Module 4
Network layer: Network Layer Design Issues, Routing Algorithm – Optimality principle - Flooding - Distance vector routing – Link state routing –Multicast Routing – Congestion Control Algorithms – General principles – Congestion prevention policies – Choke packets – Random Early Detection- Quality of Service requirements- Buffering, Traffic shaping – Leaky bucket algorithm.

NETWORK LAYER

The network layer is the third level of the Open Systems Interconnection Model (OSI Model) and the layer that provides data routing paths for network communication. Data is transferred in the form of packets via logical network paths in an ordered format controlled by the network layer.

Logical connection setup, data forwarding, routing and delivery error reporting are the network layer's primary responsibilities.

NETWORK LAYER DESIGN ISSUES

To solve the problem of delivery through several links, the network layer was designed. The network layer is responsible for host-to-host delivery and for routing the packets through the routers. Network Layer Design Issues include the service provided to the transport layer and the internal design of the subnet.

Major issues that the designers of the network layer must deal with are

i. Store-and-Forward Packet Switching
ii. Services Provided to the Transport Layer
iii. Implementation of Connectionless Service
iv. Implementation of Connection-Oriented Service

i. Store-and-Forward Packet Switching

In Store-and Forwarding Packet Switching mechanism, a host transmits a packet to the nearest router, the packet is stored there until it has fully arrived so the checksum can be verified, then it is forwarded to the next router along the path until it reaches the destination host.

The environment of the network layer protocols is shown in figure 4.1. The major components of the system are the carrier'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 carrier's routers, A, by a leased line. In contrast, H2 is on a LAN with a router, F, owned and operated by the customer. This router also has a leased line to the carrier's equipment. We have shown F as being outside the oval because it does not belong to the carrier, but in terms of construction, software, and protocols, it is probably no different from the carrier's routers

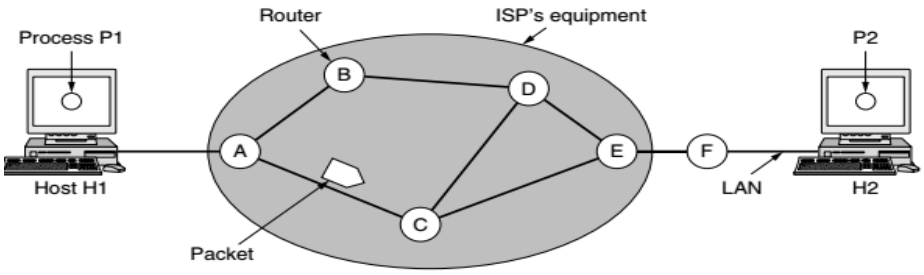


Fig. 4.1 The environment of the network layer protocols

In tore-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 carrier. The packet is stored there until it has fully arrived so the checksum can be verified. Then it is forwarded to the next router along the path until it reaches the destination host, where it is delivered.

ii. Services provided to the Transport Layer

The network layer provides services to the transport layer at the network layer/transport layer interface. The network layer services have been designed with the following goals in mind.

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

Two major services provided by the network layer to the transport layer are

- Connectionless service (Eg. Internet)
- Connection-Oriented Service (Eg. ATM Network)

iii. Implementation of Connectionless Service

- In connectionless service, packets are injected into the subnet individually and routed independently of each other.
- No advance setup is needed.
- The packets in connectionless service are called datagrams and the subnet is called a datagram subnet.

Working of datagram subnet

- Suppose that the process P1 in Fig. 4-2 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, prepends a transport header to the front of the of the message and hands the result to the network layer.

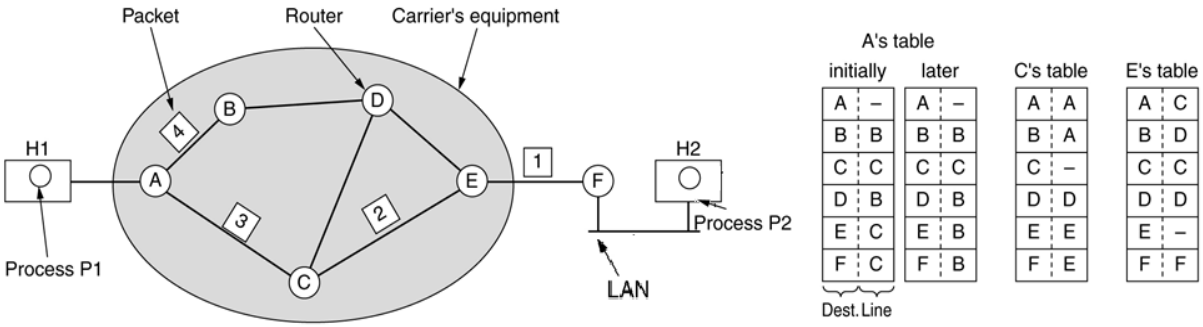


Figure 4-2. Routing within a datagram subnet.

- The transport layer code runs on H1, prepends a transport header to the front of the of the message and hands the result to the network layer.
- Assume 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 sends each of them in turn to router A using some point-to-point protocol.
- Every router has an internal table telling it where to send packets for each possible destination. Each table entry is a pair consisting of a destination and the outgoing line to use for that destination. Only directly-connected lines can be used.
- In Fig. 4-2, A has only two outgoing lines—to B and C—so every incoming packet must be sent to one of these routers
- As they arrived at A, packets 1, 2, and 3 were stored briefly (to verify their checksums). Then each was forwarded to C according to A's table as shown under the label "initially". Packet 1 was then forwarded to E and then to F. When it got to F, it was encapsulated in a data link layer frame and sent to H2 over the LAN. Packets 2 and 3 follow the same route.
- On the arrival of packet 4 to A it was sent to a different route (Router B) than that of the first three because 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.

iv. Implementation of Connection-Oriented Service

- In connection-oriented service, a path from the source router to the destination router must be established before any data packets can be sent. This connection is called a VC (virtual circuit) and the subnet is called a virtual-circuit subnet.
- In Virtual circuits, 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.
- When the connection is released, the virtual circuit is also terminated.
- With connection-oriented service, each packet carries a **flow label** (a virtual circuit identifier) that defines the virtual path the packet should follow
- As an example, consider the situation of Fig. 4-3. Here, host H1 has established connection 1 with host H2. It 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.

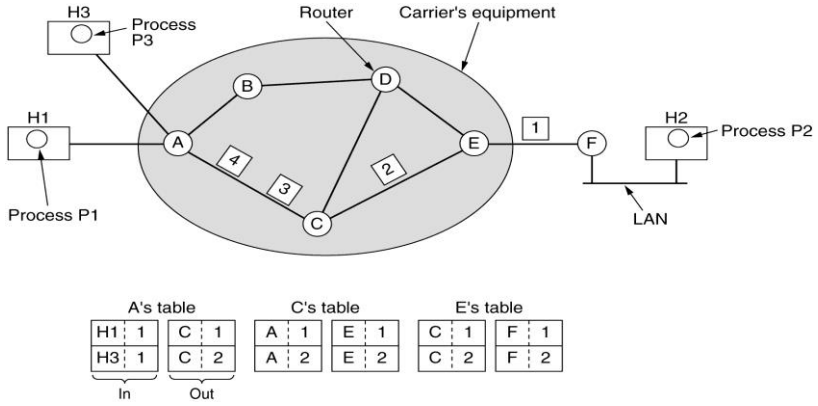


Figure 5-3. Routing within a virtual-circuit subnet.

- When H3 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 subnet 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, called label switching.

Comparison of Virtual-Circuit and Datagram Subnets

Issue	Datagram subnet	Virtual-circuit subnet
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

Inside the subnet, several trade-offs exist between virtual circuits and datagrams.

- Between router memory space and bandwidth.

    - Virtual circuits allow packets to contain circuit numbers instead of full destination addresses. A full destination address in every packet may represent a significant amount of overhead and wasted bandwidth. The price paid for using virtual circuits internally is the table space within the routers. Depending upon the relative cost of communication circuits versus router memory, one or the other may be cheaper.
- Setup time versus address parsing time.
    - Using virtual circuits requires a setup phase, which takes time and consumes resources. However, figuring out what to do with a data packet in a virtual-circuit subnet is easy: the router just uses the circuit number to index into a table to find out where the packet goes. In a datagram subnet, a more complicated lookup procedure is required to locate the entry for the destination.
- Amount of table space required in router memory
    - A datagram subnet needs to have an entry for every possible destination
    - A virtual-circuit subnet just needs an entry for each virtual circuit.
- Quality of Service & Congestion
    - Virtual circuits have advantages in guaranteeing quality of service and avoiding congestion within the subnet because resources (e.g., buffers, bandwidth, and CPU cycles) can be reserved in advance, when the connection is established.
    - With a datagram subnet, congestion avoidance is more difficult.
- Router crashes
    - If a router crashes and loses its memory in virtual circuits, even if it comes back up a second later, all the virtual circuits passing through it will have to be aborted.
    - If a datagram router goes down, only those users whose packets were queued in the router at the time will suffer, and maybe not even all those, depending upon whether they have already been acknowledged.

ROUTING ALGORITHMS

- The main function of the network layer is routing packets from the source machine to the destination machine. The routing algorithm is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on.
- For subnet 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 subnet uses virtual circuits internally, routing decisions are made only when a new virtual circuit is being set up. Thereafter, data packets just follow the previously-established route. (session routing)

Routing and forwarding

- Routing makes decision of which routes to use. It is responsible for filling in and updating the routing tables.
- Forwarding handles each packet as it arrives, looking up the outgoing line to use for it in the routing tables.

Desirable properties in a routing algorithm

- **Correctness:** The routing should be done properly and correctly so that the packets may reach their proper destination.
- **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.
- **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.
- **Stability:** The routing algorithms should be stable under all possible circumstances.
- **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.
- **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.

Conflict between fairness and optimality

In Fig. 5-4, suppose that there is enough traffic between A and A', between B and B', and between C and C' to saturate the horizontal links. To maximize the total flow, the X to X' traffic should be shut off altogether. Unfortunately, X and X' may not see it that way

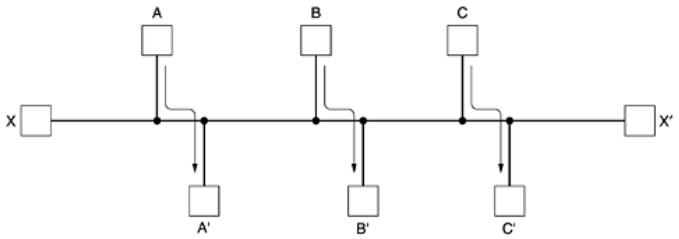
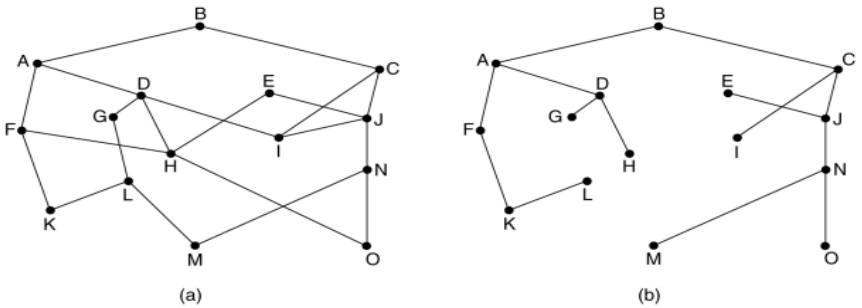


Figure 5-4. Conflict between fairness and optimality

The Optimality Principle

- A general statement can make 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.



(a) A subnet. (b) A sink tree for router B.

- **Sink tree:** Set of optimal routes from all sources to a given destination form a tree rooted at the destination; not necessarily unique; no loops. The goal of all routing algorithms is to discover and use the sink trees for all routers
- The optimality principle and the sink tree provide a benchmark against which other routing algorithms can be measured

Classification of Routing Algorithms

Routing algorithms can be grouped into two major classes - Non adaptive algorithms and Adaptive algorithms

Non adaptive algorithms (Static routing)

- Nonadaptive algorithms do not base their routing decisions on measurements or estimates of the current traffic and topology. Instead, the choice of the route to use is computed in advance, off-line, and downloaded to the routers when the network is booted.
- Eg: Shortest Path Routing and Flooding

Adaptive algorithms

- Adaptive algorithms, change their routing decisions to reflect changes in the topology, and usually the traffic as well.
- Adaptive algorithms differ in where they get their information (e.g., locally, from adjacent routers, or from all routers), when they change the routes (e.g., when the load changes or when the topology changes), and what metric is used for optimization (e.g., distance, number of hops, or estimated transit time).
- Eg. Distance Vector Routing and Link State Routing.

SHORTEST PATH ROUTING

Shortest path algorithm finds the shortest paths between routers(nodes) in a graph. The widely used shortest path algorithm is Dijkstra's shortest path algorithm.

- The idea is to build a graph of the subnet, with each node of the graph representing a router and each arc of the graph representing a communication line (often called a link).
- To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.

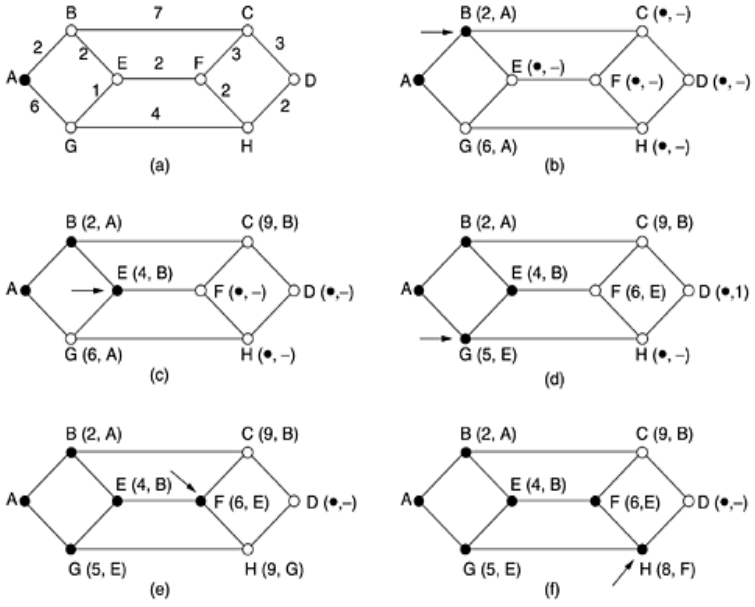


Figure 5-7. The first five steps used in computing the shortest path from A to D. The arrows indicate the working node

- The labels on the arcs could be computed as a function of the distance, bandwidth, average traffic, communication cost, mean queue length, measured delay, and other factors.
- By changing the weighting function, the algorithm would then compute the "shortest" path measured according to any one of a number of criteria or to a combination of criteria.
- An algorithm for computing the shortest path between two nodes of a graph is Dijkstra.

Working of Algorithm

- Fig. 5-7(a) contains a weighted, undirected graph, where the weights represent is distance.
- To find the shortest path from A to D, start out by marking node A as permanent, indicated by a filled-in circle.
- 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 reconstruct 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 Fig. 5-7(b). This one becomes the new working node.
- We now start at B and examine all nodes adjacent to it. If the sum of the label on B and the distance from B to the node being considered is less than the label on that node, we have a shorter path, so the node is 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. Figure 5-7 shows the first five steps of the algorithm.

FLOODING

- In flooding, every incoming packet is sent out on every outgoing line except the one it arrived on.
- Flooding generates vast numbers of duplicate packets.

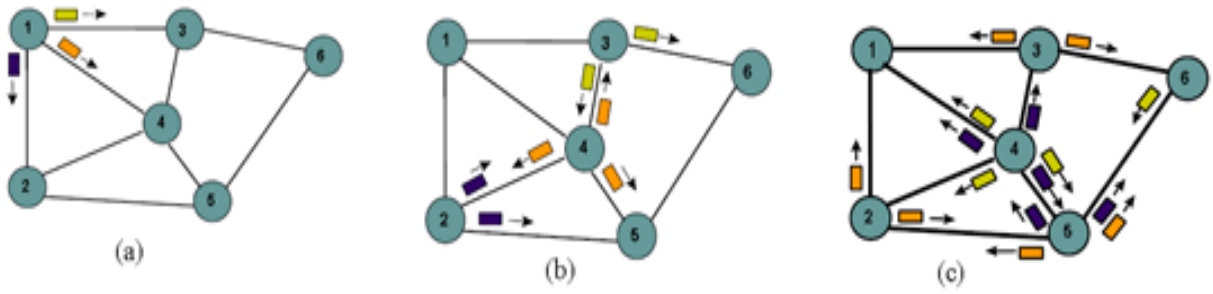


Figure: Flooding is initiated from Node 1: (a) Hop 1 transmissions (b) Hop 2 transmissions (c) Hop 3 transmissions

- To avoid a packet keep in a network forever, add a hop counter 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.
- An alternative technique for controlling the flood is to keep track of which packets have been flooded, to avoid sending them out a second time. 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 through k have been seen. When a packet comes in, it is easy to check if the packet is a duplicate; if so, it is discarded. Furthermore, the full list below k is not needed, since k effectively summarizes it.
- 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, but it does have some uses.
    - Military applications, where large numbers of routers may be blown to bits at any instant, 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, which is, in fact, flooding, and some algorithms utilize this property.
    - A metric against which other routing algorithms can be compared.

DISTANCE VECTOR ROUTING

- It is also called **Bellman-Ford routing algorithm**.
- Distance vector routing operates by having each router maintain a table (i.e. a vector) 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 distance vector routing algorithm passes periodic copies of a routing table from router to router. These regular updates between routers communicate topology changes.
- These tables are updated by exchanging information with the neighbors.
- The router is assumed to know the "distance" to each of its neighbors. If the metric is hops, the distance is just one hop. Metric may be delay time.
- Once every T msec each router sends to each neighbor a list of its estimated delays to each destination. It also receives a similar list from each neighbor.
- 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.

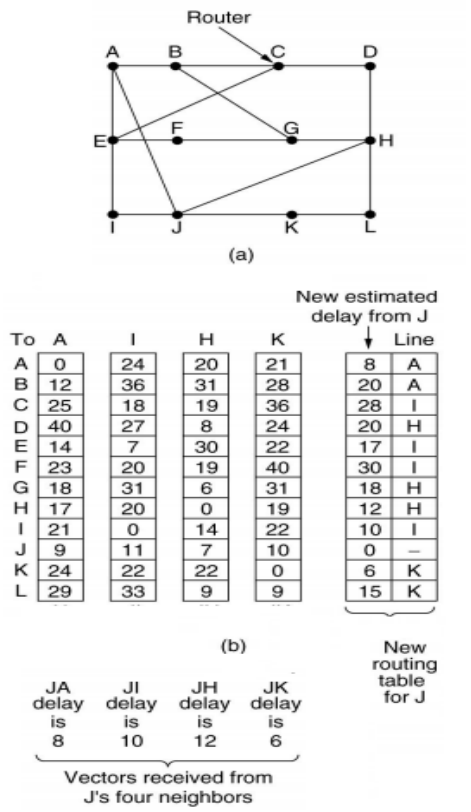


Fig. 5-9: Updating process is illustrated. Part (a) shows a subnet. The first four columns of part (b) show the delay vectors received from the neighbors of router J (Input from A, I, H, K) and the new routing table for J.

- In Fig. 5-9(b)- 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.

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.

Advantage

- Their chief advantage is the simplicity of this algorithm.

Disadvantage

- The primary drawback of this algorithm is its vulnerability to the 'Count-to-Infinity' problem. Many partial solutions have been proposed but none works under all circumstances.
- It does not take into account link bandwidth.
- It takes longer time for convergence as network size grows.

Count to infinity problem

- Counting to infinity is just another name for a routing loop. It is an important issue in Distance Vector Routing.
- In distance vector routing, routing loops usually occur when an interface goes down.
- It can also occur when two routers send updates to each other at the same time.
- Distance vector routing reacts rapidly to good news, but leisurely to bad news.

- Consider a router whose best route to destination X is large. If on the next exchange neighbor A suddenly reports a short delay to X, the router just switches over to using the line to A to send traffic to X. In one vector exchange, the good news is processed.
- Consider the five-node (linear) subnet of Fig. 5-10, where the delay metric is the number of hops. Suppose A is down initially and all the other routers know this and they have all recorded the delay to A as infinity.
- Fig. 5-10 (a) shows how fast good news propagates. 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. 5-10(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.

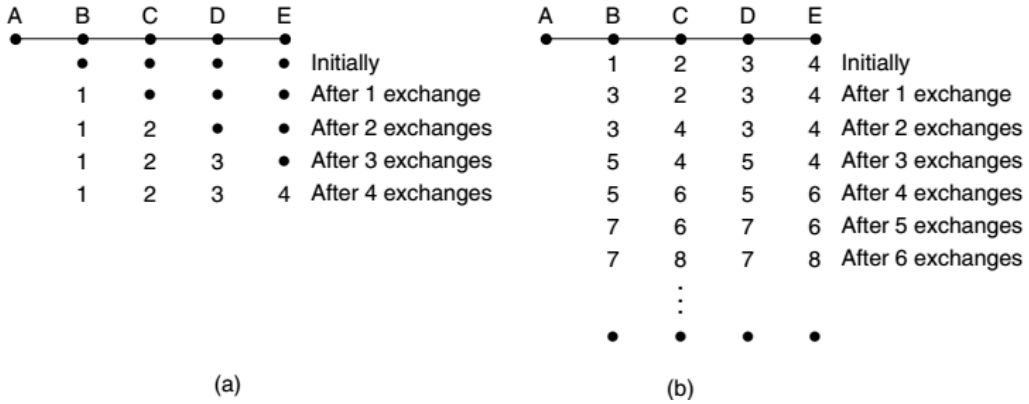


Fig. 5-10 (a) shows how fast good news propagates (b) Count to Infinity Problem

- In Fig. 5-10(b), 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.
- At the first packet exchange, B does not hear anything from A. C informs A that it have a path to A of length 2. For all B knows, C might have another line 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 them at random and makes its new distance to A 4, as shown in the third row of Fig. 5-10(b).
- Subsequent exchanges produce the history shown in the rest of Fig. 5-10(b).
- From this figure, it should be clear why bad news travels slowly: no router ever has a value more than one higher than the minimum of all its neighbors. Gradually, all routers work their way up to infinity, but the number of exchanges required depends on the numerical value used for infinity.

- For this reason, it is wise to set infinity to the longest path plus 1. If the metric is time delay, there is no well-defined upper bound, so a high value is needed to prevent a path with a long delay from being treated as down.
- There have been a few attempts to solve it (such as split horizon with poisoned reverse), but none of these work well in general.
- 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.

LINK STATE ROUTING

Link-state routers exchange messages to allow each router to learn the entire network topology. Based on this learned topology, each router is then able to compute its routing table by using a shortest path computation. For link-state routing, a network is modelled as a directed weighted graph. Each router is a node, and the links between routers are the edges in the graph. A positive weight is associated to each directed edge and routers use the shortest path to reach each destination.

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

i. Discover its neighbors and learn their network addresses.
ii. Measure the delay or cost to each of its neighbors.
iii. Construct a packet telling all it has just learned.
iv. Send this packet to all other routers.
v. Compute the shortest path to every other router.

In effect, the complete topology and all delays are experimentally measured and distributed to every router. Then Dijkstra's algorithm can be run to find the shortest path to every other router.

i. Learning about the Neighbors

- It accomplishes by sending a special HELLO packet on each point-to-point line.
- The router on the other end is send back a reply telling who it is.
- These names must be globally unique because when a distant router later hears that three routers are all connected to F, it is essential that it can determine whether all three mean the same F.
- When two or more routers are connected by a LAN, consider it as a node itself.

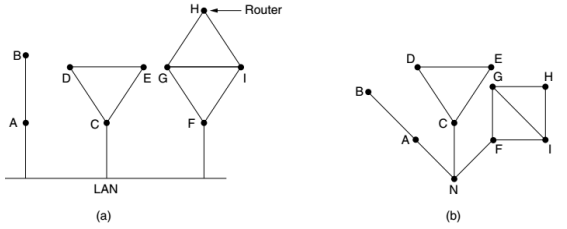


Figure 5-11. (a) Nine routers and a LAN. (b) A graph model of (a).

In Fig. 5-11(b), introduced a new, artificial node, N, to which A, C, and F are connected. The fact that it is possible to go from A to C on the LAN is represented by the path ANC here

ii. Measuring Line Cost

- The link state routing algorithm requires each router to know (or 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.

iii. Building Link State Packets

- Each router builds a packet containing the required data it collected from neighbors.
- The packet starts with the identity of the sender, followed by a sequence number and age and a list of neighbors. For each neighbor, the delay to that neighbor is given.
- An example subnet is given in Fig. 5-12(a) with delays shown as labels on the lines. The corresponding link state packets for all six routers are shown in Fig. 5-12(b).

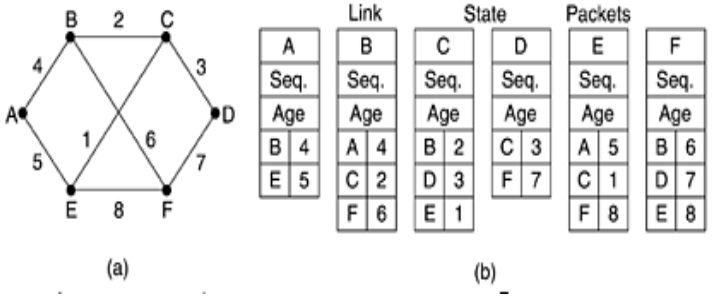


Figure 5-12. (a) A subnet. (b) The link state packets for this subnet

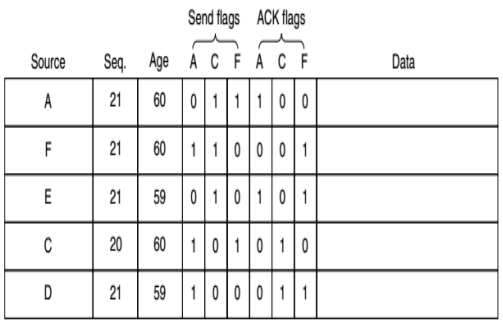
iv. Distributing the Link State Packets

- 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.

Problems

- If the sequence numbers wrap around. The solution here is to use a 32-bit sequence number. With one link state packet per second, it would take 137 years to wrap around.
- 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.
- If a sequence number is ever corrupted and 65,540 is received instead of 4 (a 1-bit error), packets 5 through 65,540 will be rejected as obsolete, since the current sequence number is thought to be 65,540.

- The 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
- The data structure used by router B for the subnet shown in Fig. 5-12(a) is depicted in Fig. 5-13.



Source	Seq.	Age	Send flags A	Send flags C	Send flags F	ACK flags A	ACK flags C	ACK flags F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Figure 5-13. The packet buffer for router B in Fig. 5-12

- Each row here corresponds to a recently-arrived, but as yet not fully-processed, link state packet.
- The table records where the packet originated, its sequence number and age, and the data.
- In addition, there are send and acknowledgement flags for each of B's three lines (to A, C, and F, respectively).
- The send flags mean that the packet must be sent on the indicated line.
- The acknowledgement flags mean that it must be acknowledged there

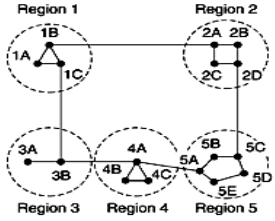
v. Computing the New Routes

- Once a router has accumulated a full set of link state packets, it can construct the entire subnet graph because every link is represented.
- Every link is, in fact, represented twice, once for each direction. The two values can be averaged or used separately.
- Dijkstra's algorithm is used 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.
- Link state routing is widely used in actual networks. The OSPF protocol, which is widely used in the Internet, uses a link state algorithm.

HIERARCHICAL ROUTING

- As networks grow in size, the router routing tables grow proportionally.
- Router memory consumed by ever-increasing tables, more CPU time is needed to scan them, more bandwidth is needed to send status reports
- 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 regions, with each router knowing all the details about how to route packets to destinations within its own region, but knowing nothing about the internal structure of other regions.
- When different networks are interconnected, consider each one as a separate region in order to free the routers in one network from having to know the topological structure of the other ones.

- Figure 5-14 gives an example of routing in a two-level hierarchy with five regions. The full routing table for router 1A has 17 entries, as shown in Fig. 5-14(b). When routing is done hierarchically, as in Fig. 5-14(c), there are entries for all the local routers as before, but all other regions have been condensed into a single router, so all traffic for region 2 goes via the 1B -2A line, but the rest of the remote traffic goes via the 1C -3B line.
- Hierarchical routing has reduced the table from 17 to 7 entries. As the ratio of the number of regions to the number of routers per region grows, the savings in table space increase.



Full table for 1A

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

(b)

Hierarchical table for 1A

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

(c)

Figure 5-14. Hierarchical routing

- **Disadvantage:** Routes are not optimal. For example, the best route from 1A to 5C is via region 2, but with hierarchical routing all traffic to region 5 goes via region 3, because that is better for most destinations in region 5.

BROADCAST ROUTING

Broadcasting refers to transmitting a packet that will be received by every device on the network. For example, a service distributing weather reports, stock market updates, or live radio programs.

Methods for Broadcasting

a. The source to simply send a distinct packet to each destination.
b. Flooding
c. Multidestination routing
d. Spanning tree based algorithm
e. Reverse path forwarding

a. The source to simply send a distinct packet to each destination

- One broadcasting method that requires no special features from the subnet is for the source to simply send a distinct packet to each destination.
- Not only is the method wasteful of bandwidth, but it also requires the source to have a complete list of all destinations.

b. Flooding

- Flooding is used especially if none of the methods described below are applicable.
- The problem with flooding as a broadcast technique is the same problem it has as a point-to-point routing algorithm: it generates too many packets and consumes too much bandwidth.

c. Multidestination routing

- If this method, each packet contains either a list of destinations or a bit map indicating the desired destinations.
- When a packet arrives at a router, the router checks all the destinations to determine the set of output lines that will be needed. (An output line is needed if it is the best route to at least one of the destinations.)
- The router generates a new copy of the packet for each output line to be used and includes in each packet only those destinations that are to use the line.
- In effect, the destination set is partitioned among the output lines. After a sufficient number of hops, each packet will carry only one destination and can be treated as a normal packet.
- Multidestination routing is like separately addressed packets, except that when several packets must follow the same route, one of them pays full fare and the rest ride free.

d. Spanning tree based algorithm

- Using a spanning tree (basically a sink tree) for the router initiating the broadcast.
- A spanning tree is a subset of the subnet that includes all the routers but contains no loops. If each router knows which of its lines belong to the spanning tree, it can copy an incoming broadcast packet onto all the spanning tree lines except the one it arrived on.
- This method makes excellent use of bandwidth, generating the absolute minimum number of packets necessary to do the job.
- The only problem is that each router must have knowledge of some spanning tree for the method to be applicable. Sometimes this information is available (e.g., with link state routing) but sometimes it is not (e.g., with distance vector routing).

e. Reverse path forwarding

- When a broadcast packet arrives at a router, the router checks to see if the packet arrived on the line that is normally used for sending packets to the source of the broadcast.
- If so, there is an excellent chance that the broadcast packet itself followed the best route from the router and is therefore the first copy to arrive at the router. This being the case, the router forwards copies of it onto all lines except the one it arrived on.
- If, the broadcast packet arrived on a line other than the preferred one for reaching the source, the packet is discarded as a likely duplicate.

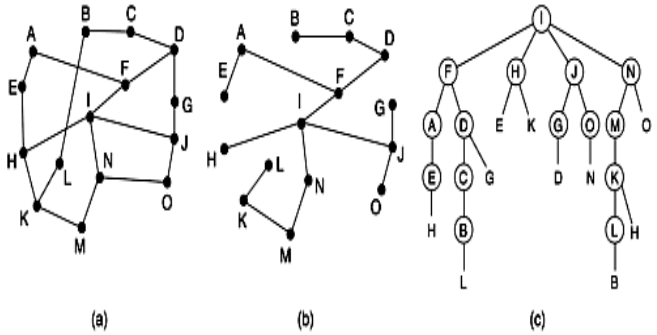


Figure 5-16. Reverse path forwarding. (a) A subnet. (b) A sink tree. (c) The tree built by reverse path forwarding

- The advantage of reverse path forwarding is that it is both reasonably efficient and easy to implement. It does not require routers to know about spanning trees, nor does it have the overhead of a destination list or bit map in each broadcast packet as does multidestination addressing. It does not require any special mechanism to stop the process, as flooding does.

MULTICAST ROUTING

- Multicast Routing are used to distribute data (for example, audio/video streaming broadcasts) to multiple recipients.
- Using multicast, a source can send a single copy of data to a single multicast address, which is then distributed to an entire group of recipients.
- Sending a message to a group is called multicasting, and its routing algorithm is called multicast routing.
- Multicasting requires group management to create and destroy groups, and to allow processes to join and leave groups.
- To do multicast routing, each router computes a spanning tree covering all other routers. For example, in Fig. 5-17(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. 5-17(b).

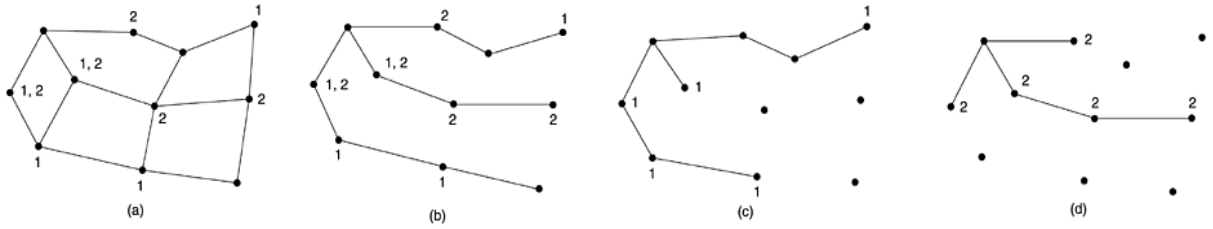


Figure 5-17. (a) A network. (b) A spanning tree for the leftmost router. (c) A multicast tree for group 1. (d) A multicast tree for group 2.

- When a process sends a multicast packet to a group, the first router examines its spanning tree and removes all lines that do not lead to group members (pruning). In Fig. 5-17(c) shows the pruned spanning tree for group 1. Similarly, Fig. 5-17(d) shows the pruned spanning tree for group 2. Multicast packets are forwarded only along the appropriate spanning tree.
- One 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. 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.

CONGESTION CONTROL ALGORITHMS

Congestion is a situation in Communication Networks in which too many packets are present in a part of the subnet, performance degrades. Congestion in a network may occur when the load on the network (i.e. the number of packets sent to the network) is greater than the capacity of the network (i.e. the number of packets a network can handle.)

When the number of packets dumped into the subnet by the hosts is within its carrying capacity, they are all delivered (except for a few that are afflicted with transmission errors) and the number delivered is proportional to the number sent.

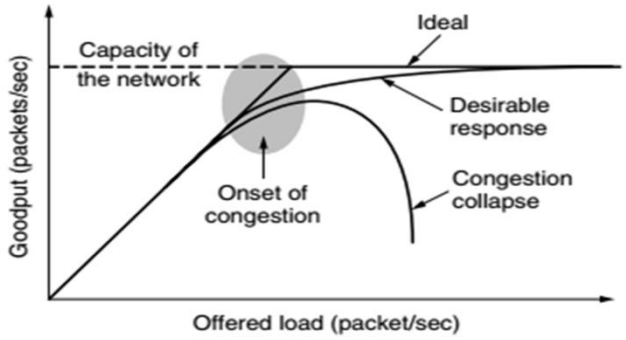


Figure 5-18. When too much traffic is offered, congestion sets in and performance degrades sharply.

As traffic increases too far, the routers are no longer able to cope and they begin losing packets. This tends to make matters worse. At very high trafffic, performance collapses completely and almost no packets are delivered.

Cause of Congestion:

The various causes of congestion in a subnet are:

1. The input traffic rate exceeds the capacity of the output lines. If suddenly, a stream of packet start arriving on three or four input lines and all need the same output line. In this case, a queue will be built up. If there is insufficient memory to hold all the packets, the packet will be lost. Increasing the memory to unlimited size does not solve the problem. This is because, by the time packets reach front of the queue, they have already timed out (as they waited the queue). When timer goes off source transmits duplicate packet that are also added to the queue. Thus same packets are added again and again, increasing the load all the way to the destination.
2. The routers are too slow to perform bookkeeping tasks (queuing buffers, updating tables, etc.)
3. The routers' buffer is too limited.
4. Congestion in a subnet can occur if the processors are slow. Slow speed CPU at routers will perform the routine tasks such as queuing buffers, updating table etc slowly. As a result of this, queues are built up even though there is excess line capacity.
5. Congestion is also caused by slow links. This problem will be solved when high speed links are used. But it is not always the case. Sometimes increase in link bandwidth can further deteriorate the congestion problem as higher speed links may make the network more unbalanced. Congestion can make itself worse. If a route!" does not have free buffers, it start ignoring/discarding the newly arriving packets. When these packets are discarded, the sender may retransmit them after the timer goes off. Such packets are transmitted by the sender again and again until the source gets the acknowledgement of these packets. Therefore multiple transmissions of packets will force the congestion to take place at the sending end.

General Principles of Congestion Control

On control theory point of view divide all solutions into two groups: open loop and closed loop.

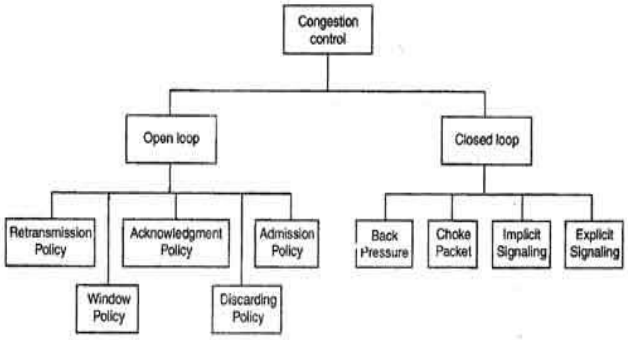
Open loop solutions

- Open loop solutions attempt to solve the problem by good design
- In this method, policies are used to prevent the congestion before it happens.
- Congestion control is handled either by the source or by the destination.
- Deciding when to accept new traffic
- Deciding when to discard packets and which ones, and
- Making scheduling decisions at various points in the network.
- All of these decisions are without regard to the current state of the network

Closed loop

- Closed loop congestion control mechanisms try to remove the congestion after it happens.
- Closed loop solutions are based on the concept of a feedback loop
- ***This approach has three parts when applied to congestion control:***

1. Monitor the system to detect when and where congestion occurs.
2. Pass this information to places where action can be taken.
3. Adjust system operation to correct the problem.



Types of Congestion Control Methods

Congestion Prevention Policies

Open loop systems are designed to minimize congestion in the first place, rather than letting it happen and reacting after the fact. They try to achieve their goal by using appropriate policies at various levels; data link, network, and transport policies that can affect congestion.

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

APPROACHES TO CONGESTION CONTROL

Congestion Control refers to techniques and mechanisms that can either prevent congestion, before it happens, or remove congestion, after it has happened. Congestion control mechanisms are divided into two categories, one category prevents the congestion from happening and the other category removes congestion after it has taken place.

Congestion Control in Virtual-Circuit Subnets

i. Admission control

- 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.
- In a telephone system, when a switch gets overloaded, it also practices admission control, by not giving dial tones.
- This approach is crude, but it is simple and easy to carry out
- Admission control can also be combined with traffic-aware routing by considering routes around traffic hotspots as part of the setup procedure. For example, consider the network illustrated in Fig. 5-20(a), in which two routers are congested, as indicated.

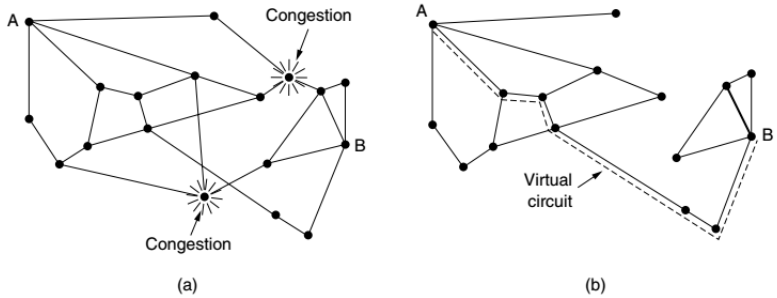


Figure 5-20. (a) A congested network. (b) The portion of the network that is not congested. A virtual circuit from *A* to *B* is also shown.

- Suppose that a host attached to router A wants to set up a connection to a host attached to router B. Normally, this connection would pass through one of the congested routers. To avoid this situation, we can redraw the network as shown in Fig. 5-20(b), omitting the congested routers and all of their lines. The dashed line shows a possible route for the virtual circuit that avoids the congested routers.

ii. Negotiating for 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 (QoS) required, and other parameters.
- To keep its part of the agreement, the subnet will reserve resources along path when the circuit is set up.
- These resources can include table and buffer space in the routers and bandwidth in the lines.
- The drawback of this method that it tends to waste resources. For example, if six virtual circuits that might use 1 Mbps all pass through the same physical 6-Mbps line, the line has to marked as full, even though it may rarely happen that all six virtual circuits are transmitting at the same time.

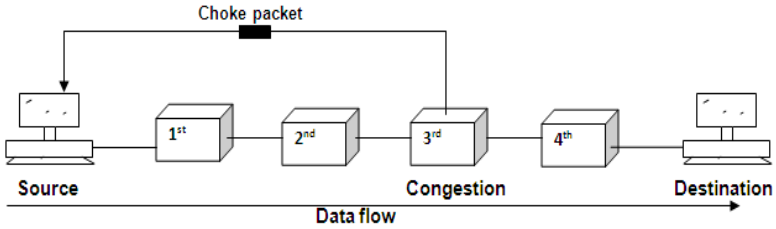
Congestion Control in Datagram Subnets

i. The Warning Bit

- A special bit in the packet header is set by the router to warn the source when congestion is detected.
- When the packet arrived at its destination, the transport entity copied the bit into the next acknowledgement sent back to the source. The source then cut back on traffic.
- As long as the router was in the warning state, it continued to set the warning bit, which meant that the source continued to get acknowledgements with it set.
- The source monitored the fraction of acknowledgements with the bit set and adjusted its transmission rate accordingly. As long as the warning bits continued to flow in, the source continued to decrease its transmission rate. When they slowed to a trickle, it increased its transmission rate.
- Note that since every router along the path could set the warning bit, traffic increased only when no router was in trouble.

ii. Choke Packets

- **Choke packets** are used in both virtual circuit and datagram subnets.
- A specialized packet that is used for flow control along a network.
- A router detects congestion by measuring the percentage of buffers in use, line utilization and average queue lengths.
- When it detects congestion, it sends choke packets across the network to all the data sources associated with the congestion.
- 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.
- In choke packet method, congested node sends a warning directly to the source station i.e. the intermediate nodes through which the packet has traveled are not warned.



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

iii. 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.
- An alternative approach is to have the choke packet take effect at every hop it passes through, as shown in the sequence of Fig. 5-22(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. In this way, congestion can be nipped in the bud without losing any packets.

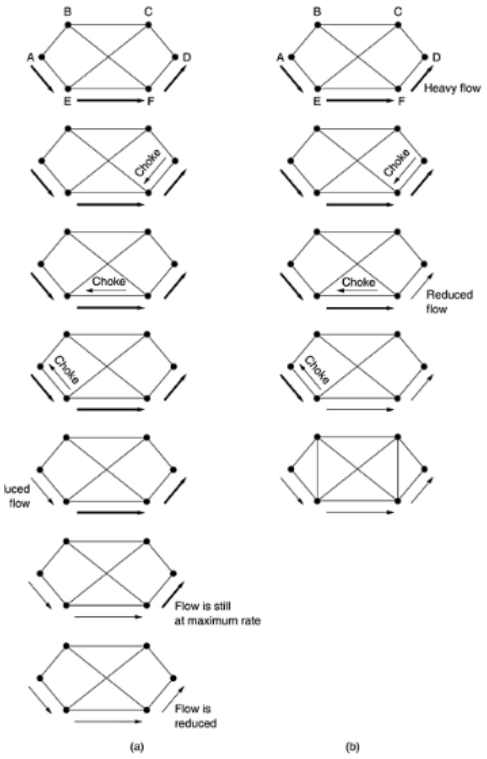


Figure 5-22. (a) A choke packet that affects only the source. (b) A choke packet that affects each hop it passes through

iv. Load Shedding

- The basic idea of **load shedding** scheme is that when routers have a lot of packets that they cannot handle, they just throw them away.
- The term comes from the world of electrical power generation, where it refers to the practice of utilities intentionally blacking out certain areas to save the entire grid from collapsing on hot summer days when the demand for electricity greatly exceeds the supply.
- A router drowning in packets can just pick packets at random to drop, may depend on the applications running.
- For file transfer, an old packet is worth more than a new one because dropping packet 6 and keeping packets 7 through 10 will cause a gap at the receiver that may force packets 6 through 10 to be retransmitted (if the receiver routinely discards out-of-order packets). In a 12-packet file, dropping 6 may require 7 through 12 to be retransmitted, whereas dropping 10 may require only 10 through 12 to be retransmitted. In contrast, for multimedia, a new packet is more important than an old one. If transmitting a document containing ASCII text and pictures. Losing a line of pixels in some image is far less damaging than losing a line of readable text.

- To implement an intelligent discard policy, applications must mark their packets in priority classes to indicate how important they are. If they do this, then when packets have to be discarded, routers can first drop packets from the lowest class, then the next lowest class, and so on.

v. Random Early Detection

- Random early detection (RED is a queueing discipline for a network scheduler suited for congestion avoidance
- RED monitors the average queue size and drops (or marks when used in conjunction with ECN) packets based on statistical probabilities.
- If the buffer is almost empty, then all incoming packets are accepted.
- As the queue grows, the probability for dropping an incoming packet grows too.
- When the buffer is full, the probability has reached 1 and all incoming packets are dropped.

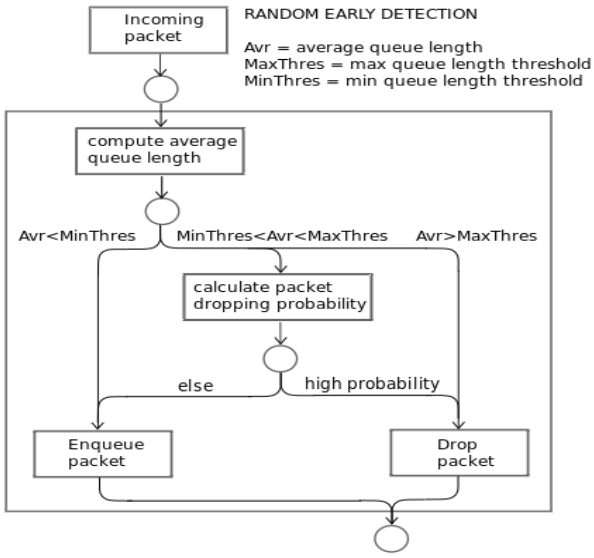


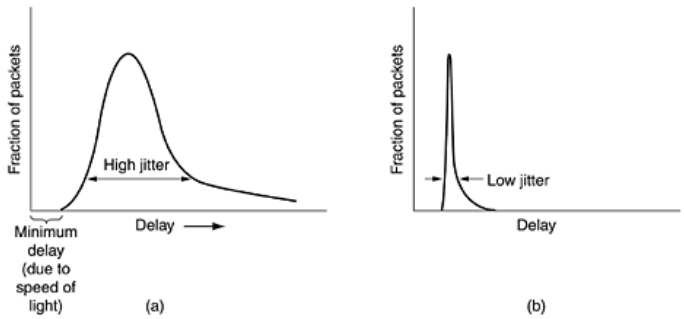
Fig. 5.23 Random Early Detection

- RED is more fair than tail drop, in the sense that it does not possess a bias against bursty traffic that uses only a small portion of the bandwidth.
- The more a host transmits, the more likely it is that its packets are dropped as the probability of a host's packet being dropped is proportional to the amount of data it has in a queue.
- Early detection helps avoid TCP global synchronization.

vi. Jitter Control

- Jitter is the amount of variation (i.e., standard deviation) in the packet arrival time.
- For applications such as audio and video streaming, it does not matter much if the packets take 20 msec or 30 msec to be delivered, as long as the transit time is constant.

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



- The jitter can be bounded by computing the expected transit time for each hop along the path. When a packet arrives at a router, the router checks to see how much the packet is behind or ahead of its schedule. The information is stored in the packet and updated at each hop.
- If the packet is ahead of schedule, it is held long enough to get it back on schedule.
- If the packet is behind schedule, the router tries to get it out quickly. The algorithm for determining which of several packets competing for an output line should go next can always choose the packet furthest behind in its schedule.

QUALITY OF SERVICE REQUIREMENTS

With the growth of multimedia networking, attempts at guaranteeing quality of service through network and protocol design are needed.

- 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.

Common applications and the stringency of their requirements are listed below.

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. If all the words in a telephone call are each delayed by exactly 2.000

seconds, the users will find the connection unacceptable. On the other hand, playing audio or video files from a server does not require low delay.

- 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. If a user is watching a video over the network and the frames are all delayed by exactly 2.000 seconds, no harm is done. But if the transmission time varies randomly between 1 and 2 seconds, the result will be terrible. For audio, a jitter of even a few milliseconds is clearly audible.

Techniques for Achieving Good Quality of Service

No single technique provides efficient, dependable QoS in an optimum way. Instead, a variety of techniques have been developed, with practical solutions often combining multiple techniques. Following are some of the techniques system designers use to achieve QoS.

i. Overprovisioning
ii. Buffering
iii. Traffic Shaping
iv. The Leaky Bucket Algorithm
v. The Token Bucket Algorithm
vi. Resource Reservation
vii. Admission Control
viii. Proportional Routing
ix. Packet Scheduling

Buffering

- Flows can be buffered on the receiving side before being delivered.
- It smooths out the jitter.
- For audio and video on demand, jitter is the main problem, so this technique helps a lot.

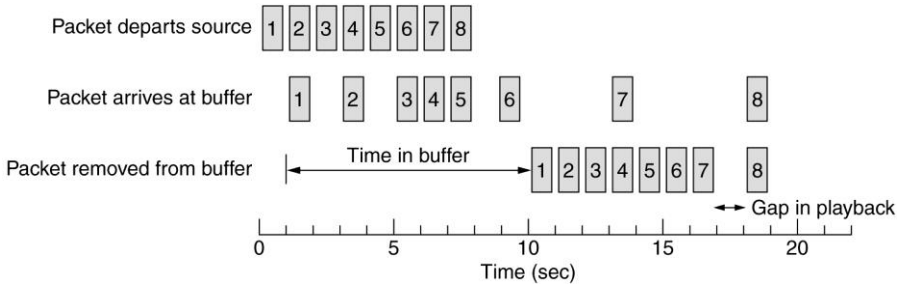


Figure 5-25. Smoothing the output stream by buffering packets

- In Figure 5-25, At t = 10 sec, playback begins. At this time, packets 1 through 6 have been buffered so that they can be removed from the buffer at uniform intervals for smooth play. Unfortunately, packet 8 has been delayed so much that it is not available when its play slot

comes up, so playback must stop until it arrives, creating an annoying gap in the music or movie.

- This problem can be alleviated by delaying the starting time even more, although doing so also requires a larger buffer. Commercial Web sites that contain streaming audio or video all use players that buffer for about 10 seconds before starting to play.

Traffic Shaping

- Traffic in data networks is bursty. It typically arrives at nonuniform rates as the traffic rate varies. e.g., videoconferencing with compression), browsing a new Web page, and computers switch between tasks.
- Traffic Shaping smooth 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. This is called a service level agreement.
- As long as the customer fulfills her part of the bargain and only sends packets according to the agreed-on contract, the carrier promises to deliver them all in a timely fashion. Traffic shaping reduces congestion and thus helps the carrier live up to its promise. 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**.

The Leaky Bucket Algorithm

- It is a single-server queueing system with constant service time.
- Imagine a bucket with a small hole in the bottom, as illustrated in Fig. 5-26(a).
- No matter the rate at which water enters the bucket, the outflow is at a constant rate, r, 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).
- The same idea can be applied to packets, as shown in Fig. 5-26(b). Conceptually, 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.
- This arrangement can be built into the hardware interface or simulated by the host operating system.

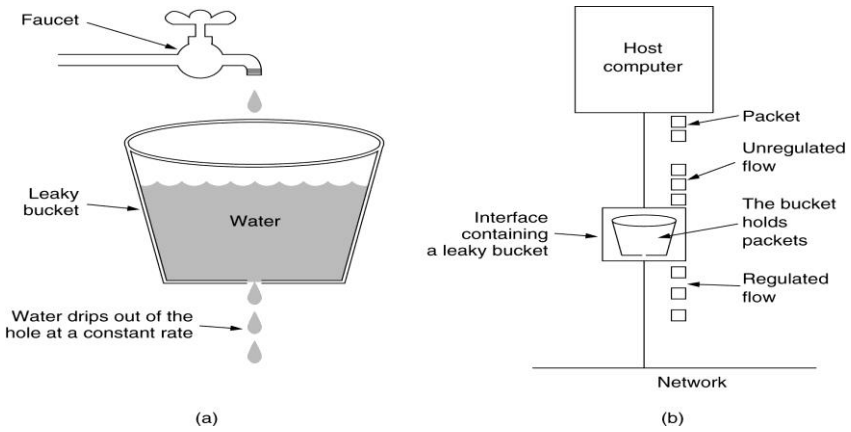


Figure 5-26. (a) A leaky bucket with water. (b) A leaky bucket with packets

- Implementing the original leaky bucket algorithm is easy. The leaky bucket consists of a finite queue. When a packet arrives, if there is room on the queue it is appended to the queue; otherwise, it is discarded. At every clock tick, one packet is transmitted.
- 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.
- In Fig. 5-27(a) we see the input to the leaky bucket running at 25 MB/sec for 40 msec. In Fig. 5-27(b) we see the output draining out at a uniform rate of 2 MB/sec for 500 msec.
- The leaky bucket algorithm enforces a rigid output pattern at the average rate, no matter how bursty the traffic is.
- The leaky bucket algorithm does not allow idle hosts to save up permission to send large bursts later

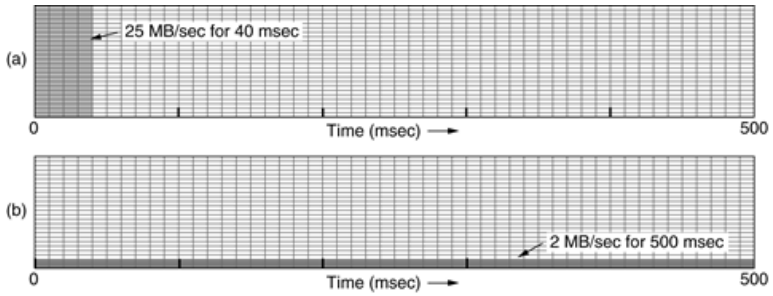


Figure 5-27. (a) Input to a leaky bucket. (b) Output from a leaky bucket

The Token Bucket Algorithm

- Token Bucket Algorithm allows the output to speed up somewhat when large bursts arrive, to avoid loses data.
- In this algorithm, the leaky bucket holds tokens, generated by a clock at the rate of one token every ΔT sec.

- In Fig. 5-28(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. 5-34(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.

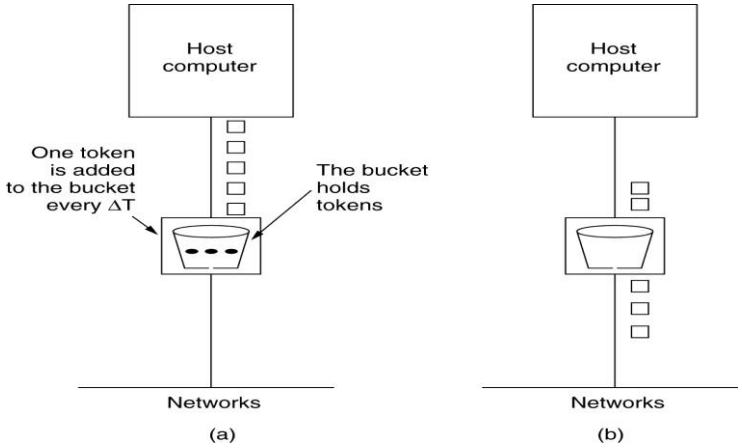


Figure 5-28. The token bucket algorithm. (a) Before. (b) After.

- The token bucket algorithm 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.
- Token bucket algorithm throws away tokens (i.e., transmission capacity) when the bucket fills up but never discards packets.
- 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 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.
- Token bucket allow bursts, up to a regulated maximum length. Figure below shows, a token bucket with a capacity of 250 KB. Tokens arrive at a rate allowing output at 2 MB/sec. Assuming the token bucket is full when the 1-MB burst arrives, the bucket can drain at the full 25 MB/sec for about 11 msec. Then it has to cut back to 2 MB/sec until the entire input burst has been sent.