UNIT-3

1.NOISY CHANNELS:

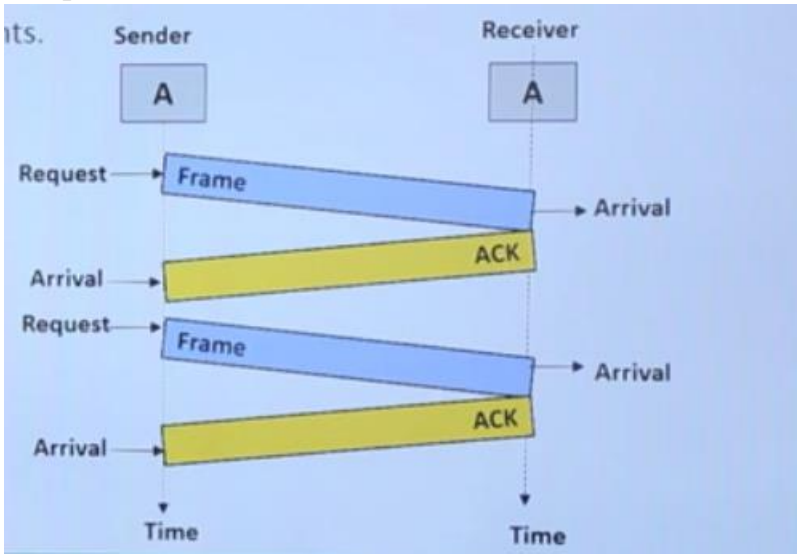
Noisy Channel is type a channel where there is a chance of frame lost, data corruption or Duplication.

Contains 4 protocols:

- Stop-and-Wait Automatic Repeat Request
- Sliding Window Protocol
  - Go-Back-N Automatic Repeat Request
  - Selective Repeat ARQ
- Piggybacking Protocol

2.Stop-and-Wait Automatic Repeat Request:

The sender sends one frame and waits for feedback from the receiver. when the ACK arrives, the sender sends the next frame. Note that sending two frames in the protocol involves the sender in four events and the receiver in two events.



Some features list:

- Error correction in stop and wait ARQ is done by keeping a copy of the sent frame and retransmitting of the frame when the timer expires i.e., when the acknowledgement is not received.
- In stop-and-wait ARQ, we use sequence numbers to number the frames.\
- The sequence numbers are 0 and 1.
- In stop-and-wait ARQ, the acknowledgement number is the sequence number of the next frame.

Design:

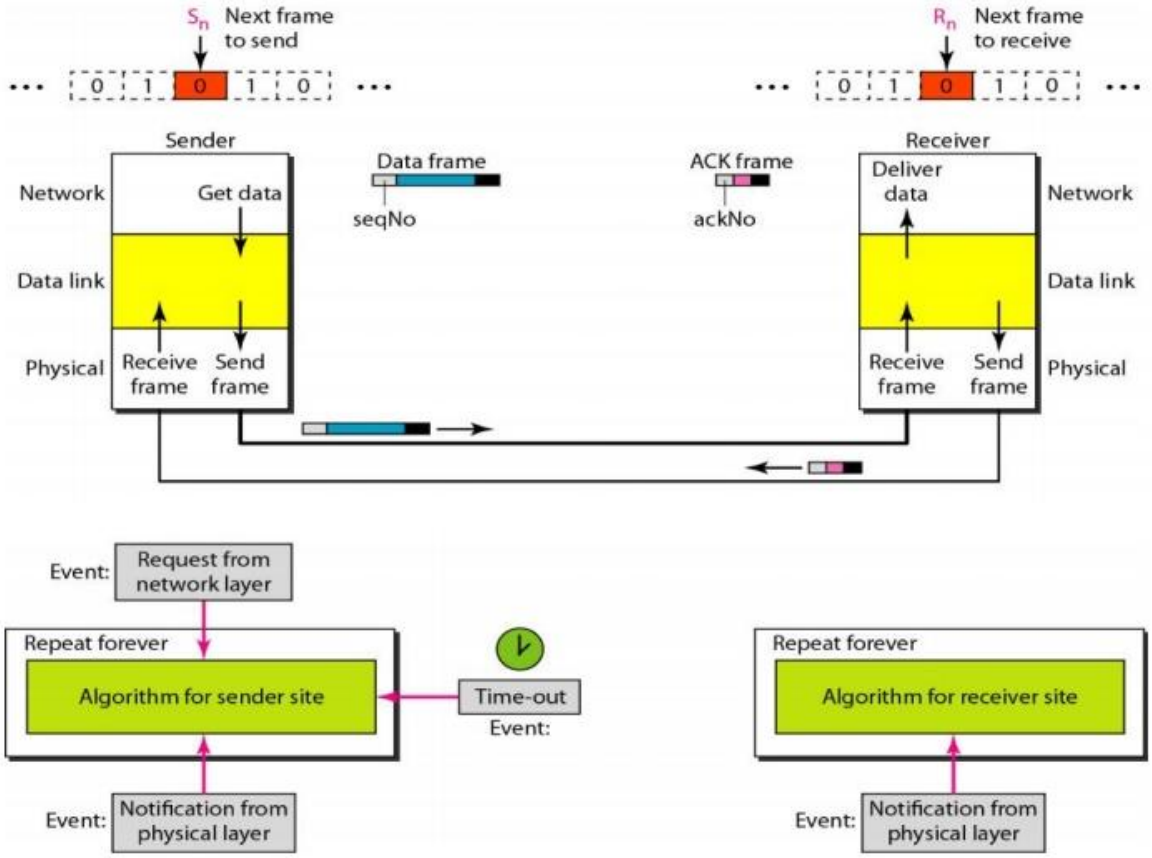


Figure 2.10 Design of the Stop-and-wait ARQ Protocol

The sending device keeps a copy of the last frame transmitted until it receives an acknowledgment for that frame. A data frame uses a seqNo(sequence number);an ACK frame uses an ackNo(acknowledgment number). The control variables for sender and receiver are Sn (next frame to send), Rn (next frame expected). If a frame is sent, the value of Sn is incremented to 1 else 0 or vice versa. And similarly, if a frame is received, the value of Rn is incremented to 1 else 0 and vice versa.

- Sn -> points to the slot that matches the sequence number of the frame sent.
- Rn -> points to the slot that matches the sequence number of the expected frame.

Example:

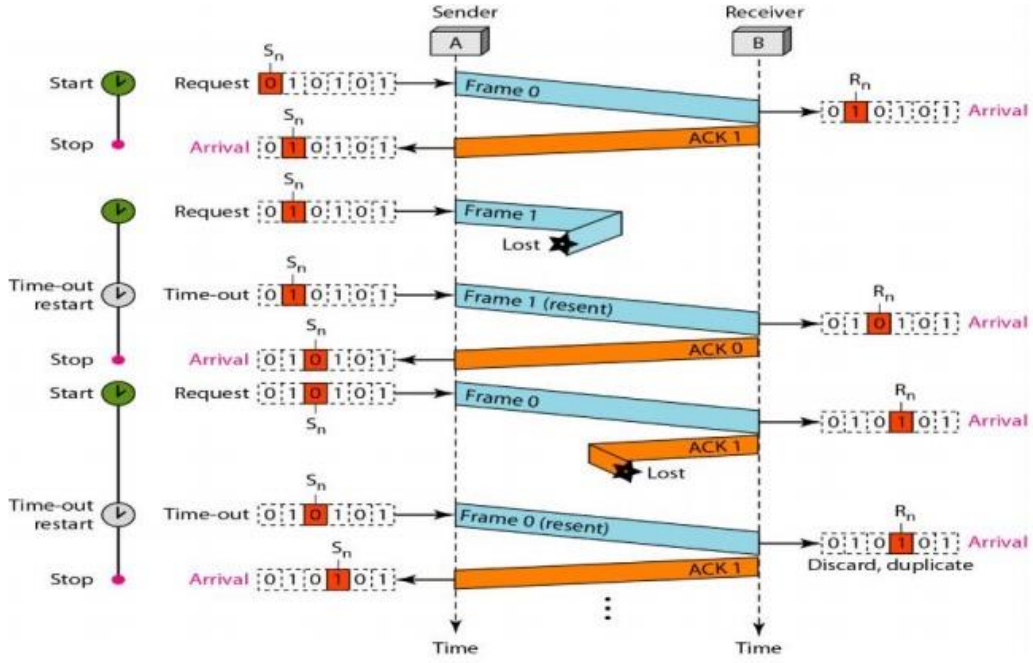


Figure 2.11 Flow diagram for Example 2.3

Frame 0 is sent and acknowledged. Frame 1 is lost and resent after the time-out. The resent frame 1 is acknowledged and the timer stops. Frame 0 is sent and acknowledged, but the acknowledgment is lost. The sender has no idea if the frame or the acknowledgment is lost, so after the time-out, it resends frame 0, which is acknowledged.

Algorithm: receiver site

Algorithm 11.6 *Receiver-site algorithm for Stop-and-Wait ARQ Protocol*

```
1  Rn = 0;                          // Frame 0 expected to arrive first
2  while(true)
3  {
4    WaitForEvent();                // Sleep until an event occurs
```


Algorithm 11.6 *Receiver-site algorithm for Stop-and-Wait ARQ Protocol* (continued)

```
5    if(Event(ArrivalNotification))   //Data frame arrives
6    {
7      ReceiveFrame();
8      if(corrupted(frame));
9        sleep();                     //Valid data frame
10     if(seqNo == Rn)
11     {
12       ExtractData();               //Deliver data
13       DeliverData();
14       Rn = Rn + 1;
15     }
16     SendFrame(Rn);                 //Send an ACK
17   }
18 }
```

sender site

Algorithms

Algorithm 11.5 is for the sender site.

Algorithm 11.5 *Sender-site algorithm for Stop-and-Wait ARQ*

```
1  Sn = 0;                          // Frame 0 should be sent first
2  canSend = true;                  // Allow the first request to go
3  while(true)                      // Repeat forever
4  {
5    WaitForEvent();                // Sleep until an event occurs
```

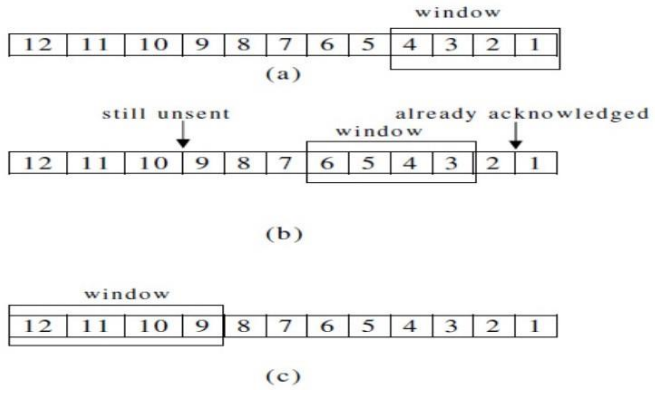

Algorithm 11.5 *Sender-site algorithm for Stop-and-Wait ARQ (continued)*

```
6    if(Event(RequestToSend) AND canSend)
7    {
8      GetData();
9      MakeFrame(Sn);
10     StoreFrame(Sn);              //The seqNo is Sn
11     SendFrame(Sn);               //Keep copy
12     StartTimer();
13     Sn = Sn + 1;
14     canSend = false;
15   }
16   WaitForEvent();                // Sleep
17     if(Event(ArrivalNotification)  // An ACK has arrived
18     {
19       ReceiveFrame(ackNo);       //Receive the ACK frame
20       if(not corrupted AND ackNo == Sn) //Valid ACK
21         {
22           Stoptimer();
23           PurgeFrame(Sn-1);      //Copy is not needed
24           canSend = true;
25         }
26      }
27
28     if(Event(TimeOut)            // The timer expired
29     {
30      StartTimer();
31      ResendFrame(Sn-1);          //Resend a copy check
32     }
33 }
```

Sliding Window Protocol:

In this flow control mechanism, both sender and receiver agree on the number of data-frames after which the acknowledgement should be sent. As we learnt, stop and wait flow control mechanism wastes resources, this protocol tries to make use of underlying resources as much as possible.

Example:



A 4-packet window sliding through outgoing data.

(a) When transmission begins

(b) after two packets has been acknowledged, and

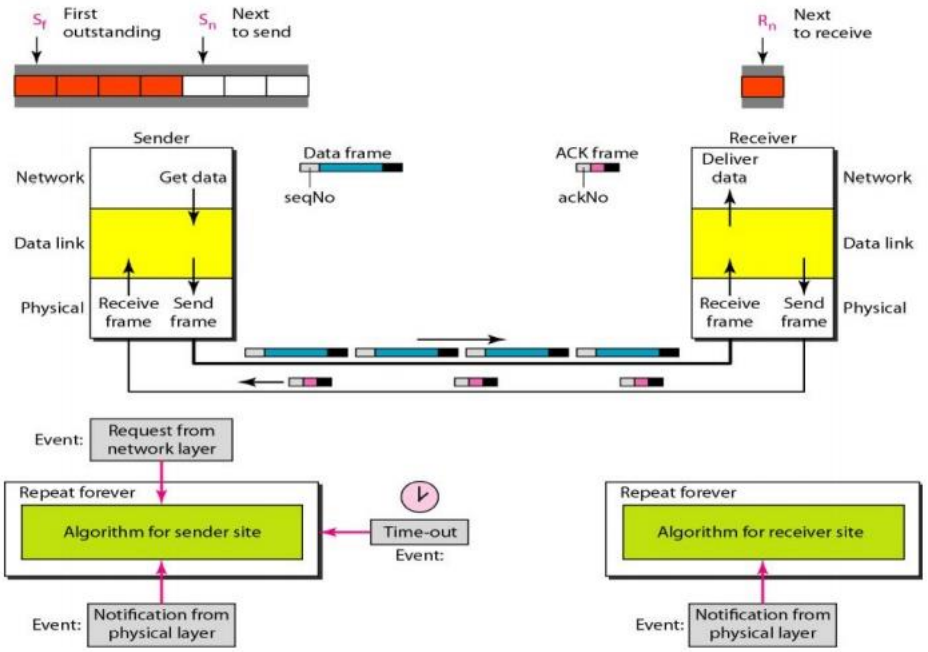
(c) after eight packets have been acknowledged.

2. Go-Back-N Automatic Repeat Request:

Go-Back-N protocol is a sliding window protocol. It is a mechanism to detect and control the error in datalink layer. In this protocol we can send several frames before receiving acknowledgments; we keep a copy of these frames until the acknowledgments arrive

When the sender sends all the frames in window, it checks for which sequence number it has received positive acknowledgement. If all frames are positively acknowledged, the sender sends next set of frames. If sender finds that it has received NACK or has not receive any ACK for a particular frame, it retransmits all the frames after which it does not receive any positive ACK.

Design:

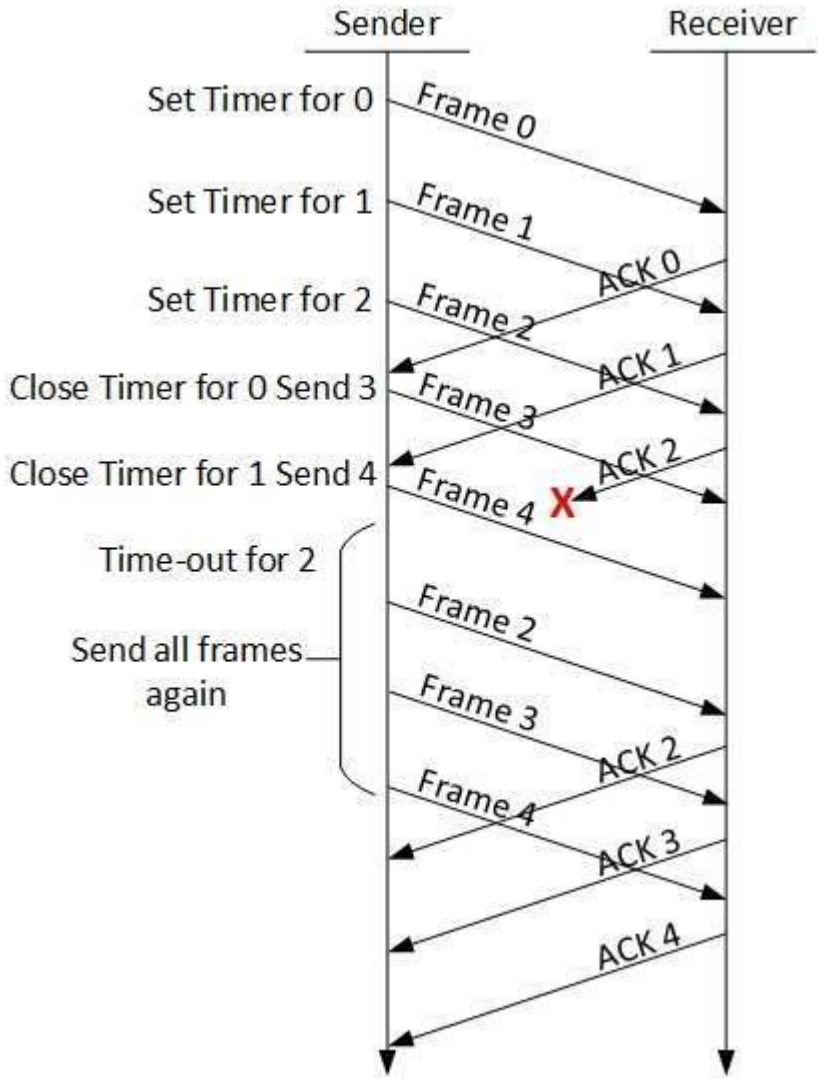


This idea is similar to stop-and-wait ARQ; the only difference is that the sender window allow us to have as many frames in transition as there are slots in the send window. The size of the send window must be less than *2m*. As an example, we choose *m* =2, which means the size of the window can be *2m*- 1, or 3. Figure 2.15 compares a window size of 3 against a window size of 4. If the size of the window is 3 (less than 22) and all three acknowledgments are lost, the frame 0 timer expires and all three frames are resent. The receiver is

now expecting frame 3, not frame 0, so the duplicate frame is correctly discarded.

Example:

The sender enables to send multiple frames without receiving the acknowledgement of the previous ones. The receiver receives multiple frames and acknowledge them. The receiver keeps track of incoming frame's sequence number.



Algorithms:

Sender site:

Algorithm 11.7 *Go-Back-N sender algorithm*

```
1  Sw = 2^m - 1;
2  Sf = 0;
3  Sn = 0;
4
5  while (true)                          //Repeat forever
6  {
7   WaitForEvent();
8   if(Event(RequestToSend))             //A packet to send
9   {
10     if(Sn-Sf >= Sw)                   //If window is full
11           Sleep();
12     GetData();
13     MakeFrame(Sn);
14     StoreFrame(Sn);
15     SendFrame(Sn);
16     Sn = Sn + 1;
17     if(timer not running)
18          StartTimer();
19  }
20
21  if(Event(ArrivalNotification)) //ACK arrives
22  {
23     Receive(ACK);
24     if(corrupted(ACK))
25          Sleep();
26     if((ackNo>Sf)&&(ackNo<=Sn))  //If a valid ACK
27     While(Sf <= ackNo)
28      {
29       PurgeFrame(Sf);
30       Sf = Sf + 1;
31      }
32      StopTimer();
33  }
34
35  if(Event(TimeOut))                   //The timer expires
36  {
37   StartTimer();
38   Temp = Sf;
39   while(Temp < Sn);
40    {
41     SendFrame(Sf);
42     Sf = Sf + 1;
43    }
44  }
45 }
```

Receiver site:

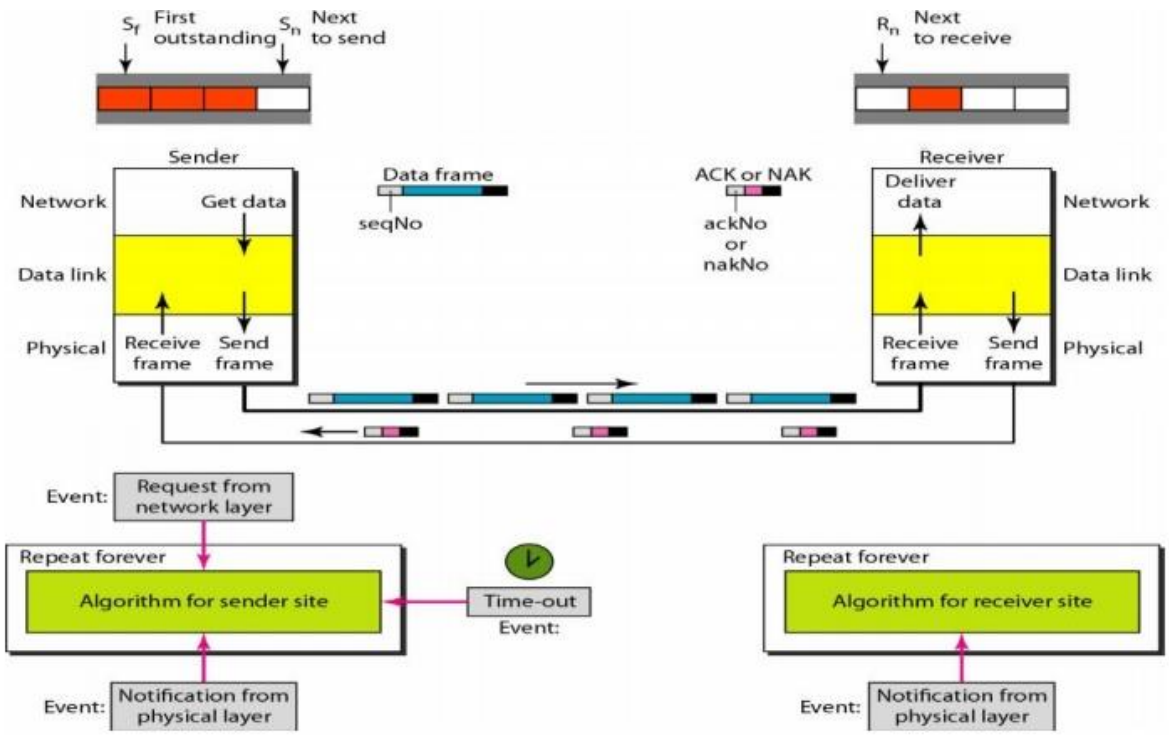
Algorithm 11.8 *Go-Back-N receiver algorithm*

```
1  Rn = 0;
2                                       //Repeat forever
3  while (true)
4  {
5    WaitForEvent();
6
7    if(Event(ArrivalNotification)) /Data frame arrives
8    {
9       Receive(Frame);
10      if(corrupted(Frame))
11           Sleep();
12      if(seqNo == Rn)                //If expected frame
13      {
14        DeliverData();               //Deliver data
15        Rn = Rn + 1;                 //Slide window
16        SendACK(Rn);
17      }
18   }
19 }
```

3.Selective Repeat ARQ:

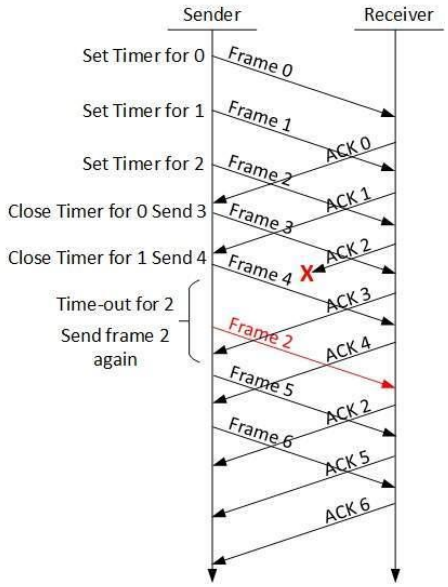
Selective repeat is also the sliding window protocol which detects or corrects the error occurred in datalink layer. The selective repeat protocol retransmits only that frame which is damaged or lost. The sender in this case, sends only packet for which NACK is received.

Design:



This is more complicated than Go-Back-N protocol. the size of the sender and receiver windows must be at most on half of *2m*. For an example, we choose *m* = 2, which means the size of the window is *2m/2*, or 2. If the size of the window is 2 and all acknowledgments are lost, the timer for frame 0 expires and frame 0 is resent. However, this time, the window of the receiver expects to receive frame 0 (0 is part of the window), so it accepts frame 0, not as a duplicate, but as the first frame in the next cycle. This is clearly an error. In Selective Repeat ARQ, the size of the sender and receiver window must be at most one-half of *2m*

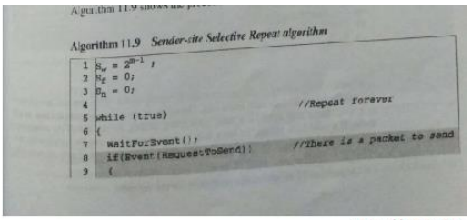
Example:



In this example, it is clear that the acknowledgement is send for which NACK is received.no n

Algorithm:

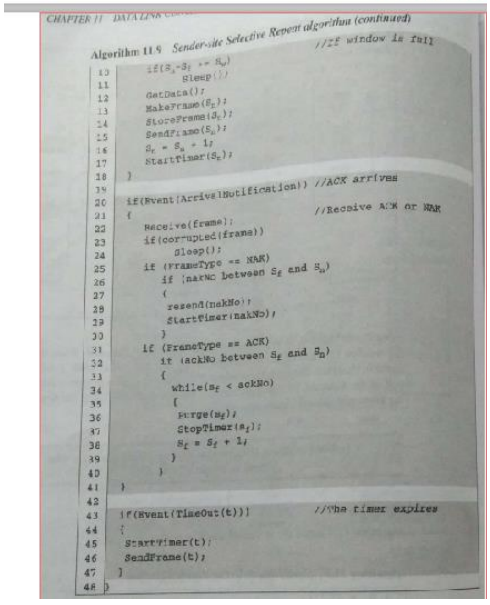
Sender site



Algorithm 11.9 *Sender-site Selective Repeat algorithm*

```
1  Sw = 2^(m-1) ;
2  Sf = 0;
3  Sn = 0;
4
5  while (true)                          //Repeat forever
6  {
7    WaitForEvent();
8    if(Event(RequestToSend))            //There is a packet to send
9    {
```

Scanned by CamScanner

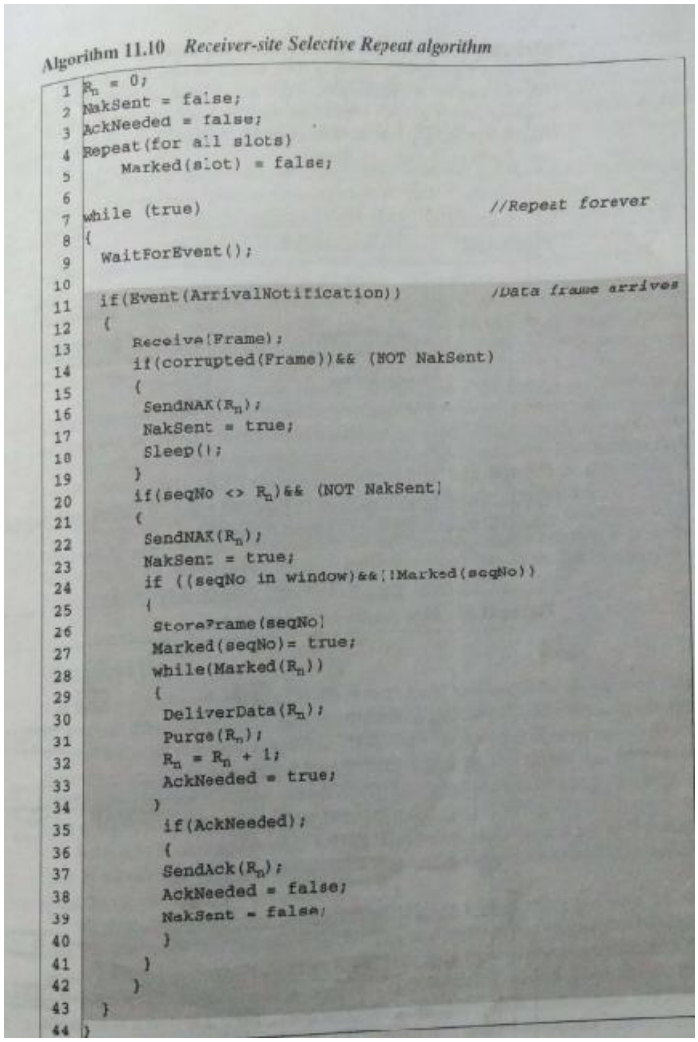


CHAPTER 11 DATA LINK ...

Algorithm 11.9 *Sender-site Selective Repeat algorithm (continued)*

```
12     if(Sn-Sf >= Sw)                   //If window is full
13       Sleep();
14     GetData();
15     MakeFrame(Sn);
16     StoreFrame(Sn);
17     SendFrame(Sn);
18     Sn = Sn + 1;
19     StartTimer(Sn);
20   }
21
22   if(Event(ArrivalNotification)) //ACK arrives
23   {                                     //Receive ACK or NAK
24     Receive(frame);
25     if(corrupted(frame))
26       Sleep();
27     if (FrameType == NAK)
28       if (nakNo between Sf and Sn)
29       {
30         resend(nakNo);
31         StartTimer(nakNo);
32       }
33     if (FrameType == ACK)
34       if (ackNo between Sf and Sn)
35       {
36         while(Sf < ackNo)
37         {
38           Purge(Sf);
39           StopTimer(Sf);
40           Sf = Sf + 1;
41         }
42       }
43   }
44
45   if(Event(TimeOut(t)))               //The timer expires
46   {
47     StartTimer(t);
48     SendFrame(t);
49   }
50 }
```

receiver site



Algorithm 11.10 *Receiver-site Selective Repeat algorithm*

```
1  Rn = 0;
2  NakSent = false;
3  AckNeeded = false;
4  Repeat(for all slots)
5      Marked(slot) = false;
6
7  while (true)                              //Repeat forever
8  {
9   WaitForEvent();
10
11  if(Event(ArrivalNotification))           /Data frame arrives
12  {
13     Receive(Frame);
14     if(corrupted(Frame))&& (NOT NakSent)
15     {
16      SendNAK(Rn);
17      NakSent = true;
18      Sleep(!;
19     }
20     if(seqNo <> Rn)&& (NOT NakSent]
21     {
22      SendNAK(Rn);
23      NakSent = true;
24      if ((seqNo in window)&&(!Marked(seqNo))
25      {
26       StoreFrame(seqNo]
27       Marked(seqNo)= true;
28       while(Marked(Rn))
29       {
30        DeliverData(Rn);
31        Purge(Rn);
32        Rn = Rn + 1;
33        AckNeeded = true;
34       }
35        if(AckNeeded);
36        {
37        SendAck(Rn);
38        AckNeeded = false;
39        NakSent = false;
40        }
41      }
42     }
43  }
44 }
```

4.Piggybacking Protocol:

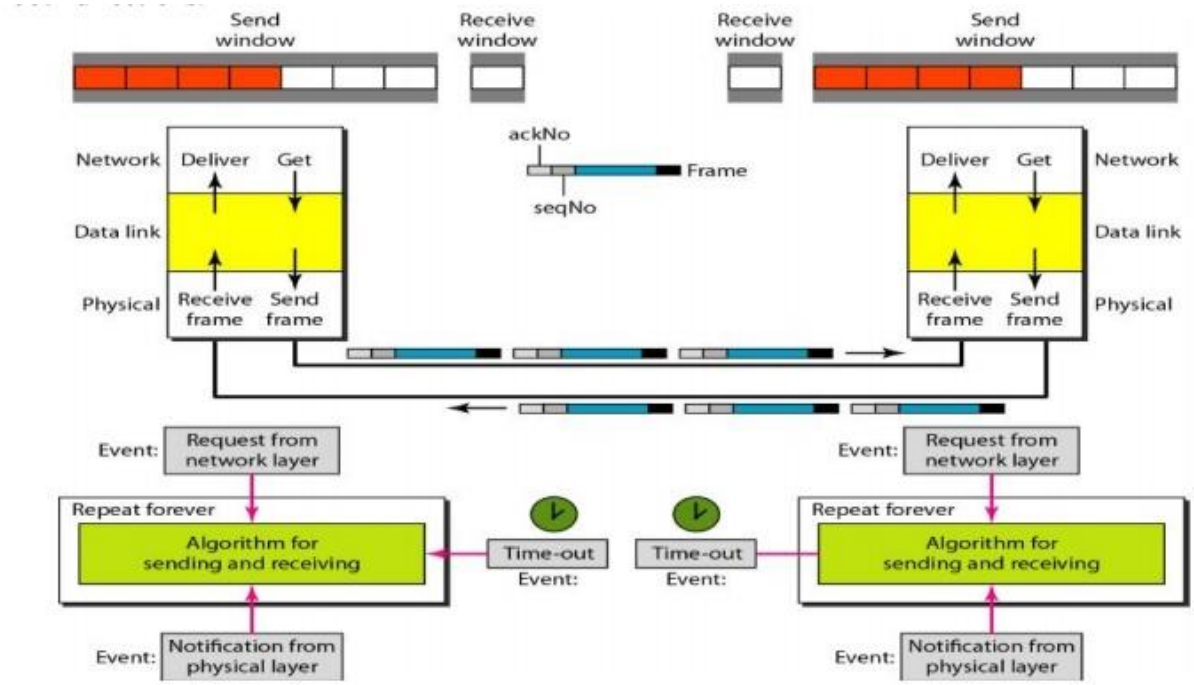
In some protocols data frames flow in only one direction although control information such as ACK and NAK frames can travel in the other direction. In real life, data frames are normally flowing in both directions, from node A to node B and from node B to node A. This means that the control information also needs to flow in both directions i.e., full – duplex in nature. A technique called piggybacking is used to improve the efficiency of the bidirectional protocols

For example, if we consider a frame is carrying data from A to B, it can also carry control information about arrived (or lost) frames from B; when a frame is carrying data from B to A, it can also carry control information about the arrived (or lost) frames from A.

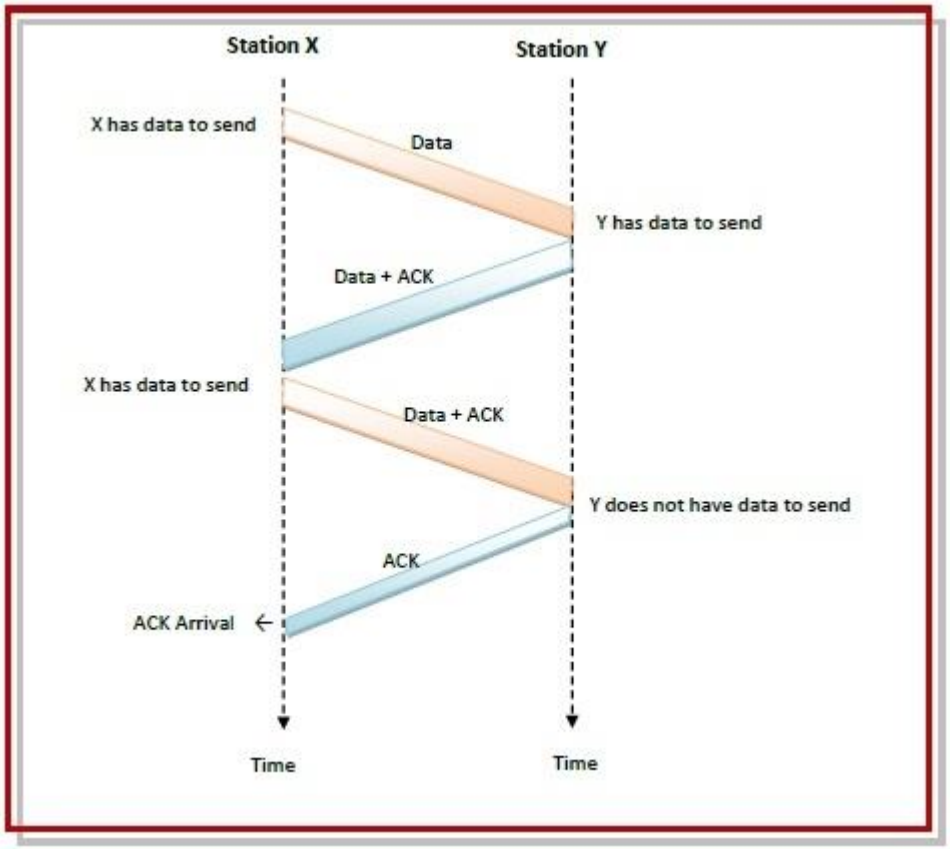
Principles:

1. If station X has both data and acknowledgment to send, it sends a data frame with the *ack* field containing the sequence number of the frame to be acknowledged.
2. If station X has only an acknowledgment to send, it waits for a finite period of time to see whether a data frame is available to be sent. If a data frame becomes available, then it piggybacks the acknowledgment with it. Otherwise, it sends an ACK frame.
3. If station X has only a data frame to send, it adds the last acknowledgment with it. The station Y discards all duplicate acknowledgments. Alternatively, station X may send the data frame with the *ack* field containing a bit combination denoting no acknowledgment.

Design:



Example:



5.NETWORK LAYER:

This layer helps to uniquely identify hosts beyond the subnets and defines the path which the packets will follow or be routed to reach the destination. The network layer is responsible for carrying data from one host to another. It provides means to allocate logical addresses to hosts, and identify them uniquely using the same. Network layer takes data units from Transport Layer and cuts them in to smaller unit called Data Packet.

Network layer defines the data path, the packets should follow to reach the destination. Routers work on this layer and provides mechanism to route data to its destination.

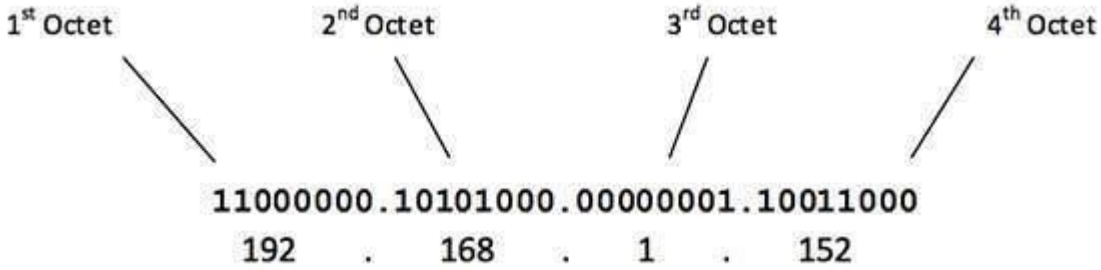
6.IPV4 protocol:

Internet Protocol version 4 (IPv4) is the fourth version in the development of the Internet Protocol. Internet Protocol version 4 uses 32-bit logical address. This protocol works at the network layer of the OSI model and at the Internet layer of the TCP/IP model. IP uses best effort delivery, i.e. it does not guarantee that packets would be delivered to the destined host, but it will do its best to reach the destination.

7.Addresses:

The IPv4 Addressing system is divided into five classes of IP Addresses. All the five classes are identified by the first octet of IP Address.

The first octet referred here is the left most of all. The octets numbered as follows depicting dotted decimal notation of IP Address −



The number of networks and the number of hosts per class can be derived by this formula −

Number of networks = 2^network_bits

Number of Hosts/Network = 2^host_bits – 2

8.Classful Addressing:

In the classful addressing, there are 5 classes in which the address space is divided: **A, B, C, D, and E**. Each class occupies some fraction of the address space. We can find the class of an address when given the address in binary notation or dotted-decimal notation by checking the first few bits or first byte.

→Class A:

The first bit of the first octet is always set to 0 (zero). Thus the first octet ranges from 1 – 127, i.e.

00000001 – 01111111
1 – 127

- Class A addresses only include IP starting from 1.x.x.x to 126.x.x.x only. The IP range 127.x.x.x is reserved for loopback IP addresses.
- **Format: 0NNNNNNN**. HHHHHHHH. HHHHHHHH. HHHHHHHH

→**Class B:** An IP address which belongs to class B has the first two bits in the first octet set to 10, i.e.

```
10000000 - 10111111
     128 - 191
```

- Class B IP Addresses range from 128.0.x.x to 191.255.x.x. The default subnet mask for Class B is 255.255.x.x.
- **Format: 10NNNNNN.NNNNNNNN**.HHHHHHHH.HHHHHHHH

→**Class C:**

The first octet of Class C IP address has its first 3 bits set to 110, that is −

```
11000000 - 11011111
     192 - 223
```

- Class C IP addresses range from 192.0.0.x to 223.255.255.x. The default subnet mask for Class C is 255.255.255.x.
- Format: **110NNNNN.NNNNNNNN.NNNNNNNN**.HHHHHHHH

→**Class D:**

Very first four bits of the first octet in Class D IP addresses are set to 1110, giving a range of −

```
11100000 - 11101111
     224 - 239
```

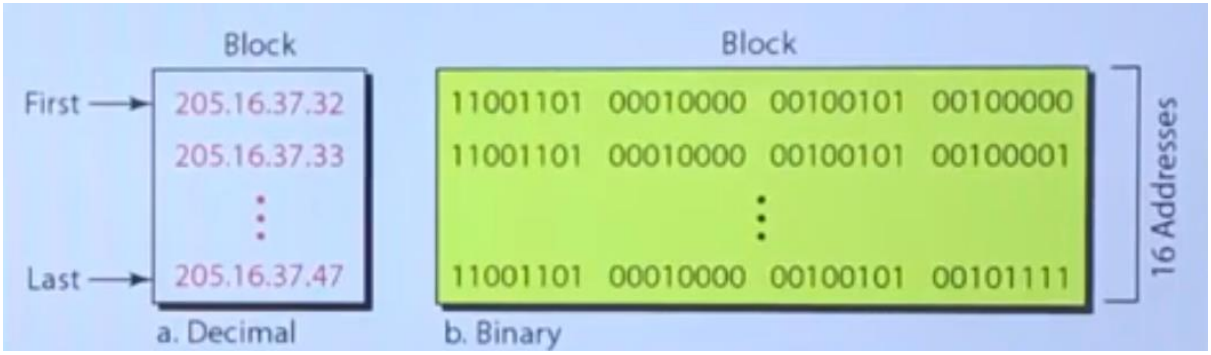
- Class D has IP address range from 224.0.0.0 to 239.255.255.255. Class D is reserved for Multicasting. In multicasting data is not destined for a particular host, that is why there is no need to extract host address from the IP address, and Class D does not have any subnet mask

→**Class E:**

This IP Class is reserved for experimental purposes only for R&D or Study. IP addresses in this class ranges from 240.0.0.0 to 255.255.255.254. Like Class D, this class too is not equipped with any subnet mask.

9.Classless Addressing:

Classless Addressing was designed and implemented to overcome address depletion and give more organizations to access the internet. Here in classless addressing there are no classes, but addresses are still granted in blocks.



- A block of 16 address combinations (consider last 4 bits) granted to a small organisation.
- In IPv4 addressing, a block of addresses can be defined as x. y. z. w / n defines the mask.
- The first addresses in the block can be found by setting the rightmost 32 – n bits to 0s.
- The last addresses in the block can be found by setting the rightmost 32 – n bits to 1s.
- So , the number of possible addresses = 2^{32-n}

Example:

➢ For the address 205.16.37.39/28, find the first and the last addresses in the block. Also find number of addresses in block.

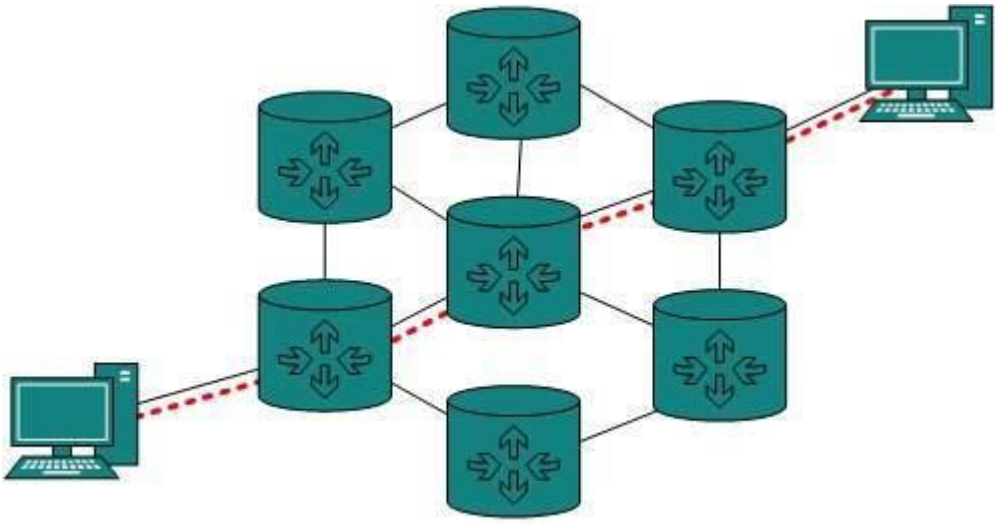
- Binary representation of the given **address and mask** are
  11001101 00010000 00100101 00100111 **(address)**
  11111111 11111111 11111111 11110000 **(mask)**
- Now to get the **first address** in the block we set 32−28=4 rightmost bits to set to 0, so we get
  11001101 00010000 00100101 00100000
  i.e. 205.16.37.32
- Now to get the **last address** in the block we set 32−28=4 rightmost bits to set to 1, so we get
  11001101 00010000 00100101 00101111
  i.e. 205.16.37.47
- So the number addresses = $2^{32-n} = 2^{32-28} = 16$

10.Internetworking:

networks under same administration are generally scattered geographically. There may exist requirement of connecting two different networks of same kind as well as of different kinds. Routing between two networks is called internetworking.

Networks can be considered different based on various parameters such as, Protocol, topology, Layer-2 network and addressing scheme.

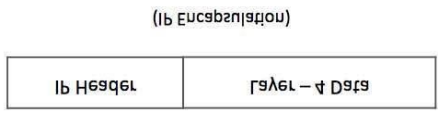
In internetworking, routers have knowledge of each other's address and addresses beyond them. They can be statically configured go on different network or they can learn by using internetworking routing protocol.



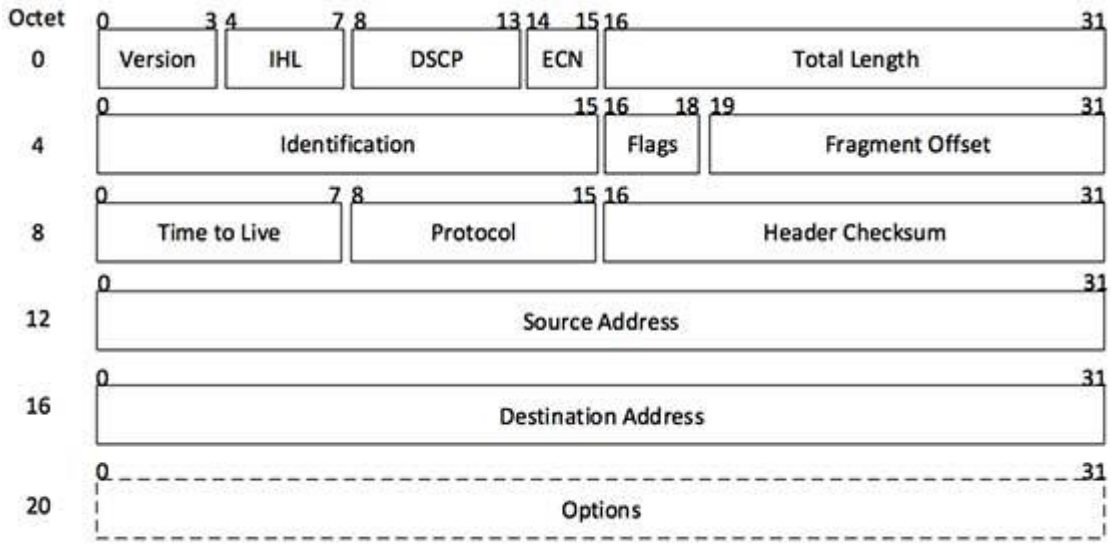
Routing protocols which are used within an organization or administration are called Interior Gateway Protocols or IGP. RIP, OSPF are examples of IGP. Routing between different organizations or administrations may have Exterior Gateway Protocol, and there is only one EGP i.e. Border Gateway Protocol.

11.IPV4 Datagram:

IP packet encapsulates data unit received from above layer and add to its own header information.



The encapsulated data is referred to as IP Payload. IP header contains all the necessary information to deliver the packet at the other end.



[Image: IP Header]

IP header includes many relevant information including Version Number, which, in this context, is 4. Other details are as follows −

- **Version** − Version no. of Internet Protocol used (e.g. IPv4).
- **IHL** − Internet Header Length; Length of entire IP header.
- **DSCP** − Differentiated Services Code Point; this is Type of Service.
- **ECN** − Explicit Congestion Notification; It carries information about the congestion seen in the route.
- **Total Length** − Length of entire IP Packet (including IP header and IP Payload).
- **Identification** − If IP packet is fragmented during the transmission, all the fragments contain same identification number. to identify original IP packet they belong to.
- **Flags** − As required by the network resources, if IP Packet is too large to handle, these ‘flags’ tells if they can be fragmented or not. In this 3-bit flag, the MSB is always set to ‘0’.
- **Fragment Offset** − This offset tells the exact position of the fragment in the original IP Packet.
- **Time to Live** − To avoid looping in the network, every packet is sent with some TTL value set, which tells the network how many routers (hops) this packet can cross. At each hop, its value is decremented by one and when the value reaches zero, the packet is discarded.

- **Protocol** − Tells the Network layer at the destination host, to which Protocol this packet belongs to, i.e. the next level Protocol. For example protocol number of ICMP is 1, TCP is 6 and UDP is 17.
- **Header Checksum** − This field is used to keep checksum value of entire header which is then used to check if the packet is received error-free.
- **Source Address** − 32-bit address of the Sender (or source) of the packet.
- **Destination Address** − 32-bit address of the Receiver (or destination) of the packet.
- **Options** − This is optional field, which is used if the value of IHL is greater than 5. These options may contain values for options such as Security, Record Route, Time Stamp, etc.

2.Fragmentation:

A data packet can have more or less packet length depending upon the application. If the data packet size is less than or equal to the size of packet the transit network can handle, it is processed neutrally. If the packet is larger, it is broken into smaller pieces and then forwarded. This is called packet fragmentation. Each fragment contains the same destination and source address and routed through transit path easily. At the receiving end it is assembled again.

If a packet with DF (don't fragment) bit set to 1 comes to a router which cannot handle the packet because of its length, the packet is dropped.

When a packet is received by a router has its MF (more fragments) bit set to 1, the router then knows that it is a fragmented packet and parts of the original packet is on the way.

If packet is fragmented too small, the overhead is increases. If the packet is fragmented too large, intermediate router may not be able to process it and it might get dropped.

13.Checksums:

The **IPv4 header checksum** is a checksum used in version 4 of the Internet Protocol (IPv4) to detect corruption in the header of IPv4 packets. It is carried in the IP packet header, and represents the 16-bit result of summation of the header words.

The checksum field is the 16-bit one's complement of the one's complement sum of all 16-bit words in the header. For purposes of computing the checksum, the value of the checksum field is zero.

The result of summing the entire IP header, including checksum, should be zero if there is no corruption. At each hop, the checksum is recalculated and the packet will be discarded upon checksum mismatch. The router must adjust the checksum if it changes part of the IP header

→Calculating the IPv4 header checksum:

Consider 4500 0073 0000 4000 4011 b861 c0a8 0001

To calculate the checksum, we can first calculate the sum of each 16 bit value within the header, skipping only the checksum field itself. Note that these values are in hexadecimal notation.
4500 + 0073 + 0000 + 4000 + 4011 + c0a8 + 0001 + c0a8 + 00c7 = 2479C
The first digit is the carry count and is added to the sum:
2 + 479C = 479E (if another carry is generated by this addition, another 1 must be added to the sum)
To obtain the checksum we take the one's complement of this result: B861

→Verifying the IPv4 header checksum:

When verifying a checksum, the same procedure is used as above, except that the original header checksum is not omitted.
4500 + 0073 + 0000 + 4000 + 4011 + b861 + c0a8 + 0001 + c0a8 + 00c7 = 2fffd
Add the carry bits: f f f d + 2 = f f f f
Taking the ones' complement (flipping every bit) yields 0000, which indicates that no error is detected.

14.Options:

- This is optional field, which is used if the value of IHL is greater than 5. These options may contain values for options such as Security, Record Route, Time Stamp, etc.

15.IPV6:

The successor of IPv4 is not designed to be backward compatible. Trying to keep the basic functionalities of IP addressing, IPv6 is redesigned entirely.

An IPv6 address is made of 128 bits divided into eight 16-bits blocks. Each block is then converted into 4-digit Hexadecimal numbers separated by colon symbols.

For example, given below is a 128 bit IPv6 address represented in binary format and divided into eight 16-bits blocks:

0010000000000001 0000000000000000 0011001000111000
1101111111100001 0000000001100011 0000000000000000
0000000000000000 1111111011111011

Each block is then converted into Hexadecimal and separated by ':' symbol:

2001 : 0000:3238:DFE1:0063:0000:0000:FEFB

Even after converting into Hexadecimal format, IPv6 address remains long. IPv6 provides some rules to shorten the address. The rules are as follows:

Rule.1: Discard leading Zero(es):

In Block 5, 0063, the leading two 0s can be omitted, such as (5th block):

2001: 0000 : 3238 :DFE1:63:0000:0000:FEFB

Rule.2: If two of more blocks contain consecutive zeroes, omit them all and replace with double colon sign ::, such as (6th and 7th block):

2001 : 0000:3238:DFE1:63::FEFB

Consecutive blocks of zeroes can be replaced only once by :: so if there are still blocks of zeroes in the address, they can be shrunk down to a single zero, such as (2nd block):

2001 :0:3238:DFE1:63::FEFB

16.Advantages: The IPv6 is the successor of IPv4 and has several advantages over IPv4 and is being widely used nowadays.

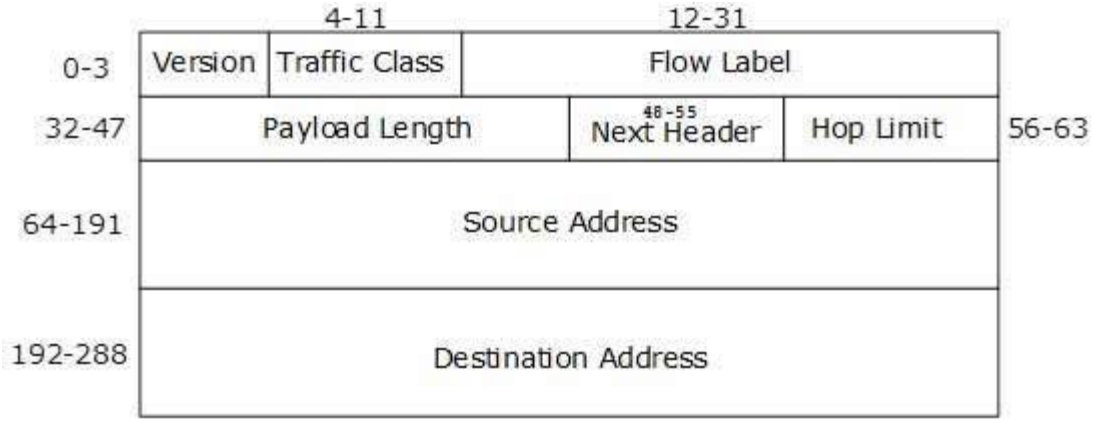
- **Larger Address Space:** As IPv6 is 128 bit, The scalability of this is huge. We can have up to 340 trillion addresses. Thus, we feel these IP addresses will never get consumed.
- **Simple Header for Router Efficiency:** The encapsulation is simpler than that of IPv4. As IPv6 does not have any checksum thus reduces the burden of processing for other endpoint devices.
- **No Broadcasts:** IPv6 does not have concept of broadcast which reduces utilization of devices in the same subnet.
- **Stateful and Stateless Auto configuration:** This means that using IPv6, host configuration is simplified and can be used in both case where a DHCPv6 server is present (Stateful) and in situations where DHCPv6 server is not available (Stateless).
- **Support for Built-in Mobile IP:** This is a major feature taking into consideration the use of mobile devices. This helps a device to roam into different wireless networks without the connection breaking.
- **Builtin IP Sec Security:** In IPv4, IP Sec was added later as an add-on but with IPv6 IP Security is built-in. Devices can negotiate the security

parameters dynamically to build a tunnel for secure communication and we get this without any user intervention.

- **Possess Transition features:** As there are already millions of devices using IPv4 addresses and it is not easy to switch to IPv6 immediately, thus there are some transition mechanisms that help devices between different IP versions i.e. IPv4 and IPv6 to communicate with each other.
- ***Dual Stack:*** Allows both protocols to be running simultaneously
- ***Tunneling:*** is used to tunnel the IPv4 under a IPv6 and vice versa
- ***NAT-PT:*** Translate the IPv4 address to IPv6 address. It stands for Network Address Translation – Protocol Translation or Proxy Translation.
- **No more need of NAT:** As sufficient IPv6 addresses are available, no need to do NAT.
- **Neighbour Discovery:** The usage of Neighbor Discovery Protocol in IPv6 is used in place of ARP used with IPv4.

17.Fixed Header

IPv6 headers have one Fixed Header and zero or more Optional (Extension) Headers. All the necessary information that is essential for a router is kept in the Fixed Header. The Extension Header contains optional information that helps routers to understand how to handle a packet/flow.



[*Image: IPv6 Fixed Header*]

IPv6 fixed header is 40 bytes long and contains the following information.

S.N.	Field & Description

1	**Version** (4-bits): It represents the version of Internet Protocol, i.e. 0110.
2	**Traffic Class** (8-bits): These 8 bits are divided into two parts. The most significant 6 bits are used for Type of Service to let the Router Known what services should be provided to this packet. The least significant 2 bits are used for Explicit Congestion Notification (ECN).
3	**Flow Label** (20-bits): This label is used to maintain the sequential flow of the packets belonging to a communication. The source labels the sequence to help the router identify that a particular packet belongs to a specific flow of information. This field helps avoid re-ordering of data packets. It is designed for streaming/real-time media.
4	**Payload Length** (16-bits): This field is used to tell the routers how much information a particular packet contains in its payload. Payload is composed of Extension Headers and Upper Layer data. With 16 bits, up to 65535 bytes can be indicated; but if the Extension Headers contain Hop-by-Hop Extension Header, then the payload may exceed 65535 bytes and this field is set to 0.
5	**Next Header** (8-bits): This field is used to indicate either the type of Extension Header, or if the Extension Header is not present then it indicates the Upper Layer PDU. The values for the type of Upper Layer PDU are same as IPv4's.
6	**Hop Limit** (8-bits): This field is used to stop packet to loop in the network infinitely. This is same as TTL in IPv4. The value of Hop Limit field is decremented by 1 as it passes a link (router/hop). When the field reaches 0 the packet is discarded.
7	**Source Address** (128-bits): This field indicates the address of originator of the packet.
8	**Destination Address** (128-bits): This field provides the address of intended recipient of the packet.

18.Extension Headers

In IPv6, the Fixed Header contains only that much information which is necessary, avoiding those information which is either not required or is rarely used. All such information is put between the Fixed Header and the Upper layer header in the form of Extension Headers. Each Extension Header is identified by a distinct value.

When Extension Headers are used, IPv6 Fixed Header's Next Header field points to the first Extension Header. If there is one more Extension Header, then the first Extension Header's 'Next-Header' field points to the second one, and so on. The last Extension Header's 'Next-Header' field points to the Upper Layer Header. Thus, all the headers points to the next one in a linked list manner.

If the Next Header field contains the value 59, it indicates that there are no headers after this header, not even Upper Layer Header.

The following Extension Headers must be supported as per RFC 2460:

Extension Header	Next Header Value	Description
Hop-by-Hop Options header	0	read by all devices in transit network
Routing header	43	contains methods to support making routing decision
Fragment header	44	contains parameters of datagram fragmentation
Destination Options header	60	read by destination devices
Authentication header	51	information regarding authenticity
Encapsulating Security Payload header	50	encryption information

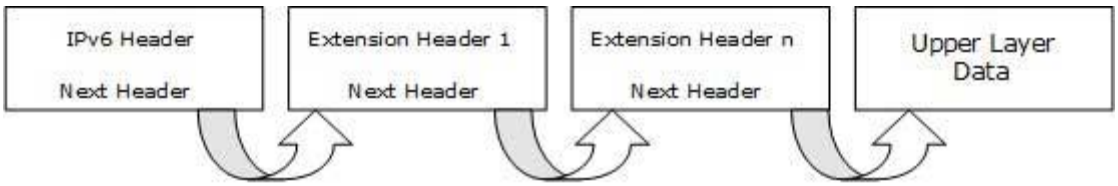
The sequence of Extension Headers should be:

IPv6 header
Hop-by-Hop Options header
Destination Options header[1]
Routing header
Fragment header
Authentication header
Encapsulating Security Payload header
Destination Options header[2]
Upper-layer header

These headers:

- 1. should be processed by First and subsequent destinations.
- 2. should be processed by Final Destination.

Extension Headers are arranged one after another in a linked list manner, as depicted in the following diagram:



[*Image: Extension Headers Connected Format*]