Unit- 5

The Network Layer

Content:

I. Introduction-

The network layer is concerned with getting packets from the source all the way to the destination. Getting to the destination may require making many hops at intermediate routers along the way. This function clearly contrasts with that of the data link layer, which has the more modest goal of just moving frames from one end of a wire to the other. Thus, the network layer is the lowest layer that deals with end-to-end transmission.

To achieve its goals, the network layer must know about the topology of the communication subnet (i.e., the set of all routers) and choose appropriate paths through it. It must also take care to choose routes to avoid overloading some of the communication lines and routers while leaving others idle. Finally, when the source and destination are in different networks, new problems occur. It is up to the network layer to deal with them. In this chapter we will study all these issues and illustrate them, primarily using the Internet and its network layer protocol, IP, although wireless networks will also be addressed.

1. Network Layer Design Issues:

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

1.1 Store-and-Forward Packet Switching:

The major components of the system are the carrier's equipment (routers connected by transmission lines), shown inside the shaded oval, and the customers' equipment, shown outside the oval.

Host *H1* is directly connected to one of the carrier's routers, *A*, by a leased line. In contrast, *H2* is on a LAN with a router, *F*, owned and operated by the customer. This router also has a leased line to the carrier's equipment. We have shown *F* as being outside the oval because it does not belong to the carrier, but in terms of construction, software, and protocols, it is probably no different from the carrier's routers.

Whether it belongs to the subnet is arguable, but for the purposes of this chapter, routers on customer premises are considered part of the subnet because they run the same algorithms as the carrier's routers (and our main concern here is algorithms).

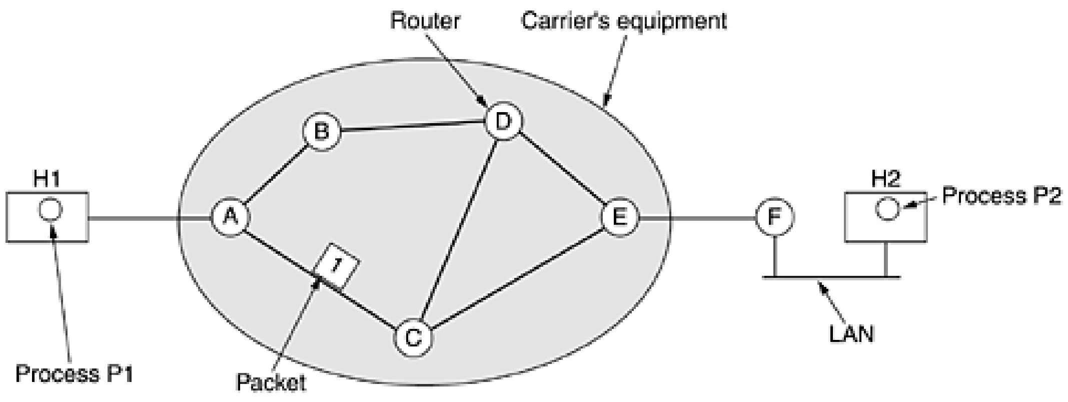


Figure-1. The environment of the network layer protocols.

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

1.2 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 what kind of services the network layer provides to the transport layer. The network layer services have been designed with the following goals in mind.

1. The services should be independent of the router technology.

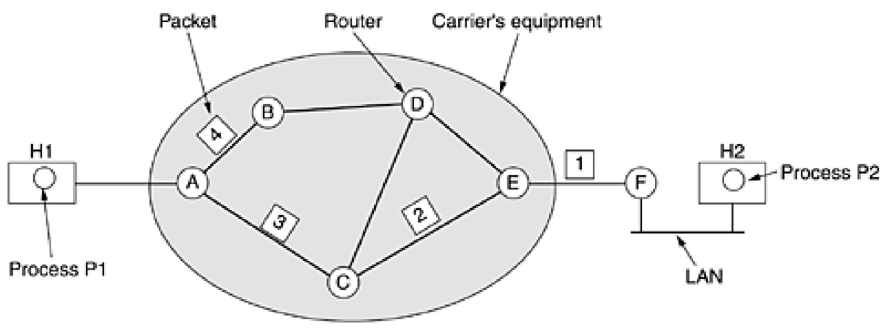
2. The transport layer should be shielded from the number, type, and topology of the routers present.

3. The network addresses made available to the transport layer should use a uniform numbering plan, even across LANs and WANs.

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 centres on whether the network layer should provide connection-oriented service or connectionless service.

5.1.3 Implementation of Connectionless Service:

If connectionless service is offered, packets are injected into the subnet individually and routed independently of each other. No advance setup is needed. In this context, the packets are frequently called **datagrams** (in analogy with telegrams) and the subnet is called a **datagram subnet**.



A's table							
initially		later		C's table		E's table	
A	–	A	–	A	A	A	C
B	B	B	B	B	A	B	D
C	C	C	C	C	–	C	C
D	B	D	B	D	D	D	D
E	C	E	B	E	E	E	–
F	C	F	B	F	E	F	F
Dest.	Line						

Figure-2. Routing within a datagram subnet.

Let us now see how a datagram subnet works. Suppose that the process *P1* in Fig-2 has a long message for *P2*. It hands the message to the transport layer with instructions to deliver it to process *P2* on host *H2*. The transport layer code runs on *H1*, typically within the operating system. It prepends a transport header to the front of the message and hands the result to the network layer, probably just another procedure within the operating system.

Let us assume that the message is four times longer than the maximum packet size, so the network layer has to break it into four packets, 1, 2, 3, and 4 and sends each of them in turn to router A using some point-to-point protocol. At this point the carrier takes over. Every router has an internal table telling it where to send packets for each possible destination. Each table entry is a pair consisting of a destination and the outgoing line to use for that destination. Only directly-connected lines can be used.

For example, in Fig-2, A has only two outgoing lines—to B and C—so every incoming packet must be sent to one of these routers, even if the ultimate destination is some other router. A's initial routing table is shown in the figure under the label ''initially.''

As they arrived at A, packets 1, 2, and 3 were stored briefly (to verify their checksums). Then each was forwarded to C according to A's table. Packet 1 was then forwarded to E and then to F. When it got to F, it was encapsulated in a data link layer frame and sent to H2 over the LAN. Packets 2 and 3 follow the same route.

However, something different happened to packet 4. When it got to A it was sent to router B, even though it is also destined for F. For some reason, A decided to send packet 4 via a different route than that of the first three. Perhaps it learned of a traffic jam somewhere along the ACE path and updated its routing table, as shown under the label ''later.'' The algorithm that manages the tables and makes the routing decisions is called the routing algorithm.

1.4 Implementation of Connection-Oriented Service:

If connection-oriented service is used, a path from the source router to the destination router must be established before any data packets can be sent. This connection is called a **VC** (**virtual circuit**), in analogy with the physical circuits set up by the telephone system, and the subnet is called a **virtual-circuit subnet**.

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

As an example, consider the situation of Fig-3. Here, host H1 has established connection 1 with host H2. It is remembered as the first entry in each of the routing tables. The first line of A's table says that if a packet bearing connection identifier 1 comes in from H1, it is to be sent to router C and given connection identifier 1. Similarly, the first entry at C routes the packet to E, also with connection identifier 1.

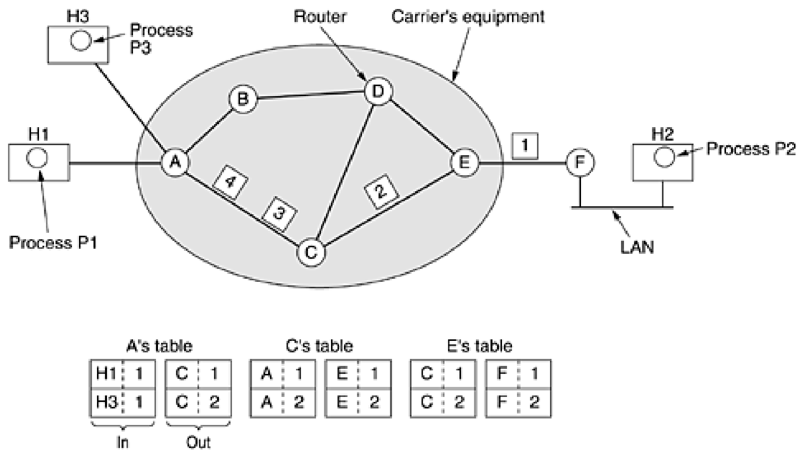
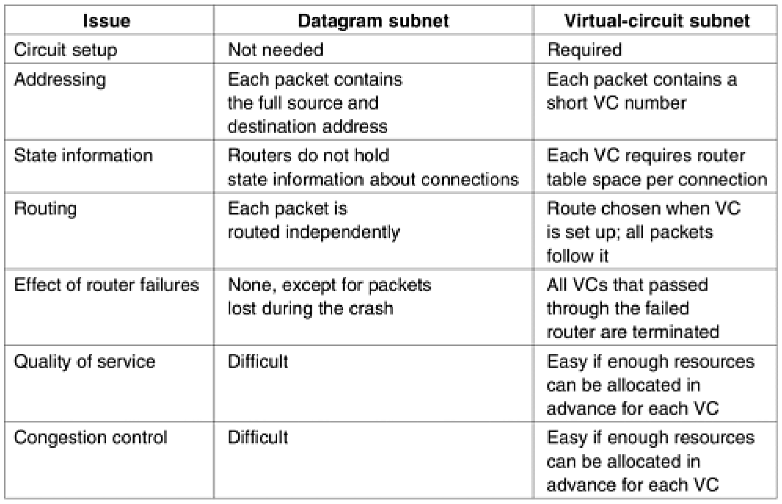


Figure-3. Routing within a virtual-circuit subnet.

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

1.5 Comparison of Virtual-Circuit and Datagram Subnets:



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

2. Routing Algorithms:

Routing is concerned with the question: Which line should router *J* use when forwarding a packet to router *K*?

There are two types of algorithms:

- *Adaptive* algorithms use such dynamic information as current topology, load, delay, etc. to select routes.
- In *nonadaptive* algorithms, routes never change once initial routes have been selected. Also called *static* routing.

Obviously, adaptive algorithms are more interesting, as nonadaptive algorithms don't even make an attempt to handle failed links.

Adaptive algorithms can be further divided in the following types:

1. *Isolated*: each router makes its routing decisions using only the *local* information it has on hand. Specifically, routers do not even exchange information with their neighbors.
2. *Centralized*: a centralized node makes all routing decisions. Specifically, the centralized node has access to *global* information.
3. *Distributed*: algorithms that use a combination of local and global information.

Non-Adaptive Algorithms

Flooding is a form of isolated routing. Does not select a specific route. When a router receives a packet, it sends a copy of the packet out on each line (except the one on which it arrived):

- To prevent packets from looping forever, each router decrements a hop count contained in the packet header. Whenever the hop count decrements to zero, the router discards the packet.
- To reduce looping even further:

1. Add a sequence number to each packet's header.
2. Each router maintains a private sequence number. When it sends a new packet, it copies the sequence number into the packet, and increments its private sequence number.
3. For each source router *S*, a router:
    - keeps track of the highest sequence number seen from *S*.
    - whenever it receives a packet from *S* containing a sequence number lower than the one stored in its table, it discards the packet.
    - otherwise, it updates the entry for *S* and forwards the packet on

Flooding has several important uses:

1. in military applications, the network must remain robust in the face of (extreme) hostility
2. sending routing updates, because updates can't rely on the correctness of a router's routing table.
3. theoretical--chooses all possible paths, so it chooses the shortest one.

In *selective flooding*, a router sends packets out only on those lines in the general direction of the destination. That is, don't send packets out on lines that clearly lead in the wrong direction.

In *random walk* a router selects one or more lines at random.

- equalize load (on network)
- robust

Shortest Path Algorithm

What if we ``know'' the complete topology of the network? Can look at computing the optimal path. What if we have the following network and we want to route a packet from node A to node G. What is the shortest path (do not initially show distance.

Use Dijkstra's algorithm (or variation). Basic idea:

- Choose the source, and put nodes connected to source in list to consider.
- From the list to consider choose the nearest node.

Algorithm results:

nodes	list to consider
-----	----------------
(A, -, 0)	B, D, E
(B, A, 2)	C, D, E
(C, B, 3)	D, E, H
(H, C, 4)	D, E, G
(D, A, 5)	E, F, G
(E, C, 6)	G, F
(G, H, 7)	F
(F, E, 7)	-

Guaranteed to get the shortest path? How to prove? If an alternate shorter path to a node then we would have already tried the path.

Static Routing (Multipath Routing)

May not be a single *best* route between two nodes. Can use a routing table. Can compute alternate routes between source and destination by first computing shortest path and then removing these links.

Problem is that these routing tables are initially created and not changed.

Adaptive/Dynamic Routing

Centralized--use a *routing control center*. Creates, modifies, and distributes routing tables to other routers. Gathers information from the routers.

Good: adaptive routing, relieve burden on the routers of computing tables.

Problems:

- Does not adapt quickly.
- Quicker the adaption, the more overhead it causes.
- Synchronization of updates (some routers change, but not others, send at each other).
- If RCC crashes the network becomes stale.
- Lines near the RCC are overloaded.

Decentralized Routing

Isolated Routing

Base decisions on local traffic and conditions.

- hot potato--choose output line with the shortest queue
- backward learning--each packet contains source address and number of hops so far. Use this information to learn shortest path to each source. Will learn shortest path to all routers. What is wrong? Only deal with good news, not bad. ``Good'' may no longer be good due to down router or congestion.

Must periodically forget and start over (with suboptimal performance after a purge).

Delta Routing

Combine centralized and isolated routing. RCC (routing control center) computes optimal paths for each router as before. Has a parameter δ to decide if two lines are equivalent. Are equivalent if the delay is within δ of each other.

$\delta \to 0$, centralized
$\delta \to \infty$, isolated, up to the routers.

Modern networks use two dynamic (adaptive) algorithms--distance vector routing and link state routing.

Distance Vector Routing

Distributed routing algorithm (Old Arpanet Routing Algorithm). routers work together.

1. Each router maintains a table (vector) giving the best-known distance to a destination and the line to use for sending there. Tables are updated by exchanging information with neighbors.
2. Each router knows the distance (cost) of reaching its neighbors (e.g. send echo requests).
3. routers periodically exchange routing tables with each of their neighbors.
4. Upon receipt of an update, for each destination in its table, a router:
    1. compares the metric in its local table with the metric in the neighbor's table plus the cost of reaching that neighbor
    2. if the path via the neighbor has a lower cost, the router updates its local table to forward packets to the neighbor

This algorithm was used in the original ARPANET. Unfortunately, it suffers from the problem: good news travels quickly, bad news travels slowly (count-to-infinity problem).

The fundamental problem with the old Arpanet algorithm is that it continues to use ``old'' information that is invalid, even after newer information becomes available.

Link State Routing

The ``old'' Arpanet routing algorithm was replaced in 1979. Problems with old algorithm included:

1. High-priority routing update packets were large, adversely affecting traffic.
2. Network was too slow in adapting to congestion, too fast to react to minor changes.
3. Average queue length was used to estimate delay. This works only if all lines have the same capacity and propagation delay, and doesn't take into account that packets have varying sizes.

In the new algorithm:

1. Each router maintains a database describing the topology and link delays between each router. That is, each router keeps track of the full graph of links and nodes.
2. Each router periodically discovers it neighbors (sends ``hello'' message on booting) and measures the delays across its links (echo requests--should load be taken into account?), then forwards that information to all other routers (link state packets)
3. Updates are propagated at high priority using flooding. Updates contain sequence numbers, and a router forwards ``new'' copies of the packet.

Why use flooding? Because that way routing updates propagate even when routing tables aren't quite correct. ACKs are sent to neighbors.

4. Each router uses an SPF algorithm to calculate shortest paths based on the current values in its database.
5. Because each router makes its calculation using the same information, better routing decisions are made.

3. Congestion Control Algorithms:

Introduction:

As Internet can be considered as a *Queue of packets,* where transmitting nodes are constantly adding packets and some of them (receiving nodes) are removing packets from the queue. So, consider a situation where too many packets are present in this queue (or internet or a part of internet), such that constantly transmitting nodes are pouring packets at a higher rate than receiving nodes are removing them. This degrades the performance, and such a situation is termed as *Congestion*. Main reason of congestion is more number of packets into the network than it can handle. So, the objective of congestion control can be summarized as to maintain the number of packets in the network below the level at which performance falls off dramatically. The nature of a Packet switching network can be summarized in following points:

A network of queues:

At each node, there is a queue of packets for each outgoing channel

If packet arrival rate exceeds the packet transmission rate, the queue size grows without bound

When the line for which packets are queuing becomes more than 80% utilized, the queue length grows alarmingly. When the number of packets dumped into the network is within the carrying capacity, they all are delivered, expect a few that have too be rejected due to transmission errors). And then the number delivered is proportional to the number of packets sent. However, as traffic increases too far, the routers are no longer able to cope, and they begin to lose packets. This tends to make matter worse. At very high traffic, performance collapse completely, and almost no packet is delivered. In the following sections, the causes of congestion, the effects of congestion and various congestion control techniques are discussed in detail.

Causes Of Congestion:

Congestion can occur due to several reasons. For example, 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. Similarly, *Low-Bandwidth* 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. The real problem is the mismatch between different parts of the system.

Congestion tends to feed upon itself to get even worse. 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.

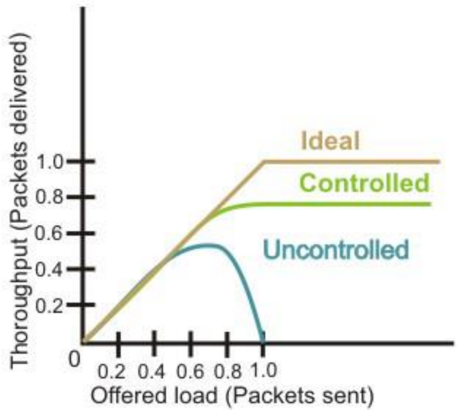
So, the major cause of congestion is often the *bursty* nature of traffic. If the hosts could be made to transmit at a uniform rate, then congestion problem will be less common and all other causes will not even led to congestion because other causes just act as an enzyme which boosts up the congestion when the traffic is bursty (i.e., other causes just add on to make the problem more serious, main cause is the bursty traffic).

This means that when a device sends a packet and does not receive an acknowledgment from the receiver, in most the cases it can be assumed that the packets have been dropped by intermediate devices due to congestion. By detecting the rate at which segments are sent and not acknowledged, the source or an intermediate router can infer the level of congestion on the network. In the following section we shall discuss the ill effects of congestion.

Effects of Congestion:

Congestion affects two vital parameters of the network performance, namely *throughput* and *delay*. In simple terms, the throughput can be defined as the percentage utilization of the network capacity. Figure (a) shows how throughput is affected as offered load increases. Initially throughput increases linearly with offered load, because utilization of the network increases. However, as the offered load increases beyond certain limit, say 60% of the capacity of the network, the throughput drops. If the offered load increases further, a point is reached when not a single packet is delivered to any destination, which is commonly known as *deadlock* situation. There are three curves in Fig.(a), the ideal one corresponds to the situation when all the packets introduced are delivered to their destination up to the maximum capacity of the network. The second one corresponds to the situation when there is no congestion control. The third one is the case when some congestion control technique is used. This prevents the throughput collapse, but provides lesser throughput than the ideal condition due to overhead of the congestion control technique.

The delay also increases with offered load, as shown in Fig.(b). And no matter what technique is used for congestion control, the delay grows without bound as the load approaches the capacity of the system. It may be noted that initially there is longer delay when congestion control policy is applied. However, the network without any congestion control will saturate at a lower offered load.



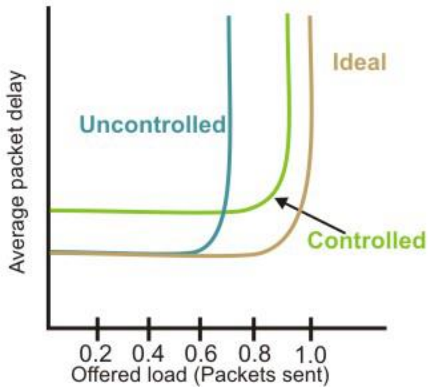


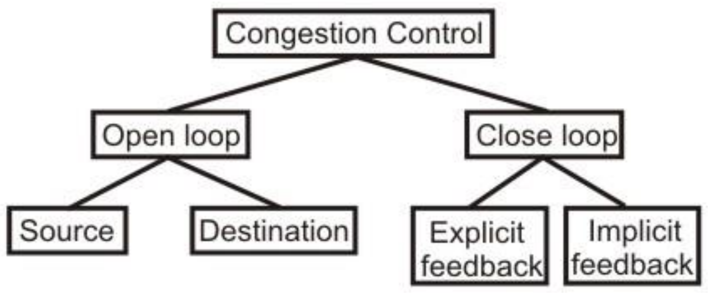
Figure (a) Effect of congestion on throughput (b) Effect of congestion on delay

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 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.

Close loop: Protocols that allow system to enter congested state, detect it, and remove it.



The first category of solutions or protocols attempt to solve the problem by a good design, at first, to make sure that it doesn't occur at all. Once system is up and running midcourse corrections are not made. These solutions are somewhat static in nature, as the policies to control congestion don't change much according to the current state of the system. Such Protocols are also known as *Open Loop* solutions. These rules or policies 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.

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.
- To pass this information to the places where actions can be taken
- Adjust the system operation to correct the problem.

These solutions are known as *Closed Loop* solutions. 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. A general feedback step would be, say 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.

Yet 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.

In the following sections we shall discuss about some of the popular algorithms from the above categories.

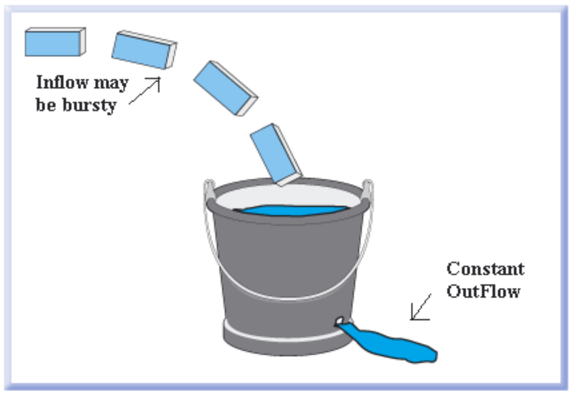
Leaky Bucket Algorithm:

Consider a Bucket with a small hole at the bottom, whatever may be the rate of water pouring into the bucket, the rate at which water comes out from that small hole is constant. This scenario is depicted in figure(a). Once the bucket is full, any additional water entering it spills over the sides and is lost (i.e. it doesn’t appear in the output stream through the hole underneath).

The same idea of leaky bucket can be applied to packets, as shown in Fig. (b). Conceptually each network interface contains a *leaky bucket*. And the following steps are performed:

When the host has to send a packet, the packet is thrown into the bucket.
The bucket leaks at a constant rate, meaning the network interface transmits packets at a constant rate.
Bursty traffic is converted to a uniform traffic by the leaky bucket.
In practice the bucket is a finite queue that outputs at a finite rate.

This arrangement can be simulated in the operating system or can be built into the hardware. Implementation of this algorithm is easy and consists of a finite queue. Whenever a packet arrives, if there is room in the queue it is queued up and if there is no room then the packet is discarded.



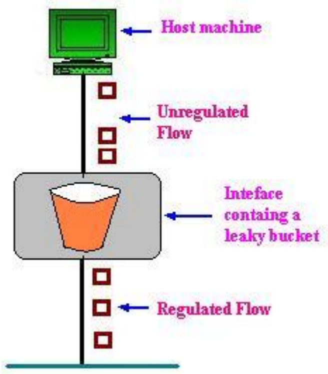


Figure (a) Leaky bucket (b) Leaky bucket implementation

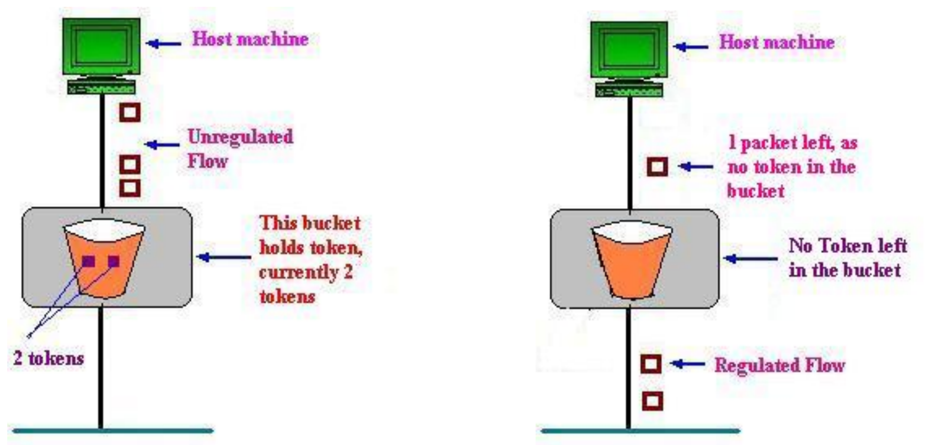
Token Bucket Algorithm:

The leaky bucket algorithm described above, enforces a rigid pattern at the output stream, irrespective of the pattern of the input. For many applications it is better to allow the output to speed up somewhat when a larger burst arrives than to loose the data. Token Bucket algorithm provides such a solution. In this algorithm leaky bucket holds token, generated at regular intervals. Main steps of this algorithm can be described as follows:

- In regular intervals tokens are thrown into the bucket.
- The bucket has a maximum capacity.
- If there is a ready packet, a token is removed from the bucket, and the packet is send.
- If there is no token in the bucket, the packet cannot be send.

Figure, shows the two scenarios before and after the tokens present in the bucket have been consumed. In Fig.(a) the bucket holds two tokens, and three packets are waiting to be sent out of the interface, in Fig.(b) two packets have been sent out by consuming two tokens, and 1 packet is still left.

The token bucket algorithm is less restrictive than the leaky bucket algorithm, in a sense that it allows bursty traffic. However, the limit of burst is restricted by the number of tokens available in the bucket at a particular instant of time.


Host machine
Unregulated Flow
This bucket holds token, currently 2 tokens
2 tokens
Host machine
1 packet left, as no token in the bucket
No Token left in the bucket
Regulated Flow

Figure(a) Token bucket holding two tokens, before packets are send out, (b) Token bucket after two packets are send, one packet still remains as no token is left

Congestion control in virtual Circuit:

Till now we have discussed two open loop algorithms, where the policy decisions are made in the beginning, irrespective of the current state. Both leaky bucket algorithm and token bucket algorithm are open loop algorithms.

In this section we shall have a look at how the congestion is tackled in a virtual-circuit network. *Admission control* is one such closed-loop technique, where action is taken once congestion is detected in the network. Different approaches can be followed:

- Simpler one being: do not set-up new connections, once the congestion is signalled. This type of approach is often used in normal telephone networks. When the exchange is overloaded, then no new calls are established.

- Another approach, which can be followed is: to allow new virtual connections, but route these carefully so that none of the congested router (or none of the problem area) is a part of this route.
- Yet another approach can be: To negotiate different parameters between the host and the network, when the connection is setup. During the setup time itself, Host specifies the volume and shape of traffic, quality of service, maximum delay and other parameters, related to the traffic it would be offering to the network. Once the host specifies its requirement, the resources needed are reserved along the path, before the actual packet follows.

Choke Packet Technique:

The *choke packet* technique, a closed loop control technique, can be applied in both virtual circuit and datagram subnets. Each router monitors its resources and the utilization at each of its output line. There is a threshold set by the administrator, and whenever any of the resource utilization crosses this threshold and action is taken to curtail down this. Actually each output line has a utilization associated with it, and whenever this utilization crosses the threshold, the output line enters a “warning” state. If so, the router sends a *choke packet* back to the source, giving it a feedback to reduce the traffic. And the original packet is tagged (a bit is manipulated in the header field) so that it will not generate other choke packets by other intermediate router, which comes in place and is forwarded in usual way. It means that the first router (along the way of a packet), which detects any kind of congestion, is the only one that sends the choke packets.

When the source host gets the choke packet, it is required to reduce down the traffic send out to that particular destination (choke packet contains the destination to which the original packet was send out). After receiving the choke packet the source reduces the traffic by a particular fixed percentage, and this percentage decreases as the subsequent choke packets are received. Figure, depicts the functioning of choke packets.

For Example, when source A receives a choke packet with destination B at first, it will curtail down the traffic to destination B by 50%, and if again after affixed duration of time interval it receives the choke packet again for the same destination, it will further curtail down the traffic by 25% more and so on. As stated above that a source will entertain another subsequent choke packet only after a fixed interval of time, not before that. The reason for this is that when the first choke packet arrives at that point of time other packets destined to the same destination would also be there in the network and they will generate other choke packets too, the host should ignore these choke packets which refer to the same destination for a fixed time interval.

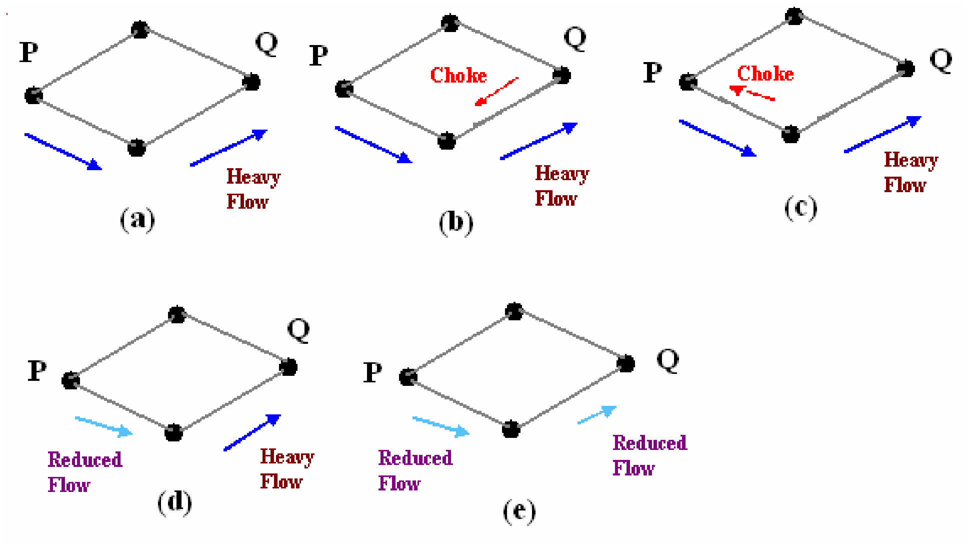


Figure: Depicts the functioning of choke packets, (a) Heavy traffic between nodes P and Q, (b) Node Q sends the Choke packet to P, (c) Choke packet reaches P, (d) P reduces the flow and send a reduced flow out, (e) Reduced flow reaches node Q

Hop-by Hop Choke Packets:

This technique is an advancement over Choked packet method. At high speed over long distances, sending a packet all the way back to the source doesn't help much, because by the time choke packet reach the source, already a lot of packets destined to the same original destination would be out from the source. So to help this, Hop-by-Hop Choke packets are used. In this approach, the choke packet affects each and every intermediate router through which it passes by. Here, as soon as choke packet reaches a router back to its path to the source, it curtails down the traffic between those intermediate routers. In this scenario, intermediate nodes must dedicate few more buffers for the incoming traffic as the outflow through that node will be curtailed down immediately as choke packet arrives it, but the input traffic flow will only be curtailed down when choke packet reaches the node which is before it in the original path. This method is illustrated in Fig.

As compared to choke packet technique, hop-by-hop choke packet algorithm is able to restrict the flow rapidly. One-step reduction is seen in controlling the traffic, this single step advantage is because in our example there is only one intermediate router. Hence, in a more complicated network, one can achieve a significant advantage by using hop-by-hop choke packet method.

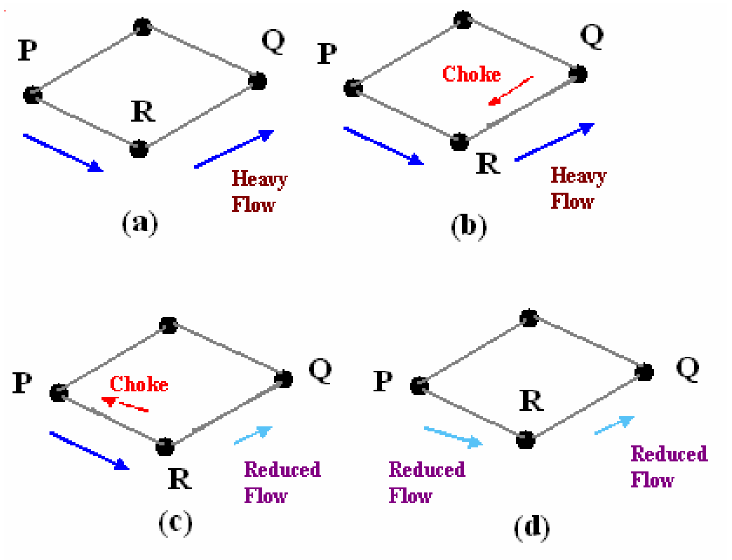


Figure: Depicts the functioning of Hop-by-Hop choke packets, (a) Heavy traffic between nodes P and Q, (b) Node Q sends the Choke packet to P, (c) Choke packet reaches R, and the flow between R and Q is curtail down, Choke packer reaches P, and P reduces the flow out

Load Shedding:

Another simple closed loop technique is *Load Shedding*; it is one of the simplest and more effective techniques. In this method, whenever a router finds that there is congestion in the network, it simply starts dropping out the packets. There are different methods by which a host can find out which packets to drop. Simplest way can be just choose the packets randomly which has to be dropped. More effective ways are there but they require some kind of cooperation from the sender too. For many applications, some packets are more important than others. So, sender can mark the packets in priority classes to indicate how important they are. If such a priority policy is implemented than intermediate nodes can drop packets from the lower priority classes and use the available bandwidth for the more important packets.

Slow Start - a Pro-active technique:

This is one of the pro-active techniques, which is used to avoid congestion. In the original implementation of TCP, as soon as a connection was established between two devices, they could each go “hog wild”, sending segments as fast as they liked as long as there was room in the other devices receive window. In a busy internet, the sudden appearance of a large amount of new traffic could aggravate any existing congestion.

To alleviate this, modern TCP devices are restrained in the rate at which they initially send segments. Each sender is at first restricted to sending only an amount of data equal to one “full-sized” segment—that is, equal to the MSS (maximum segment size) value for the connection. Each time an acknowledgment is received, the amount of data the device can send is increased by the size of another full-sized segment. Thus, the device “starts slow” in terms of how much data it can send, with the amount it sends increasing until either the full window size is reached or congestion is detected on the link. In the latter case, the congestion avoidance feature is used.

When potential congestion is detected on a TCP link, a device responds by throttling back the rate at which it sends segments. A special algorithm is used that allows the device to drop the rate at which segments are sent quickly when congestion occurs. The device then uses the *Slow Start* algorithm just above to gradually increase the transmission rate back up again to try to maximize throughput without congestion occurring again.

Flow Control Versus Congestion control:

Let’s have a look at the difference between *Flow Control* and *Congestion Control*, which are mixed up at times.

Flow control is a very important part of regulating the transmission of data between devices, but it is limited in a way that it only considers what is going on within each of the devices on the connection, and not what is happening in devices between them. It relates to the point-point traffic between a given sender and a receiver. Flow control always involves some kind of feedback from receiver to sender to tell sender how things are at other end of the network. Since we are dealing with how TCP works between a typical server and client at layer four, we don't worry about how data gets between them; that's the job of the Internet Protocol at layer three.

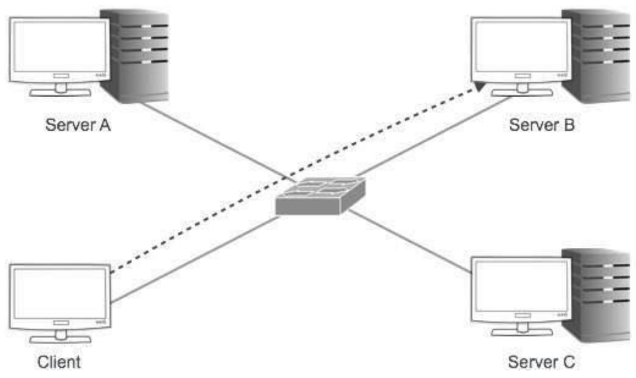
In practice, what is going on at layer three can be quite important. Considered from an abstract point of view, our server and client may be connected “directly” using TCP, but all the packets we transmit are carried across an internet and routers between different networks. These networks and routers are also carrying data from many other connections and higher-layer protocols. If the internet becomes very busy, the speed at which segments are carried between the endpoints of our connection will be reduced, and they could even be dropped. This is called *congestion control*. Congestion control has to do with making sure that subnet carry the offered traffic. It is the global issue, involving the behaviour of all the hosts, router, link, store and forward mechanism between them in the entire subnet or internet.

4. IPV4 Address:

Addressing Modes:

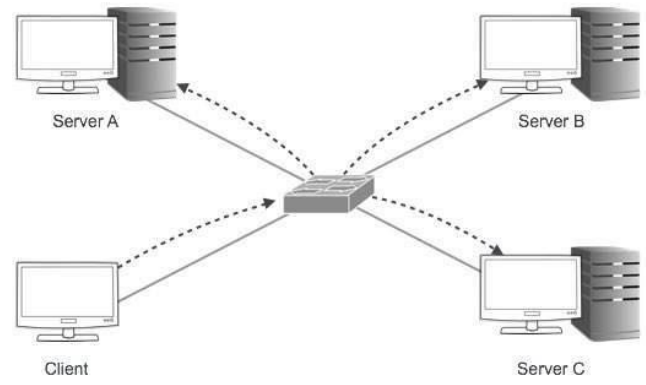
- Unicast Addressing Mode:

In this mode, data is sent only to one destined host. The Destination Address field contains 32- bit IP address of the destination host. Here the client sends data to the targeted server −



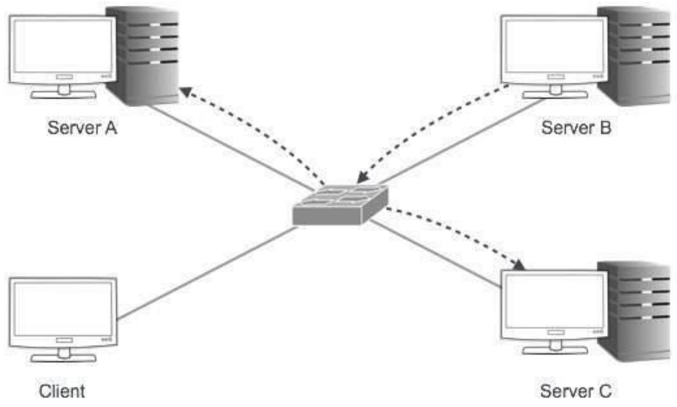
- Broadcast Addressing Mode:

In this mode, the packet is addressed to all the hosts in a network segment. The Destination Address field contains a special broadcast address, i.e. **255.255.255.255**. When a host sees this packet on the network, it is bound to process it. Here the client sends a packet, which is entertained by all the Servers −



- Multicast Addressing Mode:

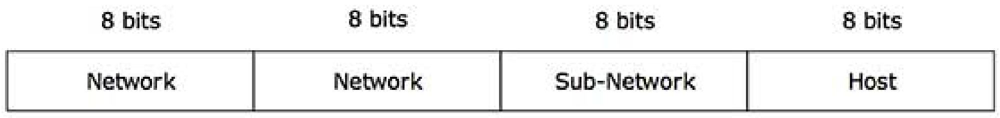
This mode is a mix of the previous two modes, i.e. the packet sent is neither destined to a single host nor all the hosts on the segment. In this packet, the Destination Address contains a special address which starts with 224.x.x.x and can be entertained by more than one host.



Here a server sends packets which are entertained by more than one servers. Every network has one IP address reserved for the Network Number which represents the network and one IP address reserved for the Broadcast Address, which represents all the hosts in that network.

- Hierarchical Addressing Scheme:

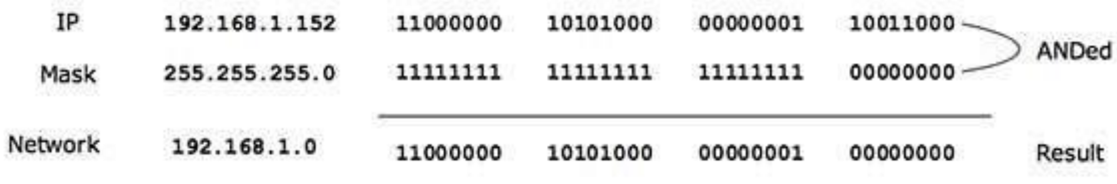
IPv4 uses hierarchical addressing scheme. An IP address, which is 32-bits in length, is divided into two or three parts as depicted −



A single IP address can contain information about the network and its sub-network and ultimately the host. This scheme enables the IP Address to be hierarchical where a network can have many sub-networks which in turn can have many hosts.

- Subnet Mask

The 32-bit IP address contains information about the host and its network. It is very necessary to distinguish both. For this, routers use Subnet Mask, which is as long as the size of the network address in the IP address. Subnet Mask is also 32 bits long. If the IP address in binary is ANDed with its Subnet Mask, the result yields the Network address. For example, say the IP Address is 192.168.1.152 and the Subnet Mask is 255.255.255.0 then −

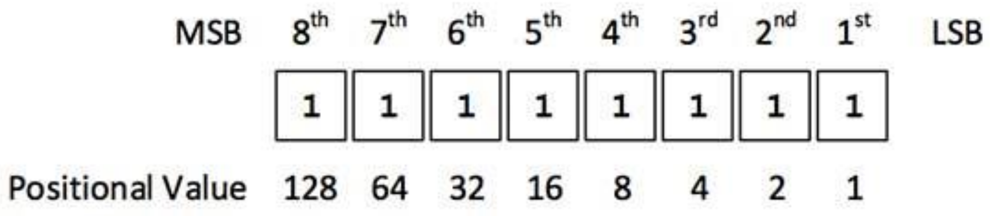


IP	192.168.1.152	11000000	10101000	00000001	10011000	ANDed
Mask	255.255.255.0	11111111	11111111	11111111	00000000	
Network	192.168.1.0	11000000	10101000	00000001	00000000	Result

This way the Subnet Mask helps extract the Network ID and the Host from an IP Address. It can be identified now that 192.168.1.0 is the Network number and 192.168.1.152 is the host on that network.

- Binary Representation

The positional value method is the simplest form of converting binary from decimal value. IP address is 32 bit value which is divided into 4 octets. A binary octet contains 8 bits and the value of each bit can be determined by the position of bit value '1' in the octet.



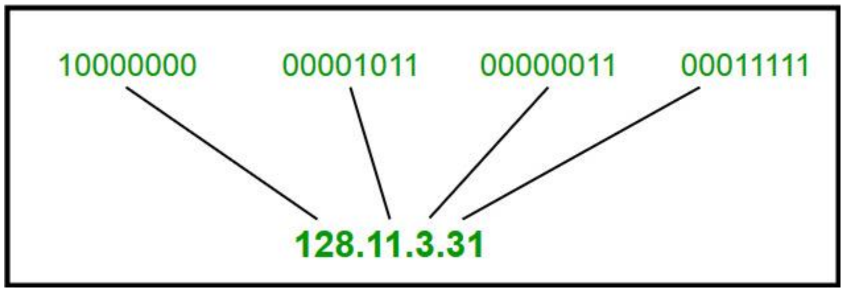
Positional value of bits is determined by 2 raised to power (position – 1), that is the value of a bit 1 at position 6 is 2^(6-1) that is 2^5 that is 32. The total value of the octet is determined by adding up the positional value of bits. The value of 11000000 is 128+64 = 192. Some examples are shown in the table below −

128	64	32	16	8	4	2	1	Value
0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	0	2
0	0	0	0	0	0	1	1	3
0	0	0	0	0	1	0	0	4
0	0	0	0	0	1	0	1	5
0	0	0	0	0	1	1	0	6
0	0	0	0	0	1	1	1	7
0	0	0	0	1	0	0	0	8
0	0	0	0	1	0	0	1	9
0	0	0	0	1	0	1	0	10
0	0	0	1	0	0	0	0	16
0	0	1	0	0	0	0	0	32
0	1	0	0	0	0	0	0	64
0	1	1	0	0	1	0	0	100
0	1	1	1	1	1	1	1	127
1	0	0	0	0	0	0	0	128
1	0	1	0	1	0	0	0	168
1	1	0	0	0	0	0	0	192
1	1	1	1	1	1	1	1	255

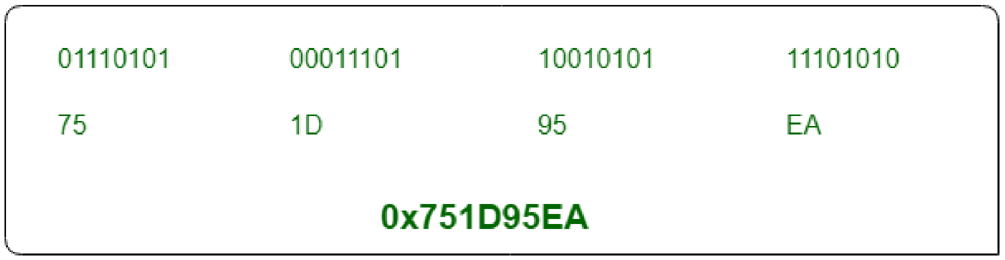
Introduction of Classful IP Addressing

IP address is an address having information about how to reach a specific host, especially outside the LAN. An IP address is a 32 bit unique address having an address space of 2^{32}. Generally, there are two notations in which IP address is written, dotted decimal notation and hexadecimal notation.

Dotted Decimal Notation:



Hexadecimal Notation:



Some points to be noted about dotted decimal notation:

1. The value of any segment (byte) is between 0 and 255 (both included).
2. There are no zeroes preceding the value in any segment (054 is wrong, 54 is correct).

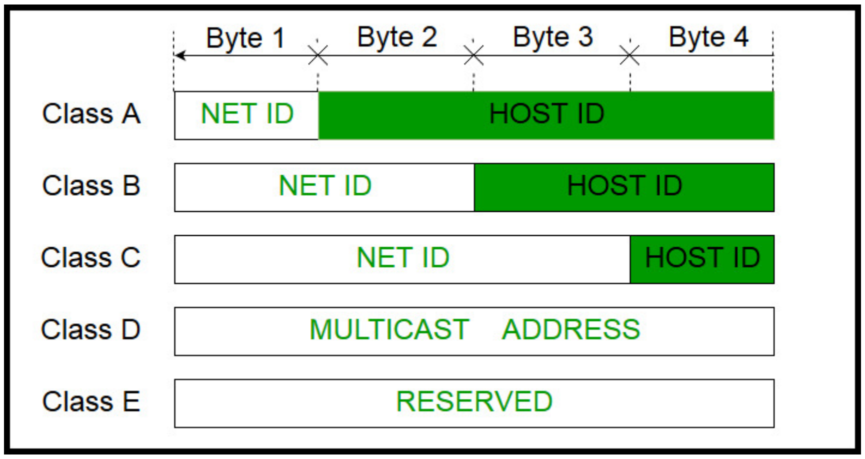
Classful Addressing
The 32 bit IP address is divided into five sub-classes. These are:

- Class A
- Class B
- Class C
- Class D
- Class E

Each of these classes has a valid range of IP addresses. Classes D and E are reserved for multicast and experimental purposes respectively. The order of bits in the first octet determine the classes of IP address.
IPv4 address is divided into two parts:

- Network ID
- Host ID

The class of IP address is used to determine the bits used for network ID and host ID and the number of total networks and hosts possible in that particular class. Each ISP or network administrator assigns IP address to each device that is connected to its network.



Note: IP addresses are globally managed by Internet Assigned Numbers Authority(IANA) and regional Internet registries(RIR).

Note: While finding the total number of host IP addresses, 2 IP addresses are not counted and are therefore, decreased from the total count because the first IP address of any network is the network number and whereas the last IP address is reserved for broadcast IP.

Class A:

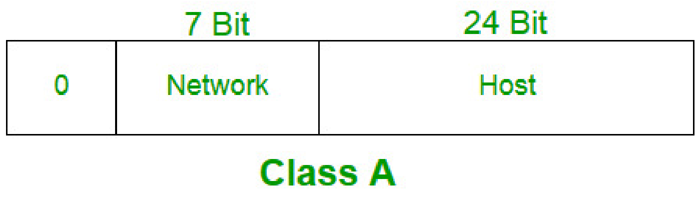
IP address belonging to class A are assigned to the networks that contain a large number of hosts.

- The network ID is 8 bits long.
- The host ID is 24 bits long.

The higher order bit of the first octet in class A is always set to 0. The remaining 7 bits in first octet are used to determine network ID. The 24 bits of host ID are used to determine the host in any network. The default subnet mask for class A is 255.x.x.x. Therefore, class A has a total of:

- 2^7-2= 126 network ID(Here 2 address is subtracted because 0.0.0.0 and 127.x.y.z are special address.)
- 2^24 – 2 = 16,777,214 host ID

IP addresses belonging to class A ranges from 1.x.x.x – 126.x.x.x



Class B:

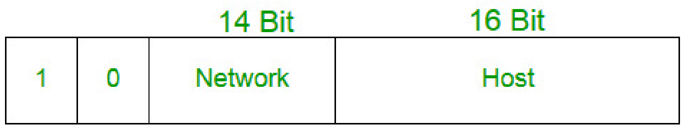
IP address belonging to class B are assigned to the networks that ranges from medium-sized to large-sized networks.

- The network ID is 16 bits long.
- The host ID is 16 bits long.

The higher order bits of the first octet of IP addresses of class B are always set to 10. The remaining 14 bits are used to determine network ID. The 16 bits of host ID is used to determine the host in any network. The default sub-net mask for class B is 255.255.x.x. Class B has a total of:

- 2^14 = 16384 network address
- 2^16 – 2 = 65534 host address

IP addresses belonging to class B ranges from 128.0.x.x – 191.255.x.x.



Class B

Class C:

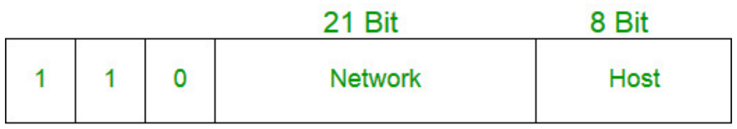
IP address belonging to class C are assigned to small-sized networks.

- The network ID is 24 bits long.
- The host ID is 8 bits long.

The higher order bits of the first octet of IP addresses of class C are always set to 110. The remaining 21 bits are used to determine network ID. The 8 bits of host ID is used to determine the host in any network. The default sub-net mask for class C is 255.255.255.x. Class C has a total of:

- 2^21 = 2097152 network address
- 2^8 – 2 = 254 host address

IP addresses belonging to class C ranges from 192.0.0.x – 223.255.255.x.

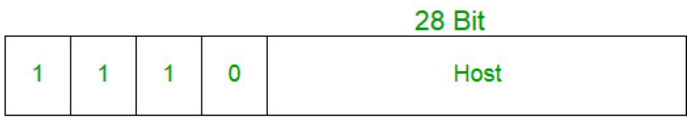


Class C

Class D:

IP address belonging to class D are reserved for multi-casting. The higher order bits of the first octet of IP addresses belonging to class D are always set to 1110. The remaining bits are for the address that interested hosts recognize.

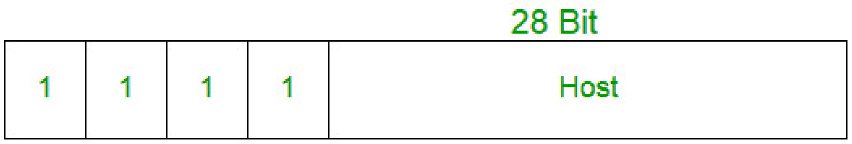
Class D does not posses any sub-net mask. IP addresses belonging to class D ranges from 224.0.0.0 – 239.255.255.255.



Class D

Class E:

IP addresses belonging to class E are reserved for experimental and research purposes. IP addresses of class E ranges from 240.0.0.0 – 255.255.255.254. This class doesn't have any sub-net mask. The higher order bits of first octet of class E are always set to 1111.



Class E

Range of special IP addresses:

169.254.0.0 – 169.254.0.16 : Link local addresses
127.0.0.0 – 127.0.0.8 : Loop-back addresses
0.0.0.0 – 0.0.0.8 : used to communicate within the current network.

Rules for assigning Host ID:

Host ID's are used to identify a host within a network. The host ID are assigned based on the following rules:

- Within any network, the host ID must be unique to that network.
- Host ID in which all bits are set to 0 cannot be assigned because this host ID is used to represent the network ID of the IP address.
- Host ID in which all bits are set to 1 cannot be assigned because this host ID is reserved as a broadcast address to send packets to all the hosts present on that particular network.

Rules for assigning Network ID:

Hosts that are located on the same physical network are identified by the network ID, as all host on the same physical network is assigned the same network ID. The network ID is assigned based on the following rules:

- The network ID cannot start with 127 because 127 belongs to class A address and is reserved for internal loop-back functions.
- All bits of network ID set to 1 are reserved for use as an IP broadcast address and therefore, cannot be used.
- All bits of network ID set to 0 are used to denote a specific host on the local network and are not routed and therefore, aren’t used.

Classless Addressing

Classless Inter-Domain Routing (CIDR) is another name for classless addressing. This addressing type aids in the more efficient allocation of IP addresses. This technique assigns a block of IP addresses based on specified conditions when the user demands a specific amount of IP addresses. This block is known as a "CIDR block", and it contains the necessary number of IP addresses.

When allocating a block, classless addressing is concerned with the following three rules.

- **Rule 1** − The CIDR block's IP addresses must all be contiguous.
- **Rule 2** − The block size must be a power of two to be attractive. Furthermore, the block's size is equal to the number of IP addresses in the block.
- **Rule 3** − The block's first IP address must be divisible by the block size.

For example, assume the classless address is 192.168.1.35/27.

- The network component has a bit count of 27, whereas the host portion has a bit count of 5. (32-27)
- The binary representation of the address is: (00100011 . 11000000 . 10101000 . 00000001).
- (11000000.10101000.00000001.00100000) is the first IP address (assigns 0 to all host bits), that is, 192.168.1.32
- (11000000.10101000.00000001.00111111) is the most recent IP address (assigns 1 to all host bits), that is, 192.168.1.63
- The IP address range is 192.168.1.32 to 192.168.1.63.

Difference Between Classful and Classless Addressing

- Classful addressing is a technique of allocating IP addresses that divides them into five categories. Classless addressing is a technique of allocating IP addresses that is intended to replace classful addressing in order to reduce IP address depletion.
- The utility of classful and classless addressing is another distinction. Addressing without a class is more practical and helpful than addressing with a class.
- The network ID and host ID change based on the classes in classful addressing. In classless addressing, however, there is no distinction between network ID and host ID. As a result, another distinction between classful and classless addressing may be made.

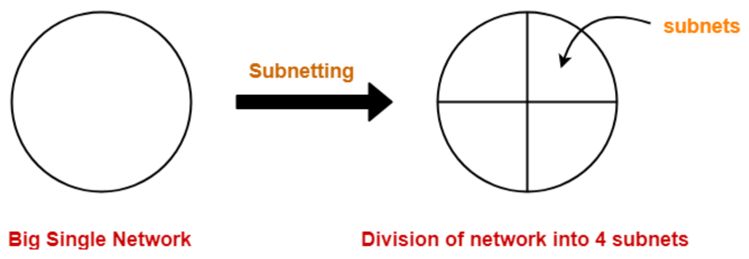
Subnetting in Networking-

In networking,

- The process of dividing a single network into multiple sub networks is called as **subnetting**.
- The sub networks so created are called as **subnets**.

Example-

Following diagram shows the subnetting of a big single network into 4 smaller subnets-



Advantages-

The two main advantages of subnetting a network are-

- It improves the security.
- The maintenance and administration of subnets is easy.

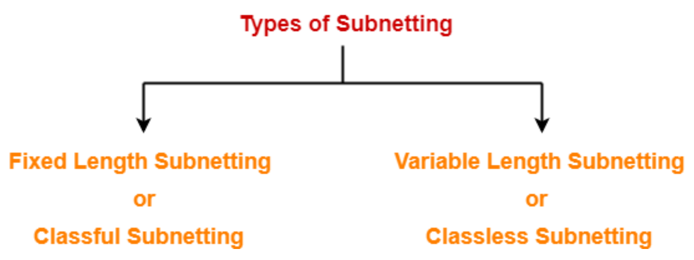
Subnet ID-

- Each subnet has its unique network address known as its **Subnet ID**.
- The subnet ID is created by borrowing some bits from the Host ID part of the IP Address.

- The number of bits borrowed depends on the number of subnets created.

Types of Subnetting-

Subnetting of a network may be carried out in the following two ways-


Types of Subnetting
Fixed Length Subnetting or Classful Subnetting
Variable Length Subnetting or Classless Subnetting

1. Fixed Length Subnetting
2. Variable Length Subnetting

1. Fixed Length Subnetting-

Fixed length subnetting also called as **classful subnetting** divides the network into subnets where-

- All the subnets are of same size.
- All the subnets have equal number of hosts.
- All the subnets have same subnet mask.

2. Variable Length Subnetting-

Variable length subnetting also called as **classless subnetting** divides the network into subnets where-

- All the subnets are not of same size.
- All the subnets do not have equal number of hosts.
- All the subnets do not have same subnet mask.

Subnetting Examples-

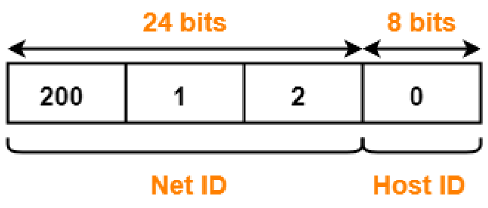
Now, we shall discuss some examples of subnetting a network-

Example-01:

Consider-

- We have a big single network having IP Address 200.1.2.0.
- We want to do subnetting and divide this network into 2 subnets.

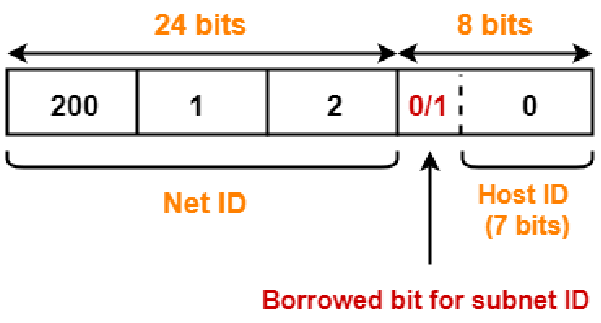
Clearly, the given network belongs to class C.



For creating two subnets and to represent their subnet IDs, we require 1 bit.

So,

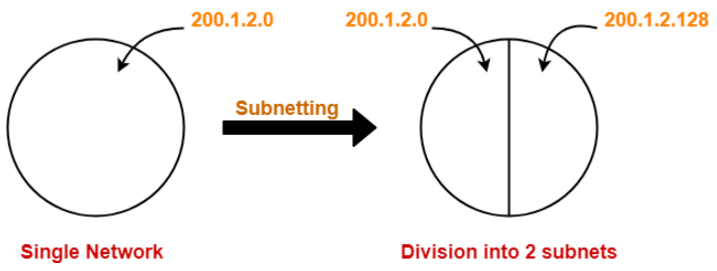
- We borrow one bit from the Host ID part.
- After borrowing one bit, Host ID part remains with only 7 bits.



- If borrowed bit = 0, then it represents the first subnet.
- If borrowed bit = 1, then it represents the second subnet.

IP Address of the two subnets are-

- 200.1.2.**0**0000000 = 200.1.2.0
- 200.1.2.**1**0000000 = 200.1.2.128



For 1st Subnet-

- IP Address of the subnet = 200.1.2.0
- Total number of IP Addresses = 2^7 = 128
- Total number of hosts that can be configured = 128 – 2 = 126
- Range of IP Addresses = [200.1.2.**0**0000000, 200.1.2.**0**1111111] = [200.1.2.0, 200.1.2.127]
- Direct Broadcast Address = 200.1.2.**0**1111111 = 200.1.2.127
- Limited Broadcast Address = 255.255.255.255

For 2nd Subnet-

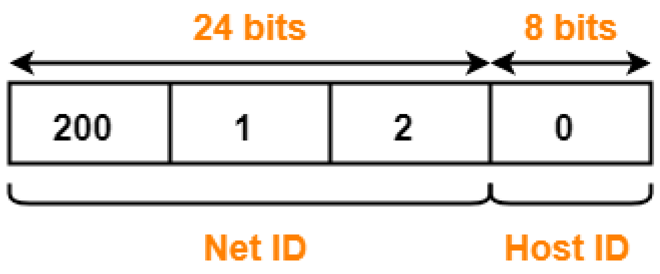
- IP Address of the subnet = 200.1.2.128
- Total number of IP Addresses = 2^7 = 128
- Total number of hosts that can be configured = 128 – 2 = 126
- Range of IP Addresses = [200.1.2.**1**0000000, 200.1.2.**1**1111111] = [200.1.2.128, 200.1.2.255]
- Direct Broadcast Address = 200.1.2.**1**1111111 = 200.1.2.255
- Limited Broadcast Address = 255.255.255.255

Example-02:

Consider-

- We have a big single network having IP Address 200.1.2.0.
- We want to do subnetting and divide this network into 4 subnets.

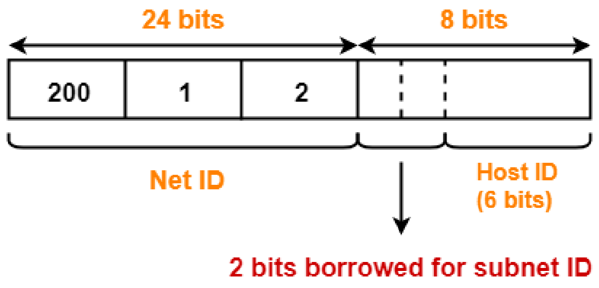
Clearly, the given network belongs to class C.



For creating four subnets and to represent their subnet IDs, we require 2 bits.

So,

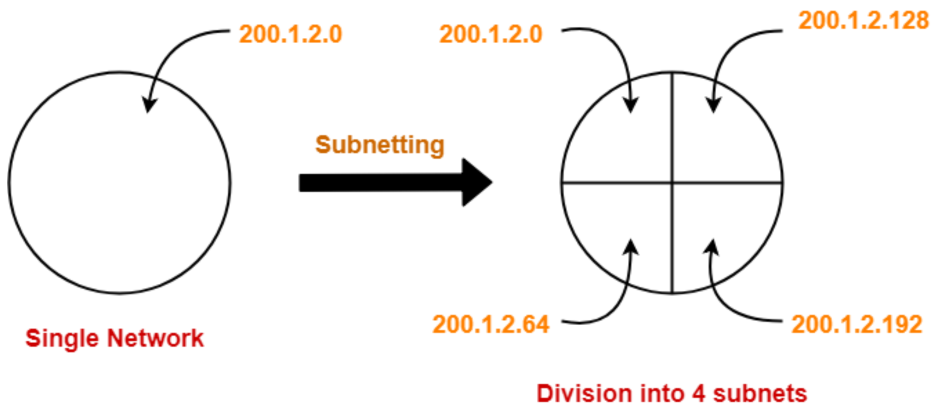
- We borrow two bits from the Host ID part.
- After borrowing two bits, Host ID part remains with only 6 bits.



- If borrowed bits = 00, then it represents the 1st subnet.
- If borrowed bits = 01, then it represents the 2nd subnet.
- If borrowed bits = 10, then it represents the 3rd subnet.
- If borrowed bits = 11, then it represents the 4th subnet.

IP Address of the four subnets are-

- 200.1.2.**00**000000 = 200.1.2.0
- 200.1.2.**01**000000 = 200.1.2.64
- 200.1.2.**10**000000 = 200.1.2.128
- 200.1.2.**11**000000 = 200.1.2.192



<u>**For 1st Subnet-**</u>

- IP Address of the subnet = 200.1.2.0
- Total number of IP Addresses = 2^6 = 64
- Total number of hosts that can be configured = 64 – 2 = 62
- Range of IP Addresses = [200.1.2.**00**000000, 200.1.2.**00**111111] = [200.1.2.0, 200.1.2.63]
- Direct Broadcast Address = 200.1.2.**00**111111 = 200.1.2.63

- Limited Broadcast Address = 255.255.255.255

For 2nd Subnet-

- IP Address of the subnet = 200.1.2.64
- Total number of IP Addresses = 2^6 = 64
- Total number of hosts that can be configured = 64 – 2 = 62
- Range of IP Addresses = [200.1.2.**01**000000, 200.1.2.**01**111111] = [200.1.2.64, 200.1.2.127]
- Direct Broadcast Address = 200.1.2.**01**111111 = 200.1.2.127
- Limited Broadcast Address = 255.255.255.255

For 3rd Subnet-

- IP Address of the subnet = 200.1.2.128
- Total number of IP Addresses = 2^6 = 64
- Total number of hosts that can be configured = 64 – 2 = 62
- Range of IP Addresses = [200.1.2.**10**000000, 200.1.2.**10**111111] = [200.1.2.128, 200.1.2.191]
- Direct Broadcast Address = 200.1.2.**10**111111 = 200.1.2.191
- Limited Broadcast Address = 255.255.255.255

For 4th Subnet-

- IP Address of the subnet = 200.1.2.192
- Total number of IP Addresses = 2^6 = 64
- Total number of hosts that can be configured = 64 – 2 = 62
- Range of IP Addresses = [200.1.2.**11**000000, 200.1.2.**11**111111] = [200.1.2.192, 200.1.2.255]
- Direct Broadcast Address = 200.1.2.**11**111111 = 200.1.2.255
- Limited Broadcast Address = 255.255.255.255

Example-03:

Consider-

- We have a big single network having IP Address 200.1.2.0.
- We want to do subnetting and divide this network into 3 subnets.

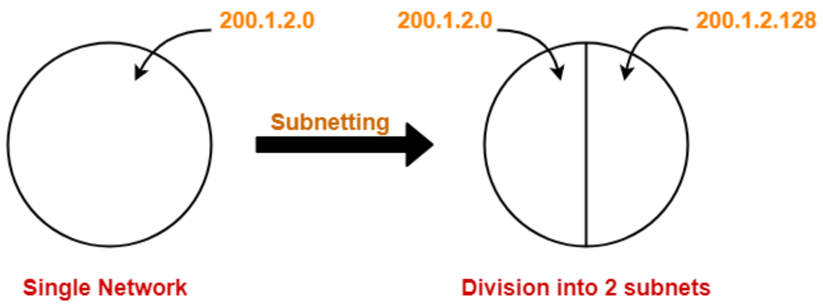
Here, the subnetting will be performed in two steps-

1. Dividing the given network into 2 subnets
2. Dividing one of the subnets further into 2 subnets

Step-01: Dividing Given Network into 2 Subnets-

The subnetting will be performed exactly in the same way as performed in Example-01.

After subnetting, we have-

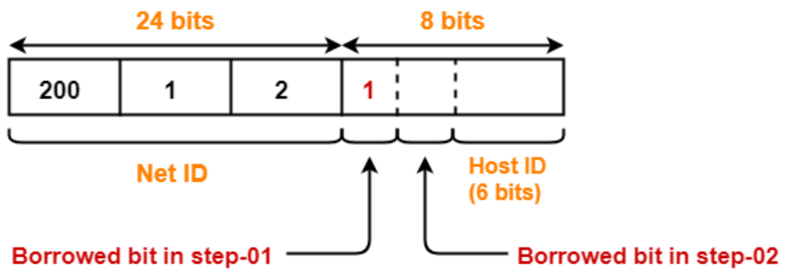


Step-02: Dividing One Subnet into 2 Subnets-

- We perform the subnetting of one of the subnets further into 2 subnets.
- Consider we want to do subnetting of the 2nd subnet having IP Address 200.1.2.128.

For creating two subnets and to represent their subnet IDs, we require 1 bit. So,

- We borrow one more bit from the Host ID part.
- After borrowing one bit, Host ID part remains with only 6 bits.

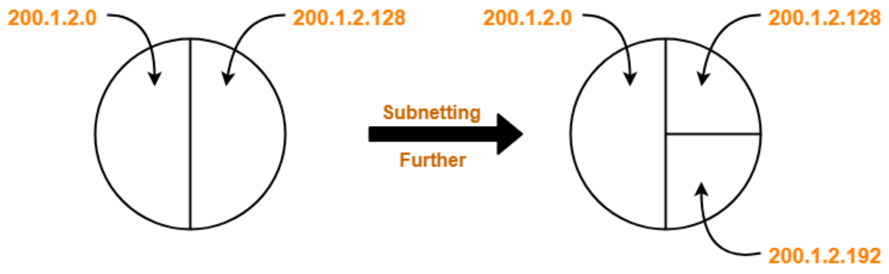


- If 2nd borrowed bit = 0, then it represents one subnet.
- If 2nd borrowed bit = 1, then it represents the other subnet.

IP Address of the two subnets are-

- 200.1.2.**10**000000 = 200.1.2.128

- 200.1.2.**11**000000 = 200.1.2.192



Finally, the given single network is divided into 3 subnets having IP Address-

- 200.1.2.0
- 200.1.2.128
- 200.1.2.192

<u>For 1st Subnet-</u>

- IP Address of the subnet = 200.1.2.0
- Total number of IP Addresses = 2^7 = 128
- Total number of hosts that can be configured = 128 – 2 = 126
- Range of IP Addresses = [200.1.2.**0**0000000, 200.1.2.**0**1111111] = [200.1.2.0, 200.1.2.127]
- Direct Broadcast Address = 200.1.2.**0**1111111 = 200.1.2.127
- Limited Broadcast Address = 255.255.255.255

<u>For 2nd Subnet-</u>

- IP Address of the subnet = 200.1.2.128
- Total number of IP Addresses = 2^6 = 64
- Total number of hosts that can be configured = 64 – 2 = 62
- Range of IP Addresses = [200.1.2.**10**000000, 200.1.2.**10**111111] = [200.1.2.128, 200.1.2.191]
- Direct Broadcast Address = 200.1.2.**10**111111 = 200.1.2.191
- Limited Broadcast Address = 255.255.255.255

<u>For 3rd Subnet-</u>

- IP Address of the subnet = 200.1.2.192
- Total number of IP Addresses = 2^6 = 64
- Total number of hosts that can be configured = 64 – 2 = 62

- Range of IP Addresses = [200.1.2.**11**000000, 200.1.2.**11**111111] = [200.1.2.192, 200.1.2.255]
- Direct Broadcast Address = 200.1.2.**11**111111 = 200.1.2.255
- Limited Broadcast Address = 255.255.255.255

****