Internet Protocols

IP Header, Fragmentation / Forwarding / Encapsulation / IPv6
IP Operation

Go to Router X

MAC address for Router X

Encapsulated with LAN protocol

Encapsulated with X.25 protocol

TCP-H = TCP header
IP-H = IP header
LLCi-H = LLC header
MACi-H = MAC header
MAC2-H = MAC header
MAC2-T = MAC trailer
MAC-T = MAC trailer
XP-H = X.25 packet header
XL-H = X.25 link header
XL-T = X.25 link trailer
TCP/IP Stack Protocol

- **Bridge**
  - IS used to connect two LANs using similar LAN protocols
  - Address filter passing on packets to the required network only
    - OSI layer 2 (Data Link)

- **Router**
  - Connects two (possibly dissimilar) networks
  - Uses internet protocol present in each router and end system
    - OSI Layer 3 (Network)
## IP Header Format

- **Source address**
- **Destination address**
- **Protocol**
  - Recipient e.g. TCP
- **Type of Service**
  - Specify treatment of data unit during transmission through networks

### Identification
- Source, destination address and user protocol
- Uniquely identifies PDU
- Needed for re-assembly and error reporting
- Send only

<table>
<thead>
<tr>
<th></th>
<th>0 - 3</th>
<th>4 - 7</th>
<th>8 - 15</th>
<th>16 - 18</th>
<th>19 - 31</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Version</td>
<td>Header length</td>
<td>Type of Service</td>
<td>Total Length</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Identification</td>
<td>Flags</td>
<td>Fragment Offset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>Time to Live</td>
<td>Protocol</td>
<td>Header Checksum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>96</td>
<td></td>
<td></td>
<td>Source Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>128</td>
<td></td>
<td>Destination Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>160</td>
<td></td>
<td>Options + padding</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>192</td>
<td>1-64K Octets</td>
<td>Data</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

20 Bytes

HL=5 rows → 20 octet (variable) / 8*20/32
IP Header Format

- **VERS**
  - Each datagram begins with a 4-bit protocol version number (the figure shows a version 4 header)

- **H.LEN (Header Length)**
  - Number of 32-bit rows in the Header → Header Length
  - 4-bit header specifies the number of 32-bit quantities in the header (in the figure we have 5 32-bit rows)
  - If no options are present, the value is 5

- **SERVICE TYPE**
  - 8-bit field that carries a class of service for the datagram
  - Seldom used in practice
  - Chapter 28 explains the DiffServ interpretation of the service type field

- **TOTAL LENGTH**
  - 16-bit integer that specifies the total number of bytes in the datagram
  - Includes both the header and the data
IP Header Format

- **IDENTIFICATION**
  - 16-bit number (usually sequential) assigned to the datagram
  - used to gather all fragments for reassembly of the datagram

- **FLAGS**
  - 3-bit field with individual bits specifying whether the datagram is a fragment
  - If so, then whether the fragment corresponds to the rightmost piece of the original datagram

- **FRAGMENT OFFSET**
  - 13-bit field that specifies where in the original datagram the data in this fragment belongs
  - the value of the field is multiplied by 8 to obtain an offset
IP Header Format

- **TIME TO LIVE**
  - 8-bit integer initialized by the original sender
  - Represents the max. number of hops the packets can visit
  - It is decremented by each router that processes the datagram
  - If the value reaches zero (0)
    - The datagram is discarded and an error message is sent back to the source

- **TYPE**
  - 8-bit field that specifies the type of the payload

- **HEADER CHECKSUM**
  - 16-bit ones-complement checksum of header fields

- **SOURCE IP ADDRESS**
  - 32-bit Internet address of the original sender
  - The addresses of intermediate routers do not appear in the header
Example:
Encapsulated IP Packet in Ethernet Frame

Important!

90 Bytes
Example:
Encapsulated IP Packet in Ethernet Frame

Ethernet Frame Carrying IP Packet

An Ethernet frame containing IP information has 08 00 in its type field.

IP starting with 45 Hex indicates IPv4 with standard HED length of 20 bytes = 5 rows x 32/8.

IP starting with 4F Hex indicates IPv4 with HED length of 60 bytes = 15 rows x 32/8.

Remember: $2^4=16$; 45 = 0100 0101 = One Byte.

Protocol Analyzer Display:

<table>
<thead>
<tr>
<th>Protocol Analyzer Display</th>
<th>Example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 00 00 C0 A0 51 24 00 C0 93 21 88 A7 08 00 45 08</td>
<td>99 is one byte 1001 1001</td>
</tr>
<tr>
<td>0010 00 5A DC 28 00 00 FF 01 88 08 C0 99 B8 01 C0 99</td>
<td></td>
</tr>
<tr>
<td>0020 B8 03 2a B4 DD .....</td>
<td></td>
</tr>
</tbody>
</table>
Example of a Single IP Packet

20 byte was used for the header

It is a single packet
No fragmentation is used

Forwarding

- The Internet uses next-hop forwarding.
- To make the selection of a next hop efficient, an IP router uses a forwarding table.
- Mask field is used to direct the incoming packet.
- Number of entries in the table can be very large.
- A forwarding table is initialized when the router boots.
  - Forwarding table must be updated if the topology changes or hardware fails.

Routing Table: e.g., If a packet with destination 30.0.0.0 arrives at R2 → The next hop will be 40.0.0.7.
Suppose a router's forwarding table contains entries for the following two network prefixes:

- 128.10.0.0/16
- 128.10.2.0/24

What happens if a datagram arrives destined to 128.10.2.3?

Matching procedure succeeds for both of the entries:

- A Boolean and of a 16-bit mask will produce 128.10.0.0
- A Boolean and with a 24-bit mask will produce 128.10.2.0

Which entry should be used?

To handle ambiguity that arises from overlapping address masks, Internet forwarding uses a longest prefix match:

- Instead of examining the entries in arbitrary order,
- Forwarding software arranges to examine entries with the longest prefix first.

In the example above, Internet forwarding will choose the entry that corresponds to 128.10.2.0/24.
Transmission Across the Internet

Header can change – Going through WiFi or Ethernet
Transmission Across the Internet

- Each hardware technology specifies the maximum amount of data that a frame can carry
  - The limit is known as a Maximum Transmission Unit (MTU)
- There is no exception to the MTU limit
  - Network hardware is not designed to accept or transfer frames that carry more data than the MTU allows
  - A datagram must be smaller or equal to the network MTU
- In an internet that contains heterogeneous networks, MTU restrictions create a problem
- A router can connect networks with different MTU values
  - A datagram that a router receives over one network can be too large to send over another network

Two networks with different MTU
(a heterogeneous network)
IP Fragmentation (1)

- IP re-assembles at destination only
- Uses fields in header
  - Data Unit Identifier (ID)
    - Identifies end system originated datagram
      - Source and destination address
      - Protocol layer generating data (e.g. TCP)
      - Identification supplied by that layer
IP Fragmentation (2)

- **Offset**
  - Position of fragment of user data in original datagram
  - In multiples of 64 bits (8 octets)

- **More flag (more is coming!)**
  - Indicates that this is not the last fragment
IP Fragmentation (3)

Data Size = Data + Header

MTU = Max Data Size = Data + Header

Total of data: $480 + 480 + 20 = 980$

$480 \times 8 / 64 = 60$

$480 \times 8 / 64 = 60 ; 60 + 60 = 120$

Offset
- Position of fragment of user data in original datagram
- In multiples of 64 bits (8 octets)

More flag
- Indicates that this is not the last fragment

Fragmentation Example

Fragmentation-related fields will be modified in each IP header.

TCP header is only required for the first fragment.

More is cleared = last fragment
Fragmentation Example

IP + TCP + Data = Data Size
Max. Datagram Size = IP + TCP + Data

Datagram Size: 424
MTU: 228
Datagram ID: 2

Compare with the previous figure!
### Fragmentation

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.10.0.3</td>
<td>192.168.0.128</td>
<td>ICMP</td>
</tr>
<tr>
<td>10.10.0.3</td>
<td>192.168.0.128</td>
<td>IPv4</td>
</tr>
<tr>
<td>10.10.0.3</td>
<td>192.168.0.128</td>
<td>IPv4</td>
</tr>
<tr>
<td>192.168.0.128</td>
<td>10.10.0.3</td>
<td>IPv4</td>
</tr>
<tr>
<td>192.168.0.128</td>
<td>10.10.0.3</td>
<td>IPv4</td>
</tr>
<tr>
<td>192.168.0.128</td>
<td>10.10.0.3</td>
<td>ICMP</td>
</tr>
</tbody>
</table>

Creating a large ICMP frame:

"ping –s 3000 -M do 192.168.1.1"
Fragmentation

Frame 2: 1514 bytes on wire (12112 bits), 1514
Ethernet II, Src: 00:25:b3:bf:91:ee (00:25:b3:b)
Internet Protocol Version 4, Src: 10.10.0.3 (10
   Version: 4
   Header length: 20 bytes
   Differentiated Services Field: 0x00 (DSCP 0x0
   Total Length: 1500
   Identification: 0x7474 (29812)
   Flags: 0x01 (More Fragments)
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..1. .... = More fragments: Set
   Fragment offset: 1480
   Time to live: 128
   Protocol: ICMP (1)
   Header checksum: 0x0000 [incorrect, should be
                     Source: 10.10.0.3 (10.10.0.3)
                     Destination: 192.168.0.128 (192.168.0.128)
                     [Source GeoIP: Unknown]
                     [Destination GeoIP: Unknown]
   Data (1480 bytes)
      Data: 6162636465666768696a6b6c6d6e6f707172737
      [Length: 1480]

Frame 3: 582 bytes on wire (4656 bits), 582 by
Ethernet II, Src: 00:25:b3:bf:91:ee (00:25:b3:b
Internet Protocol Version 4, Src: 10.10.0.3 (10
   Version: 4
   Header length: 20 bytes
   Differentiated Services Field: 0x00 (DSCP 0x0
   Total Length: 568
   Identification: 0x7474 (29812)
   Flags: 0x00
      0... .... = Reserved bit: Not set
      .0.. .... = Don't fragment: Not set
      ..0. .... = More fragments: Not set
   Fragment offset: 2960
   Time to live: 128
   Protocol: ICMP (1)
   Header checksum: 0x0000 [incorrect, should be
                     Source: 10.10.0.3 (10.10.0.3)
                     Destination: 192.168.0.128 (192.168.0.128)
                     [Source GeoIP: Unknown]
                     [Destination GeoIP: Unknown]
   Data (548 bytes)
      Data: 696a6b6c6d6e6f7071727374757677761626364
      [Length: 548]
Dealing with Failure

- Re-assembly may fail if some fragments get lost
  - Requires buffer
  - Failures must be detected
- Ways to deal with failures (two approaches)
  - Use re-assembly time-out
    - Assigned to first fragment to arrive
    - If timeout expires before all fragments arrive, discard partial data
  - Use packet lifetime (TTL in IP) -
    - If time-to-live runs out, kill partial data
    - Note: TTL can be in hops or sec.
Sub-fragments

- A receiver cannot know if an incoming fragment is the result of one router fragmenting a datagram or multiple routers fragmenting fragments
- Fragmenting fragments results in subfragments
- Having subfragments requires
  - The receiver to perform reassembly for subfragments first
  - More processing is required (more CPU time)
Why Change IPv4 Addressing?

- Address space exhaustion
  - Two level addressing (network and host) wastes space
  - Network addresses used even if not connected to Internet
  - Growth of networks and the Internet
  - Extended use of TCP/IP
  - Single address per host

- Requirements for new types of service

http://www.changeipaddress.org/how-to-change-ip-address.php
IP v6 Header vs. IPV4

Features:
- Extended address space
- Improved option mechanism
- Dynamic address assignment
- Multicasting and anycasting
- Flow routing

Note:
IPv5 used for Stream Protocol - IP-layer protocol that provides end-to-end guaranteed service across a network.
Converting IPv4 to IPv6

http://www.subnetonline.com/pages/subnet-calculators/ipv4-to-ipv6-converter.php
TCP/IP Stack Protocol

- **Bridge**
  - IS used to connect two LANs using similar LAN protocols
  - Address filter passing on packets to the required network only
  - OSI layer 2 (Data Link)

- **Router**
  - Connects two (possibly dissimilar) networks
  - Uses internet protocol present in each router and end system
  - OSI Layer 3 (Network)
TCP Header

TCP Flags
- C: 0x80 Reduced (CWR)
- E: 0x40 ECN Echo (ECE)
- U: 0x20 Urgent
- A: 0x10 Ack
- P: 0x08 Push
- R: 0x04 Reset
- S: 0x02 Syn
- F: 0x01 Fin

Congestion Notification
- ECN (Explicit Congestion Notification). See RFC 3168 for full details, valid states below.
- No Congestion
- Ack

TCP Options
- 0 End of Options List
- 1 No Operation (NOP, Pad)
- 2 Maximum segment size
- 3 Window Scale
- 4 Selective ACK ok
- 8 Timestamp

Checksum
- Checksum of entire TCP segment and pseudo header (parts of IP header)

Offset
- Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

RFC 793
- Please refer to RFC 793 for the complete Transmission Control Protocol (TCP) Specification.
TCP

- Used for reliability (RFC 793)
  - Data sequencing
  - Error recovery
  - Built-in error checking
- Layer 4 OSI model
- Contains source/Destination PORT
- Connection Oriented
  - TCP Handshake (3-way setup)
  - TCP Teardown (4-way teardown)
  - TCP Reset

---

Transmission Control Protocol, Src Port: 80 (80), Dst Port: 28

<table>
<thead>
<tr>
<th>Flags: 0x012 (SYN, ACK)</th>
</tr>
</thead>
<tbody>
<tr>
<td>000. ...... ..... = Reserved: Not set</td>
</tr>
<tr>
<td>...0 ...... ..... = Nonce: Not set</td>
</tr>
<tr>
<td>.... 0... ..... = Congestion Window Reduced (CWR): Not set</td>
</tr>
<tr>
<td>.... .0... ..... = ECN-Echo: Not set</td>
</tr>
<tr>
<td>.... ...0... ..... = Urgent: Not set</td>
</tr>
<tr>
<td>.... ....1.... = Acknowledgment: Set</td>
</tr>
<tr>
<td>.... .... 0... = Push: Not set</td>
</tr>
<tr>
<td>.... .... ....c. = Reset: Not set</td>
</tr>
</tbody>
</table>

Window size value: 5840
[Calculated window size: 5840]
Checksum: 0x9ac0 [validation disabled]
Options: (12 bytes), Maximum segment size, No-Operation (NCF)
[SEQ/ACK analysis]
TCP Handshack

3-Way
TCP Termination
4-Way

TCP Reset is sent to ACK request and Refuse connection!
Example is a server whose port 80 is not listening.
References

- Very Good Applets: 