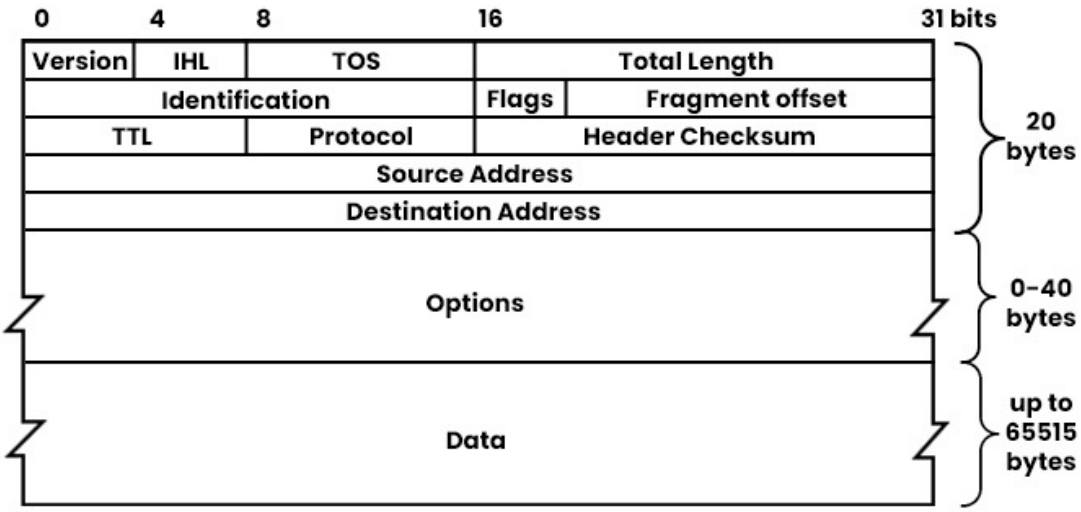
DAY 10 - IPv4 Header

Purinat33

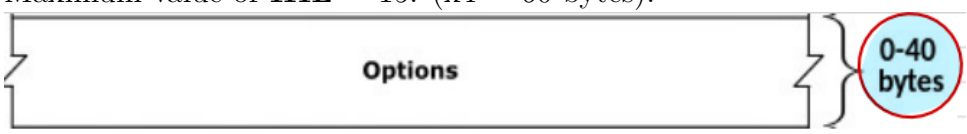
IPv4 Header



Read the image from: *left to right* then *top to bottom*

Components in the IPv4 Header:

1. **Version** (4 bits): Identifies the version of IP being used.
   (a) `0100` = IPv4
   (b) `0110` = IPv6
2. **IHL (Internet Header Length)** (4 bits):
   (a) Indicate the total length of the header using **4-Bytes Increment**
      i. eg. **IHL** = 5 multiply by `4 Bytes Increment` = 20 Bytes.
   (b) Minimum value of **IHL** = 5. (x4 = 20 bytes).
   (c) Maximum value of **IHL** = 15. (x4 = 60 bytes).



   (d) The size depends on the **Options** field:

    i. Size 0 **Options** + Minimum **IHL** size (20) = 20+0 Bytes.
    ii. Size 40 **Options** + Minimum **IHL** = 20 + 40 = 60 Bytes.
  (e) IPv4 Header size = 20 - 60 Bytes.
3. **DSCP (Differentiated Services Code Point)** (6 bits):
  (a) **DSCP** is used for **QoS (Quality of Service)**
  (b) Used to prioritize delay-sensitive data (eg. *streaming voices, videos etc.*)
4. **ECN (Explicit Congestion Notification)** (2 bits):
  (a) Provides End-to-End (between two endpoints) notification of network congestion **without dropping packets**. (Normally congestion = Packet Dropped)
  (b) Optional feature that requires both endpoints, as well as the underlying network infrastructure, to support it.
  (c) The chart combines **DSC + ECN** into **TOS (Type of Service)**.
5. **Total Length** (16 bits):
  (a) Indicates the total length of the **packet** (not just the header).
  (b) Measured in *Bytes* (Not **4-Bytes Increment** like in **IHL**)
  (c) Minimum Value = **20** (IPv4 Header with no encapsulated data).
  (d) Maximum Value = **65,535** (All `1` in 16-bits).
6. **Identification** (16 bits):
  (a) If a packet is fragmented due to being too large, this field is used to identify **which packet the fragment belongs**.
  (b) All fragments of the *same packet* will have their own IPv4 header with the *same value* in this field.
  (c) Packets are fragmented if larger than the **MTU (Maximum Transmission Unit)**.
7. **Flags** (3 bits):
  (a) Used to control/identify fragments.
  (b) **Bit 0**th: `Reserved` , always set to `0`
  (c) **Bit 1**st: `Don't Fragment` (**DF**), used to indicate a packet that should not be fragmented.
  (d) **Bit 2**nd: `More Fragment` (MF)
    i. Set to `1` if there are more fragment in the packet.
    ii. Set to `0` for the last fragment.
    iii. Unfragmented packets will always have MF bit = 0)
8. **Fragment Offset** (13 bits):
  (a) Used to indicate the position of the fragment within the original, unfragmented IP Packet.
  (b) Allows fragmented packets to be reassembled by the receiving host even if the fragments arrive out of order.
9. **TTL (Time To Live)** (8 bits):

(a) A router will drop a packet with TTL = 0
(b) Used to prevent infinite loops.
(c) Originally designed to indicate the packet's maximum lifetime in seconds.
(d) In practice, indicates a **Hop Count**:
    i. Each time the packet arrives at a router, decrement the TTL by 1.
(e) Recommend value = 64

10. **Protocol** (8 bits):
    (a) Indicates the protocol of the encapsulated **Layer 4 PDU**
        i. Value of `6` : **TCP**
        ii. Value of `17` : **UDP**
        iii. Value of `1` : **ICMP** (Ping)
        iv. Value of `89` : **OSPF** (Dynamic Routing Protocol)
11. **Header Checksum** (16 bits):
    (a) A calculated checksum used to check for errors in the IPv4 Header.
    (b) When a router receives a packet, it calculates the checksum of the header and compares it to the value in this field of the header.
    (c) If they do not match, the router drops the packet.
    (d) Used to check for errors **ONLY** in the IPv4 Header.
    (e) IP relies on the encapsulated protocol to detect errors in the encapsulated data/payload.
12. **Source IP Address** (32 bits)
    (a) IPv4 address of the sender of the packet.
13. **Destination IP Address** (32 bits)
    (a) IPv4 address of the intended receiver of the packet.
14. **Options** (0-320 bits):
    (a) Rarely used.
    (b) If the **IHL** field is greater than 5, it means that **Options** are present.
    (c) **Not Required for CCNA**

Wireshark Example: