features of RIP

1. Updates of the network are exchanged periodically.

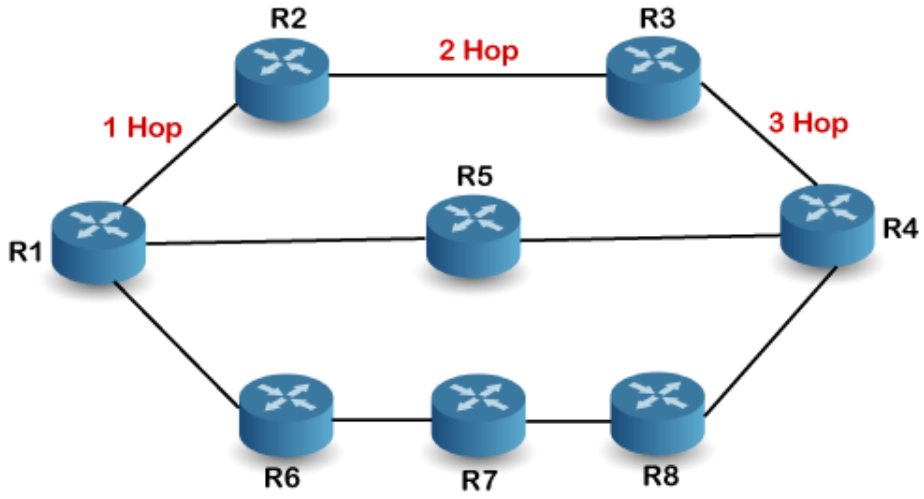
2. Updates (routing information) are always broadcast.

3. Full routing tables are sent in updates.

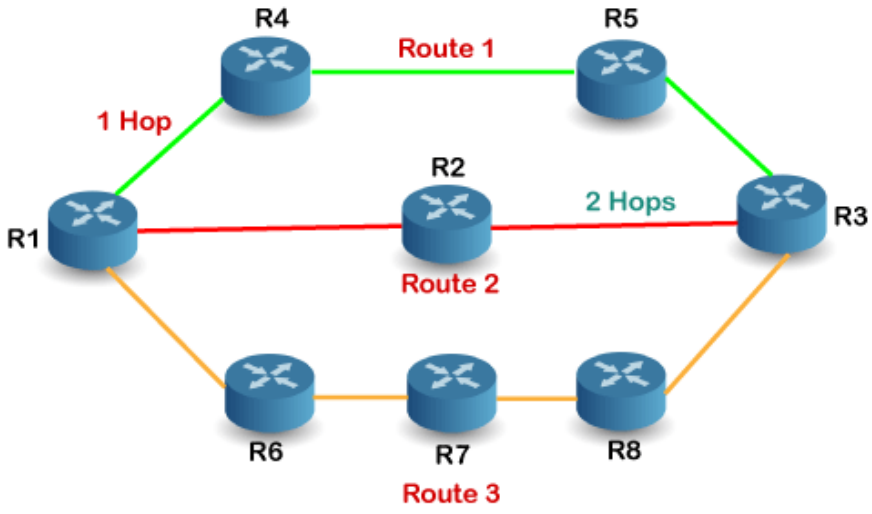
4. Routers always trust routing information received from neighbor routers. This is also known as Routing on rumors.

RIP working

When the router sends the packet to the network segment, then it is counted as a single hop.



In the above figure, when the router 1 forwards the packet to the router 2 then it will count as 1 hop count. Similarly, when the router 2 forwards the packet to the router 3 then it will count as 2 hop count, and when the router 3 forwards the packet to router 4, it will count as 3 hop count. In the same way, RIP can support maximum upto 15 hops, which means that the 16 routers can be configured in a RIP.



- If there are 8 routers in a network where Router 1 wants to send the data to Router 3. If the network is configured with RIP, it will choose the route which has the least number of hops. There are three routes in the above network, i.e., Route 1, Route 2, and Route 3.

The Route 2 contains the least number of hops, i.e., 2 where Route 1 contains 3 hops, and Route 3 contains 4 hops, so RIP will choose Route 2.

The following are the advantages of a RIP protocol:

- ✓ It is easy to configure
- ✓ It has less complexity
- ✓ The CPU utilization is less.

RIP messages

-Request

- A request message is sent by a router that has just come up or by a router that has some time-out entries
- A request can ask about specific entries or all entries

-Response

- A response can be either solicited (ask for a proposal) or unsolicited (30s or when there is a change in the routing table)

RIP Timers

-Periodic timer

- It controls the advertising of regular update message (25 ~ 30 sec)

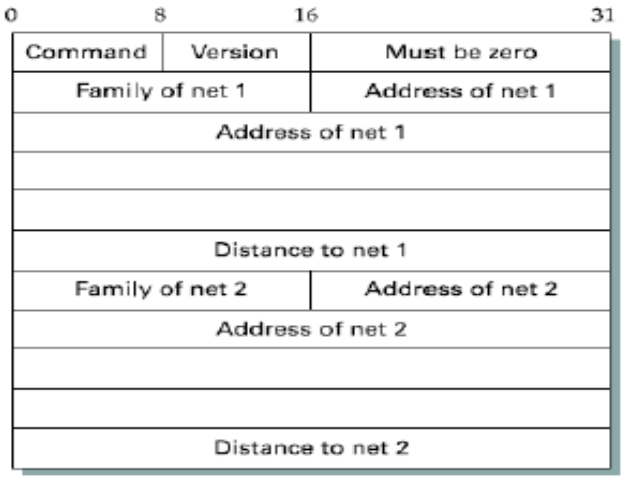
-Expiration timer

- It governs the validity of a route (180 sec)
- The route is considered expired and the hop count of the route is set to 6

-Garbage collection timer

- An invalid route is not purged(remove that not in use) from the routing table until this timer expires (120 sec)

RIP Packet format



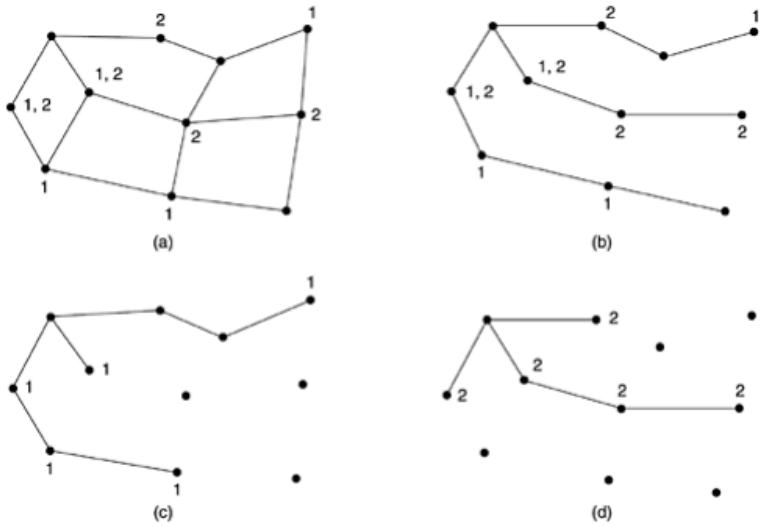
- RIP messages uses UDP datagrams on port 520
  - Implemented in Unix systems by the 'routed' daemon(manages the network routing table in the kernel).
- Size of datagram limited to 512 bytes (allow advertisement of 25 routes.
- Command: 8 bits -Request (1), reply (2)

- Version: 1 or 2
- Family: of protocol used for TCP/IP it is 2
- Network address : 32 bytes
- Distance: hop count from the advertising router to the destination network
- Response: solicited or unsolicited

Multicast Routing

- **Multicasting-** In multicast communication, there is one source and a group of destinations. The relationship is one-to-many. In this type of communication, the source address is a unicast address, but the destination address is a group address, which defines one or more destinations. The group address identifies the members of the group.
- Some applications require that widely-separated processes work together in groups, for example, a group of processes implementing a distributed database system. In these situations, it is frequently necessary for one process to send a message to all the other members of the group. If the group is small, it can just send each other member a point-topoint message. If the group is large, this strategy is expensive. Thus, we need a way to send messages to well-defined groups that are numerically large in size but small compared to the network as a whole.
- Sending a message to such a group is called **multicasting,** and its routing algorithm is called **multicast routing.**
- Multicasting requires group management. Some way is needed to create and destroy groups, and to allow processes to join and leave groups.
- How these tasks are accomplished is not based on the routing algorithm. It is based on the fact that when a process joins a group, it informs its host of this fact. It is important that routers know which of their hosts belong to which groups.
- Either hosts must inform their routers about changes in group membership, or routers must query their hosts periodically. Either way, routers learn about which of their hosts are in which groups. Routers tell their neighbors, so the information propagates through the subnet.
- To do multicast routing, each router computes a spanning tree covering all other routers. For example, in Fig.(a) we have two groups, 1 and 2. Some routers are attached to hosts that belong to one or both of these groups, as indicated in the figure. A spanning tree for the leftmost router is shown in Fig.(b).

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



- When a process sends a multicast packet to a group, the first router examines its spanning tree and prunes it, removing all lines that do not lead to hosts that are members of the group.In this example, Fig. (c) shows the pruned spanning tree for group 1. Similarly, Fig. (d) shows the pruned spanning tree for group 2. Multicast packets are forwarded only along the appropriate spanning tree.
- Various ways of pruning the spanning tree are possible. The simplest one can be used if **link state routing** is used and each router is aware of the complete topology, including which hosts belong to which groups. Then the spanning tree can be pruned, starting at the end of each path, working toward the root, and removing all routers that do not belong to the group in question.
- With **distance vector routing**, a different pruning strategy can be followed. The basic algorithm is reverse path forwarding. However, whenever a router with no hosts interested in a particular group and no connections to other routers receives a multicast message for that group, it responds with a PRUNE message, telling the sender not to send it any more multicasts for that group. When a router with no group members among its own hosts has received such messages on all its lines, it, too, can respond with a PRUNE message. In this way, the subnet is recursively pruned.
- One potential **disadvantage** of this algorithm is that it scales poorly to large networks. Suppose that a network has n groups, each with an average of m members. For each group, m pruned spanning trees must be stored, for a total of mn trees.
- When many large groups exist, considerable storage is needed to store all the trees. An alternative design uses core-based trees→ Here, a single spanning tree per group is computed, with the root (the core) near the middle of the group. To send a multicast message, a host sends it to the core, which then does the multicast along the spanning tree. Although this tree will not be optimal for all sources, the reduction in storage costs from m trees to one tree per group is a major saving.

Routing for Mobile Host

- The term mobile hosts to mean either category, as distinct from stationary hosts that never move.
- These mobile hosts introduce a new complication:

➢ To route a packet to a mobile host, the network first has to find it.

- The basic idea used for mobile routing in the Internet and cellular networks is

  -For the mobile host to tell a host at the home location where it is now. (cellular network or mobile network is a communication network where the last link is wireless)

  -This host, which acts on behalf of the mobile host, is called the home agent.

- Once it knows where the mobile host is currently located, it can forward packets so that they are delivered.

Definition of Terms

i. **Home address :** the IP address assigned to the device within its home network.
ii. **Home network :** the **network** within which the device receives its **identifying IP address** (home address).
iii. **Foreign network :** the **network** in which a mobile node is **operating** when away from its home network.
iv. **Care-of-address:** the **network-native IP address** of the device when operating in a foreign network.
v. **Stationary :** Hosts that never move are said to be ***stationary***. They are connected to the network by copper wires or fiber optics.
vi. **Home agent : It** is a **router** on a mobile node's home network which **tunnels datagrams** for delivery to the mobile node when it is away from home.

- It maintains **current location (IP address)information** for the mobile node.
- It is used with one or more foreign agents.

vii. **Foreign agent : It** is a **router** that stores information about mobile nodes visiting its network. Foreign agents also advertise care of addresses which are used by Mobile IP.
viii. **Binding :** the **association** of the home address with a care of address.

- Here we have a **WAN** consisting of routers and hosts.
- Connected to the WAN are LANs, MANs, and wireless cell.
- **Users** who never move are said to be **stationary**. They are connected to the **network** by copper wires or fibre optic.
- two other kinds of users.

(1) **Migratory users** are basically stationary users who move from one fixed site to another from time to time but use the network only when they are physically connected to it.
(2) **Roaming users** actually compute on the run and want to maintain their connections as they move around.

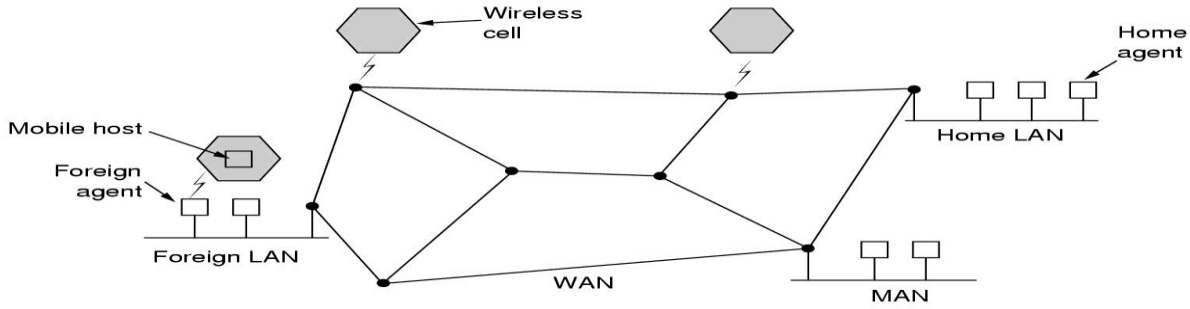


Fig: The model of the world that network designers typically use

- The **world** is divided into small areas and each area has one or more foreign agent, which keeps track of all mobile users visiting the area .
- In addition, each area has a **home agent**, which keeps the track of users whose home is in the area, but who are currently visiting another area.
- When a new user enters an area, either by connecting to it, or just wandering into the cell, his computer must register itself with the **foreign agent** there.

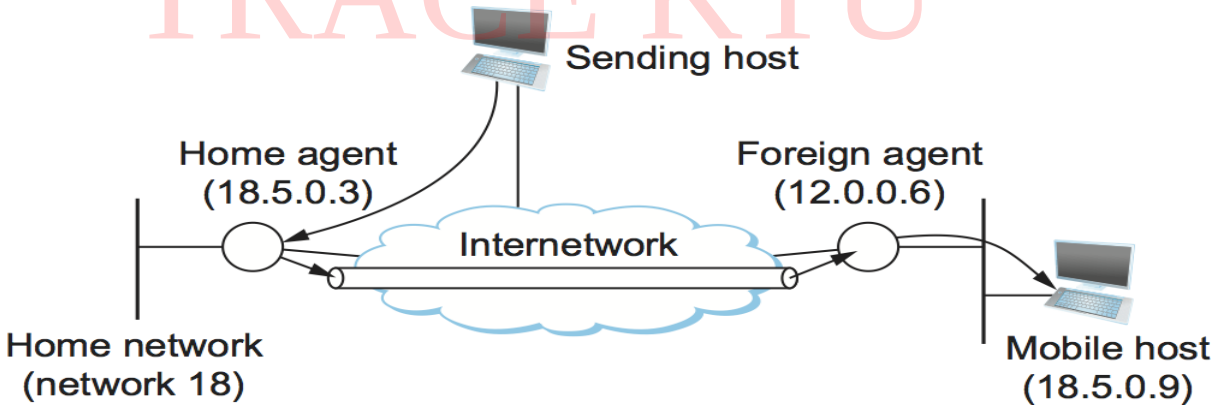


Figure-Mobile host and mobility agents

- **<u>The registration procedure typically works like this:</u>**

1) Periodically, each **foreign agent** broadcasts a packet announcing its existence and address. A newly arrived **mobile host** may wait for one these messages, but if one arrives quickly enough, the **mobile host** can broadcast a packet saying:" are there any foreign agent around? "

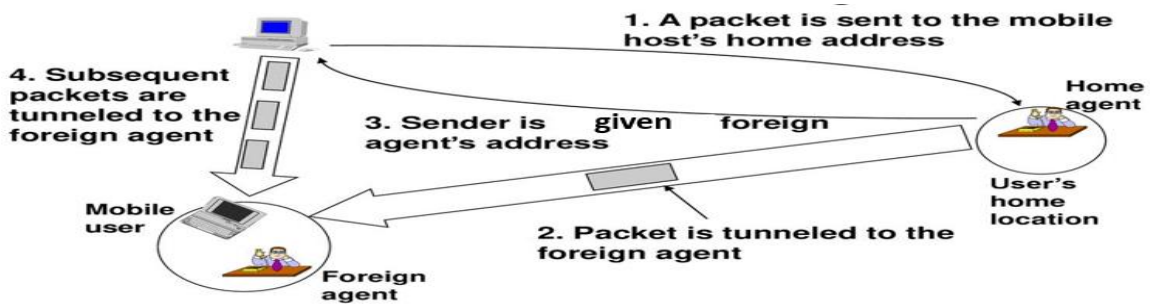
2) The mobile host just **register** with the foreign agent, giving its home address, current data link layer address, and some security information.

3) The **foreign agent** contacts the mobile hosts home agent and says, one of your hosts is over here, It also includes the security information, to convince the home agent that the mobile hosts are really there.

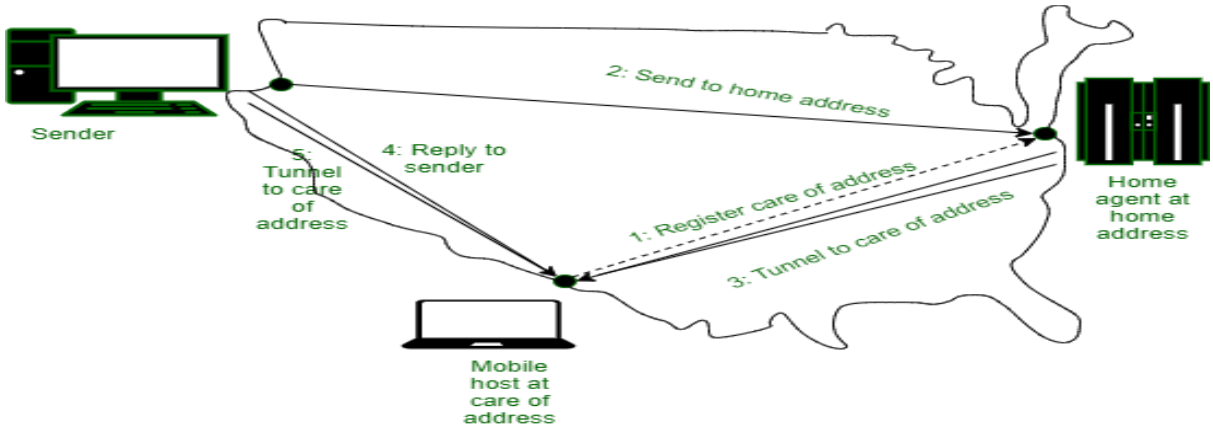
4) The **home agent** examines the security information, which contains a **timestamp**, to prove that it was generated within the past few seconds. If it is happy, it tells the foreign agent to proceed.

5) When the **foreign agent** gets the acknowledgement from the **home agent**, it makes an entry in its **table** and informs the mobile hosts that it is now registered.

Fig : Packet routing for mobile users



- When a packet is sent to a **mobile host**, it is routed to the host's home LAN because that is what the address says should be done, as illustrated in **step 1** of following figure.
- Here the **sender**, in the northwest city of **Seattle**, wants to send a packet to a **host** normally across the **United States in New York**.
- Packets sent to the mobile host on its home LAN in **New York** are intercepted by the **home agent** there. The **home agent** then looks up the mobile host's new (temporary) location and finds the address of the foreign agent handling the mobile host, in **Los Angeles**. **Fig : Packet routing for mobile users**



- The **home agent** then does two things.
- First, it encapsulates the packet in the payload field of an outer packet and sends the latter to the **foreign agent** (step 2 in Fig.).

✓ This mechanism is called **tunneling**;
✓ After getting the encapsulated packet, the foreign agent removes the original packet from the payload field and sends it to the **mobile host** as a data link frame.

- Second, the **home agent** tells the sender to henceforth send packets to **the mobile host** by encapsulating them in the payload of packets explicitly addressed to the **foreign agent** instead of just sending them to the mobile host's home address (step 3).
- Subsequent packets can now be routed directly to the host via the **foreign agent** (step 4), bypassing the home location entirely.

CONGESTION-

-When one part of the subnet (e.g. one or more routers in an area) becomes overloaded, congestion results.

- Because routers are receiving packets faster than they can forward them, one of two things must happen:
  - The subnet must prevent additional packets from entering the congested region until those already present can be processed.
  - The congested routers can discard queued packets to make room for those that are arriving.

FACTORS THAT AFFECT CONGESTION

- Packet arrival rate exceeds the outgoing link capacity.
- Insufficient memory to store arriving packets
- Bursty traffic
- Slow processor

Insufficient memory to store arriving packets: If all of a sudden a stream of packets arrive on several input lines and need to be out on the same output line, then a long queue will be build up for that output. If there is insufficient memory to hold these packets, then packets will be lost (dropped). Adding more memory also may not help in certain situations. If router have an infinite amount of memory even then instead of congestion being reduced, it gets worse; because by the time packets gets at the head of the queue, to be dispatched out to the output line, they have already timed-out (repeatedly), and duplicates may also be present. All the packets will be forwarded to next router up to the destination, all the way only increasing the load to the network more and more. Finally when it arrives at the destination, the packet will be discarded, due to time out, so instead of been dropped at any intermediate router (in case memory is restricted) such a packet goes all the way up to the destination, increasing the network load throughout and then finally gets dropped there.

Slow processors: also cause Congestion. If the router CPU is slow at performing the task required for them (Queuing buffers, updating tables, reporting any exceptions etc.), queue can build up even if there is excess of line capacity.

LowBandwidth lines can also cause congestion. Upgrading lines but not changing slow processors, or vice-versa, often helps a little; these can just shift the bottleneck to some other point. Routers respond to overloading by dropping packets. When these packets contain TCP segments, the segments don't reach their destination, and they are therefore left unacknowledged, which eventually leads to timeout and retransmission.

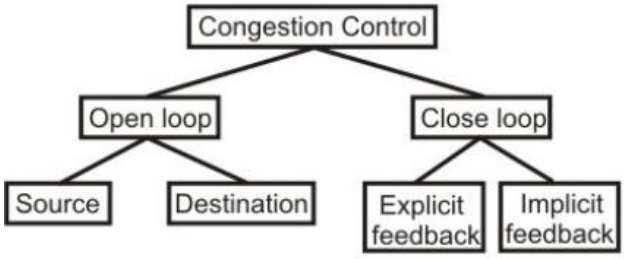
Bursty nature of traffic. If the hosts could be made to transmit at a uniform rate, then congestion problem will be less common and all other causes will not even led to congestion .

Congestion control Vs flow control

- Congestion control is a global issue – involves every router and host within the subnet
- Flow control – scope is point-to-point; involves just sender and receiver

Congestion Control Techniques

Congestion control refers to the mechanisms and techniques used to control congestion and keep the traffic below the capacity of the network. As shown in the following Fig., the congestion control techniques can be broadly classified two broad categories:



• Open loop: Protocols to prevent or avoid congestion, ensuring that the system (or network under consideration) never enters a Congested State. Open Loop solutions are rules or policies that include deciding upon when to accept traffic, when to discard it, making scheduling decisions and so on. Main point here is that they make decision without taking into consideration the current state of the network. The open loop algorithms are further divided on the basis of whether these acts on source versus that act upon destination

• Closed loop: Protocols that allow system to enter congested state, detect it, and remove it. The second category is based on the concept of feedback. During operation, some system

parameters are measured and feed back to portions of the subnet that can take action to reduce the congestion. This approach can be divided into 3 steps:

• Monitor the system (network) to detect whether the network is congested or not and what's the actual location and devices involved.

• Pass this information to the places where actions can be taken

• Adjust the system operation to correct the problem.

Various Metrics can be used to monitor the network for congestion. Some of them are: the average queue length, number of packets that are timed-out, average packet delay, number of packets discarded due to lack of buffer space, etc.

In the second step, a router, which detects the congestion send special packets to the source (responsible for the congestion) announcing the problem. These extra packets increase the load at that moment of time, but are necessary to bring down the congestion at a later time. Other approaches are also used at times to curtail down the congestion. For example, hosts or routers send out probe packets at regular intervals to explicitly ask about the congestion and source itself regulate its transmission rate, if congestion is detected in the network. This kind of approach is a pro-active one, as source tries to get knowledge about congestion in the network and act accordingly.

Another approach may be where instead of sending information back to the source an intermediate router which detects the congestion send the information about the congestion to rest of the network, piggy backed to the outgoing packets. This approach will in no way put an extra load on the network (by not sending any kind of special packet for feedback).

Once the congestion has been detected and this information has been passed to a place where the action needed to be done, then there are two basic approaches that can overcome the problem. These are: either to increase the resources or to decrease the load. For example, separate dial-up lines or alternate links can be used to increase the bandwidth between two points, where congestion occurs. Another example could be to decrease the rate at which a particular sender in transmitting packets out into the network.

The closed loop algorithms can also be divided into two categories, namely explicit feedback and implicit feedback algorithms. In the explicit approach, special packets are sent back to the sources to curtail down the congestion. While in implicit approach, the source itself acts pro-actively and tries to deduce the existence of congestion by making local observations.

Congestion Prevention Policies

At the data link layer,

- The retransmission policy is concerned with how fast a sender times out and what it transmits upon timeout. A jumpy sender that times out quickly and retransmits all outstanding packets using go back n will put a heavier load on the system than will a leisurely sender that uses selective repeat.
- If receivers routinely discard all out-of-order packets, these packets will have to be transmitted again later, creating extra load. With respect to congestion control, selective repeat is clearly better than go back n.
- Acknowledgement policy also affects congestion. If each packet is acknowledged immediately, the acknowledgement packets generate extra traffic. However, if acknowledgements are saved up to piggyback onto reverse traffic, extra timeouts and retransmissions may result.
- A tight flow control scheme (e.g., a small window) reduces the data rate and thus helps fight congestion.

At the network layer,

- The choice between using virtual circuits and using datagram affects congestion since many congestion control algorithms work only with virtual-circuit subnets.
- Packet queuing and service policy relates to whether routers have one queue per input line, one queue per output line, or both. It also relates to the order in which packets are processed (e.g., round robin or priority based).
- Discard policy is the rule telling which packet is dropped when there is no space. A good policy can help alleviate congestion and a bad one can make it worse.
- A good routing algorithm can help avoid congestion by spreading the traffic over all the lines, whereas a bad one can send too much traffic over already congested lines.
- Packet lifetime management deals with how long a packet may live before being discarded. If it is too long, lost packets may clog up the works for a long time, but if it is too short, packets may sometimes time out before reaching their destination, thus inducing retransmissions.

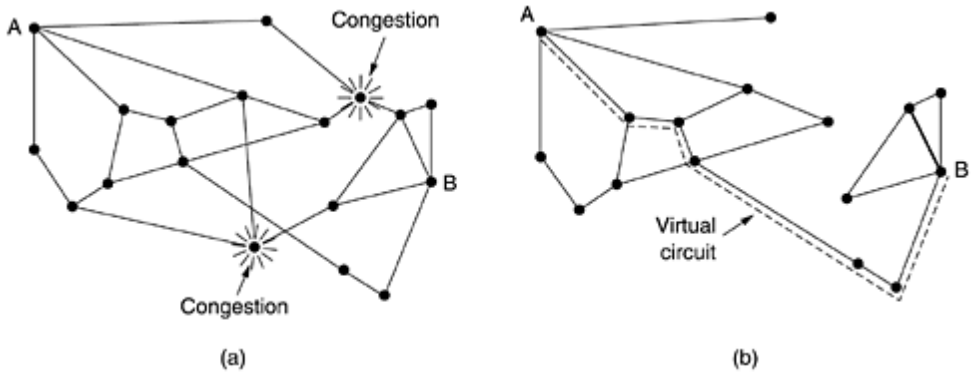
In the transport layer, the same issues occur as in the data link layer, but in addition, determining the timeout interval is harder because the transit time across the network is less predictable than the transit time over a wire between two routers. If the timeout interval is too short, extra packets will be sent unnecessarily. If it is too long, congestion will be reduced but the response time will suffer whenever a packet is lost.

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

Congestion control in virtual Circuit

One technique is **admission control,** the idea is, once congestion has been signaled, no more virtual circuits are set up until the problem has gone away. Thus, attempts to set up new transport layer connections fail. While this approach is crude, it is simple and easy to carry out.

An alternative approach is to allow new virtual circuits but carefully route all new virtual circuits around problem areas. For example, consider the subnet of Fig. (a), in which two routers are congested, as indicated.



(a) A congested subnet. (b) A redrawn subnet that eliminates the congestion. A virtual circuit from* A *to* B *is also shown.

Another strategy relating to virtual circuits is to negotiate an agreement between the host and subnet when a virtual circuit is set up. This agreement normally specifies the volume and shape of the traffic, quality of service required, and other parameters. To keep its part of the agreement, the subnet will typically reserve resources along the path when the circuit is set up. These resources can include table and buffer space in the routers and bandwidth on the lines. In this

way, congestion is unlikely to occur on the new virtual circuits because all the necessary resources are guaranteed to be available. Disadvantage is wastage of resources.

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.

The Warning Bit

The old DECNET architecture signaled the warning state by setting a special bit in the packet's header.

- 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 decreases the flow of transmission.
- 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 small stream, 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 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.
- Hosts can reduce traffic by adjusting their policy parameters, for example, their window size.

- 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.

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 router *A* in Fig.(a) that is sending traffic to a router *D* at 155 Mbps. If D begins to run out of buffers, it will take about 30 msec for a choke packet to get back to A to tell it to slow down. The choke packet propagation is shown as the second, third, and fourth steps in Fig. (a). In those 30 msec, another 4.6 megabits will have been sent. Even if the host A 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. (a) will the router D notice a slower flow.

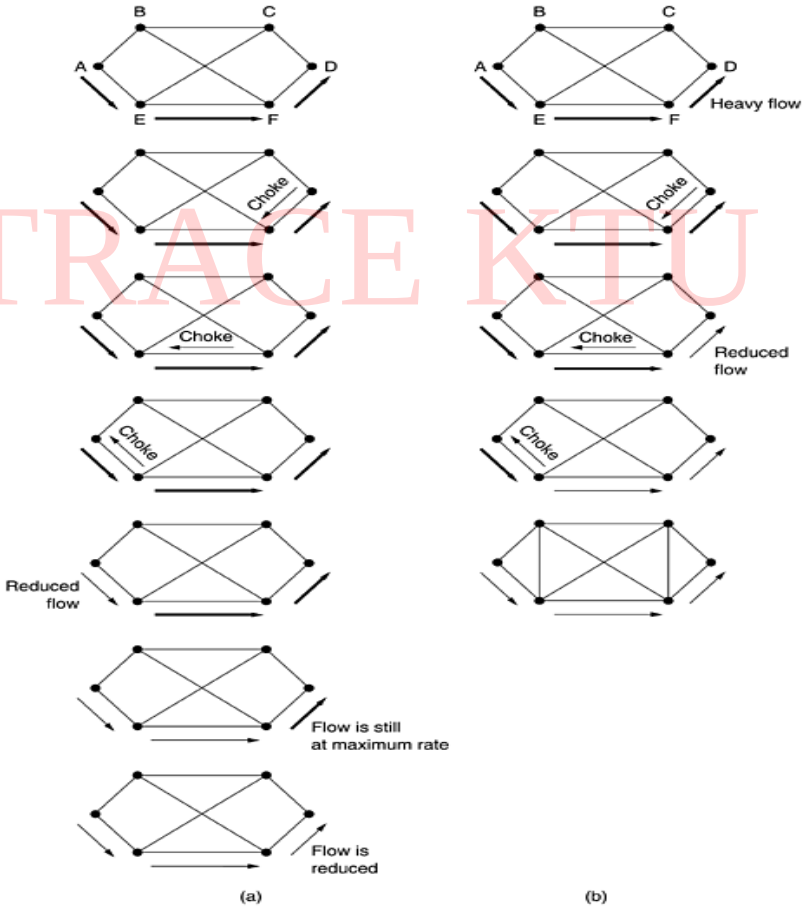


Fig.(a) A choke packet that affects only the source. (b) A Choke packet that affects each hop it passes through.

An alternative approach is to have the choke packet take effect at every hop it passes through, as shown in the sequence of Fig. (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. 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. The net effect of this hop-by-hop scheme is to provide quick relief at the point of congestion at the price of using up more buffers upstream.

Load Shedding

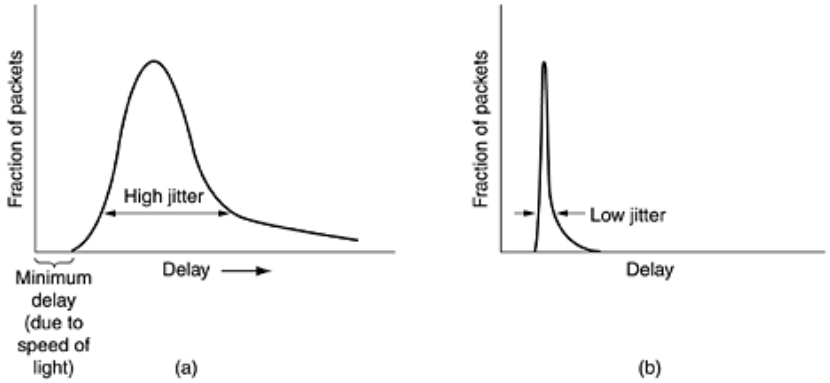
- Another simple closed loop technique
- When buffers become full, routers simply discard packets.
- Which packet is chosen to be the victim depends on the application and on the error strategy used in the data link layer.
- For a file transfer, for, e.g. cannot discard older packets since this will cause a gap in the received data. This policy is often called as *wine*.
- For real-time voice or video it is probably better to throw away old data and keep new packets. This policy is often known as *milk*
- Get the application to mark packets with discard priority
- Another option is to allow the hosts to exceed the limits specified in the agreement negotiated when the virtual circuit was set up.

Random Early Detection

- This is a proactive approach in which the router discards one or more packets *before* the buffer becomes completely full.
- Each time a packet arrives, the RED algorithm computes the average queue length, ***avg***.
- If *avg* is lower than some lower threshold, congestion is assumed to be minimal or non-existent and the packet is queued.
- If *avg* is greater than some upper threshold, congestion is assumed to be serious and the packet is discarded.
- If *avg* is between the two thresholds, this might indicate the onset of congestion. The probability of congestion is then calculated.

Jitter control

The variation (i.e., standard deviation) in the packet arrival times is called **jitter**. High jitter, for example, having some packets taking 20 msec and others taking 30 msec to arrive will give an uneven quality to the sound or movie.



(a) High jitter. (b) Low jitter.

Quality of Service

A stream of packets from a source to a destination is called a **flow**. In a connection-oriented network, all the packets belonging to a flow follow the same route; in a connectionless network, they may follow different routes. The needs of each flow can be characterized by four primary parameters: reliability, delay, jitter, and bandwidth. Together these determine the **QoS** (**Quality of Service**) the flow requires. Several common applications and the stringency of their requirements are listed in Fig.

Application	Reliability	Delay	Jitter	Bandwidth
E-mail	High	Low	Low	Low
File transfer	High	Low	Low	Medium
Web access	High	Medium	Low	Medium
Remote login	High	Medium	Medium	Low
Audio on demand	Low	Low	High	Medium
Video on demand	Low	Low	High	High
Telephony	Low	High	High	Low
Videoconferencing	Low	High	High	High

- The first four applications have stringent requirements on reliability. No bits may be delivered incorrectly. This goal is usually achieved by checksumming each packet and verifying the checksum at the destination. If a packet is damaged in transit, it is not acknowledged and will be retransmitted eventually. This strategy gives high reliability.
- The four final (audio/video) applications can tolerate errors, so no checksums are computed or verified.
- File transfer applications, including e-mail and video, are not delay sensitive. If all packets are delayed uniformly by a few seconds, no harm is done.
- Interactive applications, such as Web surfing and remote login, are more delay sensitive. Real-time applications, such as telephony and videoconferencing have strict delay requirements.
- The first three applications are not sensitive to the packets arriving with irregular time intervals between them. Remote login is somewhat sensitive to that, since characters on the screen will appear in little bursts if the connection suffers much jitter.
- Video and especially audio are extremely sensitive to jitter.
- Finally, the applications differ in their bandwidth needs, with e-mail and remote login not needing much, but video in all forms needing a great deal.

Techniques for Achieving Good Quality of Service

1. ***Overprovisioning***

An easy solution is to provide so much router capacity, buffer space, and bandwidth that the packets flow easily. The trouble with this solution is that it is expensive.

2. ***Buffering***

Flows can be buffered on the receiving side before being delivered. Buffering them does not affect the reliability or bandwidth, and increases the delay, but it smooths out the jitter. For audio and video on demand, jitter is the main problem, so this technique helps a lot.

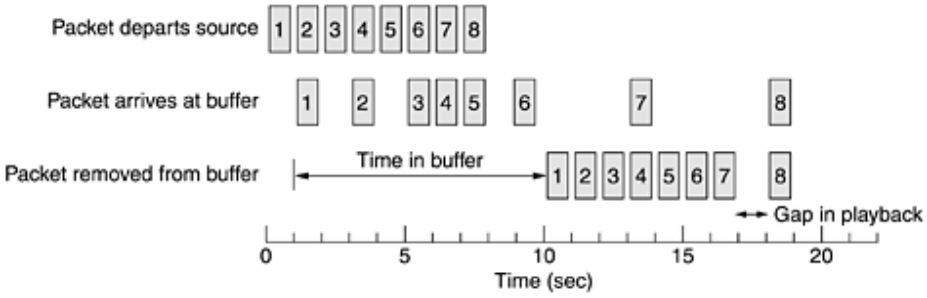


Figure . example for Smoothing the output stream by buffering packets.

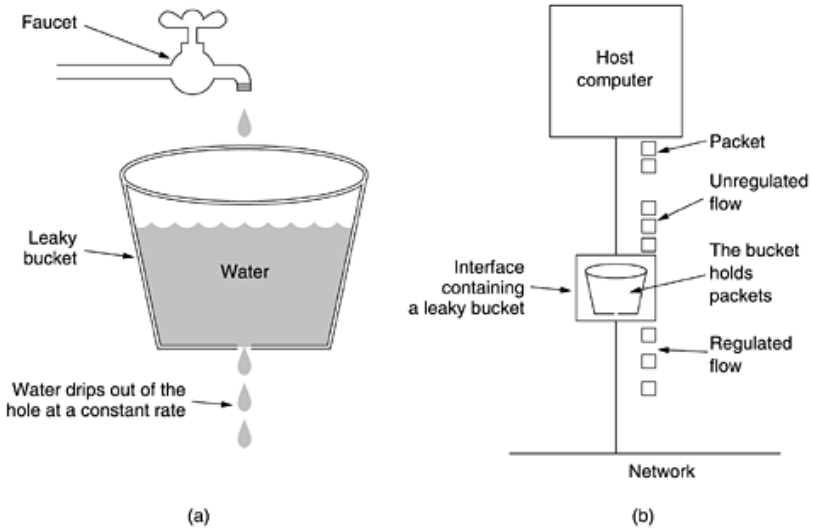
3. Traffic Shaping

Traffic shaping make the server (and hosts in general) transmit at a uniform rate, smooths out the traffic on the server side, rather than on the client side.

- Traffic shaping is about regulating the average *rate* (and burstiness) of data transmission. When a connection is set up, the user and the subnet (i.e., the customer and the carrier) agree on a certain traffic pattern (i.e., shape) for that circuit, which is called a **service level agreement**.
- Traffic shaping reduces congestion. Such agreements are not so important for file transfers but are of great importance for real-time data, such as audio and video connections, which have stringent quality-of-service requirements.
- Monitoring a traffic flow is called **traffic policing**. Agreeing to a traffic shape and policing it afterward are easier with virtual-circuit subnets than with datagram subnets.

3.1 The Leaky Bucket Algorithm

It is a single-server queuing system with constant service time Imagine a bucket with a small hole in the bottom, as illustrated in Fig. (a). No matter the rate at which water enters the bucket, the outflow is at a constant rate, ρ, when there is any water in the bucket and zero when the bucket is empty. Also, once the bucket is full, any additional water entering it spills over the sides and is lost (i.e., does not appear in the output stream under the hole)



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

The same idea can be applied to packets, as shown in Fig. (b).

- Each host is connected to the network by an interface containing a leaky bucket, that is, a finite internal queue.
- If a packet arrives at the queue when it is full, the packet is discarded. In other words, if one or more processes within the host try to send a packet when the maximum number is already queued, the new packet is discarded. This arrangement can be built into the hardware interface or simulated by the host operating system.
- The host is allowed to put one packet per clock tick onto the network. This can be enforced by the interface card or by the operating system.
- This mechanism turns an uneven flow of packets from the user processes inside the host into an even flow of packets onto the network, smoothing out bursts and greatly reducing the chances of congestion.
- When the packets are all the same size (e.g., ATM cells), this algorithm can be used as described. However, when variable-sized packets are being used, it is often better to allow a fixed number of bytes per tick, rather than just one packet. Thus, if the rule is 1024 bytes per tick, a single 1024-byte packet can be admitted on a tick, two 512-byte packets, four 256-byte packets, and so on. If the residual byte count is too low, the next packet must wait until the next tick.
- The byte-counting leaky bucket is implemented almost the same way. At each tick, a counter is initialized to *n*. If the first packet on the queue has fewer bytes than the current value of the counter, it is transmitted, and the counter is decremented by that number of bytes. Additional packets may also be sent, as long as the counter is high enough. When the counter drops below the length of the next packet on the queue, transmission stops until the next tick, at which time the residual byte count is reset and the flow can continue.

3.2 The Token Bucket Algorithm

The leaky bucket algorithm enforces a rigid output pattern at the average rate, no matter how bursty the traffic is. For many applications, it is better to allow the output to speed up when large bursts arrive, so a more flexible algorithm is needed, preferably one that never loses data. One such algorithm is the **token bucket algorithm**.

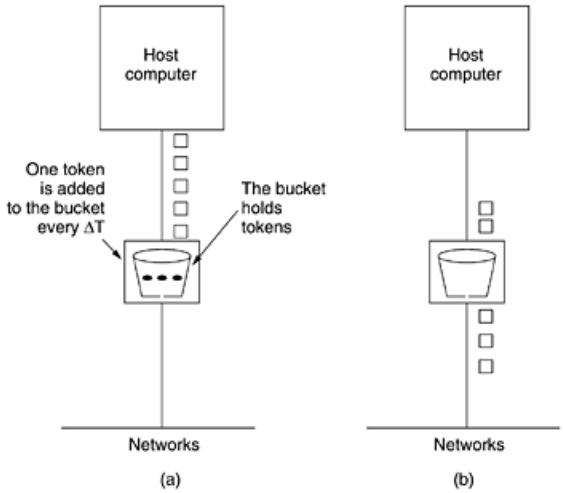


Fig. The tocken bucket algorithm a)before b) after

- In this algorithm, the leaky bucket holds tokens, generated by a clock at the rate of one token every ΔT sec. In Fig. (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 Fig. (b) we see that three of the five packets have gotten through, but the other two are stuck waiting for two more tokens to be generated.

- The token bucket algorithm provides a different kind of traffic shaping than that of the leaky bucket algorithm. The leaky bucket algorithm does not allow idle hosts to save up permission to send large bursts later. The token bucket algorithm does allow saving, up to the maximum size of the bucket, *n*. This property means that bursts of up to *n* packets can be sent at once, allowing some burstiness in the output stream and giving faster response to sudden bursts of input.
- Another difference between the two algorithms is that the token bucket algorithm throws away tokens (i.e., transmission capacity) when the bucket fills up but never discards packets. In contrast, the leaky bucket algorithm discards packets when the bucket fills up.
- Here, a minor variant is possible, in which each token represents the right to send not one packet, but *k* bytes. A packet can only be transmitted if enough tokens are available to cover its length in bytes. Fractional tokens are kept for future use.
- The leaky bucket and token bucket algorithms can also be used to smooth traffic between routers, as well as to regulate host output as in our examples.
- The implementation of the basic token bucket algorithm is just a variable that counts tokens. The counter is incremented by one every ΔT and decremented by one whenever a packet is sent. When the counter hits zero, no packets may be sent. In the byte-count variant, the counter is incremented by *k* bytes every ΔT and decremented by the length of each packet sent.

4.Resource Reservation

Once we have a specific route for a flow, it becomes possible to reserve resources along that route to make sure the needed capacity is available. Three different kinds of resources can potentially be reserved:

1. Bandwidth.
2. Buffer space.
3. CPU cycles.

- Bandwidth is the most obvious. If a flow requires 1 Mbps and the outgoing line has a capacity of 2 Mbps, trying to direct three flows through that line is not going to work. Thus, reserving bandwidth means not oversubscribing any output line.
- Buffer space-When a packet arrives, it is usually deposited on the network interface card by the hardware itself. The router software then has to copy it to a buffer in RAM and queue that buffer for transmission on the chosen outgoing line. If no buffer is available, the packet has to be discarded since there is no place to put it. For a good quality of service, some buffers can be reserved for a specific flow so that flow does not have to compete for buffers with other flows.
- Finally, CPU cycles are also a scarce resource. It takes router CPU time to process a packet, so a router can process only a certain number of packets per second. Making sure that the CPU is not overloaded is needed to ensure timely processing of each packet.

5.Admission Control

Admission control refers to the mechanism used by a router, or a switch, to accept or reject a flow based on predefined parameters called flow specifications. Before a router accepts a flow

for processing, it checks the flow specifications to see if its capacity (in terms of bandwidth, buffer size, CPU speed, etc.) and its previous commitments to other flows can handle the new flow.

6.Proportional Routing

Most routing algorithms try to find the best path for each destination and send all traffic to that destination over the best path. A different approach that has been proposed to provide a higher quality of service is to split the traffic for each destination over multiple paths. Since routers generally do not have a complete overview of network-wide traffic, the only feasible way to split traffic over multiple routes is to use locally-available information. A simple method is to divide the traffic equally or in proportion to the capacity of the outgoing links.

7.Packet Scheduling

Packets from different flows arrive at a switch or router for processing. A good scheduling technique treats the different flows in a fair and appropriate manner. Several scheduling techniques are designed to improve the quality of service. Three of them here: FIFO queuing, priority queuing, and weighted fair queuing.

1) FIFO Queuing: In first-in, first-out (FIFO) queuing, packets wait in a buffer (queue) until the node (router or switch) is ready to process them. If the average arrival rate is higher than the average processing rate, the queue will fill up and new packets will be discarded. Following Figure shows a conceptual view of a FIFO queue.

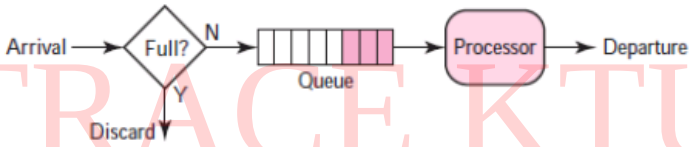


Fig: FIFO queue

2) Priority Queuing: In priority queuing, packets are first assigned to a priority class. Each priority class has its own queue. The packets in the highest-priority queue are processed first. Packets in the lowest-priority queue are processed last. Note that the system does not stop serving a queue until it is empty. Figure shows priority queuing with two priority levels (for simplicity).

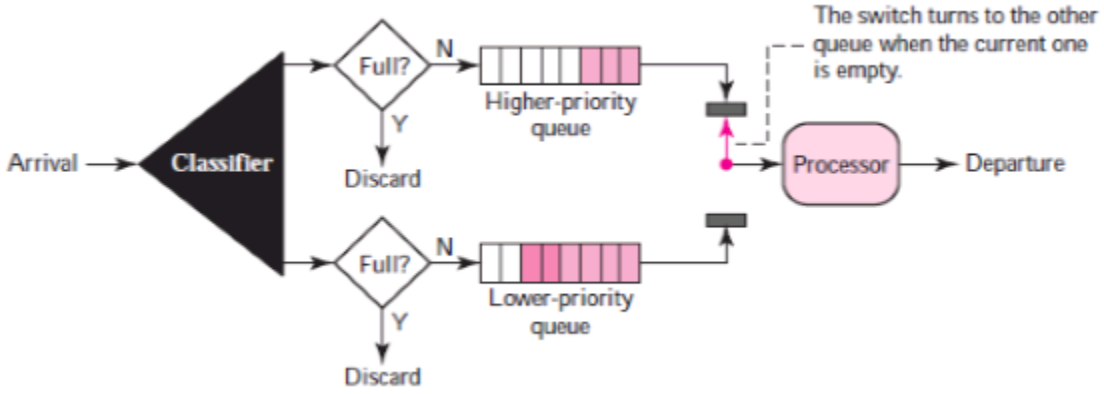


Fig: Priority queuing

A priority queue can provide better QoS than the FIFO queue because higher priority traffic, such as multimedia, can reach the destination with less delay.

3) Weighted Fair Queuing: A better scheduling method is weighted fair queuing. In this technique, the packets are still assigned to different classes and admitted to different queues. The queues, however, are weighted based on the priority of the queues; higher priority means a higher weight. The system processes packets in each queue in a round-robin fashion with the number of packets selected from each queue based on the corresponding weight.

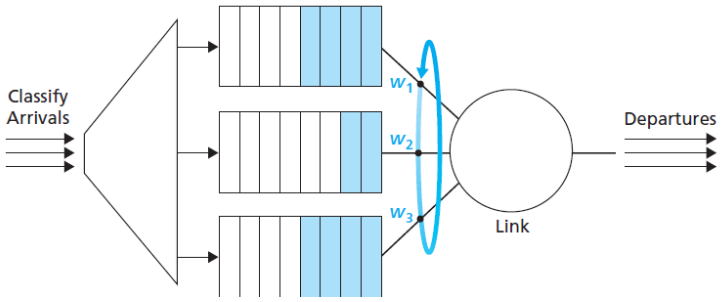


Figure Weighted fair queuing (WFQ)

RSVP: Resource Reservation Protocol

The RSVP protocol allows applications to reserve bandwidth for their data flows, other protocols are used for sending the data. RSVP allows multiple senders to transmit to multiple groups of receivers, permits individual receivers to switch channels freely, and optimizes bandwidth use while at the same time eliminating congestion. To implement RSVP the RSVP software must be present on the receivers, senders and routers.

Principle characteristics:

- Provides reservations for bandwidth in multicast spanning trees (unicast is handled as a degenerate case of multicast). Each group is assigned a group address. To send to a group, a sender puts the group's address in its packets. The standard multicast routing algorithm then builds a spanning tree covering all group members. The routing algorithm is not part of RSVP. The only difference from normal multicasting is a little extra information that is multicast to the group periodically to tell the routers along the tree to maintain certain data structures in their memories.
- Is receiver-oriented, that is, the receiver of the data flow initiates and maintains the resource reservation used for that flow. To get better reception and eliminate congestion, any of the receivers in a group can send a reservation message up the tree to the sender. The message is propagated using the reverse path forwarding algorithm discussed earlier. At each hop, the router notes the reservation and reserves the necessary bandwidth. If insufficient bandwidth is available, it reports back failure. By the time the message gets back to the source, bandwidth has been reserved all the way from the sender to the receiver making the reservation request along the spanning tree.

  When making a reservation, a receiver can (optionally) specify one or more sources that it wants to receive from. It can also specify whether these choices are fixed for the duration of the reservation or whether the receiver wants to keep open the option of

changing sources later. The routers use this information to optimize bandwidth planning. In particular, two receivers are only set up to share a path if they both agree not to change sources later on.

Differentiated Services

Difficulties associated with the Intserv model of per-flow reservation of resources:

- Scalability. Intermediate routers have to maintain per-flow state;
- Flexible service models. Intserv provides small number of prespecified service classes. Need for qualitative or relative definitions of service classes.

Diffserv is a architecture for providing scalable and flexible service differentiation ñ that is the ability to handle different classes of traffic in different ways within the Internet. Diffserv arch. have two sets of functional elements: Edge functions: Packet Classification and traffic conditioning and Core function: Forwarding.

Differentiated services (DS) can be offered by a set of routers forming an administrative domain (e.g., an ISP or a telco). The administration defines a set of service classes with corresponding forwarding rules. If a customer signs up for DS, customer packets entering the domain may carry a *Type of Service* field in them, with better service provided to some classes (e.g., premium service) than to others. Traffic within a class may be required to conform to some specific shape, such as a leaky bucket with some specified drain rate. An operator with a nose for business might charge extra for each premium packet transported or might allow up to *N* premium packets per month for a fixed additional monthly fee. Note that this scheme requires no advance setup, no resource reservation, and no time-consuming end-to-end negotiation for each flow, as with integrated services. This makes DS relatively easy to implement.

Expedited Forwarding

-Two classes of service are available: regular and expedited. The vast majority of the traffic is expected to be regular, but a small fraction of the packets are expedited. The expedited packets should be able to transit the subnet as though no other packets were present.

Assured Forwarding

-A somewhat more elaborate scheme for managing the service classes is called **assured forwarding**. It is described in RFC 2597. It specifies that there shall be four priority classes, each class having its own resources. In addition, it defines three discard probabilities for packets that are undergoing congestion: low, medium, and high. Taken together, these two factors define 12 service classes.

-Step 1 is to classify the packets into one of the four priority classes. This step might be done on the sending host (as shown in the figure) or in the ingress (first) router. The advantage of doing classification on the sending host is that more information is available about which packets belong to which flows there.

-Step 2 is to mark the packets according to their class. A header field is needed for this purpose. Fortunately, an 8-bit *Type of service* field is available in the IP header, as we will see shortly.

-Step 3 is to pass the packets through a shaper/dropper filter that may delay or drop some of them to shape the four streams into acceptable forms, for example, by using leaky or token buckets. If there are too many packets, some of them may be discarded here, by discard category.