UNIT – 5
NETWORK LAYER

NETWORK LAYER DESIGN ISSUES

In the following sections we will provide an introduction to some of the issues that the designers of the network layer must grapple with. These issues include the service provided to the transport layer and the internal design of the subnet.

STORE-AND-FORWARD PACKET SWITCHING

- The network layer protocols operation can be seen in Fig. 5-1.
- The major components of the system are the carrier's equipment (routers connected by transmission lines), shown inside the shaded oval.
- 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. Whether it belongs to the subnet is arguable, but for the purposes of this chapter, routers on customer premises are considered part of the subnet.

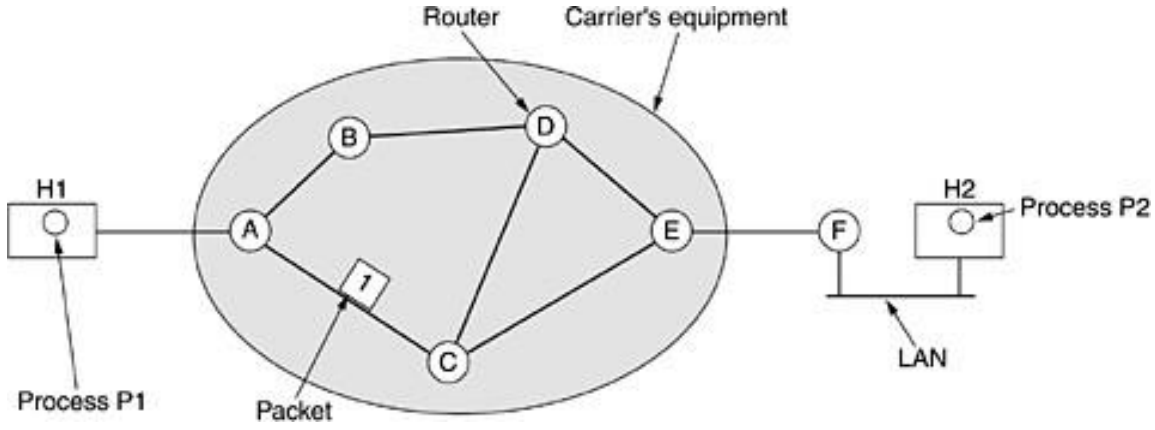


Figure 5-1. The environment of the network layer protocols.

- 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.
- This mechanism is **store-and-forward packet switching**.

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.

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.

- There is a discussion centers on whether the network layer should provide connection-oriented service or connectionless service.
- In their view (based on 30 years of actual experience with a real, working computer network), the subnet is inherently unreliable, no matter how it is designed. Therefore, the hosts should accept the fact that the network is unreliable and do error control (i.e., error detection and correction) and flow control themselves.
- This viewpoint leads quickly to the conclusion that the network service should be connectionless, with primitives SEND PACKET and RECEIVE PACKET and little else.
- In particular, no packet ordering and flow control should be done, because the hosts are going to do that anyway, and there is usually little to be gained by doing it twice.
- Furthermore, each packet must carry the full destination address, because each packet sent is carried independently of its predecessors, if any.
- The other camp (represented by the telephone companies) argues that the subnet should provide a reliable, connection-oriented service.
- These two camps are best exemplified by the Internet and ATM. The Internet offers connectionless network-layer service; ATM networks offer connection-oriented network-layer service. However, it is interesting to note that as quality-of-service guarantees are becoming more and more important, the Internet is evolving. In particular, it is starting to acquire properties normally associated with connection-oriented service, as we will see later.

IMPLEMENTATION OF CONNECTIONLESS SERVICE

- Two different organizations are possible, depending on the type of service offered.
- If connectionless service is offered, packets are injected into the subnet individually and routed independently of each other. No advance setup is needed. In this context, the packets are frequently called **datagrams** (in analogy with telegrams) and the subnet is called a **datagram subnet**.
- If connection-oriented service is used, 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**), in analogy with the physical circuits set up by the telephone system, and the subnet is called a **virtual-circuit subnet**.
- Let us now see how a datagram subnet works. Suppose that the process *P1* in **Fig. 5-2** 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*, 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.

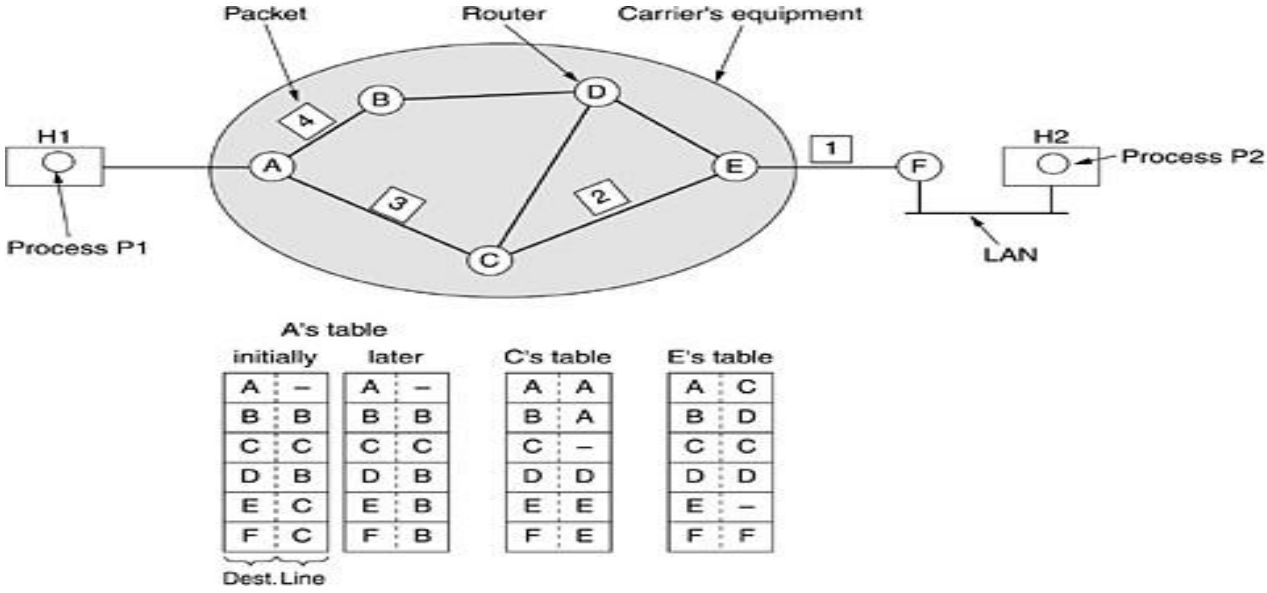


Figure 5-2. Routing within a datagram subnet

- Let us 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,
- For example, PPP. At this point the carrier takes over. 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.
- For example, in Fig. 5-2, *A* has only two outgoing lines—to *B* and *C*—so every incoming packet must be sent to one of these routers, even if the ultimate destination is some other router. *A*'s initial routing table is shown in the figure under the label "initially". 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. 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.
- However, something different happened to packet 4. When it got to *A* it was sent to router *B*, even though it is also destined for *F*. For some reason, *A* decided to send packet 4 via a different route than that of the first three. Perhaps it learned of a traffic jam somewhere along the *ACE* path and 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**.

IMPLEMENTATION OF CONNECTION-ORIENTED SERVICE

- For connection-oriented service, we need a virtual-circuit subnet.
- Let us see how that works.
- The idea behind virtual circuits is to avoid having to choose a new route for every packet sent, as in Fig. 5-2. Instead, when a connection is established, a route from the source machine to the destination machine is chosen as part of the connection setup and stored in tables inside the routers.
- That route is used for all traffic flowing over the connection, exactly the same way that the telephone system works.
- When the connection is released, the virtual circuit is also terminated. With connection-oriented service, each packet carries an identifier telling which virtual circuit it belongs to.
- As an example, consider the situation of **Fig. 5-3**. 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.

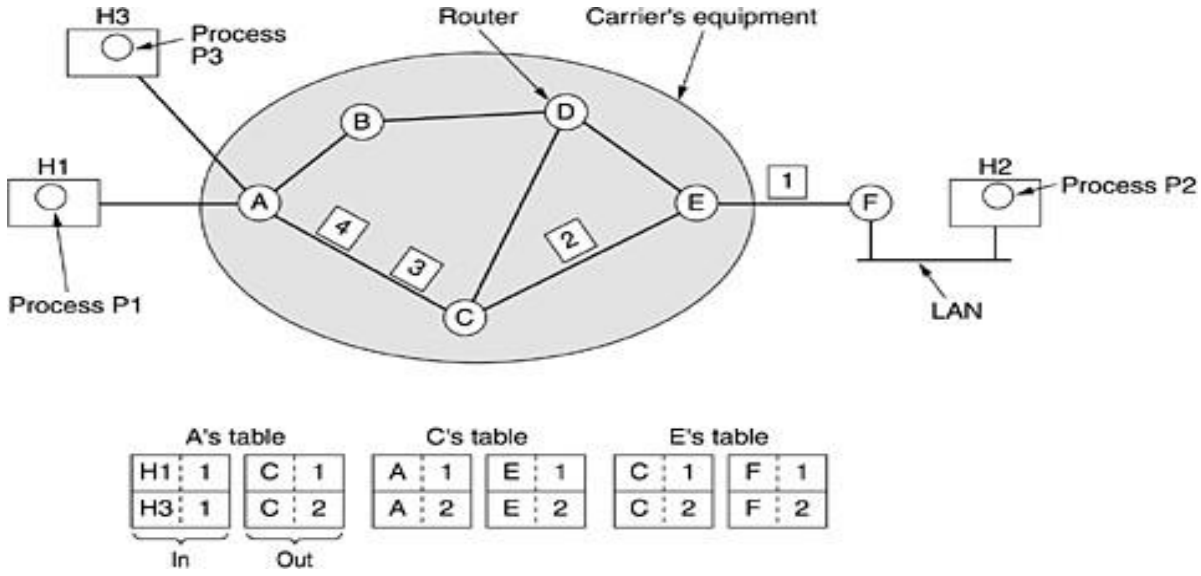


Figure 5-3. Routing within a virtual-circuit subnet.

- Now let us consider what happens if *H3* also wants to establish a connection to *H2*. It chooses connection identifier 1 (because it is initiating the connection and this is its only connection) and tells the 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. In some contexts, this is called label switching.

COMPARISON OF VIRTUAL-CIRCUIT AND DATAGRAM SUBNETS

The major issues are listed in Fig. 5-4, although purists could probably find a counter example for everything in the figure.

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

Figure 5-4. Comparison of datagram and virtual-circuit subnets.

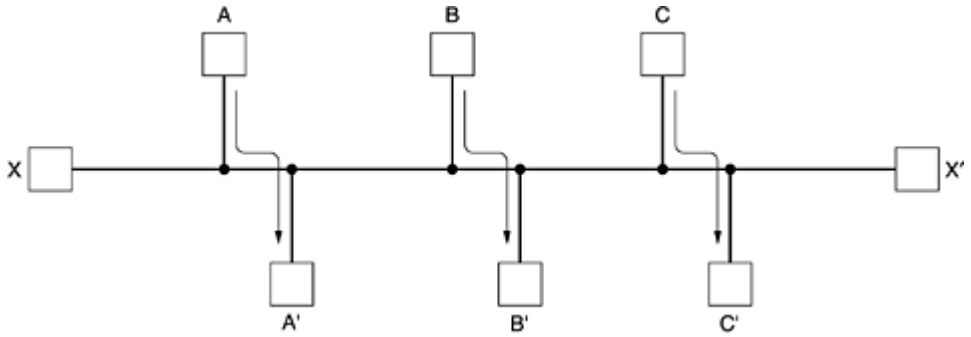
ROUTING ALGORITHMS

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

PROPERTIES OF ROUTING ALGORITHM:

Correctness, simplicity, robustness, stability, fairness, and optimality

FAIRNESS AND OPTIMALITY.



Fairness and optimality may sound obvious, but as it turns out, they are often contradictory goals. 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. Evidently, some compromise between global efficiency and fairness to individual connections is needed.

CATEGORY OF ALGORITHM

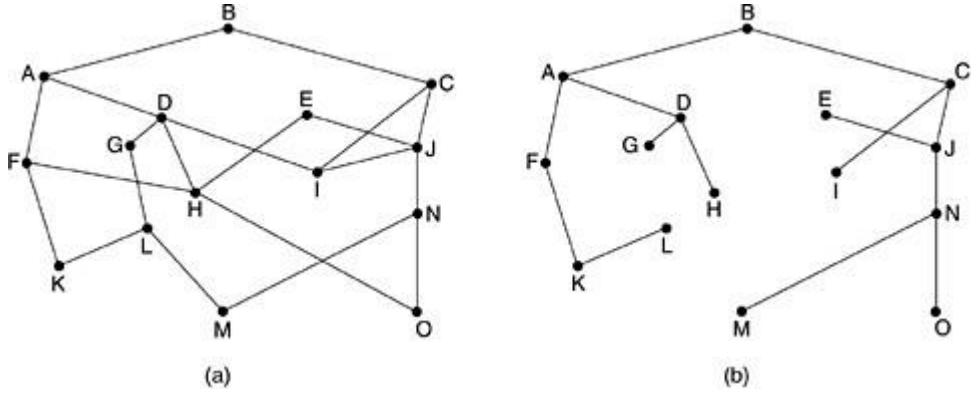
- Routing algorithms can be grouped into two major classes: **nonadaptive and adaptive.**
- **Nonadaptive algorithms** do not base their routing decisions on measurements or estimates of the current traffic and topology. Instead, the choice of the route to use to get from I to J is computed in advance, off-line, and downloaded to the routers when the network is booted.
- This procedure is sometimes called **Static routing**.

- **Adaptive algorithms**, in contrast, change their routing decisions to reflect changes in the topology, and usually the traffic as well
- This procedure is sometimes called **dynamic routing**

THE OPTIMALITY PRINCIPLE

(a) **I**f 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.

(b) 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.



***(c) A subnet. (b) A sink tree for router* B.**

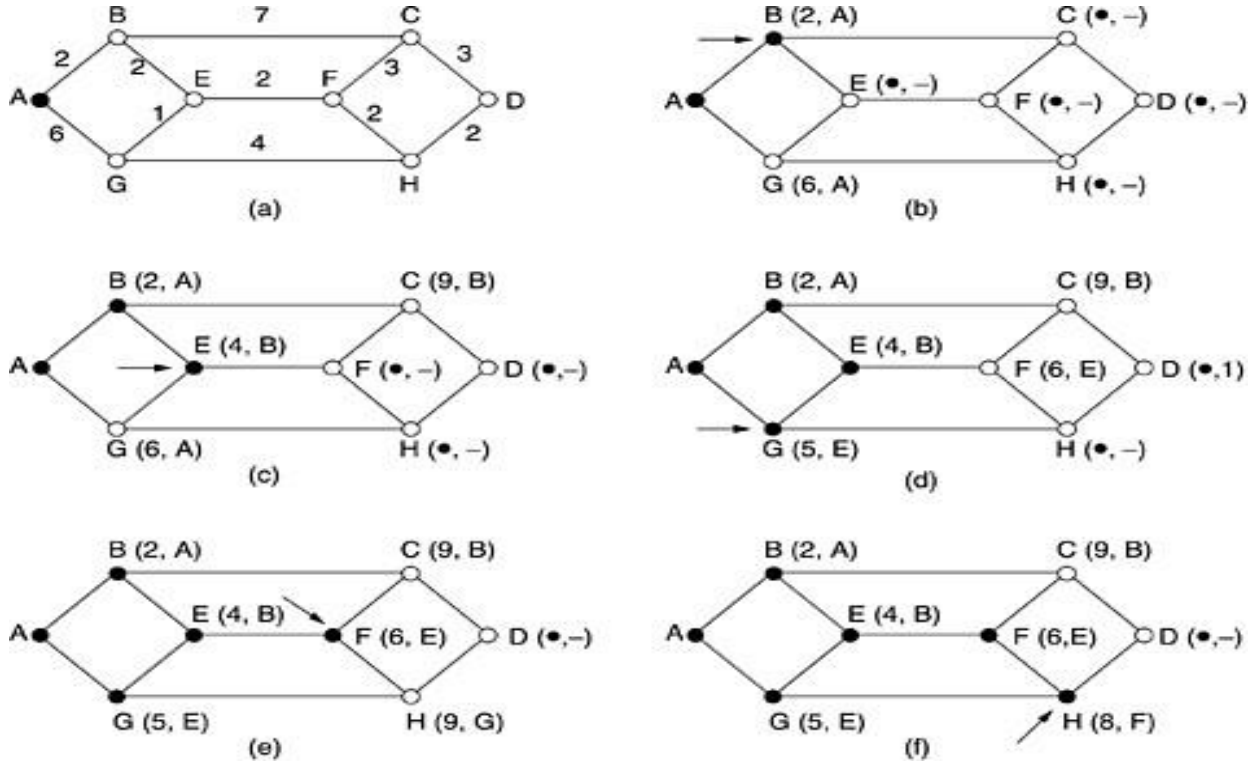
- As a direct consequence of the optimality principle, we can see that the set of optimal routes from all sources to a given destination form a tree rooted at the destination.
- Such a tree is called a **sink tree** where the distance metric is the number of hops. Note that a sink tree is not necessarily 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.

SHORTEST PATH ROUTING

- A technique to study routing algorithms: The idea is to build a graph of the subnet, with each node of the graph representing a router and each arc of the graph representing a communication line (often called a link).
- To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.
- One way of measuring path length is the number of hops. Another metric is the geographic distance in kilometers. Many other metrics are also possible. For example,

each arc could be labeled with the mean queuing and transmission delay for some standard test packet as determined by hourly test runs.

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



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

- To illustrate how the labelling algorithm works, look at the weighted, undirected graph of Fig. 5-7(a), where the weights represent, for example, distance.
- We want to find the shortest path from *A* to *D*. We start out by marking node *A* as permanent, indicated by a filled-in circle.
- Then we examine, in turn, each of the nodes adjacent to *A* (the working node), relabeling each one with the distance to *A*.
- Whenever a node is relabelled, we also label it with the node from which the probe was made so that we can reconstruct the final path later.
- Having examined each of the nodes adjacent to *A*, we examine all the tentatively labelled nodes in the whole graph and make the one with the smallest label permanent, as shown in Fig. 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 relabelled.

After all the nodes adjacent to the working node have been inspected and the tentative labels changed if possible, the entire graph is searched for the tentatively-labelled node with the smallest value. This node is made permanent and becomes the working node for the next round. Figure 5-7 shows the first five steps of the algorithm.

- To see why the algorithm works, look at Fig. 5-7(c). At that point we have just made *E* permanent. Suppose that there were a shorter path than *ABE*, say *AXYZE*. There are two possibilities: either node *Z* has already been made permanent, or it has not been. If it has, then *E* has already been probed (on the round following the one when *Z* was made permanent), so the *AXYZE* path has not escaped our attention and thus cannot be a shorter path.

- Now consider the case where *Z* is still tentatively labelled. Either the label at *Z* is greater than or equal to that at *E*, in which case *AXYZE* cannot be a shorter path than *ABE*, or it is less than that of *E*, in which case *Z* and not *E* will become permanent first, allowing *E* to be probed from *Z*.
- This algorithm is given in Fig. 5-8. The global variables *n* and *dist* describe the graph and are initialized before *shortest path* is called. The only difference between the program and the algorithm described above is that in Fig. 5-8, we compute the shortest path starting at the terminal node, *t*, rather than at the source node, *s*. Since the shortest path from *t* to *s* in an undirected graph is the same as the shortest path from *s* to *t*, it does not matter at which end we begin (unless there are several shortest paths, in which case reversing the search might discover a different one). The reason for searching backward is that each node is labelled with its predecessor rather than its successor. When the final path is copied into the output variable, *path*, the path is thus reversed. By reversing the search, the two effects cancel, and the answer is produced in the correct order.

CONGESTION CONTROL ALGORITHMS

- When too many packets are present in (a part of) the subnet, performance degrades. This situation is called **congestion**.
- Figure 5-25 depicts the symptom. When the number of packets dumped into the subnet by the hosts is within its carrying capacity, they are all delivered (except for a few that are afflicted with transmission errors) and the number delivered is proportional to the number sent.

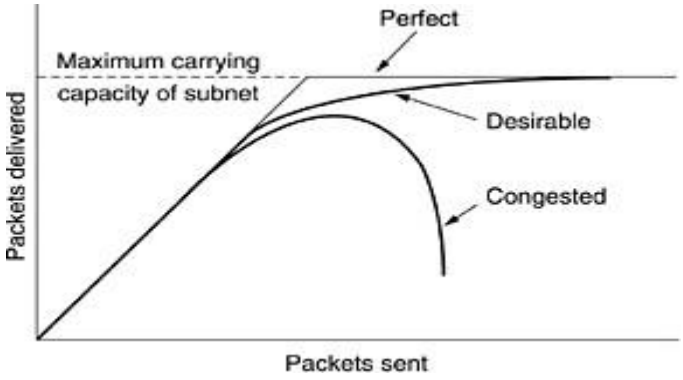


Figure 5-25. When too much traffic is offered, congestion sets in and performance degrades sharply.

- However, as traffic increases too far, the routers are no longer able to cope and they begin losing packets. This tends to make matters worse. At very high traffic, performance collapses completely and almost no packets are delivered.
- Congestion can be brought on by several factors. If all of a sudden, streams of packets begin arriving on three or four input lines and all need the same output line, a queue will build up.
- If there is insufficient memory to hold all of them, packets will be lost.
- Slow processors can also cause congestion. If the routers' CPUs are slow at performing the bookkeeping tasks required of them (queuing buffers, updating tables, etc.), queues can build up, even though there is excess line capacity. Similarly, low-bandwidth lines can also cause congestion.

APPROACHES TO CONGESTION CONTROL

- Many problems in complex systems, such as computer networks, can be viewed from a control theory point of view. This approach leads to dividing all solutions into two groups: open loop and closed loop.

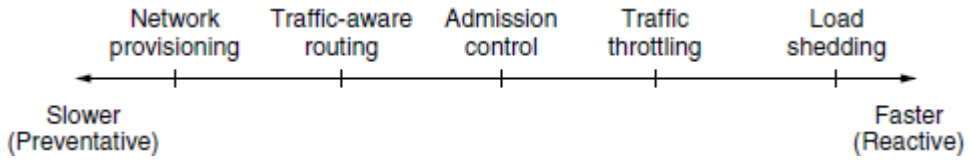


Figure: Timescales Of Approaches To Congestion Control

- Open loop solutions attempt to solve the problem by good design.
- Tools for doing open-loop control include deciding when to accept new traffic, deciding when to discard packets and which ones, and making scheduling decisions at various points in the network.
- 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.
- A variety of metrics can be used to monitor the subnet for congestion. Chief among these are the percentage of all packets discarded for lack of buffer space, the average queue lengths, the number of packets that time out and are retransmitted, the average packet delay, and the standard deviation of packet delay. In all cases, rising numbers indicate growing congestion.
- The second step in the feedback loop is to transfer the information about the congestion from the point where it is detected to the point where something can be done about it.

- In all feedback schemes, the hope is that knowledge of congestion will cause the hosts to take appropriate action to reduce the congestion.
- The presence of congestion means that the load is (temporarily) greater than the resources (in part of the system) can handle. Two solutions come to mind: increase the resources or decrease the load.

CONGESTION PREVENTION POLICIES

The methods to control congestion by looking at open loop systems. These 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. In Fig. 5-26 we see different data link, network, and transport policies that can affect congestion (Jain, 1990).

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

Figure 5-26. Policies that affect congestion.

The **data link layer Policies.**

- 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.
- Closely related to this is the **buffering policy**. 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.

The **network layer Policies.**

- The choice between using **virtual circuits and using datagrams** affects congestion since many congestion control algorithms work only with virtual-circuit subnets.
- **Packet queueing 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 **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.

The **transport layer Policies**,

- 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.

ADMISSION CONTROL

- One technique that is widely used to keep congestion that has already started from getting worse is **admission control**.
- Once congestion has been signaled, no more virtual circuits are set up until the problem has gone away.
- 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. 5-27(a), in which two routers are congested, as indicated.

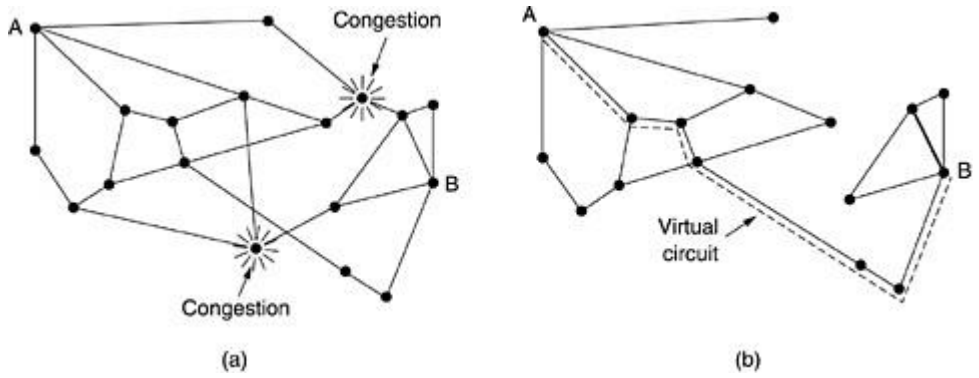


Figure 5-27. (a) A congested subnet. (b) A redrawn subnet that eliminates the congestion. A virtual circuit from* A *to* B *is also shown.

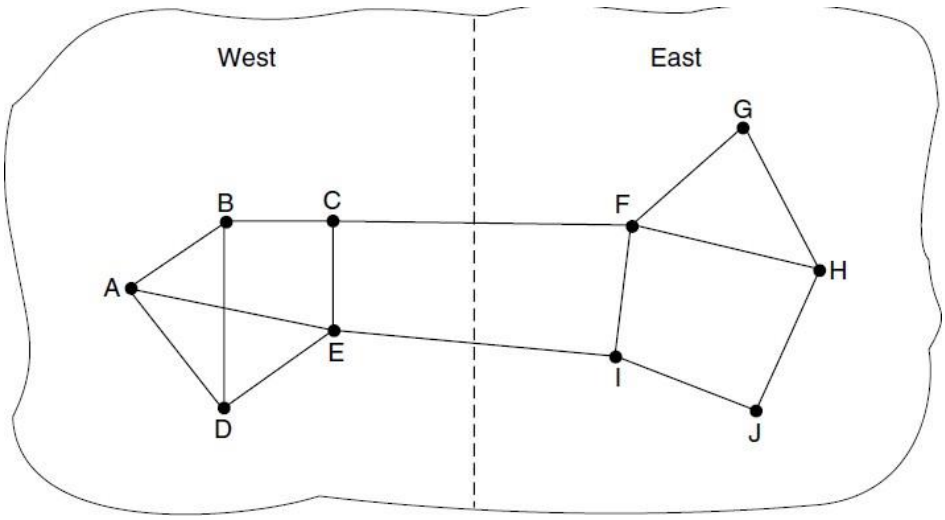
Suppose that a host attached to router *A* wants to set up a connection to a host attached to router *B*. Normally, this connection would pass through one of the congested routers. To avoid this situation, we can redraw the subnet as shown in Fig. 5-27(b), omitting the congested routers and all of their lines. The dashed line shows a possible route for the virtual circuit that avoids the congested routers.

TRAFFIC AWARE ROUTING

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

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

Consider the network of Fig. 5-23, which is divided into two parts, East and West, connected by two links, *CF* and *EI*. Suppose that most of the traffic between East and West is using link *CF*, and, as a result, this link is heavily loaded with long delays. Including queueing delay in the weight used for the shortest path calculation will make *EI* more attractive. After the new routing tables have been installed, most of the East-West traffic will now go over *EI*, loading this link. Consequently, in the next update, *CF* will appear to be the shortest path. As a result, the routing tables may oscillate wildly, leading to erratic routing and many potential problems.



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

TRAFFIC THROTTLING

- Each router can easily monitor the utilization of its output lines and other resources. For example, it can associate with each line a real variable, *u*, whose value, between 0.0 and 1.0, reflects the recent utilization of that line. To maintain a good estimate of *u*, a sample of the instantaneous line utilization, *f* (either 0 or 1), can be made periodically and *u* updated according to

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

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

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

THE WARNING BIT

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

CHOKE PACKETS

- In this approach, the router sends a **choke packet** back to the source host, giving it the destination found in the packet.
- The original packet is tagged (a header bit is turned on) so that it will not generate any more choke packets farther along the path and is then forwarded in the usual way.
- When the source host gets the choke packet, it is required to reduce the traffic sent to the specified destination by *X* percent. Since other packets aimed at the same destination are probably already under way and will generate yet more choke packets, the host should ignore choke packets referring to that destination for a fixed time interval. After that period has expired, the host listens for more choke packets for another interval. If one arrives, the line is still congested, so the host reduces the flow still more and begins ignoring choke packets again. If no choke packets arrive during the listening period, the host may increase the flow again.

- The feedback implicit in this protocol can help prevent congestion yet not throttle any flow unless trouble occurs.
- Hosts can reduce traffic by adjusting their policy parameters.
- Increases are done in smaller increments to prevent congestion from reoccurring quickly.
- 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.

HOP-BY-HOP BACK PRESSURE

- At high speeds or over long distances, sending a choke packet to the source hosts does not work well because the reaction is so slow.

Consider, for example, a host in San Francisco (router *A* in Fig. 5-28) that is sending traffic to a host in New York (router *D* in Fig. 5-28) at 155 Mbps. If the New York host begins to run out of buffers, it will take about 30 msec for a choke packet to get back to San Francisco to tell it to slow down. The choke packet propagation is shown as the second, third, and fourth steps in Fig. 5-28(a). In those 30 msec, another 4.6 megabits will have been sent. Even if the host in San Francisco completely shuts down immediately, the 4.6 megabits in the pipe will continue to pour in and have to be dealt with. Only in the seventh diagram in Fig. 5-28(a) will the New York router notice a slower flow.

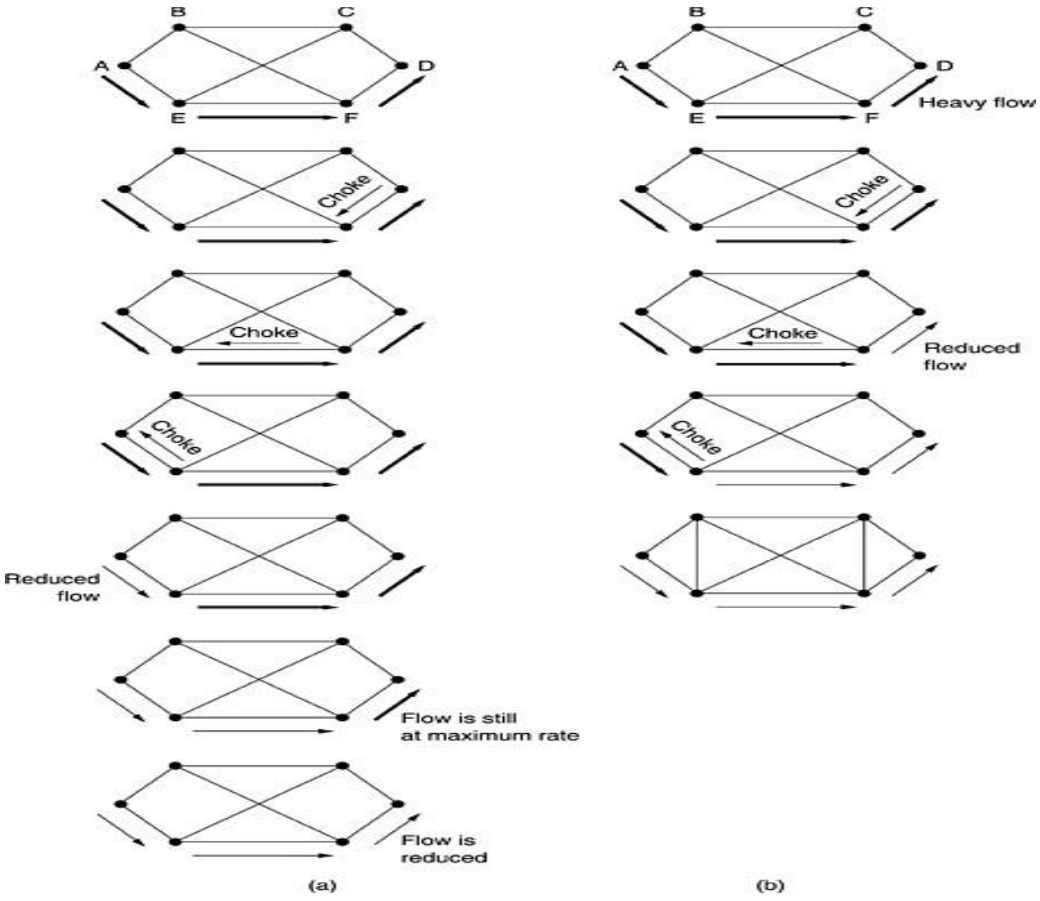


Figure 5-28. (a) A choke packet that affects only the source. (b) A choke packet that affects each hop it passes through.

An alternative approach is to have the choke packet take effect at every hop it passes through, as shown in the sequence of Fig. 5-28(b). Here, as soon as the choke packet reaches *F*, *F* is required to reduce the flow to *D*. Doing so will require *F* to devote more buffers to the flow, since the source is still sending away at full blast, but it gives *D* immediate relief, like a headache remedy in a television commercial. In the next step, the choke packet reaches *E*, which tells *E* to reduce the flow to *F*. This action puts a greater demand on *E*'s buffers but gives *F* immediate relief. Finally, the choke packet reaches *A* and the flow genuinely slows down.

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.

LOAD SHEDDING

- When none of the above methods make the congestion disappear, routers can bring out the heavy artillery: load shedding.
- **Load shedding** is a fancy way of saying that when routers are being in undated by packets that they cannot handle, they just throw them away.
- A router drowning in packets can just pick packets at random to drop, but usually it can do better than that.
- Which packet to discard may depend on the applications running.
- 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.

RANDOM EARLY DETECTION

- It is well known that dealing with congestion after it is first detected is more effective than letting it gum up the works and then trying to deal with it. This observation leads to the idea of discarding packets before all the buffer space is really exhausted. A popular algorithm for doing this is called **RED** (**Random Early Detection)**.
- In some transport protocols (including TCP), the response to lost packets is for the source to slow down. The reasoning behind this logic is that TCP was designed for wired networks and wired networks are very reliable, so lost packets are mostly due to buffer overruns rather than transmission errors. This fact can be exploited to help **reduce congestion**.
- By having routers drop packets before the situation has become hopeless (hence the "early" in the name), the idea is that there is time for action to be taken before it is too late. To determine when to start discarding, routers maintain a running average of their queue lengths. When the average queue length on some line exceeds a threshold, the line is said to be congested and action is taken.

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. Jitter is illustrated in Fig. 5-29. In contrast, an agreement that 99 percent of the packets be delivered with a delay in the range of 24.5 msec to 25.5 msec might be acceptable.

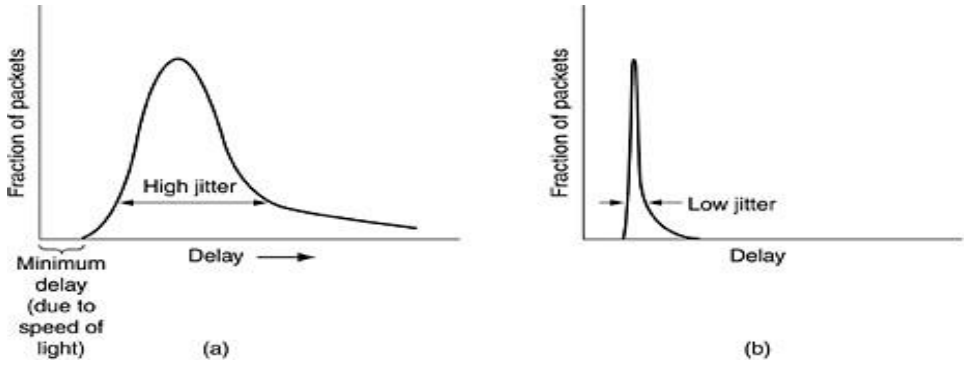


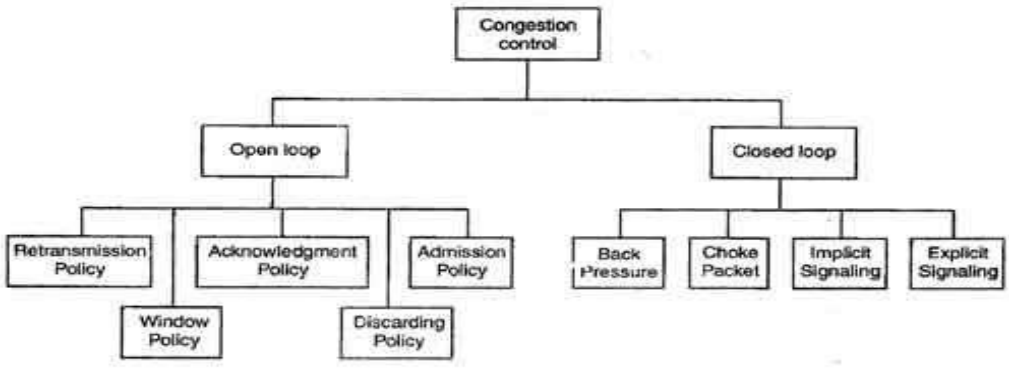
Figure 5-29. (a) High jitter. (b) Low jitter.

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. This information is stored in the packet and updated at each hop. If the packet is ahead of schedule, it is held just long enough to get it back on schedule. If it is behind schedule, the router tries to get it out the door quickly.

- In fact, 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.
- In this way, packets that are ahead of schedule get slowed down and packets that are behind schedule get speeded up, in both cases reducing the amount of jitter.
- In some applications, such as video on demand, jitter can be eliminated by buffering at the receiver and then fetching data for display from the buffer instead of from the network in real time. However, for other applications, especially those that require real-time interaction between people such as Internet telephony and videoconferencing, the delay inherent in buffering is not acceptable.

How to correct the Congestion Problem:

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.



Types of Congestion Control Methods

These two categories are:

1. Open loop
2. Closed loop

Open Loop Congestion Control

• 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.

• The various methods used for open loop congestion control are:

1. Retransmission Policy

• The sender retransmits a packet, if it feels that the packet it has sent is lost or corrupted.

• However retransmission in general may increase the congestion in the network. But we need to implement good retransmission policy to prevent congestion.

• The retransmission policy and the retransmission timers need to be designed to optimize efficiency and at the same time prevent the congestion.

2. Window Policy

• To implement window policy, selective reject window method is used for congestion control.

• Selective Reject method is preferred over Go-back-n window as in Go-back-n method, when timer for a packet times out, several packets are resent, although some may have arrived safely at the receiver. Thus, this duplication may make congestion worse.

• Selective reject method sends only the specific lost or damaged packets.

3. Acknowledgement Policy

• The acknowledgement policy imposed by the receiver may also affect congestion.

• If the receiver does not acknowledge every packet it receives it may slow down the sender and help prevent congestion.

• Acknowledgments also add to the traffic load on the network. Thus, by sending fewer acknowledgements we can reduce load on the network.

• To implement it, several approaches can be used:

1. A receiver may send an acknowledgement only if it has a packet to be sent.
2. A receiver may send an acknowledgement when a timer expires.
3. A receiver may also decide to acknowledge only *N* packets at a time.

4. Discarding Policy

• A router may discard less sensitive packets when congestion is likely to happen.

• Such a discarding policy may prevent congestion and at the same time may not harm the integrity of the transmission.

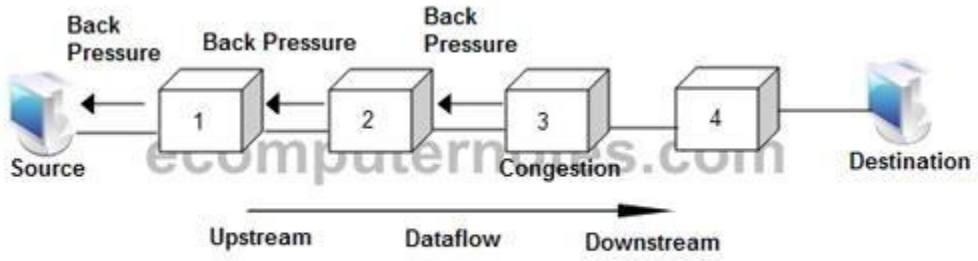
5. Admission Policy

• An admission policy, which is a quality-of-service mechanism, can also prevent congestion in virtual circuit networks.

• Switches in a flow first check the resource requirement of a flow before admitting it to the network.
• A router can deny establishing a virtual circuit connection if there is congestion in the "network or if there is a possibility of future congestion.

Closed Loop Congestion Control
• Closed loop congestion control mechanisms try to remove the congestion after it happens.
• The various methods used for closed loop congestion control are:

1. Backpressure
• Backpressure is a node-to-node congestion control that starts with a node and propagates, in the opposite direction of data flow.

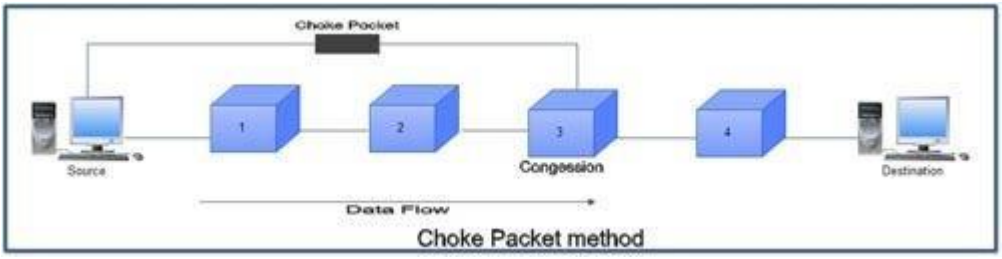


Backpressure Method

• The backpressure technique can be applied only to virtual circuit networks. In such virtual circuit each node knows the upstream node from which a data flow is coming.
• In this method of congestion control, the congested node stops receiving data from the immediate upstream node or nodes.
• This may cause the upstream node on nodes to become congested, and they, in turn, reject data from their upstream node or nodes.

As shown in fig node 3 is congested and it stops receiving packets and informs its upstream node 2 to slow down. Node 2 in turns may be congested and informs node 1 to slow down. Now node 1 may create congestion and informs the source node to slow down. In this way the congestion is alleviated. Thus, the pressure on node 3 is moved backward to the source to remove the congestion.

2. Choke Packet
• In this method of congestion control, congested router or node sends a special type of packet called choke packet to the source to inform it about the congestion.
• Here, congested node does not inform its upstream node about the congestion as in backpressure method.
• 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.



Choke Packet method

3. Implicit Signaling

• In implicit signaling, there is no communication between the congested node or nodes and the source.

• The source guesses that there is congestion somewhere in the network when it does not receive any acknowledgment. Therefore the delay in receiving an acknowledgment is interpreted as congestion in the network.

• On sensing this congestion, the source slows down.

• This type of congestion control policy is used by TCP.

4. Explicit Signaling

• In this method, the congested nodes explicitly send a signal to the source or destination to inform about the congestion.

• Explicit signaling is different from the choke packet method. In choke packed method, a separate packet is used for this purpose whereas in explicit signaling method, the signal is included in the packets that carry data .

• Explicit signaling can occur in either the forward direction or the backward direction .

• In backward signaling, a bit is set in a packet moving in the direction opposite to the congestion. This bit warns the source about the congestion and informs the source to slow down.

• In forward signaling, a bit is set in a packet moving in the direction of congestion. This bit warns the destination about the congestion. The receiver in this case uses policies such as slowing down the acknowledgements to remove the congestion.