Indian Institute of Information Technology
Design and Manufacturing, Kancheepuram
Chennai 600 127, India
An Autonomous Institute under MHRD, Govt of India
An Institute of National Importance
www.iiitdm.ac.in
COM302T Computer Networks-Lecture Notes

Instructor
N.Sadagopan
Scribe:
P.Renjith

Forwarding and Routing

Objective: We shall explore two primitive terms in computer networking, namely *routing* which is a process of determining a path between a pair of nodes in a network and *forwarding* which is a process of determining the 'right' output port for the incoming packet by consulting the routing (forwarding) table. Also, we discuss IP (Internet Protocol) addressing schemes, Subnetting, designing local area networks with subnets, etc.

1 Transition from MAC address to IP address

Local area networking using IEEE standards (802.3,802.5) used MAC (media access control) addresses for identifying a node in a network and for transmitting a data packet from one node to the other. Although MAC address is unique, it cannot be used as an addressing scheme for a larger network. We below highlight some of the limitations of MAC based addressing scheme.

– In a 48 bit MAC address, the first 6 bits are reserved for identifying the manufacturer whereas the other 42 bits are reserved for identifying the host. Theoretically, we can have an INTERnet (network of network of ... nodes) of size 2^{42}. Is this number sufficient to meet the future requirements ? Is MAC addressing scalable ? Let us do a quick estimate of the number of nodes in INTERnet. The world's population is approximately 100 billion. Assuming each person has at most 10 systems (10 MAC addresses), then the number of MACs required is $100 \times 10^9 \times 10$ which is approximately 2^{40}. This rate of growth suggests that we may exhaust 2^{42} MAC addresses in near future.

– There is no hierarchy in 42 bits of MAC addresses, and therefore, one cannot identify the origin of the packet (for example; the country from which the packet is generated) using MAC addresses. We thus, need a scheme using which we can identify the geography of the node that generated the packet. This is important from the perspective of network security and security related attacks.

– Assuming as many as 2^{30} out of 2^{42} systems are active at a given point in time, then all switches (routers) in the network must be configured with a routing table containing 2^{30} entries each of size 48 bits. For an incoming packet at a switch, one has to perform the linear search by performing 48 bit comparisons at each stage of the search. Can we overcome this scenario by maintaining a smaller routing table and less number of bit comparisons at each stage of the search ? Can we look for a different addressing scheme to meet these requirements ?

– It is important to note that in a campus network (academia/corporate network), not all systems in the network is connected to INTERnet. This means that the addressing scheme need not assign a unique address to each machine in this world. Typically, a campus network having 1000 systems has 10 global addresses (unique addresses) and only these 10 systems are known to the systems connected to other networks. Thus, some of the addresses which are local to a network can be used in other networks as well. Therefore, if we come up with a hierarchical addressing scheme, we may achieve (i) address reusuability (ii) identification of a node based on geography (iii) reduction in the size of the forwarding table (iv) less number of bits in the address is used for search.

– Internet Protocol (IP) address is an addressing scheme which precisely achieves all of the above parameters of interest. IPv4 (IP version 4) is 32 bits which is currently in use in many campus networks. Next generation campus networks are switching to IPv6 (IP version 6) which is of size 128 bits and offers a good scalability compared to IPv4. A typical IPv4 address looks like *192.168.2.1* and the format of the address naturally brings in hierarchy among networks and nodes in the network.

Road map: We shall next discuss forwarding in detail using forwarding tables which contain either MAC or IP addresses. Further, we shall be discussing forwarding in a switched network containing loops (cycles). We close this section by discussing routing algorithms in detail.

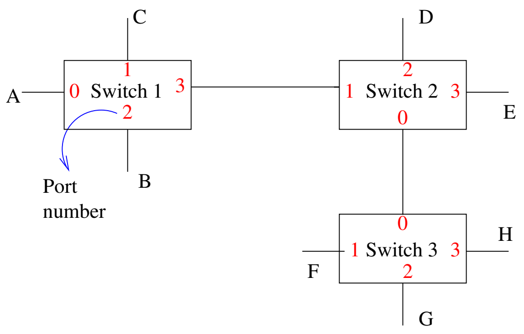
Network Gadgets: Commonly used network components to interface two or more networks are Hubs, Bridges, Switches, Routers, and Gateways. Hubs and bridges are also known as Layer 2 switches as the forwarding is based on MAC addresses. Further, a hub and bridge follow broadcasting strategy while forwarding a data packet, and unicasting is not possible with these devices. A Switch, Router and Gateway are Layer 3 switches, and by switch we always mean a Layer 3 switch. All three follow unicasting strategy while forwarding a data packet. Typically, the term 'switch' is used when we create a small network of 200 systems (Eg., a laboratory network in an academia), the term 'router' is used when we interface two campus networks (Eg., two academia networks), and 'gateway' is used when we interface two different networks of large size (Eg., to interface two states, two countries). Although, switch, router, gateway are Layer 3 switches, the internal architecture (buffer size, processor used, the number of NICs) is different as it supports different applications of varying complexity.

2 Forwarding

Having highlighted the need for IP based addressing scheme, we shall now present two approaches for forwarding data packets between a pair of nodes in a network. Recall that forwarding is a process of identifying the right out port by the switch for the incoming packet. Given a static network with a set of nodes and switches in place, to forward a packet from node A to node B, we follow one of the following three schemes.

(i) Datagram or Connectionless Approach
(ii) Source routing
(iii) Virtual circuit or Connection-oriented Approach
For a static network, the routing (forwarding) table at each switch is manually configured by the network administrator. If the size of the data to be transmitted is small and fits in one packet (for example: email, status update packet), then it is sufficient to work with datagram approach, whereas, virtual circuit is preferred when node A wishes to transmit a large file containing many packets (for example: ftp/telnet). The forwarding schemes discussed here can also be used in Layer 2 switches; networks which follow MAC based addressing scheme.

Consider a static network shown in Figure 1. The forwarding tables of switches are also il-



Forwarding Table at Switch 1

MAC address (Destination node)	O/P port
A	0
B	2
C	1
D	3
E	3
F	3
G	3
H	3

Forwarding Table at Switch 2

MAC address (Destination node)	O/P port
A	1
B	1
C	1
D	2
E	3
F	0
G	0
H	0

Forwarding Table at Switch 3

MAC address (Destination node)	O/P port
A	0
B	0
C	0
D	0
E	0
F	1
G	2
H	3

Fig. 1. A Static Network with nodes and switches

lustrated in Figure 1. For simplicity, in the figure we use node labels in place of 48-bit MAC addresses or 32-bit IP addresses.

Datagram Approach: Suppose node A wishes to transmit a packet to node G. The packet from A reaches switch 1 via port 0, $S1$ consults the forwarding table and identifies that the outgoing port is 3. Thus, the packet is forwarded through port 3 which inturn reaches switch 2 via port 1. Further, $S2$ identifies the 'right' output port; port 0 by looking at its forwarding table. Finally, switch 3, forwards the packet through port 2. It is clear that each forwarding table acts a travel guide for a packet in transit until it reaches the destination. Whenever there is a change in the topology of the network due to the insertion/deletion of a node/switch, the network administrator accordingly updates all forwarding tables to reflect the changes done on the network. In practice, a 48-bit comparison between the incoming packet and the entries of the table is done at every switch to decide upon the outgoing port. This scheme is good as long as the size of the table (network) is small.

Source Routing: The major drawback of datagram approach is that it involves 48-bit comparison across the entries of forwarding tables. To overcome this limitation, in source routing, the source node includes all the port numbers in order that would be visited while the packet is in transit. The port numbers are included as part of the header field of the packet. For the above data transfer (A to G), the header contains 3 0 2 which is an ordered sequence of output ports at respective switches. Here again, there is a overhead due to the augmentation of output ports as part of the header, and thus the scheme is good for a small network. Although, this avoids 48-bit comparison completely at intermediate switches, a sizable portion of data part of the packet is used to store the port information.

Virtual Circuit: Both datagram and source routing techniques are good if the data transfer fits in just one packet. Supposing, a data transfer involves sending a large file consisting of 100 packets, then, one can trivially follow datagram approach (source routing) for each of the 100 packets, however, the search time at each switch is more as 48-bit (32-bit) comparisons must be done at all switches. It is natural to ask: can we reduce the search time by performing less number of bit comparisons at each switch ? As part of virtual circuit scheme, we perform 48-bit (32-bit) comparisons only for the connection request packet (zeroth packet) and for the subsequent data packets $P_1, \ldots, P_{100}$, we perform less number of comparisons by making use of *virtual circuit identifiers (VCI).*

Virtual circuit scheme has two phases; (i) virtual circuit creation phase (ii) data transfer phase using the virtual circuit. We first establish a connection known as *virtual circuit* between the source and destination for which virtual circuit identifiers are used, which are a small integer chosen randomly to create such a connection. Further, using the virtual circuit, we perform the data transfer of 100 packets between nodes A and G. During virtual circuit creation phase, we create a virtual circuit table at each switch in addition to the existing forwarding table. For the data transfer phase of A-G, we make use of virtual circuit table which is unique for this connection. Note that virtual circuit creation phase is like a datagram scheme. Suppose $P_1, \ldots, P_{100}$ are the packets to be transmitted from A to G, then, A initiates a connection set up request which is a special packet (P_0) that flows from A to G following the datagram scheme. While the special packet visits a switch, the switch assigns a label *virtual circuit identifier* for that connection and populates the virtual circuit table available at that switch.

For A-G data transfer, the node A assigns a VCI 10, $S1$ assigns 11, $S2$ assigns 12, and finally $S3$ assigns 13. In reality, the assignment happens in reverse when ACK signal is generated by the destination. That is, when G is willing to perform A-G data transfer in response to connection setup request, it sends an ACK which contains VCI 13 assigned by the node G. $S3$ updates the VCI table of $S3$ mentioning every packet meant for A-G data transfer leaves $S3$ at port 2 with VCI label 13.
Further, ACK packet is sent to $S2$ by $S3$ after assigning a VCI label 12. This means that, every packet of A-G data transfer leaves $S2$ at port 0 with VCI label 12. This process continues until ACK reaches the node A. That is, $S2$ assigns a VCI 11, $S1$ assigns a VCI 10. On receiving ACK containing VCIs $(13, 12, 11, 10)$, the node A knows that the path from A to G is active and all intermediate switches are willing to forward the packets by consulting their VCI tables, and have reserved a buffer in the switch for this virtual circuit based data transfer.

For $P_1, \ldots, P_{100}$ of A-G data transfer, the scheme works as follows: for each P_i, the node

Table 1. Virtual Circuit Table at Switch 1

Data transfer	Incoming Port	Incoming VCI	Outgoing Port	Outgoing VCI
A-G	0	10	3	11
A-E	0	20	3	21

A assigns VCI 10 before sending it to the switch 1 on port 0. $S1$ consults the VCI table and finds that the corresponding output port 3 and the number to be assigned is 11. VCI 10 is removed from the packets's header and VCI 11 is placed in place of 10, and it is sent to $S2$ on

Table 2. Virtual Circuit Table at Switch 2

Data transfer	Incoming Port	Incoming VCI	Outgoing Port	Outgoing VCI
A-G	1	11	0	12
A-E	1	21	3	22

Table 3. Virtual Circuit Table at Switch 3

Data transfer	Incoming Port	Incoming VCI	Outgoing Port	Outgoing VCI
A-G	0	12	2	13

port 3. This process continues at every switch for all 100 packets. It is important to note that, each time, VCI table entries are compared against the incoming VCI label which requires less number of bit comparisons to identify the right outgoing port. A 48-bit or 32-bit comparisons of MAC or IP addresses in datagram approach is reduced to 8-bit or 16-bit comparisons of VCI labels. At the end of the data transmission, the source generates a VCI termination request and on seeing the termination packet, the switches remove the VCI entries of that connection from VCI tables. The removed VCI labels are available in the VCI pool for reuse by the subsequent virtual circuits.

Remarks:
1. For VCI scheme, the sender waits for one RTT before it begins data transfer whereas datagram or source routing is a connection less approach where in there is no wait time.
2. In VCI scheme, each packet follow the same path and the number of distinct virtual circuits at a given point in time is determined based on the buffer capacity available at switches and the delay bandwidth products of links. Since there is an acknowledgement phase associated with VCI creation, on seeing the ACK, the sender knows that there will not be any contention or congestion related issues along the transmission line when data transfer phase begins.
3. **Contention:** For a 100 Mbps line with $t_{prop} = 10$ micro secs, the DB product is $100 \times 10^6 \times 10 \times 10^{-6} = 1000$ bits. If packet size is 100 bits, then at most 10 packets can be there along the transmission line at a given point of time. This implies that, no more than 10 virtual circuits (connections) are possible. If more than 10 connections compete for 100 Mbps link, then contention arises. In datagram or source routing, contention issue is quite common as the flow of packets is arbitrary, more than 10 datagrams may compete for the link leading to contention. In VCI scheme, contention will not be there as virtual circuit itself will not be established beyond 10 requests.
4. **Congestion:** Each switch has a finite buffer, say 50 cells of size 100 bits and switches are store and forward type. Essentially, at the switch; each packet is stored, processed (identifying the 'right' output port), and forwarded to the next switch. If there are more than 50 simultaneous connections, then the switch buffer is congested and the switch is forced to drop packets beyond 50. Congestion is a common problem in networks as many datagram or source routing packets flow across the network. In VCI scheme, there is no congestion issue at any of the switches as the number of VCs is restricted to 50.
5. In datagram or source routing, if more than one packet is sent to the same destination, the paths taken by the packets may be different for each iteration, whereas, in VCI, all packets follow the same virtual circuit until all packets of that circuit are transferred.
6. Applications such as Email, DHCP (Dynamic Host Control Protocol - the protocol responsi-

ble for releasing dynamic IPs) follow datagram approach as the data associated with the request is fit in just one packet. Also, IP (internet protocol) packets follow datagram approach as part of routing algorithms such as RIP and OSPF.
7. Applications such as Telnet, FTP (File Transfer Protocol) follow VCI scheme as many data packets have to be transferred in a reliable fashion.
8. Since VCI and the incoming port uniquely identifies the packet and the entry in the VCI table, it is perfectly fine to use same VCI labels at all switches for a given connection. That is, VCI label 10 can be used by all switches for assigning both incoming VCI and outgoing VCI labels for *A*-*G* data transfer.

2.1 Spanning Tree Approach: Forwarding in Bridges with loops

The switched LAN illustrated in Figure 1 employs Layer 3 switches which means switches have some intelligence using which it can identify the right outgoing port for an incoming packet. However, the forwarding scenario is different if it is a Layer 2 switch. Layer 2 switches (bridges) are by nature broadcast-type switches as it has no intelligence component in it. It is important to note that bridges are basically unmanageable switches; switches with no intelligence and hence, for the incoming packet, it cannot identify the right output port by consulting the forwarding table. Instead, bridges forward the incoming packet to all output ports; thereby, flooding the network with many data packets. Due to this flooding of packets, the useful component of delay bandwidth product is ineffectively used by flooding packets, and most importantly the destination node may receive more than one copy of the same packet sent by the sender. This scenario is illustrated in Figure 2.
In Figure 2, the packet generated by LAN A reaches bridges 1 and 2 which inturn forwards it to LAN B. LAN B receive two copies of the same packet due to loops in the network.

In the first place, why are the loops in a switched LAN or LAN with bridges. Loops pro-

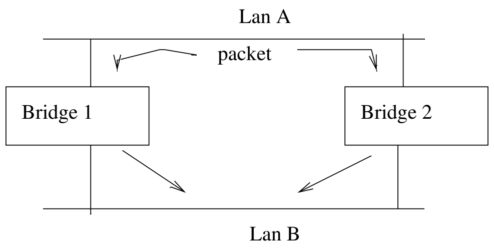


Fig. 2. A network with loops among bridges

vide an additional path between a pair of nodes, thereby providing redundancy, and thus if a link or bridge fails, the network is still active. There exists at least two distinct paths between any pair of bridges and hence the network is resilient to a single bridge/link failure. Having highlighted the importance of loops among bridges, we thus need a strategy to avoid flooding of packets in the network if the underlying network has a loop involving bridges.
To mitigate the issues due to flooding, we shall now discuss a scheme using which we can obtain a spanning tree (an acyclic subgraph) of the underlying network before data transmission. Since spanning trees do not contain loops, packets broadcasted by bridges do not circulate in the network unnecessarily. Let us consider the switched LAN using bridges, illustrated in Figure 2. Note that, if the underlying network is a static network, then we can simply run well-known

Prim's or Kruskal's minimum spanning tree algorithm. However, the network on bridges is a dynamic network. We thus need a scheme or an algorithm that works on dynamic graphs. A natural choice for any dynamic network is a `distributed algorithm`. As the name suggests, the distributed algorithm is implemented in all switches, and all switches act in synchronization to obtain a spanning tree.

Distributed Algorithm to output a Spanning Tree: Consider the dynamic network illustrated in Figure 3. There is a loop among bridges and each bridge interconnects two or more LANs. The objective is to extract a spanning tree on bridges so that packets, on travel, will not loop round the network unnecessarily. An algorithm that is run on all bridges in a consistent manner by exchanging packets is a distributed algorithm. Bridges exchange packets so that when the algorithm terminates, all bridges have consistent information about the network. The following properties are ensured at the termination of the algorithm.

- A bridge with the least index is designated as the root bridge. For Figure 3, B_1 is the root.
- The root bridge maintains a shortest path from the root to all other bridges.
- If a LAN has two shortest paths to reach the bridge, i.e., LAN A is connected to B_6 and B_7, both are at equi-distant from the root, then the path through B_6 is preferred. In general, the bridge with the least index is given preference over the other.

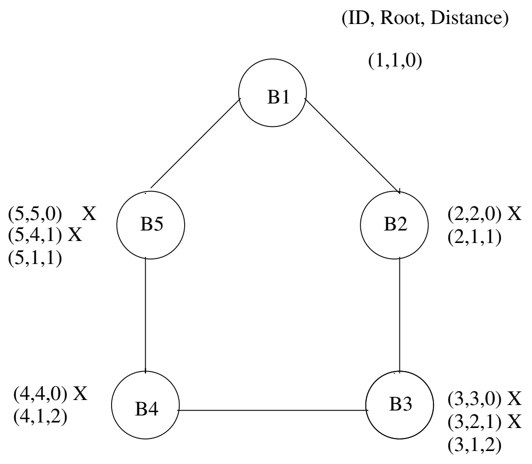


Fig. 3. Distributed Algorithm on Bridges with loops

Trace of the Algorithm:

- As part of the algorithm, the bridges exchange status packets consisting of (Bridge ID, the ID of the root bridge, the distance from the root).
- At the start of the algorithm, each bridge assumes themselves as the root and broadcasts the status vector to its neighbors. Let us assume the following sequence of exchanges take place as we do not the exact timing sequence. For example, B_5 generates (5,5,0) and B_3 generates (3,3,0) initially. The packet from B_5 is sent to B_1 and B_4. Since they both have smaller index compared to B_5, the packet (5,5,0) is discarded.
- The exact sequence of exchanges may not be known as the bridge can transmit the status vector at arbitrary timings. Let us assume, (1,1,0) packet is sent to B_2 and B_5, on seeing

this packet, they both learn and assume that B_1 is the root and updates its status vector to (2,1,1) and (5,1,1) which is inturn sent to the other neighbors of B_2 and B_5.

- Subsequently, B_3 and B_4 get to learn about B_1 and update its vector accordingly. Further, B_3 stops sending packet to B_4 as it learnt that reaching B_1 via B_4 takes 3 hops. Similarly, B_4 stops sending its status vector to B_3 as reaching B_1 via B_3 takes 3 hops. This implies that the link (B_3, B_4) is not used for data traffic and every other link is used for data traffic. Thus, the above network without the link (B_3, B_4) yields a spanning tree which will be used for data traffic.
- The above algorithm is re-run whenever there is a change in topology. For example, if a link is added between (B_1, B_4) or the link (B_3, B_2) fails.

3 Routing

How are forwarding (routing) tables built? How do we identify a shortest path between a pair of nodes in a dynamic network ? Routing is a process of building routing tables containing shortest path between any two nodes in the network. Two commonly followed strategies are (i) Distance Vector Routing (ii) Link State Routing. Distance vector routing is implemented by Routing Information Protocol (RIP) and Link state routing is implemented by Open Shortest Path First (OSPF) protocol. We shall discuss these two protocols in detail.

Routing Information Protocol (RIP)

RIP is based on distance vector routing strategy, i.e., a node shares its neighborhood information to all its neighbors. By this way, a neighbor gets to know about other neighbors of the node and this process of exchanges continues for a while. After a few exchanges, the network reaches a stable state and each node in the network knows how to reach the other node, the cost, the next hop, etc. For the dynamic network shown in Figure 4, the content of the routing tables are listed below.

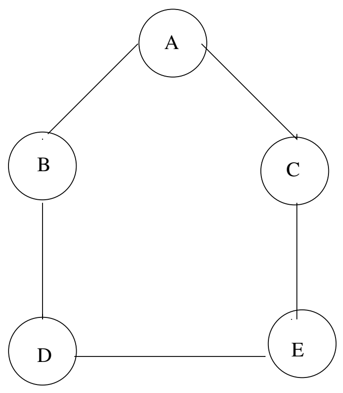


Fig. 4. Distance Vector Routing and RIP

- A sends its neighbor information to B and C, C learns about B and updates its routing table as C to B via A with hop count=2. Similarly, B learns about C. B broadcasts its table to its neighbors. D learns about A and C. A learns about D that D is reachable in two hops. D learns from B that C is reachable in 3 hops via B. Subsequently, it learns from E that C is reachable via E in 2 hops. Similar updates at E for B. After a few exchanges, the entire routing table is constructed.

- Whenever there is a change in topology, for example, a link is added or a link fails, then distance vector routing algorithm is re-run to get the updated status of the underlying network. However, a failure of a link and subsequent exchanges may bring in some inconsistency in routing table entries which we shall highlight now in detail.

- Updates happen either at regular intervals (periodic updates) or when there is a change in topology (triggered updates).

Table at A	Destination	Next Hop	Count
	B	B	1
	C	C	1
	D	B	2
	E	C	2
Table at B			
	A	A	1
	D	D	1
	C	A	2
	E	D	2
Table at C			
	A	A	1
	E	E	1
	B	A	2
	D	E	2
Table at D			
	D	D	1
	E	E	1
	A	B	2
	C	B	3 (old)
	C	E	2 (new)
Table at E			
	D	D	1
	C	C	1
	A	C	2
	B	C	3 (old)
	B	D	2 (new)

Let us consider a scenario where the link (D,E) fails in Figure 4. D learns that E is unreachable and set the cost of D to E to infinity. Similarly E sets the cost of reaching D to infinity. Assuming RIP follows triggered updates, then D as well as E wishes to broadcast this to its neighbors so that they can learn about this failure. After a while, the updates takes place at all routing tables. As a result, all switches are aware of the new topology.

For periodic updates, if the synchronization is not done correctly, then a wrong picture of the topology may get circulated among nodes. For example, assume that even before D sends the status packet to B informing (D,E) link failure, A informs B that E is reachable in 2 hops. B learns from D that E is unreachable but learns from A that E is reachable, therefore, B concludes that E is reachable in 3 hops. This happens due to synchronization issue between periodic update packets.

After a while, again due to periodic updates, E informs C, inturn A that E is unreachable. However, B updates A that it can reach E in 3 hops; A concludes that E is reachable in 4 hops. This cyclic argument due to synchronization can lead to `count-to-infinity` problem. To mitigate this issue, we fix a cap on the hop count, say 16. This way, even if count-to-infinity problem occurs, the packet will be dropped after a maximum of 16 iterations. Because of count-to-infinity problem, RIP is preferred for a small network. Moreover, the convergence of RIP algorithm usually takes many iterations and this number increases with the number of nodes. By convergence, we mean the number of iterations required to learn the entire topology. The next strategy *link state routing* works well for large networks.

Open Shortest Path First (OSPF)

Very similar to RIP, OSPF broadcasts neighborhood information (link state) to other neighbors, however, this happens in a peculiar fashion. For the Figure 4, let us assume OSPF is triggered first at node A; (i) B learns about A and C, and sent to D (ii) C learns about A and C, and sent to E (iii) Finally, E gets information from D and C which contains the entire topology of the dynamic network. No further forwarding at E as no new neighbors other than D and C are connected to E. Secondly, assume that OSPF is triggered at node B; similar to the above exchanges, the nodes D,E,A learn the network and finally concludes at C. Now C learns the entire topology. This triggering takes place at every node, after a while, each node in the network learns the entire topology.

At this point of time, each node has a static network of the underlying dynamic graph, it is a weighted graph with weights being either the hop count or the propagation delay of the link.
Further, to discover shortest paths from a node to all other nodes, the classical shortest path (Dijkstra's Algorithm) is implemented at every node. Based on the output of shortest path algorithm, the routing tables are populated with information such as destination, next hop and cost. Any change in topology triggers OSPF packets using which a new static network is generated at all nodes and shortest path information is recomputed to reflect the change in topology at all routing tables.

Recap: We have presented two routing schemes using which one can build routing tables at switches. Forwarding schemes make use of routing tables to identify the right out port for an incoming packet. Throughout our discussions, we have used labels to identify the nodes, however, in practice MAC/IP addresses are used depending upon whether the underlying network uses bridges or switches. We shall next discuss IP addressing schemes in detail.

4 Internet Protocol Addressing Scheme

How are machines identified in a network ? How are campus networks identified in internet ? How big is the size of the routing table at routers ? We shall explore these questions in this section in detail. An Internet Protocol (IP) address is represented like 172.16.1.2 with a 'dot' after every byte. The commonly used addressing schemes are as follows;

Class	Break-up of 32 bits	Public IPs	Private IPs	Hosts
Class A	$\boxed{0 \mid \text{Network ID} \mid \text{Host}}$ 7 bits for network and 24 bits for hosts			
		0.0.0.0 to 127.255.255.255	10.0.0.0 to 10.255.255.255	2^{24} hosts in each network 127 Public networks and one private network
Class B	$\boxed{1 \mid 0 \mid \text{Network ID} \mid \text{Host}}$ 14 bits for network and 16 bits for hosts			
		128.0.0.0 to 191.255.255.255	172.16.0.0 to 172.31.255.255	2^{16} hosts in each network 2^{14}-16 Public networks and 16 private networks
Class C	$\boxed{1 \mid 1 \mid 0 \mid \text{Network ID} \mid \text{Host}}$ 21 bits for network and 8 bits for hosts			
		192.0.0.0 to 223.255.255.255	192.168.0.0 to 192.168.255.255	2^{8} hosts in each network 2^{21}-256 Public networks and 256 private networks
Class D	$\boxed{1 \mid 1 \mid 1 \mid 0 \mid \text{Network ID} \mid \text{Host}}$ 20 bits for network and 8 bits for hosts			
		224.0.0.0 to 239.255.255.255		2^{8} hosts in each network 2^{20} Public networks
Class E	$\boxed{1 \mid 1 \mid 1 \mid 1 \mid \text{Network ID} \mid \text{Host}}$ 20 bits for network and 8 bits for hosts			
		240.0.0.0 to 255.255.255.255		2^{8} hosts in each network 2^{20} Public networks

Class A is preferred when a network has a large number of hosts connected to it. Class B is a moderate network which supports 65,536 hosts. Class C is largely preferred for campus networks

as it supports many small networks of size 256. Class D is reserved for multicast purpose and largely used by department of defense. Class E is reserved for experimental and research use. Class A,B and C are used for assigning global IPs to campus networks.
Typically, a campus network has one or more global IP addresses (public IPs) and makes use of local IP addresses (private IPs) for assigning IPs to machines inside the network. This is done with the intent to effectively utilize the link bandwidth. For example, IIITDM network has one global IP (14.110.11.291) and uses Class B private IPs to configure the campus network. A typical campus network distribution is given below;

Location	Number of Hosts	Sample IPs
Academic Bay	1000	172.16.1.0 to 172.16.4.255
Hostel Bay	1000	172.17.1.0 to 172.17.4.255
Recreational Spots	1000	172.18.1.0 to 172.18.4.255

Extracting the network id from the IP address: For routing purpose, the routers extract the network part of the IP address so that the packet can be routed to the appropriate network. For this purpose, `IP mask` is defined at all routers. IP mask of class A, B and C are given below.

Class	IP Mask	Example
Class A	255.0.0.0	Incoming IP: 14.1.2.3; perform (14.1.2.3 AND 255.0.0.0) which yields 14.0.0.0 which means the packet belongs to 14.0 network
Class B	255.255.0.0	Incoming IP: 172.17.1.4; perform (172.17.1.4 AND 255.255.0.0)=172.17.0.0, the packet belongs to 172.17 network.
Class C	255.255.255.0	Incoming IP: 192.168.1.4; perform (192.168.1.4 AND 255.255.255.0)=192.168.1.0, the packet belongs to 192.168.1 network.

An example of a routing table with IP masks is illustrated below.

IP Mask	Network ID	Outport	Example IPs
255.0.0.0	15.0.0.0	1	15.1.2.3, 15.0.1.191
255.255.0.0	139.0.0.0	2	139.0.1.2, 139.0.1.8
255.255.0.0	181.10.0.0	3	181.10.1.2, 181.10.10.11
255.255.255.0	200.1.2.0	4	200.1.2.5, 200.1.2.50

Remarks:
1. Consider the following requirements; (i) configure a laboratory network with 300 systems. (ii) configure a research laboratory network with 30 systems. For the first requirement, we need at least a Class B network, however, address space efficiency is very poor. Address space efficiency= $\frac{300}{65,536}$ which is approximately 5%. If we go for two class C networks with 150 systems each, then the address space efficiency=$\frac{150}{256}$ which is approximately 60%. For the second requirement, the efficiency is 12% if we follow Class C scheme.
2. If address space is not assigned wisely, we may exhaust the IP address space soon. Class A is generally preferred for country's gateway; within a country, class B is used to assign IP addresses to campus networks, and within a campus network, class C scheme must be followed. Although, this strategy works fine in a few case studies, the address wastage is still high. We thus need a scheme which assigns IPs effectively with less wastage in IP address assignment.

3. Inefficiency in Address space assignment is due to the fact that classful schemes follow a rigid break up for network ids and hosts. One approach is to bring in second level network (subnetwork) in the host part of an IP address. For example, for the second requirement mentioned above, we can follow class C scheme by reserving the last five bits for hosts (32 systems) and the other 3 bits of host acts as a subnetwork identifier.

Sub networking or Subnetting

In subnetting, the host part of IP address is split into subnet id and host id, and the network id remains the same. This brings two-level network hierarchy among IP addresses. Let consider a case study: A campus network is to be designed with 8 laboratories having 30 systems each. A simple scheme is to use 8 class C addresses; 192.168.0.1 to 192.168.0.30, 192.168.1.1 to 192.168.1.30, and so on. As mentioned above, this is not a wise assignment as it wastes 225 addresses in each network. Since, we need 30 systems, the last five bits of host id is reserved for identifying hosts and the first three bits of host id is designated as subnet id. By this assignment, we just make use of one class C IP address, say 192.168.0. The detailed assignment is given below.

Subnet Id	Subnet distribution	Subnet Mask	Subnet Number	Example IPs
Id=0	192.168.1.0 to 192.168.31	255.255.255.224	192.168.1.0	192.168.1.10, 192.168.1.21
Id=32	192.168.1.32 to 192.168.1.63	255.255.255.224	192.168.1.32	192.168.1.40, 192.168.1.51
Id=64	192.168.1.64 to 192.168.1.95	255.255.255.224	192.168.1.64	192.168.1.76, 192.168.1.93
Id=96	192.168.1.96 to 192.168.1.127	255.255.255.224	192.168.1.96	192.168.1.100, 192.168.1.113
Id=128	192.168.1.128 to 192.168.1.159	255.255.255.224	192.168.1.128	192.168.1.130, 192.168.1.153
Id=160	192.168.1.160 to 192.168.1.191	255.255.255.224	192.168.1.160	192.168.1.170, 192.168.1.183
Id=192	192.168.1.192 to 192.168.1.223	255.255.255.224	192.168.1.192	192.168.1.198, 192.168.1.218
Id=224	192.168.1.224 to 192.168.1.255	255.255.255.224	192.168.1.224	192.168.1.228, 192.168.1.248

Typically, in each subnet, the first IP (Eg., 192.168.1.32) is reserved for the switch that is used to interconnect hosts in that subnet, and other addresses starting from 192.168.1.33 is assigned to hosts. By subnetting, for the above case study, exactly one IP is unused for each subnet, overall 8 IPs are unused. This brings 95% address space efficiency. This design is practically feasible as 48-port switches can be used to create subnets. The routing table for the above case study is given below. For an incoming packet with IP address, 192.168.1.76, the switch performs (192.168.1.76 AND 255.255.255.224)=192.168.1.64; this means that the host is part of the subnet 192.168.1.64. The network id is 192.168.1 and the subnet id is 64, thus the packet is routed through port 3.

Subnet Mask	Subnet Number	Out port
255.255.255.224	192.168.1.0	1
255.255.255.224	192.168.1.32	2
255.255.255.224	192.168.1.64	3
255.255.255.224	192.168.1.96	4

Remark:
1. Suppose, we are instructed to use only 24 port switches, not 48 port switches as part of design, then one can bring in one more level of subnetworking (subsubnetworking) within 192.168.1.64 subnet. In the first level of subnet, the first three bits identify the subnet and in the second level, the first four bits identify the subsubnet. That is, 192.168.1.64 is split into
`subsubnet 1:` 192.168.1.64 to 192.168.1.79
`subsubnet 2:` 192.168.1.80 to 192.168.1.95
Each subsubnet interconnects 16 hosts. IP mask to identify the subnet remains the same; 255.255.255.224 and for the subsubnet it is 255.255.255.240. Level 1 switch, by performing (192.168.1.86 AND 255.255.255.224)=192.168.1.64, identifies that the packet is meant for subnet 64. Level 2 switch, by performing (192.168.1.86 AND 255.255.255.240)=192.168.1.80, knows that the packet is meant for subsubnetwork 2.

2. We can also use a Class B address, say 172.16.1.2 for designing a campus network satisfying the above requirements. In such a case, mask used is either 255.255.0.224 or 255.255.255.224.

4.1 Some Noteworthy Protocols

We shall now discuss commonly used protocols while designing a typical campus network. We shall also highlight the role of domain spaces and the mapping between domain names and IP addresses.
Network Address Translation (NAT) Protocol: Assigning a unique global IP to each machine in a campus network may not be a wise choice as it may not effectively utilize the bandwidth offered by its campus network. Consider a campus network with 1 Gbps link speed; i.e., the campus network is connected to the external world with 1 Gbps link. Since each IP packet is of size 1500 bytes, a 1 Gbps link with effective bandwidth 500 Mbps, can contain approximately 2^{16} packets. A campus network with 1000 hosts, assuming each host generates 10 IP packets at a given time frame, then there can be approximately 2^{13} packets along the internet pipe which is just 12.5% of what the campus network offers.

It is natural to ask, can we identify all 2^{13} packets of a given campus network using just one global IP and let there be some local scheme to identify the end host inside the network. The answer to this is to reserve a portion IP addresses under the banner local IPs (private IPs) for configuring machines inside the network and the campus network (the master switch) is identified using just one global IP. By this way, we make use of global IPs effectively and bring in more campus networks into the internet. Private IPs can be reused, i.e., two campus networks may follow the same private IP scheme for configuring hosts inside their network. Moreover, private IPs are reserved only for local use and will not be used by any node in the global network.

Given that a node in a network can be assigned a local IP, how does it identify itself in the global network. The answer to this question is NAT. NAT provides a mapping between a local IP and global IP, and the entries of NAT table are given below. NAT table is maintained at the core switch; before sending the packet, a NATID is assigned and included in the packet so that when a response packet comes back to the master switch, the local host can be easily traced with NATID.

Local IP	Packet Label	Global IP	NAT ID
172.16.1.2	P1	14.1.178.1	ID1
172.16.1.2	P2	14.1.178.1	ID2
172.16.1.3	P1	14.1.178.1	ID3

Address Resolution Protocol (ARP): While creating a campus network, we may use bridges (layer 2) or switches (layer 3) to interconnect machines. If bridges are used for interconnection, then forwarding of packets is based on MAC addresses as bridges can understand only MAC addresses. When a bridge based subnetwork is interfaced with a switch based subnetwork, then mapping between MAC and IP address is necessary which is accomplished using ARP. Similar to NAT, ARP maintains a table mapping MAC and IP addresses.
Internet Control Message Protocol (ICMP): Protocol responsible for sending error messages such as `host unreachable`, `destination not found` is sent by ICMP which operates at Layer 3. When an IP datagram cannot discover the destination host or the next router, then such error messages are sent to the node that triggered the data transfer.

Domain Name Service (DNS): User application, often specifies the name of the machine (www.google.com, www.iiitdm.ac.in) for performing data transfer between end hosts. However, the network gadgets (routers) can understand only IP addresses for routing packets in internet. Thus, there must be a centralized table mapping names and its corresponding IP addresses. This table is called DNS table and maintained by centralized DNS servers. It is important to note that each application is implemented in a machine and it runs in some logical port of that machine. Thus, identifying a machine with name is equivalent to identifying a machine with its IP address. Commonly used DNS servers are `8.8.8.8` and `8.8.8.4` whose sample entries are given below;

Name	IP Address
www.iiitdm.ac.in	202.92.1.2
www.iitm.ac.in	219.96.17.198

Dynamic Host Configuration Protocol (DHCP): For a campus network with 50 departments each having 200 systems, how does IP address assignment take place. Do network administrators follow static or dynamic scheme for address assignment. Typically, in a campus network, core switches and intermediate switches get static IP as it helps in monitoring and troubleshooting a network, whereas, for end machines, dynamic scheme is followed. To release dynamic IPs to end hosts, each campus network has a DHCP server which runs DHCP application using which dynamic IPs are released to the requested machine. While assigning an IP to the requested machine, DHCP server consults its table that contains a list of used and free IPs. Also, DHCP stores the MAC address of the requested machine before releasing an IP. A snapshot of a table at DHCP server is shown below;

MAC Address	IP Address	Status
08:9e:01:38:49:a1	172.16.1.2	In use
08:9e:01:38:49:a2	172.16.1.3	In use
	172.16.1.4	Free
	172.17.1.3	Free
	172.18.1.3	Free
	172.18.1.3	Free

Authorities: Who controls (monitors) the internet ? Who performs the mapping between names and addresses ? Who releases public IPs to campus networks ? The Internet Corporation for Assigned Names and Numbers (ICANN) is the central authority situated in US who monitors the entire internet. ICANN has three major wings; Domain Name assignment, IP address assignment, protocol standards. Internet Engineering Task Force (IETF) looks after protocol standards. Whenever a protocol is discovered, after experimentation with Class E IP addresses, a white page explaining the protocol is prepared and submitted to IETF for comments. A document with a label RFC (Request For Comments) is generated and IETF assigns a number, for example RFC 791 (white page of Internet Protocol) which will be circulated among researchers to get their opinions before approval. All protocols in use today has RFC labels approved by IETF after taking concurrence from researchers.

Internet Assigned Numbers Authority (IANA), an organization under ICANN looks after public IP address assignment to campus networks. To ensure uniqueness among IP addresses assigned to campus networks, ICANN closely works with IANA. In reality, each country has a central authority (IANA) who is responsible for releasing IPs to campus networks within that country.

DNS facility is provided by ICANN and a central DNS server is also maintained by them. In practice, there are many DNS servers having the same copy of the map, and to ensure consistency among DNS servers, the central DNS works in coordination with other DNS servers.

More on IP : IP Header
As IP plays a crucial role in routing, we shall revisit IP and present the details of its header. The IP header is illustrated below.
1. The version bits identifies whether the packet is IPv4 or IPv6 packet. By default, each IP packet has a header of size at least 20 bytes. Since the header is variable in size due to options field, the header length field is also included in the header.

2. Type of service include `high throughput` service for FTP applications, `high delay` service for HTTP applications. For FTP based application, typically, a sequence of packets are sent across the network and hence a sizable portion of network bandwidth must be reserved by IP for FTP traffic. In case of HTTP applications, the application server is busy handling large number of requests and hence, the client must have a reasonable time out to avoid frequent retransmissions. Therefore, high delay service is demanded to IP.

3. Time to Live (TTL): An IP packet generated by the source visits many routers before it reaches the destination. What if the receiver is down or does not exist at all in the internet. In such a case, IP packet may simply loop round the network wasting the bandwidth. To handle this situation, a cap is fixed on each packet which dictates the maximum number of routers a packet can visit in a network. Suppose, the sender sets the TTL value to 254, then whenever the packet visits a router, the router decrements the TTL by one and pass it to the next router. On decrementing, if the TTL becomes zero, then the packet is discarded by that router and no further transmission of that packet into the network. This ensures, the packet does not cycle round the network unnecessarily.

Version(4 bits) | Header Length(4 bits) | Type of Service(8 bits)	Data Length(16 bits)
Fragment Identifier(8 bits)	Flags(4 bits) | Fragment offset(4 bits)
TTL(8 bits) | Transport Protocol (8 bits)	Checksum (16 bits)
Source	IP Address (32 bits)
Destination	IP Address (32 bits)
Options(variable in size)	Padding bits
Application Data(variable)	Application Data(variable)

4. **IP Checksum:** This checksum is computed only for the header to ascertain whether destination IP is corrupted during the data transmission. We highlight the fact that the checksum included in TCP(UDP) also includes source and destination IP addresses as part of the computation.
5. **Options Field:** If NAT protocol is employed at the source, then NAT ID is included as part of the options using which response packet from the receiver can be routed back to the intended source.
6. **Fragmentation Reassembly:** IP packet, on travel, visits many heterogeneous networks, and the maximum packet size fixed by the network varies from one network to the other. The Maximum Transmission Unit (MTU) of Ethernet is 1500 bytes, for fiber it is 4500 bytes, and for point-point network, it is 512 bytes. Since there is no uniformity in MTUs, it is the responsibility of the IP to pass the packet with correct MTU into the network. Suppose, two hosts A and B are connected via three routers having the following configurations;
(i) (A, R_1) is an Ethernet Link with MTU=1500 bytes
(ii) (R_1, R_2) is a point-point link with MTU=512 bytes
(iii) (R_2, R_3) is an Ethernet Link with MTU=1500 bytes
(iv) (R_3, B) is a point-point link with MTU=512 bytes
Note that MTU includes both the header and data. That is, if it is a simple IP header with no options field, then 20 bytes is reserved for the header and the rest 1480 bytes is reserved for the data. Each IP packet of maximum size 1500 bytes at R_1 is fragmented into a smaller packets of size 512 bytes before it is sent across (R_1, R_2) link. Further, reassembly or combining of smaller packets takes place at R_2 to reconstruct the original packet.

Packet 1	At R_1	At R_2
Header: 20 bytes Data: 1480 bytes	Fragmented into P_{11}: Header: 20 bytes, Data: 492 bytes, Flag=1, Fragment ID=123, Offset=0	
	P_{12}: Header: 20 bytes, Data: 492 bytes, Flag=1, Fragment ID=123, Offset=512	
	P_{13}: Header: 20 bytes, Data: 492 bytes, Flag=1, Fragment ID=123, Offset=1024	
	P_{14}: Header: 20 bytes, Data: 4 bytes, Flag=0, Fragment ID=123, Offset=1476	
		Header is removed from P_{11} to P_{14}, data portion is combined inorder
		new packet with Header: 20 bytes and data: 1480 bytes, sent to R_3

When fragmentation takes place, to ease the reassembly process, each fragment contains a fragment ID, offset and a flag bit. If a flag bit is set in a fragment, then some more fragments of the same packet follow the current fragment. For the first fragment, the offset is zero and the next fragment is offset by 512 bytes, and the third is 1024 and so on which helps in reconstructing the packet. Fragment ID helps to group all fragments of the same packet. Similar to the above table, R_3 does fragmentation for each packet that visits R_3 before it is sent across (R_3, B) link.

4.2 Directions for Further Research

IPv6: IPv4 is of size 32-bits which can support a maximum of 2^{32} hosts. As the internet is witnessing exponential increase in growth, the network researchers believe that we may soon exhaust IP address space. Further, IPv4 makes use NAT, ARP which is increasing processing time at local switches. In an attempt to address these scenarios, IPv5, subsequently IPv6 was discovered. IPv6 supports a 128 bit addressing scheme which is sufficient enough to handle the current demands and the growing demands. One can also remove the role of NAT by making each host unique in the global network. On the similar line, if IPv6 has a space for storing MAC address of a machine, then the role of ARP cease to exist.

Classless Inter Domain Routing (CIDR): Class based (classful) addressing scheme is rigid, in the sense that, the number of networks and hosts in each network is strictly dictated by the underlying class. What if, we follow a variable sized network ID which may reduce wastage of IP addresses further and increase the address space efficiency. Consider the following routing table; the entry 14.0.0.0/8 represents a network whose first 8 bits represents the network part of IP address.

Mask	Network Number	Out Port
255.0.0.0	14.0.0.0/8	1
255.255.0.0	14.1.0.0/16	2
255.255.192.0	14.1.128.0/18	3
255.0.0.0	15.0.0.0/8	4

In classful scheme, all of the above entries are Class A addresses with two networks; 14.0 and 15.0. However, in classless scheme, there is no classification among IP addresses, and hence masking and identifying the out port is slightly tricky. In classful scheme, the first three entries refer to 14.0 network and when we configure a routing table, there will not be entries such as 14.0, 14.1 and 14.1.128. However, these entries are possible in classless scheme. Suppose, a packet with IP address 14.1.129.32 visits a router; the router performs masking with all entries of the routing table. It turns out that 14.1.129.32 on masking with 255.0.0.0 yields 14.0.0.0, when masked with 255.255.0.0 yields 14.1.0.0, on masking 255.255.192.0 yields 14.1.128.0, this implies that the incoming IP has got match with three entries of the routing table. Which port to choose to forward the packet next when multiple match occurs ? Choose the entry satisfying the `longest prefix match` between the IP address and network number. That is, the packet will be routed via port 3 in this case. In other words, out of the three networks, we pick the network whose network label has the longest match with the incoming IP. This resolves the conflict and uniquely identifies the out port for any incoming IP.

5 Beyond Network Layer: Transport and Application Layers

The fundamental problem in computer networking is to transfer a data between two systems which are physically far apart. The data exchanged is application dependent; for which, hosts on the network run specific protocols using which exchange takes place. A protocol is a set of rules (a sequence of steps or an algorithm) that accomplishes the desired task. Commonly used application dependent protocols which are run on application layer are;

(i) Hyper Text Transfer Protocol (HTTP) - meant for exchanging website contents between two nodes
(ii) File Transfer Protocol (FTP) - to facilitate transfer of a file between two nodes on the network
(iii) Telnet - supports remote login
(iv) Simple Mail Transfer Protocol (SMTP) - facilitates exchange of emails between an email client and server
(v) Simple Network Management Protocol (SNMP) - supports monitoring and maintenance of a network
(vi) Domain Naming Service (DNS) - mapping between domain names and IP addresses (vii) Dynamic Host Configuration Protocol (DHCP) - Responsible for releasing dynamic IPs.

Typically, each application is run on a port (logical or software port); for example, port 80 for HTTP, port 21 for FTP, port 53 for DNS, etc. A client who wishes to have the HTTP service connects to the server at port 80. Typically, a server supports listening to multiple clients at port 80. We shall now discuss in detail, how the port assignment is done in practice, who provides the end-end connectivity between a client and server, who is responsible for flow control/error control. To accomplish these tasks, we make use of *transport protocols*, namely, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), and these protocols are implemented in transport layer which is below the application layer.

Services expected out of the transport layer: (i) Assignment of a port to an application requested by the application layer (ii) providing connection less or connection oriented service between end hosts (iii) Reliable service - ensuring error/flow control between hosts (iv) handling multiple clients at the server requesting the same service (v) Running multiple applications at the server in different ports.

UDP: This provides a connection less service between hosts. That is, no explicit path (virtual circuit) is created between end hosts before data transmission. Applications such as DNS, DHCP, SNMP, Multimedia (Real-time Transport Protocol) are run on top of UDP. Typically, these application data fits in one packet and these packets are having relatively less priority over HTTP/FTP packets. Error detection is implemented using checksum algorithm.
Remarks:
1. UDP packets, on travel, may get dropped by any of the intermediate routers due to congestion which will not be notified to the client. Since it is a connection less service, no acknowledgements (ACK/NACK) will be sent to the sender intimating the packet drop.
2. If there are multiple UDP packets from the same client, then each packet may take a different path, depending upon the network traffic to reach the same server.
3. If SNMP packets such as periodic update on network traffic, statistics on congestion at routers, etc., get dropped, then it will not have any serious impact on the monitoring node. Since these update packets are generated at regular intervals, the monitoring node can learn from the next update packet even if the current packet gets dropped due to network traffic.
4. On the similar line, if DNS or DHCP packet get dropped, the user application can refresh (making a re-request) after a time-out and thus, not much impact on the application.

TCP: TCP provides a connection oriented and reliable service between end hosts. That is, a virtual circuit is created between end hosts, sliding window protocol is operated between end hosts for flow control and error detection is performed using checksum.
Remarks:
1. Applications such as world wide web (WWW) that uses HTTP, email uses SMTP, etc., are implemented using TCP. Similar to UDP, TCP runs these applications at specific ports and all clients who wish to access WWW, Email, etc., must connect to these ports.
2. For all TCP based applications, virtual circuit is created for each TCP request between end hosts. The acknowledgment packet generated at the receiver contains the receiver window size allotted for this request, say 8. On receiving ACK, the sender operates selective reject sliding window scheme with SW=8 and RW=8 for all data packets.
3. Due to selective reject scheme, the sender gets ACK/NACK from the receiver and experience time outs in case of packet/ack/nack drop.
4. Since all data packets of a given application follow the same VCI, the inorder delivery of packets is implicitly guaranteed at the receiver.

End-End Data Transmission:
Application layer protocols pass the data to be transferred, referred to as *message* to the transport layer. Based on the application, appropriate transport protocol (TCP or UDP) is identified and the corresponding port of that application is assigned. Port number and other fields such as sequence number, acknowledgment, etc are included in the header of the transport data unit. For TCP data units, sequence number field, the space for acknowledgment are included in the header as it provides a connection oriented service. The size of the TCP header is at least 20 bytes and it is a variable due to the field *options* in the header. Flags bit helps in identifying whether acknowledgment sent is positive or negative. The other fields in flags include
SYN bit for synchronization; TCP follows three-way handshake mechanism to establish connection between a client and server. A client who wishes to perform TCP transfer makes a request by setting SYN bit. In response, the sender responds with ACK which includes the receiver window size reserved for this connection. The client confirms the connection by sending another signal to the server. This completes three handshakes.

RESET bit for reset or to abort the connection; when the data transmission experiences too many packet drops, either the sender or receiver can opt for RESET.

For TCP packets, the receiver window size is also sent to the sender so that selective reject sliding window with appropriate SWS and RWS can be implemented between the sender and receiver.

Urgent pointer gives the beginning of the non-urgent data in the data unit. In the data unit, just after the header, a portion is reserved for urgent data followed by non-urgent data (the actual data). The urgent data includes the change in the actual window size against the advertised receiver window so that the receiver can learn about the network traffic.
TCP Header:

Source Port (16 bits)	Destination Port (16 bits)
Sequence	Number (32 bits)
Positive or Negative	Acknowledgment (32 bits)
Header Length (8 bits) | Flags (8 bits)	Receiver Window (16 bits)
Checksum (16 bits)	Urgent Pointer (16 bits)
Options (variable in size)	Padding (16 bits)
Application Data (variable in size)	Application Data (variable in size)

UDP Header

Source Port (16 bits)	Destination Port (16 bits)
Message Length (16 bits)	Checksum (16 bits)
Application Data (variable in size)	Application Data (variable in size)

TCP and UDP Checksum
The checksum algorithm used in TCP and UDP is computed on TCP (UDP) data, header, and part of IP header which includes source IP address, destination IP address and the protocol number. The protocol number field of IP header intimates IP whether the transport layer protocol is TCP or UDP. Note that IP checksum is computed on IP header but not on the data, whereas TCP (UDP) checksum is computed on both header and data. The computed checksum is included in the TCP (UDP) header which is 16-bits.

Handling multiple requests: Port Switching Application such as DNS, DHCP and Multimedia which are UDP based, the server typically receive many requests for these applications. The operating system at the server listens continuously for requests at specific ports, and assigns process id (pid) once the request is accepted. If the size of pid is 16 bits, then the server can accept at most 2^{16} requests. This is very small considering the size of the internet. Thus, we need a different strategy to handle a large number of requests.
Strategy: Operating systems reserve logical port numbers 0 to 1024 for use by specific applications and the other port numbers from 1025 to 65,536 are free, and can be used by user applications. A client who wishes to make a request, say DNS request, first uses the standard port number (53) to connect to the server. Further, the server switches to any one of the free ports in the range 1025 to 65,536 and intimates the client that any further communication will happen at the new port. This way, the standard port number is not blocked due to network traffic which is used only for connection establishment, and the port can be used by other clients easily. Further, to each of these newly created ports one can have a maximum of 2^{16} requests. Typically, each campus network is assigned one port from 1025 to 65536 during connection establishment phase of the very first client request, say 1025, and all other clients of the same network will be switched to the port 1025 with different pids.

5.1 Congestion in TCP(UDP) traffic

Hosts in the internet generate many TCP/UDP packets (for example; multimedia packets generated by RTP, WWW packets by HTTP) which are routed to the destination node by intermediate routers by consulting the routing table. Since routers have finite buffer, it is possible that the number of TCP/UDP packets visiting a router may exceed the capacity of the buffer, resulting in dropping of TCP/UDP packets. The congestion due to flooding of TCP/UDP packets is a common scenario in the internet and a serious concern. Congestion must be controlled to ease the network traffic. It is important to note that for UDP packets, there is no acknowledgment sent to the sender intimating the packet drop, instead, the sender learns at the end of time-out and retransmits the packet. Moreover, the intermediate routers do not aware of the number of UDP packets generated by a host and the arrival time of the packets at the router. As both are random variables, the routers do not have choice but to drop the packets. Since there is no flow control for UDP packets, no intimation is sent to the sender. Thus, congestion cannot be controlled for UDP based end-end data transfer.

For TCP traffic, although, an explicit connection is created using VCI and each of the intermediate routers know the number of VCI circuits, there is still a scope for congestion at routers. It is possible that each VCI circuit can transmit more than one packet as it depends on the size of RW advertised by the receiver for that TCP connection. Moreover, the advertised window is contained in the TCP header which intermediate routers cannot access as their access boundary is limited to the IP header. Further, routers can get to see both UDP and TCP packets at a given point of time and hence, allotting a buffer of size at least two for each TCP connection may not be a feasible choice. This calls for a congestion control scheme for TCP based traffic. Since TCP performs three-way handshake and establishes a virtual circuit between end hosts by reserving a buffer cell at each of the intermediate routers, there is a scope for controlling the congestion to ease network traffic.

TCP Congestion Control Scheme
TCP follows four strategies to control the network traffic; (i) slow start (ii) exponential increase (iii) multiplicative decrease (iv) additive increase. Let us assume that a TCP connection with advertised receiver window size 8. Thus, the sender implements selective reject with SW=8 and RW=8. It is possible that if the sender sends 8 frames at a time, it may experience congestion as there may other TCP packets injected into the network. For explanation purpose, let us assume that there 5 simultaneous TCP connections between a client and server which are separated by 10 routers with each router has a buffer of size 30 cells (can store 30 packets). Clearly, 5 connections can send 40 packets which intermediate routers cannot handle leading to packet drops.
TCP congestion control scheme works as follows; though RW=8, the client does not transmit all 8 frames simultaneously. Instead, it performs a slow start after connection establishment phase.

Slow start: Each TCP connection at a host works with RW=1 and thus invokes stop and wait scheme to begin with. Initially, all TCP connections send exactly one packet and wait for the acknowledgment. Suppose, the ACKs for all 5 data packets arrive before the time-out, then the client knows that the traffic is under control and decide to go for exponential increase.

Exponential Increase: For each TCP connection, the client increase the window to two (referred to as *congestion window*) and thus works with RW=2. If ACK is received for the two packets, then it increases the congestion window to four and works with RW=4. This process of increasing the congestion window continues until congestion window becomes RW=8. It may so happen that some of the TCP connections eventually reach RW=8 and the rest works with less than RW$<$ 8. For this example, during the second iteration, the client sends two packets for each of the applications and thus routers witness 10 packets. Assuming, 10 ACKs (or 5 cumulative ACKs) are received at the client, then the client increase the window to four and sends four packets for each TCP connection. Assuming 20 ACKs are received, then the sender attempts a data transmission with congestion window=8 packets. However, this time TCP 4 (4^{th} TCP connection) and TCP 5 may not receive ACKs for all the packets if operating system sends 8 packets of TCP 1 followed by TCP 2, TCP 3, TCP 4, and finally TCP 5. If operating system follows time slicing which means the first packet of all five TCP connections is sent first followed by the second packet of all five connections and so on. If time slicing is followed, then packet drop will be experienced by all five connection for the 7^{th} and 8^{th} packets. At the end of this strategy, the sender learns that network traffic is smooth for RW=4 but not for RW=8. Hence, the sender must operate its congestion window in the range 4 to 8 and thus go for multiplicative decrease.

Multiplicative Decrease: On learning from the network that RW=8 creates congestion in the network, the sender reduces the congestion window by half and works with RW=4. In general, it may so happen that out of x TCP connections, y TCP connections experience congestion for congestion window 8 and the rest work fine. In such a case, for y connections, the congestion window is reduced by half and the rest work with RW=8. To determine the optimal window size, the sender go for additive increase strategy.

Additive Increase: For each of the five connections, the sender works with RW=4, assuming there is no congestion, then the sender increases the window by one and works with RW=5. Assuming the network traffic is smooth, then on receiving the ACK, the sender increases the window again by one and works with RW=6. If suppose, the sender experiences congestion for RW=6, then multiplicative decrease strategy is invoked, and the sender works with RW=3. Further, it follows additive increase. Thus, multiplicative decrease and additive increase strategy helps us to identify the 'right' congestion window for end-end data transmission.

Flow and Error control in TCP/UDP
For TCP traffic, both flow and error control are taken care of by end hosts. Checksum in the header identifies the error in the header as well as the data. For example, if port number or IP address get corrupted during the transmission, it will be detected by the end host. Further, checksum in IP header is of use for intermediate routers to detect if there is corruption in destination IP address. Sliding window algorithm is implemented with appropriate RWS for flow control. Since TCP takes care of both flow and error control, it is optional for the data link layer whether to perform CRC-32 and sliding window or stop and wait protocols.
For UDP traffic, although checksum is included in the header, the receiver cannot send NACK to the sender if there is an error in transmission as it is a connection less service. If there is an error, such a packet is discarded by the receiver. Further no flow control scheme is implemented by UDP. Thus, it is the responsibility of the lower layer (data link layer) to implement both flow and error control for UDP networks. At the lower layer, flow and error control is point-point rather than end-end. Hence, between any two adjoining nodes, data link layer implements CRC-32 and stop and wait (or sliding window).

6 The Internet: A big picture in a nutshell

It is time to integrate various concepts discussed in this course to get a big picture of the internet. The internet is simply a interconnection of various campus/academia networks using network gadgets such as bridges (Layer 2 switches), switches (Layer 3), routers (Layer 3) and gateways (Layer 3). The fundamental problem is to perform data transfer (email, multimedia, file, web content) between a pair of hosts in the network. As internet is a collection of heterogeneous networks, various protocols (set of rules or algorithms) were developed to accomplish desired tasks and to bring in uniformity in data communication. Let us consider a pair of hosts H_1 and H_2 between them, say a file transfer would take place; we shall list the sequence of activities that happen at H_1 and H_2 for a successful data transfer.

1. The first task is to assign IP addresses for H_1 and H_2. Typically, H_1 and H_2 are connected to a campus network; on booting H_1 and H_2, the machine first assigns the IP address `0.0.0.0`, subsequently, sends a DHCP request to the local DHCP server. By default, the machine generates a broadcast packet with IP address `255.255.255.255`, and this packet, on travel, consults the routine table at intermediate routers to reach the DHCP server. On seeing a DHCP request packet, the DHCP server releases a private IP to H_1 and H_2. DHCP server maintains table of free IPs and used IPs. For all used IPs, the server stores the MAC address of the machine that requested a dynamic IP. At this point in time, both H_1 and H_2 are connected to the local campus network.

2. Based on the application request, the operating system identifies the appropriate application layer protocol (FTP, HTTP, etc.). Often, the domain name (www.google.com, ftp://lab.cs.iiitdm.ac.in) of the destination host is specified instead of the IP address of the destination host. Application layer makes use of DNS service to map the domain name to the IP address by sending a DNS request to global DNS servers (`8.8.8.8 or 8.8.8.4`). The port number of the application can also be supplied to the requested host by the DNS server.

3. The application data at the application layer is referred to as `message` which is given to the next layer (transport layer) in the protocol stack. Based on the application, the operating system invokes either TCP or UDP which contains the port number of the application. Further, if it is a TCP based application, then TCP offers both flow control and error control between H_1 and H_2. If it is a UDP traffic, then flow/error control is offered by the lower layers. The data unit of transport layer is referred to as `segment` in the literature.

4. At this point in time, we know the IP address of the destination, the port at which the application is running, whether to follow connection oriented or connection less service. How do we identify the destination (H_2) in such a big network. Network layer takes the lead in identifying the destination by employing routing algorithms. Firstly, H_1 which is part of a campus network must obtain a global IP before it initiates the route discovery. Typically, a campus network has a master switch (router) which is connected to the service provider switch, has a global IP. The service provider ensures campus networks are interlinked via its switch and inturn provide access to the external world. The network layer runs NAT protocol using which the mapping between local IP and global IP of H_1 is done. Further, the packet from H_1 consults the routing table of the master switch to identify the next router, consultation is done again at the next router to identify the next next router, and so on. Finally, the packet from H_1 reaches the master switch to which H_2 is connected. The network layer employs RIP, OSPF to discover a path between H_1 and H_2. If end host is unreachable, then the message is communicated to H_1 by the ICMP protocol. The data unit in the network layer is referred to as `packet`.

5. The packet from the network layer is given to data link layer which provides point-point flow and error control. It typically employs CRC-32 for error detection and implements sliding window algorithm between adjoining nodes. The data unit of this layer is referred to as `frame` in the literature.

6. Finally, the physical layer, gets a sequence of frames to be transmitted across the network which it does by sending bit by bit using some encoding techniques. Manchester or Differential Manchester is used in practice for encoding of bits into signals. The encoded bits are transferred via the one or more of Ethernet, Fiber links, wireless, satellite mediums to reach H_2. The end host H_2 does the above processes in reverse to reconstruct the message.

7. The data link layer of H_2 groups bits into frames and each frame header contains information about CRC, Sequence numbers, etc., using which it validates the frame. Assuming the frame is perfect, it discards the header and pass it to the network layer of H_2.

8. The network layer cross verifies its IP with the destination address stored in the packet, and also validates the IP header by running checksum at H_2. Assuming no error in checksum, the IP header is removed and the rest of the packet is passed on to transport layer as a segment.

9. The transport layer validates the segment first by performing checksum and extract the port information from the header. The header is removed by the transport layer protocol and passes the rest of the segment to the application layer. The application layer passes the message to the appropriate application listening at specific ports to respond to H_1. Now, the response message generated by H_2 will follow the above steps to reach H_1 and this process continues till the connection termination packet is generated by the sender.

10. It is important to note that if there are 10 routers between H_1 and H_2, then application and transport layers are visited twice; once at H_1 and the other at H_2. All intermediate routers do not have application and transport layers as it operates upto layer 3. Network layer is visited 1+10+1=12 times during this process. Once at each intermediate router and the end hosts, altogether 10 visits at routers and 2 visits at hosts. At intermediate routers data link and physical layer is visited twice for each packet, and since there are 10 routers, the total visits to data link layer is 2*10+2=22 and the total visits to physical layer is also 2*10+2=22. The reason for two visits at routers is given as follows; during the first visit, CRC check is done followed by the header removal, and finally the data is handed over to the upper layer (network layer). The network layer, reduces the TTL value and identifies the next router along with the outport. The modified packet is handed over again to the lower layer (data link layer). The modified packet visits the data link layer for which CRC computation is done before it is given to the physical layer. In general, if there are k routers separating two hosts, then the number of visits to (i) application layer = 2 (ii) transport layer = 2 (iii) network layer = k+2 (iv) Data link and Physical layers = 2k+2.