History

- Developed in late 1970's – early 1980's
  - DARPA Internetworking Program
- Goal: create a "catenet" = network of networks
  - LAN technologies were coming on the scene
  - Several wide-area packet networks already existed
    - ARPANet, Tymnet, Telenet, DataPAC
  - Needed: Common global address space
- "Flag Day" cutover to TCP/IP: January 1, 1981
  - Note: Originally (late 70's) TCP and IP were one protocol
What IPv4 Provides

• Unified global, hierarchical address space
  – 32-bit addresses
    • depicted as "dotted quads": 128.113.23.44

• Datagram service: each packet forwarded independently
  – Gateways (routers) can be "stateless" (not really)
  – Requires little from underlying link layers
    • "Best-effort" service
    • Runs over everything

• Fragmentation and Reassembly
  – Datagrams can be up to 64K bytes can be sent
  – IP layer will

• Bounded Packet lifetime
  – Packets will be dropped instead of delivered after long time

• Different Types of Service (never implemented)
IP Version 4 Header

Version        Header Length   Type of Service  Total Length
Identification  Flags           Frag Offset
TTL            Protocol        Header Checksum
Source Address
Destination Address
Options

16 bits

32 bits
IP Version 4 Header

Valid Version Values: 4, 6
IP Version 4 Header

- **Version**: 4 bits indicating the version of IPv4.
- **Header Length**: 4 bits indicating the length of the header in 32-bit words.
- **Type of Service**: 8 bits indicating the service type.
- **Total Length**: 16 bits indicating the total length of the IP packet in bytes.
- **Identification**: 16 bits identifying the data packet.
- **Flags**: 3 bits indicating whether the packet is fragmentable and its fragment number.
- **Frag Offset**: 13 bits indicating the offset of the fragment in bytes.
- **Time to Live (TTL)**: 8 bits indicating the number of hops a packet can travel before being discarded.
- **Protocol**: 8 bits indicating the upper-layer protocol.
- **Header Checksum**: 16 bits indicating the checksum of the header.
- **Source Address**: 32 bits indicating the source IP address.
- **Destination Address**: 32 bits indicating the destination IP address.
- **Options**: Additional options for the packet.
IP Version 4 Header

- **Version**: Indicates the IP version (IPv4).
- **Header Length**: Specifies the length of the header in bytes.
- **Type of Service**: Specifies how the packet should be treated by the network and nodes.
- **Total Length**: 16-bit total length: up to 65,535 bytes can be sent in one datagram.
- **Identification**: Identifies the packet among multiple fragments.
- **Flags**: Determines if the packet is fragmented and if it is the last fragment.
- **Fragment Offset**: Indicates the offset of the fragment within the datagram.
- **Protocol**: Specifies the upper-layer protocol for the packet.
- **TTL (Time To Live)**: Indicates how long the packet can be forwarded by the network.
- **Header Checksum**: Ensures the header data is correct.
- **Source Address**: The source IP address of the packet.
- **Destination Address**: The destination IP address of the packet.
- **Options**: Additional information that can be carried in the packet.
IP Version 4 Header

Time-to-Live: 8 bits
Original semantics: # seconds datagram can remain in the network (≤ 255)
Current semantics: hop limit
### IP Version 4 Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version Length</td>
<td>The version number of IPv4 protocol and the header length in 32-bit units.</td>
</tr>
<tr>
<td>Type of Service</td>
<td>The protocol identifier of the higher-level protocol, such as TCP or UDP.</td>
</tr>
<tr>
<td>Total Length</td>
<td>The total length of the IP packet in bytes.</td>
</tr>
<tr>
<td>Identification</td>
<td>The identification number for fragment management.</td>
</tr>
<tr>
<td>Offset</td>
<td>The fragment offset in 32-bit units.</td>
</tr>
<tr>
<td>TTL</td>
<td>The time to live for this packet.</td>
</tr>
<tr>
<td>Protocol</td>
<td>The protocol identifier of the higher-level protocol.</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>The checksum of the header.</td>
</tr>
<tr>
<td>Source Address</td>
<td>The IP address of the sender.</td>
</tr>
<tr>
<td>Destination Address</td>
<td>The IP address of the receiver.</td>
</tr>
<tr>
<td>Options</td>
<td>Additional options relating to the packet.</td>
</tr>
</tbody>
</table>

*Address* of higher-level protocol, e.g.:

- 6 = TCP
- 17 = UDP
- 4 = IPv4

See [IANA](https://www.iana.org) for more details.
IP Version 4 Header

- **Version**: 4 bits indicating the IP version.
- **Header Length**: 4 bits indicating the length of the header in 32-bit words.
- **Type of Service**: 8 bits indicating the type of service required by the application.
- **Total Length**: 16 bits indicating the total length of the IP packet in bytes.
- **Identification**: 16 bits identifying the packet.
- **Flags**: 3 bits indicating:
  - 0: must be 0
  - 1: Don't Fragment (DF)
  - 2: More Fragments (MF)
  - MF=0 means "this fragment contains the last byte of this datagram"
- **Fragment Offset**: 13 bits indicating the offset of the first byte of the payload of this fragment from the beginning of the original payload (units: 8 bytes).
- **Source Address**: 32 bits indicating the source IP address.
- **Destination Address**: 32 bits indicating the destination IP address.
- **Protocol**: 8 bits indicating the protocol used in the packet.
- **Header Checksum**: 16 bits indicating the checksum of the header.
- **Options**: 32 bits indicating any options present.
- **Fragmentation Fields**: Indicate whether the packet needs to be fragmented and which fragments are left.

Unique per (source, dest, protocol) per maximum segment lifetime (MSL).
Fragmentation Example

MTU = 4010 bytes

Header: 20 bytes

DG Len = 13007

ID: 35689

Offset: 0

Payload: 12987 bytes
Fragmentation Example

MTU = 4010 bytes

Payload: 12987 bytes

DG Len = 4004
ID: 35689 001 Offset: 498
... bytes 3984-7967

DG Len = 4004
ID: 35689 001 Offset: 996
... bytes 7968-11951

DG Len = 4004
ID: 35689 001 Offset: 0
... bytes 0-3983

DG Len = 1055
ID: 35689 000 Offset: 1494
... bytes 11952-12986
Internet Addresses

• 32-bit addresses assigned to interfaces (not hosts)
  
  Axiom: for each IP address there is an underlying link (or physical) address
  
  – IP provides network-to-network service
  – The underlying link protocols provide host-to-host service
    (Ethernet, PPP, WiFi, ... -- more on this later)

• Addresses are hierarchical and linked to network topology
  
  – Addresses assigned to interfaces "close" to each other in the topology generally share a common prefix
  – In fact, individual addresses are not assigned; prefixes are!
Hierarchical Addressing

128.163.0.0/16
128.160.0.0/16
128.155.20.0/22
128.161.0.0/16
128.162.0.0/16

10000000101000110000000000000000

128.160.0.0/14 = 10000000101000000000000000000000

128.163.3.4
128.162.1.1

128.163.0.0/16
128.160.0.0/16
128.162.0.0/16
128.161.0.0/16

128.155.20.0/22
Network Numbers

- In the old days, addresses were self-describing
  - Boundary between network # and host # was indicated by first two bits of address
- That turned out to be too limiting
  - Not enough flexibility w.r.t. network size (cf. "Goldilocks")
  - Only two levels of hierarchy – inadequate
- Now the separation is indicated explicitly
  - It takes a pair of numbers to specify a network number: \(204.198.76.0/24\)

Network number

Prefix Length
Longest-Prefix Matching

- Recall that routers do longest-prefix matching
  - To find most-specific forwarding table entry
- Each table entry has two parts:
  - Prefix = bit string that defines the "network number"
  - Mask = 1 bits indicate part of the prefix, 0's elsewhere

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>204.198.76.0</td>
<td>255.255.255.0</td>
</tr>
<tr>
<td>128.163.0.0</td>
<td>255.255.192.0</td>
</tr>
<tr>
<td>128.163.0.0</td>
<td>255.255.0.0</td>
</tr>
</tbody>
</table>
Longest-Prefix Matching

Boolean Match(IPAddr dest, IPAddr prefix, IPAddr mask)
{
  return ((dest & mask) == prefix);
}
Longest-Prefix Matching

Boolean Match(IPAddr dest, IPAddr prefix, IPAddr mask)
{
  return ((dest & mask) == prefix);
}

dest addr = 128.163.13.1

Match!
Where does my address come from?

- **How do I get an IP address?**
  - From your Internet Service Provider (ISP)
  - If you have a single machine, provider assigns you a single address
  - If you have a network, provider assigns a prefix (set of addresses)
    - E.g., network with 6 hosts: get a /29
    - Host #'s 0...0 and 1...1 are reserved ("any", broadcast)

- **How does my provider get an address?**
  - From a Registrar (ARIN, APNIC, RIPE, ...)
  - Provider must "make the case" to get address space
  - IPv4 address space is more than 50% used
Private Address Space

• Some prefixes are set aside for networks not connected to the "capital I" Internet
  - 192.168.0.0/16
  - 172.16.0.0/12
  - 10.0.0.0/8

• This address space is often used behind Network Address Translation (NAT) boxes
  - Such boxes make it possible for many devices on the private side of the NAT box to "masquerade" as a single IP address on the public side
IP-in-IP Tunneling Example

Source=192.168.1.52
Dest=192.168.100.3
Protocol=TCP

Internet

192.168.1.52

192.168.100.3
IP-in-IP Tunneling Example

192.168.1.52

Source=65.7.22.5
Dest=150.37.2.5
Protocol=IPv4

65.7.22.5
150.37.2.5

Internet

192.168.100.3
IP-in-IP Tunneling Example

192.168.1.52

192.168.100.3

150.37.2.5

65.7.22.5
IP-in-IP Tunneling Example

192.168.1.52

65.7.22.5

150.37.2.5

192.168.100.3

Internet
Mapping IP to Lower-level

- Routing protocols (therefore forwarding tables) identify next-hop with an IP address.
- This address must be mapped to a lower-level address in order to actually forward a datagram!
- For point-to-point channels, this mapping may be statically configured:
  - Lower-level address doesn't matter much
    - After all, there's only one "other end"!
- For shared channels like Ethernet, it is a big deal.
Address Resolution Protocol

- ARP (RFC 826) designed to solve the problem of mapping IP addresses to lower-level address over broadcast channels
- Station that needs to resolve an IP address broadcasts "ARP Request" for the address
- Each station listens for such requests, responds with a message containing its "hardware" address when it hears its own IP address
# ARP Packet Format

<table>
<thead>
<tr>
<th>Hardware Type</th>
<th>Protocol Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>H/W/Len</td>
<td>Protocol Len</td>
</tr>
<tr>
<td>Source H/W Addr</td>
<td>Source Protocol Addr</td>
</tr>
<tr>
<td>Source Protocol Addr</td>
<td>Target H/W Addr</td>
</tr>
<tr>
<td>Target H/W Address</td>
<td>Target Protocol Addr</td>
</tr>
</tbody>
</table>

**Example Ethernet Frame**

- Ether Dest
- Ether Src
- Ethertype 0x0806=ARP
- ARP Message
- CRC
ARP Operation

Ozark's Cache
128.163.140.43 → 00:13:C4:80:93:3E

ARP Request
Src HW = 00:12:3F:74:6D:08
Src IP = 128.163.14.119
Target HW = 00:00:00:00:00:00
Target IP = 128.163.140.1

IP: 128.163.140.119
H/W: 00:12:3F:74:6D:08

IP: 128.163.140.43
H/W: 00:13:C4:80:93:3E

IP: 128.163.140.1
H/W: 00:30:96:33:C9:A0

IP: 128.163.140.44
H/W: 00:E0:18:F7:60:CC

Ozark
Magneto
Escalade
Yosemite
ARP Operation

Ozark's Cache
128.163.140.43 → 00:13:C4:80:93:3E
128.163.140.1 → 00:30:96:33:C9:A0

Ozark
IP: 128.163.140.119
H/W: 00:12:3F:74:6D:08

Yosemite
IP: 128.163.140.43
H/W: 00:13:C4:80:93:3E

Magneto
IP: 128.163.140.1
H/W: 00:30:96:33:C9:A0

Escalade
IP: 128.163.140.44
H/W: 00:E0:18:F7:60:CC
1. x.3 Looks up y.5 in forwarding table, finds next hop is x.1
2. x.3 resolves x.1 to Ethernet address b
3. x.3 transmits datagram as payload of Ethernet frame from a to b
4. Router receives Ethernet frame, strips header, passes payload to IP
5. Router looks up y.5 in forwarding table, finds next hop = y.5
6. Router resolves y.5 to Ethernet address d
7. Router transmits datagram as payload of Ethernet frame from c to d
8. y.5 receives frame, strips header, passes to IP