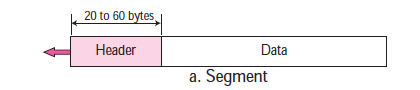
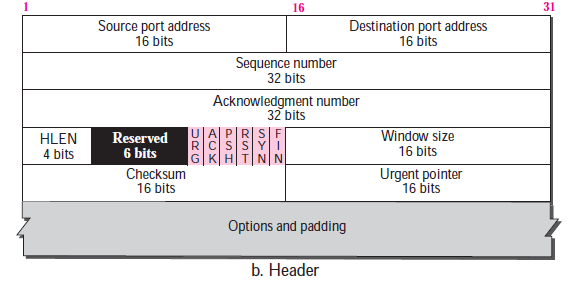
**TCP Segment**

A Packet in TCP is called a segment.



The Segment consists of 20 to 60 bytes header, followed by data from the application program. The Header is 20 bytes if there are no options and up to 60 bytes if it contains options.

**Header:**



**Source port address:**

This is a 16 bit field address that defines the port number of the application program in the host that is the sending the segment.

**Destination port address:**

This is a 16 bit field address that defines the port number of the application program in the host that is the receiving the segment.

**Sequence number:**

This 32 bit field defines the number assigned to the first byte of data contained in this segment. The sequence number tells the destination which byte in this sequence is the first byte in the segment.

**Acknowledgment number:**

This 32 bit field defines the byte number that the receiver of the segment is expecting to receive from the sender. If the receiver of the segment has successfully received byte number X from the sender, it returns X+1 as ack number.

**Header length:**

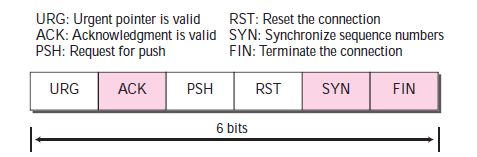
This 4 bit field indicates the number of 4 byte words in the TCP Header. The length of the header can be 20 to 60 bytes. Therefore this value is always between 5(5\*4=20) and 15 (15\*4=60).

**Reserved:**

This is a 6 bit field reserved for future use.

**Control:**

This filed defines the 6 different control bits or flags , one or more of these bits can be set at a time. These bits enable flow control, connection establishment and termination, connection abortion, and the mode of the data transfer in TCP.



Flags : There are six one-bit flags.

1. URG :  This bit indicates whether the urgent pointer field in this packet is being used.
2. ACK : This bit is set to indicate the ACK number field in this packet is valid.
3. PSH :  This bit indicates PUSHed data. The receiver is requested to deliver the data to the application upon arrival and not buffer it until a full buffer has been received.
4. RST :  This flag is used to reset a connection that has become confused due to a host crash or some other reason .It is also used to reject an invalid segment or refuse an attempt to open a connection. This causes an abrupt end to the connection, if it existed.
5. SYN :  This bit is used to establish connections. The connection request(1st packet in 3-way handshake) has SYN=1 and ACK=0. The connection reply (2nd packet in 3-way handshake) has SYN=1 and ACK=1.
6. FIN :  This bit is used to release a connection. It specifies that the sender has no more **fresh** data to transmit. However, it will retransmit any lost or delayed packet. Also, it will continue to receive data from other side. Since SYN and FIN packets have to be acknowledged, they must have a sequence number even if they do not contain any data.

**Window size:**

This field defines the window size of the sending TCP in bytes. Note that the length of this field is 16 bits, which means that the maximum size of the window is 65535 bytes.

**Checksum:**

The 16 bit field contains the checksum of data.

**Urgent Pointer:**

This 16 bit field which is valid only if the urgent flag is set, is used when the segment contains the urgent data. It defines a value that must be added to the sequence number to obtain the number of the last urgent byte in the data section of the segment.

**Options:**

There can be up to 40 bytes of optional information in the TCP Header.

These are to Enhance the TCP Protocol by introducing new features.

**Important:**

TCP Options are a multiple of **8 bits** in length. This means that if we use one TCP Option that is **4 bits** in length, there must be another **4 bits** of padding in order to comply with the TCP RFC(Request for Comments). So the TCP Option length must be in multiple of 8 bits i.e 8,16,24,..etc.

Different types of options are:

* Maximum Segment Size (MSS)
* Window Scaling
* Selective Acknowledgements (SACK)
* Timestamps
* Nop

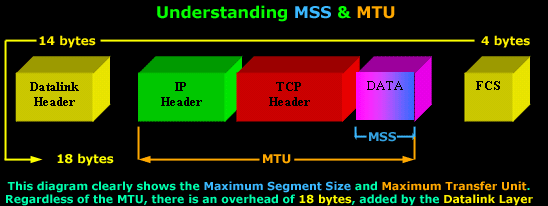
**Maximum Segment Size(MSS):**

It is used to define the maximum segment that will be used during a connection between hosts. You can see this option during SYN and SYN/ACK phase of the 3 way handshake.

Generally MTU(Maximum Transfer Unit) Size is either 576 bytes or 1500 bytes.

MSS Size=MTU Size-Header Length(IP Header Length+ TCP Header Length). (Generally 536 or 1460 bytes).

Understanding MTU & MSS:



* The TCP Header and Data is called a Segment (Layer 4), while the IP Header and the Segment is called an IP Datagram (Layer 3).
* Regardless of the MTU Size you will have additional **18 bytes** overhead placed by the **Datalink layer**. This overhead consists of Source MAC, Destination MAC ,..etc.
* This is the Reason why we can only have maximum MTU of **1500** bytes. Since the maximum size of an **Ethernet ||** frame is **1518** bytes.(It may be different on latest Ethernets).
* If the MSS option is omitted by one or both ends of the connection, then the value of **536 bytes** will be used. The MSS value of **536** bytes is defined by **RFC 1122** and is calculated by taking the **default value** of an IP Datagram, **576** bytes, minus the standard length of the IP and TCP Header (40 bytes), which gives us 536 bytes.

In general, it is very important to use the best possible **MSS** value for your network because your network performance could be extremely poor if this value is too large or too small.

Example:

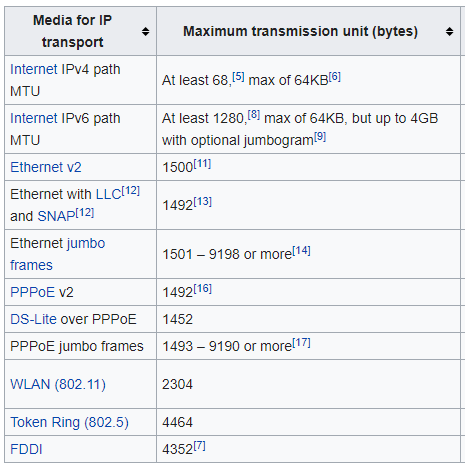
1)

If you wanted to transfer 1 byte of data through the network, you would need to create a datagram with 40 bytes of overhead, 20 for the IP Header and 20 for the TCP Header. This means that your using 1/41 of your available network bandwidth for data. The rest is nothing but overhead!

2)

On the other hand, if the MSS is **very large**, your IP Datagrams should also be very large, meaning that they will most probably fail to fit into one packet if MTU is too small. Therefore they will require to be fragmented, increasing the overhead by a factor of 2.

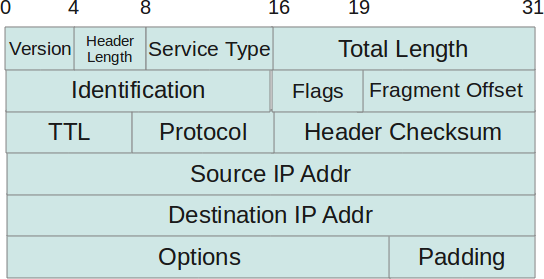
 In many cases **network MTU** is dependent on underlying network capabilities . for Example Ethernet’s MTU is 1500 bytes & WIFI’s MTU is 2300 bytes.



If your Media MTU size is less than the MTU size (Application) then **IP fragmentation** is needed.

**IP Fragmentation:**

* The **Identification field** and **Fragment offset field** along with **Don't Fragment** and **More Fragment** flags in the IP protocol header are used for fragmentation and reassembly of IP datagrams.



* **Identification field** is same for all fragments of IP datagram.
* If **Don’t Fragment** field set then fragmentation will not be done.
* If **More Fragment** flag set then receiver will got to know that there are fragments it has to wait for to receive complete IP datagram.
* **Fragment offset field** represents the original byte number of the TCP segment.
* Under [**IPv4**](https://en.wikipedia.org/wiki/IPv4), a [router](https://en.wikipedia.org/wiki/Router_(computing)) that receives a [protocol data unit](https://en.wikipedia.org/wiki/Protocol_data_unit) (PDU) larger than the next hop's MTU **has two options**: drop the PDU and send an [Internet Control Message Protocol](https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol) (ICMP) message which indicates the condition Packet too Big, or fragment the IP packet and send it over the link with a smaller MTU.
* IP fragmentation can cause excessive retransmissions when fragments encounter [packet loss](https://en.wikipedia.org/wiki/Packet_loss) and reliable protocols such as TCP must retransmit all of the fragments in order to recover from the loss of a single fragment. Thus, senders typically use **two approaches** to decide the size of IP datagrams to send over the network.

The **first** is for the sending host to send an IP datagram of size equal to the MTU of the first hop of the source destination pair.

The **second** is to run the [path MTU discovery](https://en.wikipedia.org/wiki/Path_MTU_discovery) algorithm, described in [RFC 1191](https://tools.ietf.org/html/rfc1191), to determine the **path MTU** between two IP hosts, so that IP fragmentation can be avoided.

NOTE:

This **slows down the connection speed** as seen by the computer user. In some cases the slowdown is dramatic. The likelihood of such fragmentation can be minimized by keeping the MSS as small as reasonably possible. For most computer users, the MSS is set automatically by the operating system.

Destination host will do the hard work of reassembling the fragments to produce the original packet and then that original packet will be handed up to the **Transport Layer.**

<https://www.youtube.com/watch?v=DgcbVsGIYfE>

While sending TCP segments to the other end, an inter-layer communication is done as follows:

* TCP should determine the Maximum Segment Data Size (MSDS) from either the **default** or the received value of the MSS option.
* TCP should determine if source fragmentation is possible (by asking the IP) and desirable.
* IP checks the length of data passed to it by TCP. If the length is less than or equal MDDS, IP attaches the IP header and hands it to the ND. Otherwise the IP must do source fragmentation.

Configuration:

Options have up to three fields: Option-Kind (1 byte), Option-Length (1 byte), Option-Data (variable).

**For Maximum Segment Size:**

**Option-Kind** byte=0x02 (to represent which type of Option MSS or Nop or SACK).

**Option-Length**=0x04 ( to represent the length of the Option for MSS option length is 4 bytes(kind+length+data).

**Option-Data**=0x05B4( It is of 2 bytes which represents the Maximum segment size).

MSS Option should be used at connection establishment only.