MODULE IV

Syllabus:
Congestion control algorithms – QoS. Internetworking – Network layer in internet. IPv4 - IP Addressing – Classless and Classfull Addressing. Sub-netting.

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

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

FACTORS THAT AFFECT CONGESTION

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

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

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

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

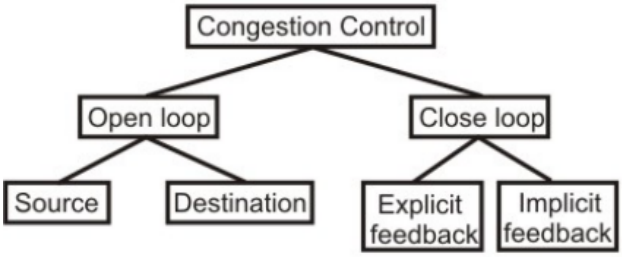
Bursty nature of traffic. If the hosts could be made to transmit at a uniform rate, then congestion problem will be less common and all other causes will not even led to congestion .

Congestion control Vs flow control

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

Congestion Control Techniques

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



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

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

subnet that can take action to reduce the congestion. This approach can be divided into 3 steps:

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

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

• Adjust the system operation to correct the problem.

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

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

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

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

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

Congestion Prevention Policies

At the data link layer,

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

At the network layer,

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

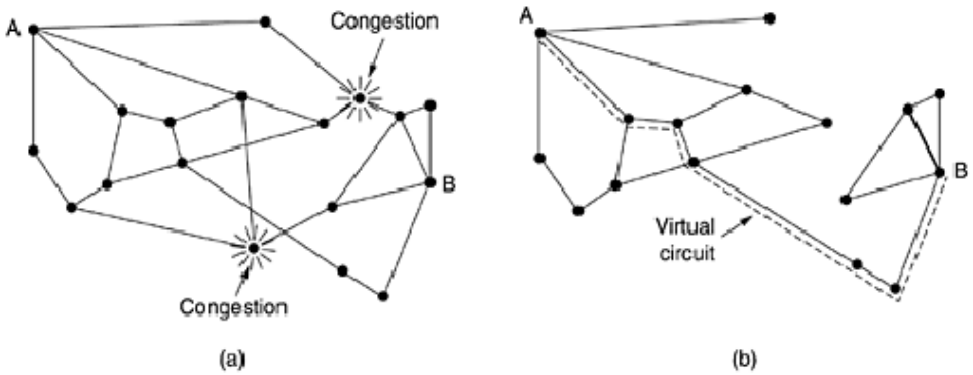
In the transport layer, the same issues occur as in the data link layer, but in addition, determining the timeout interval is harder because the transit time across the network is less
predictable than the transit time over a wire between two routers. If the timeout interval is too short, extra packets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

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

Congestion control in virtual Circuit

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

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



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

Another strategy relating to virtual circuits is to negotiate an agreement between the host and subnet when a virtual circuit is set up. This agreement normally specifies the volume and shape of the traffic, quality of service required, and other parameters. To keep its part of the agreement, the subnet will typically reserve resources along the path

when the circuit is set up. These resources can include table and buffer space in the routers and bandwidth on the lines. In this way, congestion is unlikely to occur on the new virtual circuits because all the necessary resources are guaranteed to be available. Disadvantage is wastage of resources.

Congestion Control in Datagram Subnets

Let us now turn to some approaches that can be used in datagram subnets (and also in virtual circuit subnets). Each router can easily monitor the utilization of its output lines and other resources.

The Warning Bit

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

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

Choke Packets

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

- Hosts can reduce traffic by adjusting their policy parameters, for example, their window size.
- Several variations on this congestion control algorithm have been proposed. For one, the routers can maintain several thresholds. Depending on which threshold has been crossed, the choke packet can contain a mild warning, a stern warning, or an ultimatum.
- Another variation is to use queue lengths or buffer utilization instead of line utilization as the trigger signal.

Hop-by-Hop Choke Packets

At high speeds or over long distances, sending a choke packet to the source hosts does not work well because the reaction is so slow. Consider, for example, a host router *A* in Fig.(a) that is sending traffic to a router *D* at 155 Mbps. If D begins to run out of buffers, it will take about 30 msec for a choke packet to get back to A to tell it to slow down. The choke packet propagation is shown as the second, third, and fourth steps in Fig. (a). In those 30 msec, another 4.6 megabits will have been sent. Even if the host A completely shuts down immediately, the 4.6 megabits in the pipe will continue to pour in and have to be dealt with. Only in the seventh diagram in Fig. (a) will the router D notice a slower flow.

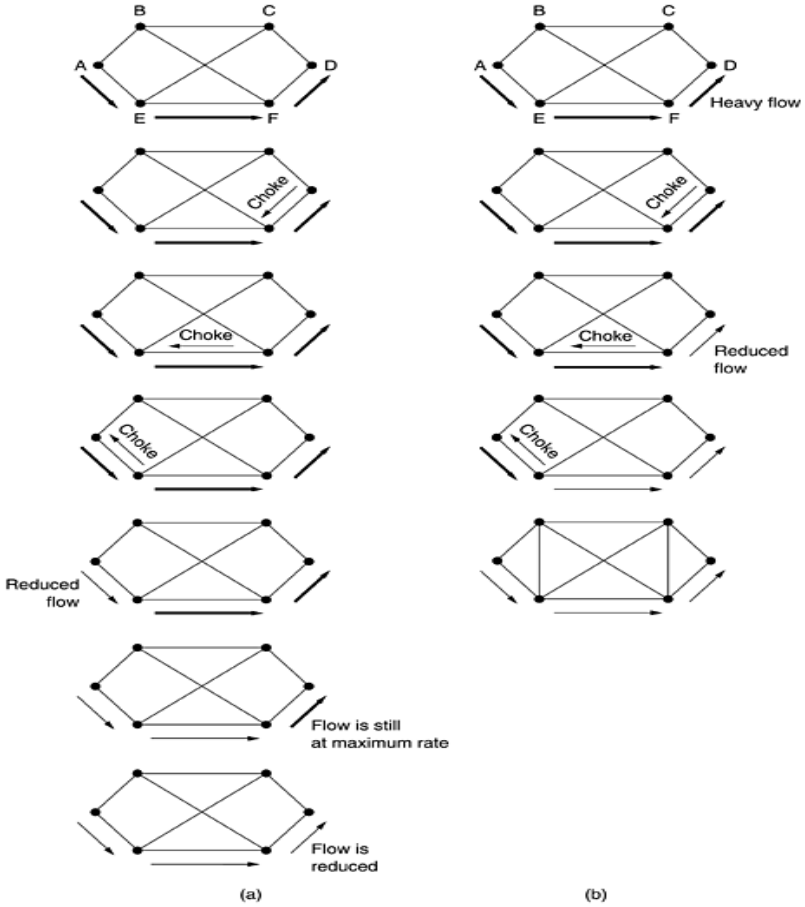


Fig.(a) A choke packet that affects only the source. (b) A

Choke packet that affects each hop it passes through.

An alternative approach is to have the choke packet take effect at every hop it passes through, as shown in the sequence of Fig. (b). Here, as soon as the choke packet reaches *F*, *F* is required to reduce the flow to *D*. Doing so will require *F* to devote more buffers to the flow, since the source is still sending away at full blast, but it gives *D* immediate relief. In the next step, the choke packet reaches *E*, which tells *E* to reduce the flow to *F*. This action puts a greater demand on *E*'s buffers but gives *F* immediate relief. Finally, the choke packet reaches *A* and the flow genuinely slows down. The net effect of this hop-by-hop scheme is to provide quick relief at the point of congestion at the price of using up more buffers upstream.

Load Shedding

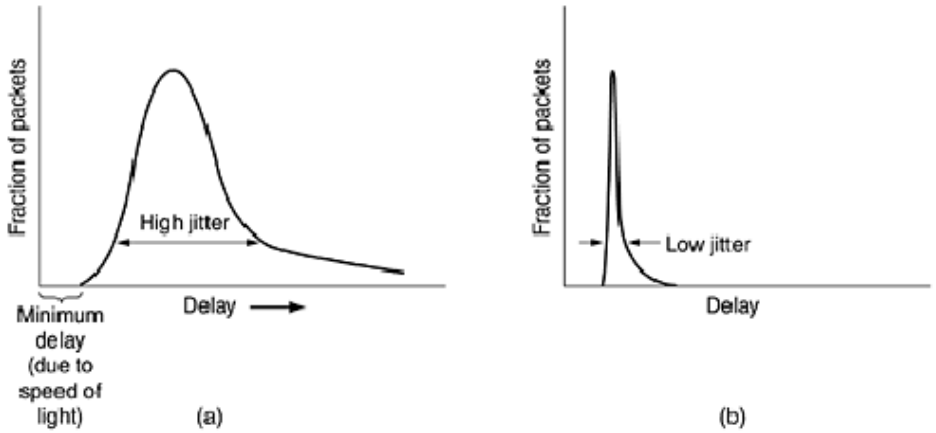
- Another simple closed loop technique
- When buffers become full, routers simply discard packets.
- Which packet is chosen to be the victim depends on the application and on the error strategy used in the data link layer.
- For a file transfer, for, e.g. cannot discard older packets since this will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

Random Early Detection

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

Jitter control

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



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

Quality of Service

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

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

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

Techniques for Achieving Good Quality of Service

1. ***Overprovisioning***

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

2. ***Buffering***

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

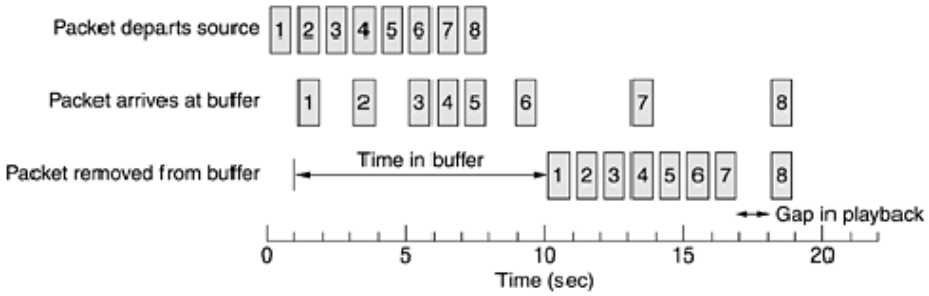


Figure . example for Smoothing the output stream by buffering packets.

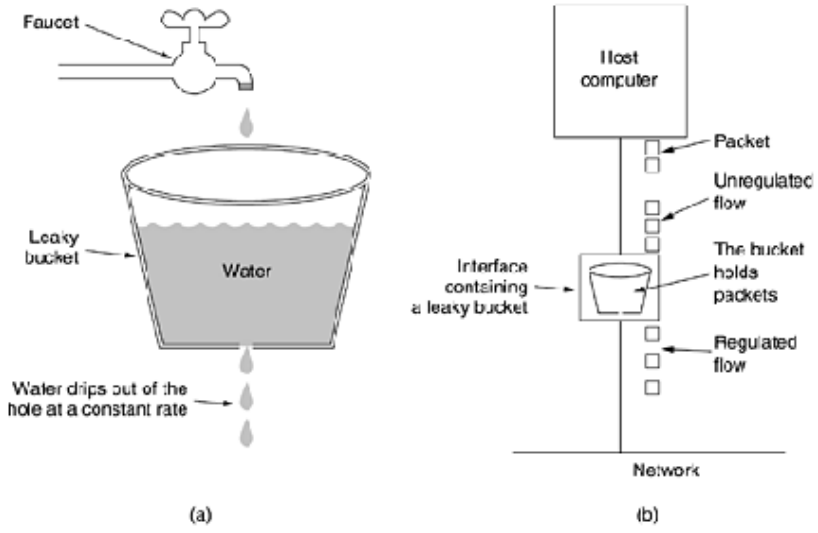
3. Traffic Shaping

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

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

3.1 The Leaky Bucket Algorithm

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



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

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

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

3.2 The Token Bucket Algorithm

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

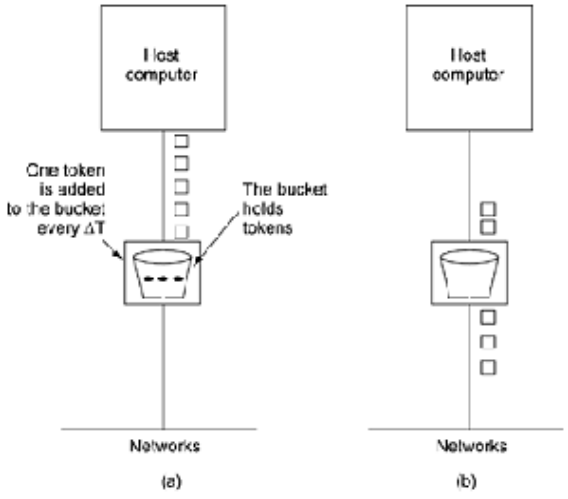


Fig. The tocken bucket algorithm a)before b) after

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

Resource Reservation

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

- Bandwidth is the most obvious. If a flow requires 1 Mbps and the outgoing line has a capacity of 2 Mbps, trying to direct three flows through that line is not going to work. Thus, reserving bandwidth means not oversubscribing any output line.
- Buffer space-When a packet arrives, it is usually deposited on the network interface card by the hardware itself. The router software then has to copy it to a buffer in RAM and queue that buffer for transmission on the chosen outgoing line.

If no buffer is available, the packet has to be discarded since there is no place to put it. For a good quality of service, some buffers can be reserved for a specific flow so that flow does not have to compete for buffers with other flows.

- Finally, CPU cycles are also a scarce resource. It takes router CPU time to process a packet, so a router can process only a certain number of packets per second. Making sure that the CPU is not overloaded is needed to ensure timely processing of each packet.

Admission Control

Admission control refers to the mechanism used by a router, or a switch, to accept or reject a flow based on predefined parameters called flow specifications. Before a router accepts a flow for processing, it checks the flow specifications to see if its capacity (in terms of bandwidth, buffer size, CPU speed, etc.) and its previous commitments to other flows can handle the new flow.

Proportional Routing

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

Packet Scheduling

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

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

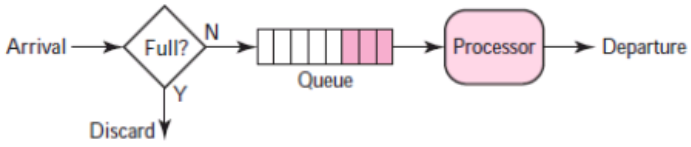


Fig: FIFO queue

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

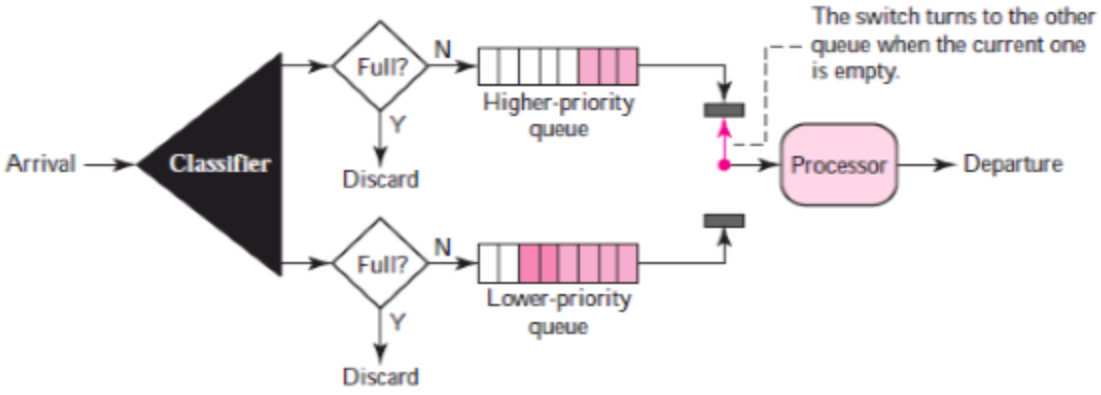


Fig: Priority queuing

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

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

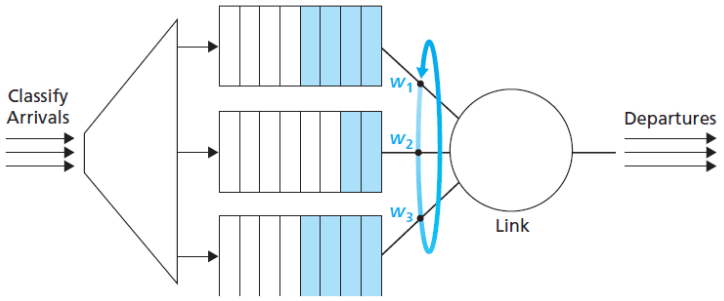


Figure Weighted fair queuing (WFQ)

Integrated services

Intserv is a framework developed by the IETF to provide individualized QoS guarantees to individual application sessions. This is also called flow-based algorithms.it was aimed at both unicast and muticast operations. Two key features of Intserv are:

- Reserved resources :A router is supposed to know what amounts of its resources (buffers, link b/w) are already reserved for ongoing sessions
- Call setup : A session requiring QoS guarantees must first be able to reserve sufficient resources at each network router on its source-to destination path to ensure that its end-to-end QoS requirement is met.

RSVP: Resource Reservation Protocol

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

- Provides reservations for bandwidth in multicast spanning trees (unicast is handled as a degenerate case of multicast). Each group is assigned a group address. To send to a group, a sender puts the group's address in its packets. The standard multicast routing algorithm then builds a spanning tree covering all group members. The routing algorithm is not part of RSVP. The only difference from normal multicasting is a little extra information that is multicast to the group periodically to tell the routers along the tree to maintain certain data structures in their memories.
- Is receiver-oriented, that is, the receiver of the data flow initiates and maintains the resource reservation used for that flow. To get better reception and eliminate congestion, any of the receivers in a group can send a reservation message up the tree to the sender. The message is propagated using the reverse path forwarding algorithm discussed earlier. At each hop, the router notes the reservation and reserves the necessary bandwidth. If insufficient bandwidth is available, it reports back failure. By the time the message gets back to the source, bandwidth has been reserved all the way from the sender to the receiver making the reservation request along the spanning tree.
  When making a reservation, a receiver can (optionally) specify one or more sources that it wants to receive from. It can also specify whether these choices are fixed for the duration of the reservation or whether the receiver wants to keep open the option of changing sources later. The routers use this information to optimize bandwidth planning. In particular, two receivers are only set up to share a path if they both agree not to change sources later on.

Differentiated Services

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

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

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

Differentiated services (DS) can be offered by a set of routers forming an administrative domain (e.g., an ISP or a telco). The administration defines a set of service classes with corresponding forwarding rules. If a customer signs up for DS, customer packets entering the domain may carry a *Type of Service* field in them, with better service provided to some classes (e.g., premium service) than to others. Traffic within a class may be required to conform to some specific shape, such as a leaky bucket with some specified drain rate. An operator with a nose for business might charge extra for each premium packet transported or might allow up to *N* premium

packets per month for a fixed additional monthly fee. Note that this scheme requires no advance setup, no resource reservation, and no time-consuming end-to-end negotiation for each flow, as with integrated services. This makes DS relatively easy to implement.

Expedited Forwarding

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

Assured Forwarding

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

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

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

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

Internetworking

Responsibilities of Network Layer:

- **Packet forwarding/Routing of packets:** Relaying of data packets from one network segment to another by nodes in a computer network
- **Connectionless communication(IP):** A data transmission method used in packet switched networks in which each data unit is separately addressed and routed based on information carried by it
- **Fragmentation of data packets:** Splitting of data packets that are too large to be transmitted on the network

The Network Layer in the Internet

The top 10 principles

1. **Make sure it works.** Do not finalize the design or standard until multiple prototypes have successfully communicated with each other.
2. **Keep it simple.** When in doubt, use the simplest solution. If a feature is not absolutely essential, leave it out, especially if the same effect can be achieved by combining other features.
3. **Make clear choices.** If there are several ways of doing the same thing, choose one.
4. **Exploit modularity.** This principle leads directly to the idea of having protocol stacks,

each of whose layers is independent of all the other ones. In this way, if circumstances that require one module or layer to be changed, the other ones will not be affected.
5. **Expect heterogeneity.** Different types of hardware, transmission facilities, and applications will occur on any large network. To handle them, the network design must be simple, general, and flexible.
6. **Avoid static options and parameters.** If parameters are unavoidable (e.g., maximum packet size), it is best to have the sender and receiver negotiate a value than defining fixed choices.
7. **Look for a good design; it need not be perfect.** Often the designers have a good design but it cannot handle some weird special case. Rather than messing up the design, the designers should go with the good design and put the burden of working around it on the people with the strange requirements.
8. **Be strict when sending and tolerant when receiving.** In other words, only send packets that rigorously comply with the standards, but expect incoming packets that may not be fully conformant and try to deal with them.
9. **Think about scalability.** If the system is to handle millions of hosts and billions of users effectively, no centralized databases of any kind are tolerable and load must be spread as evenly as possible over the available resources.
10. **Consider performance and cost.** If a network has poor performance or outrageous costs, nobody will use it.

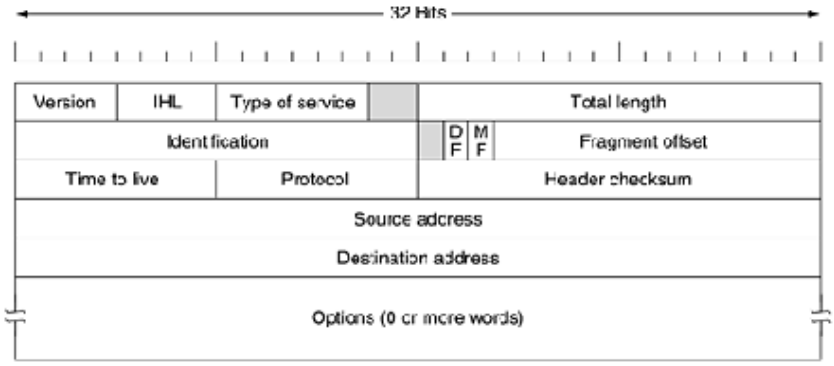
The Internet can be viewed as a collection of subnetworks or **Autonomous Systems** (**ASes**) that are interconnected. There is no real structure, but several major backbones exist. These are constructed from high-bandwidth lines and fast routers. Attached to the backbones are regional (midlevel) networks, and attached to these regional networks are the LANs at many universities, companies, and Internet service providers.

The IP Protocol

An IP datagram consists of a header part and a text part. The header has a 20-byte fixed part and a variable length optional part. The header format is shown in Fig. It is transmitted in big-endian order: from left to right, with the high-order bit of the *Version* field going first. On little endian machines, software conversion is required on both transmission and reception.

IPv4:

IPv4 is a connectionless protocol used for packet switched networks. It operates on best effort delivery model, in which neither delivery is guaranteed, nor proper sequencing or avoidance of duplicate delivery is assured. IPv4 uses 32-bit (4 byte) addressing, which gives 2^{32} addresses. IPv4 addresses are written in the dot decimal notation, which comprises of four octets of the address expressed individually in decimal and separated by periods, for instance, 192.168.1.5.



The IPv4 header

- The ***Version*** field keeps track of which version of the protocol the datagram belongs to. by including the version in each datagram
- The ***IHL*** is provided to tell how long the header is, in 32-bit words. The minimum value is 5, which applies when no options are present. The maximum value of this 4-bit field is 15, which limits the header to 60 bytes, and thus the *Options* field to 40 bytes.
- The ***Type of service*** field is one of the few fields that has changed its meaning (slightly) over the years. It was and is still intended to distinguish between different classes of service. Various combinations of reliability and speed are possible. For digitized voice, fast delivery beats accurate delivery. For file transfer, error-free transmission is more important than fast transmission Originally, the 6-bit field contained (from left to right), a three-bit *Precedence* field and three flags, *D*, *T*, and *R*. The *Precedence* field was a priority, from 0 (normal) to 7 (network control packet). The three flag bits allowed the host to specify what it cared most about from the set {Delay, Throughput, Reliability}.
- The ***Total*** **length** is the length of header and data. The maximum length is 65,535 bytes. At present, this upper limit is tolerable, but with future gigabit networks, larger datagrams may be needed.
- The ***Identification*** field is needed to allow the destination host to determine which datagram a newly arrived fragment belongs to. All the fragments of a datagram contain the same *Identification* value.
- Next comes an unused bit and then two 1-bit fields. ***DF*** stands for Don't Fragment. It is an order to the routers not to fragment the datagram because the destination is incapable of putting the pieces back together again. ***MF*** stands for More Fragments. All fragments except the last one have this bit set. It is needed to know when all fragments of a datagram have arrived.
- The ***Fragment offset*** tells where in the current datagram this fragment belongs. All fragments except the last one in a datagram must be a multiple of 8 bytes, the elementary fragment unit. Since 13 bits are provided, there is a maximum of 8192 fragments per datagram, giving a maximum datagram length of 65,536 bytes, one more than the *Total length* field.
- The ***Time to live*** field is a counter used to limit packet lifetimes. It is supposed to count time in seconds, allowing a maximum lifetime of 255 sec. It must be decremented on each hop and is supposed to be decremented multiple times when queued for a long time in a router. This feature prevents datagrams from

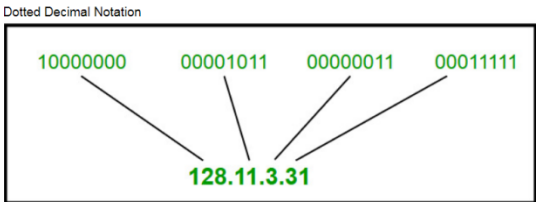
wandering around forever.

- The ***Protocol*** field tells it which transport process to give it to. TCP /UDP/or some others,
- The ***Header checksum*** verifies the header only. Such a checksum is useful for detecting errors generated by bad memory words inside a router. The algorithm is to add up all the 16-bit halfwords as they arrive, using one's complement arithmetic and then take the one's complement of the result. The *Header checksum* must be recomputed at each hop because at least one field always changes
- The ***Source address*** and ***Destination address*** indicate the network number and host number.
- The *Options* field was designed to allow subsequent versions of the protocol to include information not present in the original design, to permit experimenters to try out new ideas, and to avoid allocating header bits to information that is rarely needed. The options are variable length. Each begins with a 1-byte code identifying the option. Some options are followed by a 1-byte option length field, and then one or more data bytes. The *Options* field is padded out to a multiple of four bytes. Originally, five options were defined, as listed in following table.

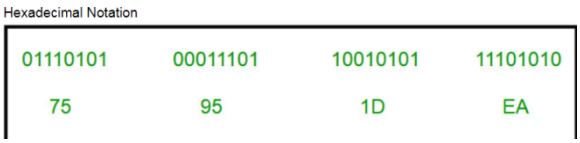
Option	Description
Security	Specifies how secret the datagram is
Strict source routing	Gives the complete path to be followed
Loose source routing	Gives a list of routers not to be missed
Record route	Makes each router append its IP address
Timestamp	Makes each router append its address and timestamp

IP addresses

- Every host and router on the Internet has an IP address, which encodes its network number and host number.
- no two machines on the Internet have the same IP address.
- All IP addresses are 32 bits long and are used in the *Source address* and destination *address* fields of IP packets.
- an IP address does not actually refer to a host, it really refers to a network interface, so if a host is on two networks, it must have two IP addresses. However, in practice, most hosts are on one network and thus have one IP address.
- IP addresses are hierarchical, unlike Ethernet addresses.
- Each 32-bit address is comprised of a variable-length network portion in the top bits and a host portion in the bottom bits.
- The network portion has the same value for all hosts on a single network, such as an Ethernet LAN. This means that a network corresponds to a contiguous block of IP address space. This block is called a **prefix**.
- IP addresses are written in **dotted decimal notation**. In this format, each of the 4 bytes is written in decimal, from 0 to 255.



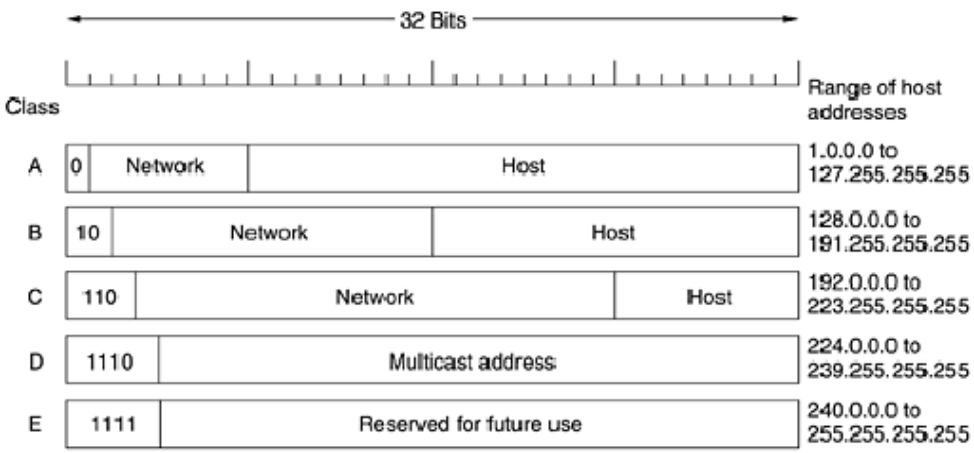
- For example, the 32-bit hexadecimal address 80D00297 is written as 128.208.2.151. Another example for hexadecimal notation is shown in the following figure.



- Prefixes are written by giving the lowest IP address in the block and the size of the block. The size is determined by the number of bits in the network portion; the remaining bits in the host portion can vary.
- This means that the size must be a power of two. By convention, it is written after the prefix IP address as a slash followed by the length in bits of the network portion. In our example, if the prefix contains 28 addresses and so leaves 24 bits for the network portion, it is written as 128.208.0.0/24.
- Since the prefix length cannot be inferred from the IP address alone, routing protocols must carry the prefixes to routers. Sometimes prefixes are simply described by their length, as in a "/16" which is pronounced "slash 16." The length of the prefix corresponds to a binary mask of 1s in the network portion. When written out this way, it is called a **subnet mask**. It can be ANDed with the IP address to extract only the network portion.
- Hierarchical addresses have significant advantages and disadvantages. The key advantage of prefixes is that routers can forward packets based on only the network portion of the address, as long as each of the networks has a unique address block. The host portion does not matter to the routers because all hosts on the same network will be sent in the same direction. It is only when the packets reach the network for which they are destined that they are forwarded to the correct host. This makes the routing tables much smaller than they would otherwise be.

Classful addressing

IP addresses were divided into the five categories listed in Fig. This allocation has come to be called **classful addressing**



- The class A, B, C, and D formats allow for up to 128 networks with 16 million hosts each, 16,384 networks with up to 64K hosts, and 2 million networks (e.g., LANs) with up to 256 hosts each (although a few of these are special).
- Also supported is multicast, in which a datagram is directed to multiple hosts.
- Addresses beginning with 1111 are reserved for future use.
- Over 500,000 networks are now connected to the Internet, and the number grows every year. Network numbers are managed by a nonprofit corporation called **ICANN** (**Internet Corporation for Assigned Names and Numbers**) to avoid conflicts. In turn, ICANN has delegated parts of the address space to various regional authorities, which then dole out IP addresses to ISPs and other companies.
- Network addresses, which are 32-bit numbers, are usually written in **dotted decimal notation**. In this format, each of the 4 bytes is written in decimal, from 0 to 255. For example, the 32-bit hexadecimal address C0290614 is written as 192.41.6.20. The lowest IP address is 0.0.0.0 and the highest is 255.255.255.255.

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

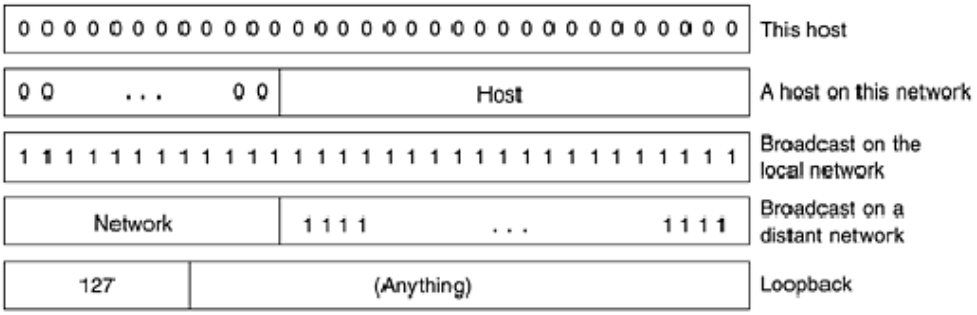


Fig. Special IP addresses

- The IP address 0.0.0.0 is used by hosts when they are being booted.
- IP addresses with 0 as network number refer to the current network. These addresses allow machines to refer to their own network without knowing its number (but they have to know its class to know how many 0s to include).
- The address consisting of all 1s allows broadcasting on the local network, typically a LAN.

- The addresses with a proper network number and all 1s in the host field allow machines to send broadcast packets to distant LANs anywhere in the Internet
- all addresses of the form 127.*xx.yy.zz* are reserved for loopback testing. Packets sent to that address are not put out onto the wire; they are processed locally and treated as incoming packets. This allows packets to be sent to the local network without the sender knowing its number.

Subnets

As we have seen, all the hosts in a network must have the same network number. This property of IP addressing can cause problems as networks grow. For example, consider a university that started out with one class B network used by the Computer Science Dept. for the computers on its Ethernet. A year later, the Electrical Engineering Dept. wanted to get on the Internet, so they bought a repeater to extend the CS Ethernet to their building. As time went on, many other departments acquired computers and the limit of four repeaters per Ethernet was quickly reached. A different organization was required. Getting a second network address would be hard to do since network addresses are scarce and the university already had enough addresses for over 60,000 hosts. The problem is the rule that a single class A, B, or C address refers to one network, not to a collection of LANs. As more and more organizations ran into this situation, a small change was made to the addressing system to deal with it.

The solution is to allow a network to be split into several parts for internal use but still act like

a single network to the outside world. A typical campus network nowadays might look like that

of following Fig. , with a main router connected to an ISP or regional network and numerous

Ethernets spread around campus in different departments. Each of the Ethernets has its own router connected to the main router (possibly via a backbone LAN, but the nature of the interrouter connection is not relevant here).

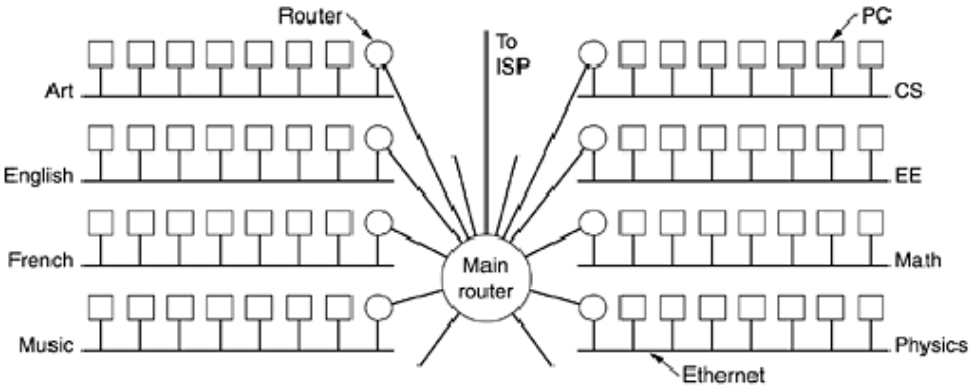


Fig.A campus network consisting of LANs for various departments

- When a packet comes into the main router, how does the router know which subnet to give it to?
- When a packet arrives, the router looks at the destination address of the packet

and checks which subnet it belongs to. The router can do this by ANDing the destination address with the mask for each subnet and checking to see if the result is the corresponding prefix.

- For example, consider a packet destined for IP address 128.208.2.151. To see if it is for the Computer Science Dept., we AND with 255.255.128.0 to take the first 17 bits (which is 128.208.0.0) and see if they match the prefix address (which is 128.208.128.0). They do not match. Checking the first 18 bits for the Electrical Engineering Dept., we get 128.208.0.0 when ANDing with the subnet mask. This does match the prefix address, so the packet is forwarded onto the interface which leads to the Electrical Engineering network
- Outside the network, the subnetting is not visible, so allocating a new subnet does not require contacting ICANN or changing any external databases.

Network Address and Mask

- Network address – It identifies a network on internet. Using this, we can find range of addresses in the network and total possible number of hosts in the network.
- Mask – It is a 32-bit binary number that gives the network address in the address block when AND operation is bitwise applied on the mask and any IP address of the block.
- The default mask in different classes are :

  Class A – 255.0.0.0

  Class B – 255.255.0.0

  Class C – 255.255.255.0
- Example : Given IP address 132.6.17.85 and default class B mask, find the beginning address (network address).
- Solution : The default mask is 255.255.0.0, which means that the only the first 2 bytes are preserved and the other 2 bytes are set to 0. Therefore, the network address is 132.6.0.0.

  Mask: 11111111 11111111 00000000 00000000
  IP: 10000100 00000110 00010001 01010101
  Then do an AND operation, it will produce
  10000100 00000110 00000000 00000000, which is equivalent to 132.6.0.0

CIDR—Classless InterDomain Routing

In principle, over 2 billion addresses exist, but the practice of organizing the address space by classes wastes millions of them. In particular, the real villain is the class B network. For most organizations, a class A network, with 16 million addresses is too big, and a class C network, with 256 addresses is too small. A class B network, with 65,536, is just right. In Internet folklore, this situation is known as the **three bears problem.**
Even if blocks of IP addresses are allocated so that the addresses are used efficiently,

there is still a problem that remains: routing table explosion. Routers in organizations at the edge of a network, such as a university, need to have an entry for each of their subnets, telling the router which line to use to get to that network. For routes to destinations outside of the organization, they can use the simple default rule of sending the packets on the line toward the ISP that connects the organization to the rest of the Internet. The other destination addresses must all be out there somewhere.
In addition, various routing algorithms require each router to transmit its tables periodically (e.g., distance vector protocols).

The solution that was implemented is **CIDR** (**Classless Inter Domain Routing**). The basic idea behind CIDR, which is described in RFC 1519, is to allocate the remaining IP addresses in variable-sized blocks, without regard to the classes. If a site needs, say, 2000 addresses, it is given a block of 2048 addresses on a 2048-byte boundary.

With CIDR, this simple algorithm of classful addressing no longer works. Instead, each routing table entry is extended by giving it a 32-bit mask. Thus, there is now a single routing table for all networks consisting of an array of (IP address, subnet mask, outgoing line) triples. When a packet comes in, its destination IP address is first extracted. Then (conceptually) the routing table is scanned entry by entry, masking the destination address and comparing it to the table entry looking for a match. It is possible that multiple entries (with different subnet mask lengths) match, in which case the longest mask is used. Thus, if there is a match for a /20 mask and a
/24 mask, the /24 entry is used.

To reduce the wastage of IP addresses in a block, we use sub-netting. What we do is that we use host id bits as net id bits of a classful IP address. We give the IP address and define the number of bits for mask along with it (usually followed by a '/' symbol), like, 192.168.1.1/28. Here, subnet mask is found by putting the given number of bits out of 32 as 1, like, in the given address, we need to put 28 out of 32 bits as 1 and the rest as 0, and so, the subnet mask would be 255.255.255.240.

Some values calculated in subnetting :

1. Number of subnets : Given bits for mask – No. of bits in default mask

2. Subnet address : AND result of subnet mask and the given IP address

3. Broadcast address : By putting the host bits as 1 and retaining the network bits as in the IP address

4. Number of hosts per subnet : $2^{(32 - \text{Given bits for mask})} - 2$

5. First Host ID : Subnet address + 1 (adding one to the binary representation of the subnet address)

6. Last Host ID : Subnet address + Number of Hosts

Example : Given IP Address – 172.16.0.0/25, find the number of subnets and the number of hosts per subnet. Also, for the first subnet block, find the subnet address, first host ID, last host ID and broadcast address.

Solution : This is a class B address. So, no. of subnets = $2^{(25-16)} = 2^9 = 512$.
No. of hosts per subnet = $2^{(32-25)} - 2 = 2^7 - 2 = 128 - 2 = 126$
For the first subnet block, we have subnet address = 0.0, first host id = 0.1, last host id = 0.126 and broadcast address = 0.127