

UNIT-5

NETWORK LAYER

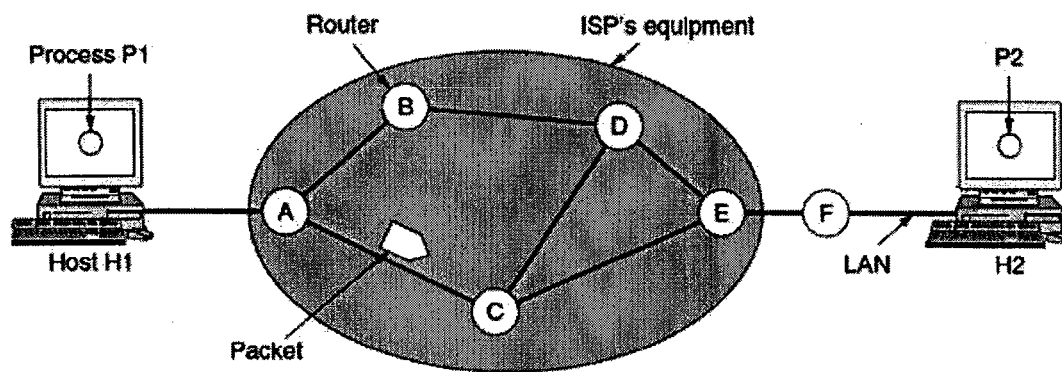
Design Issues – Network layer design issues: In the following sections, we will give an introduction to some of the issues that the designers of the network layer must grapple with. These issues include the service provided to the transport layer and the internal design of the network.

Store-and-Forward Packet Switching:

Before starting to explain the details of the network layer, it is worth restating the context in which the network layer protocols operate. This context can be seen in. The major components of the network are the ISP's equipment (routers connected by transmission lines), shown inside the shaded oval, and the customers' equipment, shown outside the oval. Host *H1* is directly connected to one of the ISP's routers, *A*, perhaps as a home computer that is plugged into a DSL modem.

In contrast, *H2* is on a LAN, which might be an office Ethernet, with a router, *F*, owned and operated by the customer. This router has a leased line to the ISP's equipment.

We have shown *F* as being outside the oval because it does not belong to the ISP. For the purposes of this chapter, however, routers on customer premises are considered part of the ISP network because they run the same algorithms as the ISP's routers (and our main concern here is algorithms).



The environment of the network layer protocols.

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

Services Provided to the Transport Layer: The network layer provides services to the transport layer at the network layer/transport layer interface. An important question is precisely what kind of services the network layer provides to the transport layer. The services need to be carefully designed with the following goals in mind:

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

Given these goals, the designers of the network layer have a lot of freedom in writing detailed specifications of the services to be offered to the transport layer.

This freedom often degenerates into a raging battle between two warring factions. The discussion centers on whether the network layer should provide connection-oriented service or connectionless service.

One camp (represented by the Internet community) argues that the routers' job is moving packets around and nothing else. In this view (based on 40 years of experience with a real computer network), the network is inherently unreliable, no matter how it is designed. Therefore, the hosts should accept this fact and do error control (i.e., error detection and correction) and flow control themselves.

This viewpoint leads 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.

This reasoning is an example of the **end-to-end argument**, a design principle that has been very influential in shaping the Internet. 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 network should provide a reliable, connection-oriented service. They claim that 100 years of successful experience with the worldwide telephone system is an excellent guide.

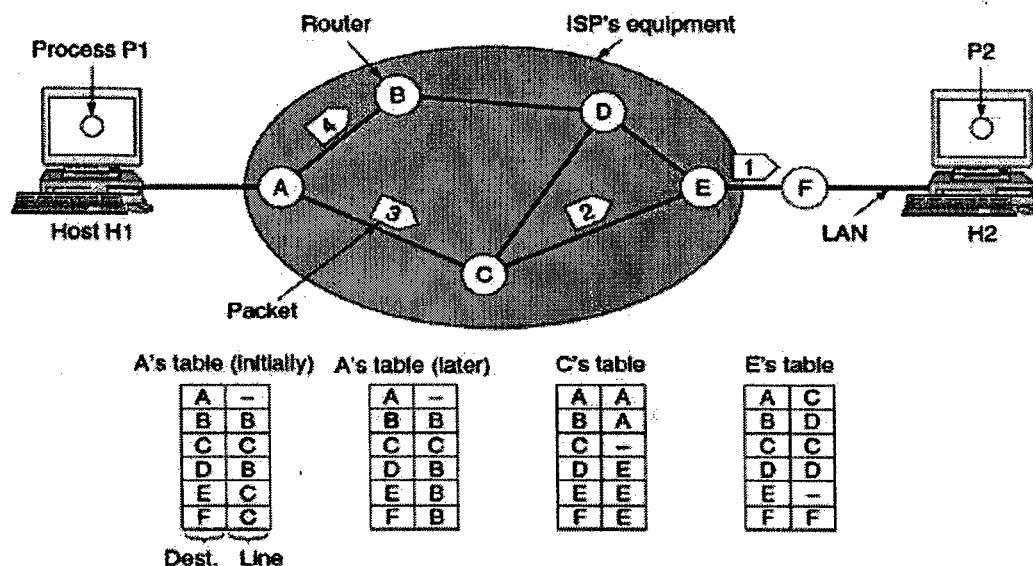
Implementation of Connectionless Service: Having looked at the two classes of service the network layer can provide to its users, it is time to see how this layer works inside. Two different organizations are possible, depending on the type of service offered. If connectionless service is offered, packets are injected into the network individually and routed independently of each other. No advance setup is needed.

In this context, the packets are frequently called **datagrams** (in analogy with telegrams) and the network is called a **datagram network**.

If connection-oriented service is used, a path from the source router all the way 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 network is called a **virtual-circuit network**.

In this section, we will examine datagram networks; in the next one, we will examine virtual-circuit networks.

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



Routing within a datagram network.

Let us assume for this example that the message is four times longer than the maximum packet size, so the network layer has to break it into four packets, 1, 2, 3, and 4, and send each of them in turn to router *A* using some point-to-point protocol, for example, PPP. At this point the ISP takes over. Every router has an internal table telling it where to send packets for each of the possible destinations.

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., *A* has only two outgoing lines—to *B* and to *C*—so every incoming packet must be sent to one of these routers, even if the ultimate destination is to some other router. *A*'s initial routing table is shown in the figure under the label "initially."

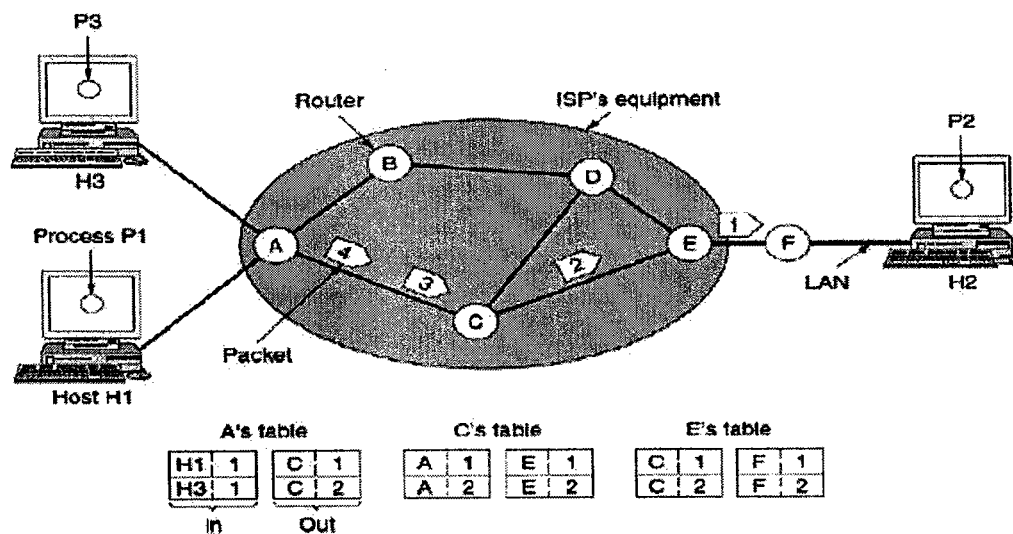
At *A*, packets 1, 2, and 3 are stored briefly, having arrived on the incoming link and had their checksums verified. Then each packet is forwarded according to *A*'s table, onto the outgoing link to *C* within a new frame. Packet 1 is then forwarded to *E* and then to *F*. When it gets to *F*, it is sent within a frame over the LAN to *H2*.

Packets 2 and 3 follow the same route. However, something different happens to packet 4. When it gets to *A* it is sent to router *B*, even though it is also destined for *F*. For some reason, *A* decided to send packet 4 via a different route than that of the first three packets. Perhaps it has 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**.

Routing algorithms are one of the main topics we will study in this chapter. There are several different kinds of them, as we will see. IP (Internet Protocol), which is the basis for the entire Internet, is the dominant example of a connectionless network service. Each packet carries a destination IP address that routers use to individually forward each packet. The addresses are 32 bits in IPv4 packets and 128 bits in IPv6 packets.

Implementation of Connection-Oriented Service: For connection-oriented service, we need a virtual-circuit network. 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. 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 shown in Fig. Here, host *H1* has established connection 1 with host *H2*. This connection is remembered as the first entry in each of the routing tables. The first line of *A*’s table says that if a packet bearing connection identifier 1 comes in from *H1*, it is to be sent to router *C* and given connection identifier 1. Similarly, the first entry at *C* routes the packet to *E*, also with connection identifier 1.



Routing within a virtual-circuit network.

Now let us consider what happens if *H3* also wants to establish a connection to *H2*. It chooses connection identifier 1 (because it is initiating the connection and this is its only connection) and tells the network to establish the virtual circuit. This leads to the second row in the tables. Note that we have a conflict here because although *A* can easily distinguish connection 1 packets from *H1* from connection 1 packets from *H3*, *C* cannot do this. For this reason, *A* assigns a different connection identifier to the outgoing traffic for the second connection. Avoiding conflicts of this kind is why routers need the ability to replace connection identifiers in outgoing packets. In some contexts, this process is called **label switching**.

Comparison of Virtual-Circuit and Datagram Networks:

Both virtual circuits and datagrams have their supporters and their detractors. We will now attempt to summarize both sets of arguments. The major issues are listed in Fig, although purists could probably find a counterexample for everything in the figure.

Issue	Datagram network	Virtual-circuit network
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

Comparison of datagram and virtual-circuit networks.

Inside the network, several trade-offs exist between virtual circuits and data grams. One trade-off is setup time versus address parsing time. Using virtual circuits requires a setup phase, which takes time and consumes resources. However, once this price is paid, figuring out what to do with a data packet in a virtual-circuit network is easy: the router just uses the circuit number to index into a table to find out where the packet goes. In a datagram network, no setup is needed but a more complicated lookup procedure is required to locate the entry for the destination.

A related issue is that the destination addresses used in datagram networks are longer than circuit numbers used in virtual-circuit networks because they have a global meaning. If the packets tend to be fairly short, including a full destination address in every packet may represent a significant amount of overhead and hence a waste of bandwidth. Yet another issue is the amount of table space required in router memory.

A datagram network needs to have an entry for every possible destination, whereas a virtual-circuit network just needs an entry for each virtual circuit. However, this advantage is somewhat illusory since connection setup packets have to be routed too, and they use destination addresses, the same as datagrams do. Virtual circuits have some advantages in guaranteeing quality of service and avoiding congestion within the network because resources (e.g., buffers, bandwidth, and CPU cycles) can be reserved in advance, when the connection is established. Once the packets start arriving, the necessary bandwidth and router capacity will be there.

With a datagram network, congestion avoidance is more difficult. For transaction processing systems (e.g., stores calling up to verify credit card purchases), the overhead required to set up and clear a virtual circuit may easily dwarf the use of the circuit. If the majority of the traffic is expected to be of this kind, the use of virtual circuits inside the network makes little sense. On the other hand, for long-running uses such as VPN traffic between two corporate offices, permanent virtual circuits (that are set up manually and last for months or years) may be useful. Virtual circuits also have a vulnerability problem. If a router crashes and loses its memory, even if it comes back up a second later, all the virtual circuits passing through it will have to be aborted.

In contrast, if a datagram router goes down, only those users whose packets were queued in the router at the time need suffer (and probably not even then since the sender is likely to retransmit them shortly). The loss of a communication line is fatal to virtual circuits using it, but can easily be compensated for if datagrams are used. Datagrams also allow the routers to balance the traffic throughout the network, since routes can be changed partway through a long sequence of packet transmissions.

ROUTING ALGORITHMS: The main function of the network layer is routing packets from the source machine to the destination machine. In most networks, packets will require multiple hops to make the journey. The only notable exception is for broadcast networks, but even here routing is an issue if the source and destination are not on the same network segment. The algorithms that choose the routes and the data structures that they use are a major area of network layer design.

The **routing algorithm** is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on. If the network uses datagrams internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time. If the network uses virtual circuits internally, routing decisions are made only when a new virtual circuit is being set up. Thereafter, data packets just

follow the already established route. The latter case is sometimes called **session routing** because a route remains in force for an entire session (e.g., while logged in over a VPN). It is sometimes useful to make a distinction between routing, which is making the decision which routes to use, and forwarding, which is what happens when a packet arrives.

One can think of a router as having two processes inside it. One of them handles each packet as it arrives, looking up the outgoing line to use for it in the routing tables. This process is **forwarding**. The other process is responsible for filling in and updating the routing tables. That is where the routing algorithm comes into play.

Regardless of whether routes are chosen independently for each packet sent or only when new connections are established, certain properties are desirable in a routing algorithm: correctness, simplicity, robustness, stability, fairness, and efficiency. Correctness and simplicity hardly require comment, but the need for robustness may be less obvious at first. Once a major network comes on the air, it may be expected to run continuously for years without system-wide failures. During that period there will be hardware and software failures of all kinds.

Hosts, routers, and lines will fail repeatedly, and the topology will change many times. The routing algorithm should be able to cope with changes in the topology and traffic without requiring all jobs in all hosts to be aborted. Imagine the havoc if the network needed to be rebooted every time some router crashed! Stability is also an important goal for the routing algorithm.

There exist routing algorithms that never converge to a fixed set of paths, no matter how long they run. A stable algorithm reaches equilibrium and stays there. It should converge quickly too, since communication may be disrupted until the routing algorithm has reached equilibrium.

Routing algorithms can be grouped into two major classes: non adaptive and adaptive. **Non adaptive algorithms** do not base their routing decisions on any measurements or estimates of the current topology and traffic. Instead, the choice of the route to use to get from I to J (for all I and J) is computed in advance, offline, and downloaded to the routers when the network is booted. This procedure is sometimes called **static routing**.

Because it does not respond to failures, static routing is mostly useful for situations in which the routing choice is clear. For example, router F in Fig. should send packets headed into the network to router E regardless of the ultimate destination.

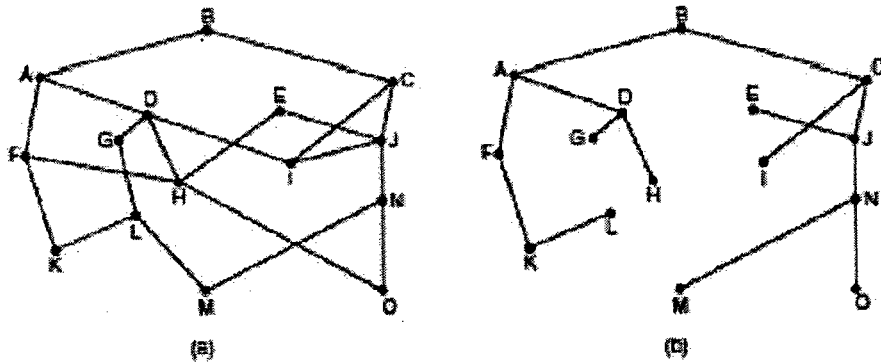
Adaptive algorithms, in contrast, change their routing decisions to reflect changes in the topology, and sometimes changes in the traffic as well.

The Optimality Principle : Before we get into specific algorithms, it may be helpful to note that one can make a general statement about optimal routes without regard to network topology or traffic. This statement is known as the **optimality principle** (Bellman, 1957). It states that if router J is on the optimal path from router I to router K , then the optimal path from J to K also falls along the same route. To see this, call the part of the route from I to J r_1 and the rest of the

route r_2 . If a route better than r_2 existed from J to K , it could be concatenated with r_1 to improve the route from I to K , contradicting our statement that $r_1 r_2$ is optimal.

As a direct consequence of the optimality principle, we can see that the set of optimal routes from all sources to a given destination form a tree rooted at the destination. Such a tree is called a **sink tree** and is illustrated in Fig. where the distance metric is the number of hops. The goal of all routing algorithms is to discover and use the sink trees for all routers.

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



Note that a sink tree is not necessarily unique; other trees with the same path lengths may exist. If we allow all of the possible paths to be chosen, the tree becomes a more general structure called a **DAG (Directed Acyclic Graph)**. DAGs have no loops.

Since a sink tree is indeed a tree, it does not contain any loops, so each packet will be delivered within a finite and bounded number of hops. Nevertheless, the optimality principle and the sink tree provide a benchmark against which other routing algorithms can be measured.

Shortest Path Algorithm : Let us begin our study of routing algorithms with a simple technique for computing optimal paths given a complete picture of the network. These paths are the ones that we want a distributed routing algorithm to find, even though not all routers may know all of the details of the network. The idea is to build a graph of the network, with each node of the graph representing a router and each edge of the graph representing a communication line, or link. To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.

The concept of a **shortest path** deserves some explanation. One way of measuring path length is the number of hops. Using this metric, the paths ABC and ABE in Fig. are equally long. Another metric is the geographic distance in kilometers, in which case ABC is clearly much longer than ABE (assuming the figure is drawn to scale).

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

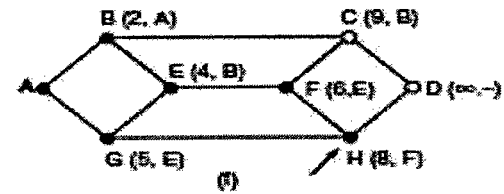
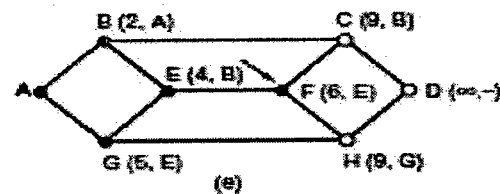
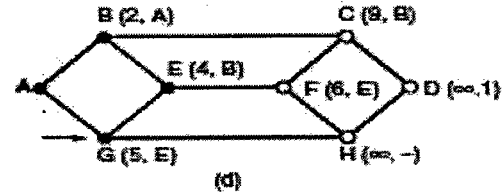
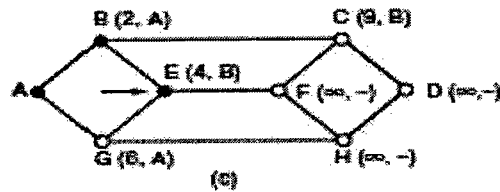
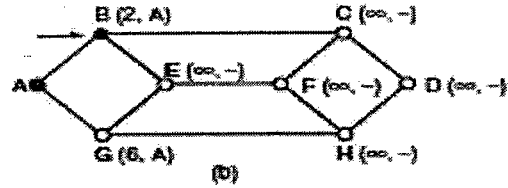
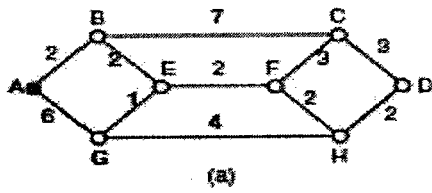
However, many other metrics besides hops and physical distance are also possible. For example, each edge could be labeled with the mean delay of a standard test packet, as measured by hourly runs. With this graph labeling, the shortest path is the fastest path rather than the path with the fewest edges or kilometers. In the general case, the labels on the edges could be computed as a function of the distance, bandwidth, average traffic, communication cost, 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. Several algorithms for computing the shortest path between two nodes of a graph are known.

This one is due to Dijkstra (1959) and finds the shortest paths between a source and all destinations in the network. Each node is labeled (in parentheses) with its distance from the source node along the best known path. The distances must be non-negative, as they will be if they are based on real quantities like bandwidth and delay. Initially, no paths are known, so all nodes are labeled with infinity.

As the algorithm proceeds and paths are found, the labels may change, reflecting better paths. A label may be either tentative or permanent. Initially, all labels are tentative. When it is discovered that a label represents the shortest possible path from the source to that node, it is made permanent and never changed thereafter. To illustrate how the labeling algorithm works, look at the weighted, undirected graph of Fig., where the weights represent, for example, distance. We want to find the shortest path from *A* to *D*. We start out by marking node *A* as permanent, indicated by a filled-in circle.

Then we examine, in turn, each of the nodes adjacent to *A* (the working node), relabeling each one with the distance to *A*. Whenever a node is relabeled, we also label it with the node from which the probe was made so that we can reconstruct the final path later. If the network had more than one shortest path from *A* to *D* and we wanted to find all of them, we would need to

remember all of the probe nodes that could reach a node with the same distance.



congestion: when too many packets rushing to nodes on a part of network, the network performance degrades, and this situation is called as congestion

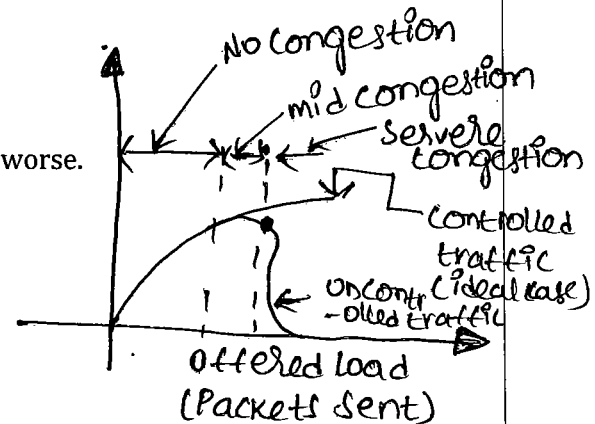
Congestion control algorithms: A state occurring in network layer when the message traffic is so heavy that it slows down network response time.

Effects of Congestion

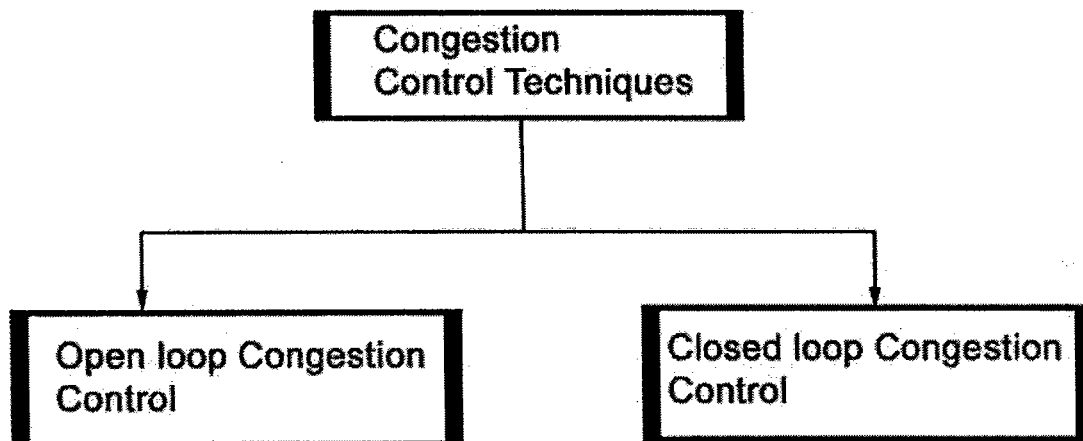
- As delay increases, performance decreases.
- If delay increases, retransmission occurs, making situation worse.

Factors for congestion:

- ✓ Three input lines and one output line.
- ✓ Insufficient memory.
- ✓ Sufficient memory.
- ✓ Slow processors.
- ✓ Low bandwidth lines.



Approaches to congestion control: Congestion control refers to the techniques used to control or prevent congestion. Congestion control techniques can be broadly classified into two categories:



Open Loop Congestion Control

Open loop congestion control policies are applied to prevent congestion before it happens. The congestion control is handled either by the source or the destination.

Policies adopted by open loop congestion control -

- ✓ **Retransmission Policy:**
It is the policy in which retransmission of the packets are taken care. If the sender feels that a sent packet is lost or corrupted, the packet needs to be retransmitted. This transmission may increase the congestion in the network. To prevent congestion, retransmission timers must be designed to prevent congestion and also able to optimize efficiency.
- ✓ **Window Policy:**
The type of window at the sender side may also affect the congestion. Several packets in the Go-back-n window are resent, although some packets may be received successfully at the receiver side. This duplication may increase the congestion in the network and making it worse.

Therefore, Selective repeat window should be adopted as it sends the specific packet that may have been lost.

✓ **Discarding Policy:**

A good discarding policy adopted by the routers is that the routers may prevent congestion and at the same time partially discards the corrupted or less sensitive package and also able to maintain the quality of a message.

In case of audio file transmission, routers can discard less sensitive packets to prevent congestion and also maintain the quality of the audio file.

✓ **Acknowledgment Policy:**

Since acknowledgement is also the part of the load in network, the acknowledgment policy imposed by the receiver may also affect congestion. Several approaches can be used to prevent congestion related to acknowledgment.

The receiver should send acknowledgement for N packets rather than sending acknowledgement for a single packet. The receiver should send a acknowledgment only if it has to sent a packet or a timer expires.

✓ **Admission Policy:**

In admission policy a mechanism should be used to prevent congestion. Switches in a flow should first check the resource requirement of a network flow before transmitting it further. If there is a chance of a congestion or there is a congestion in the network, router should deny establishing a virtual network connection to prevent further congestion.

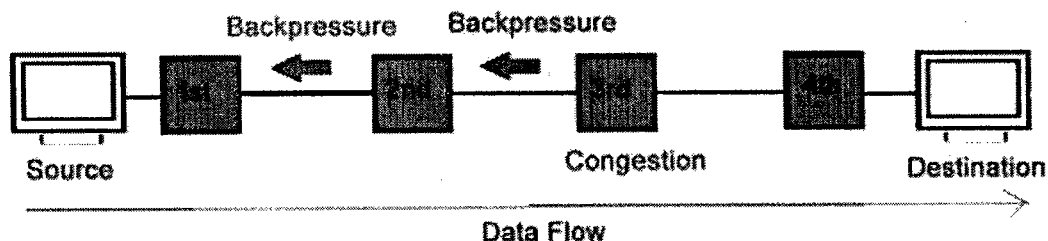
- ✓ All the above policies are adopted to prevent congestion before it happens in the network.

Closed Loop Congestion Control

Closed loop congestion control technique is used to treat or alleviate congestion after it happens. Several techniques are used by different protocols; some of them are:

✓ **Backpressure:**

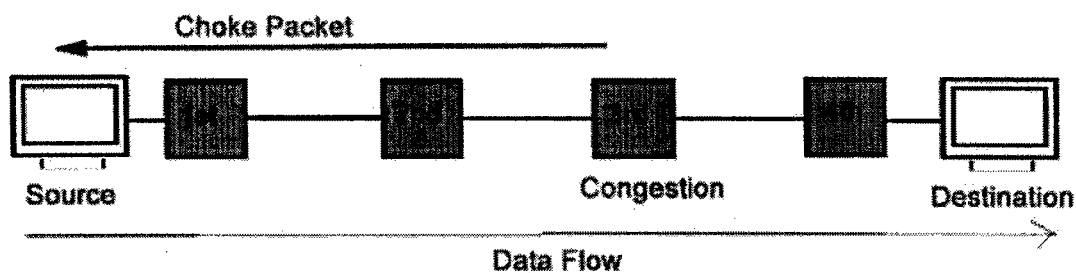
Backpressure is a technique in which a congested node stops receiving packet from upstream node. This may cause the upstream node or nodes to become congested and rejects receiving data from above nodes. Backpressure is a node-to-node congestion control technique that propagates in the opposite direction of data flow. The backpressure technique can be applied only to virtual circuit where each node has information of its above upstream node.



In above diagram the 3rd node is congested and stops receiving packets as a result 2nd node may get congested due to slowing down of the output data flow. Similarly 1st node may get congested and informs the source to slow down.

✓ **Choke Packet Technique:**

Choke packet technique is applicable to both virtual networks as well as datagram subnets. A choke packet is a packet sent by a node to the source to inform it of congestion. Each router monitors its resources and the utilization at each of its output lines. Whenever the resource utilization exceeds the threshold value which is set by the administrator, the router directly sends a choke packet to the source giving it a feedback to reduce the traffic. The intermediate nodes through which the packets have traveled are not warned about congestion.



✓ **Implicit Signaling:**

In implicit signaling, there is no communication between the congested nodes and the source. The source guesses that there is congestion in a network. For example when sender sends several packets and there is no acknowledgment for a while, one assumption is that there is congestion.

✓ **Explicit Signaling:**

In explicit signaling, if a node experiences congestion it can explicitly send a packet to the source or destination to inform about congestion. The difference between choke packet and explicit signaling is that the signal is included in the packets that carry data rather than creating different packet as in case of choke packet technique. Explicit signaling can occur in either forward or backward direction.

- **Forward Signaling:** In forward signaling signal is sent in the direction of the congestion. The destination is warned about congestion. The receiver in this case adopts policies to prevent further congestion.
- **Backward Signaling:** In backward signaling signal is sent in the opposite direction of the congestion. The source is warned about congestion and it needs to slow down.

Load Shedding: This is what routers do when they run out of buffers. If they must throw away a packet then they can at least try to pick the best packets to pitch. This depends on the application and on the error strategy used in the data link layer. Older packets will be worse to throw away for a file transfer than newer ones, since they will cause a gap in the received packets and a go-back-n strategy may be employed by the data link layer.

For real-time voice or video it is probably better to throw away old data (already obsolete?) and keep new packets. Who knows best which packets can be dropped? The application. Getting the application to mark packets with priority requires some incentive like cheaper transmission rates for lower priority packets.

It appears to be better for routers to start dropping packets as soon as congestion seems likely, rather than wait for congestion to take over.

Three levels of priority are there for packets.

- ✓ **VERY IMPORTANT**
- ✓ **NEVER**
- ✓ **DISCARD**

Admission control: Here the incoming traffic from some flow is well shaped and can potentially follow a single route in which capacity can be reserved in advance on the routers along the path. The decision to accept or reject a flow is not a simple matter of comparing the (bandwidth, buffers, and cycles) requested by the flow with the routers excess capacity in those three dimensions.

A set of these parameters is called **Flow specification**. As the specification propagates along the route, each router examines it and modifies the parameters as need be. The modifications can only reduce the flow but cannot increase.

We have 5 parameters

Parameter	Unit
Token bucket rate	Bytes/sec
Token bucket size	Bytes
Peak data rate	Bytes/sec
Minimum packet size	Bytes
Maximum packet size	Bytes

- Token bucket rate is number of bytes per second that are put into the bucket.
- Token bucket size is size of the bucket in bytes.
- Peak data rate means is the maximum tolerated transmission rate, even for brief intervals. The sender must never exceed this rate.
- The minimum packet size is important because processing each packet takes fixed time, no matter how short.
- The maximum packet size is important due to internal network limitations that may not be exceeded.