# Internet Protocol (Version 4)

CS 571

Fall 2006

# History

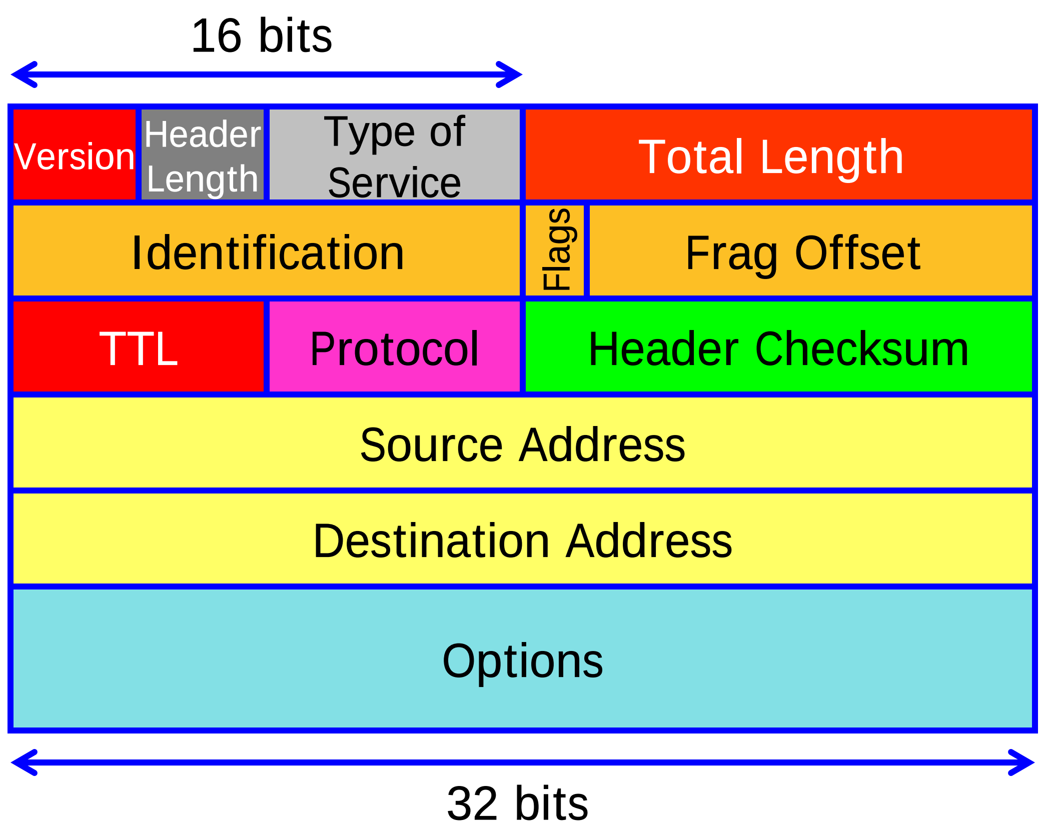
- Developed in late 1970's – early 1980's
  - DARPA Internetworking Program
- Goal: create a "catenet" = network of networks
  - LAN technologies were coming on the scene
  - Several wide-area packet networks already existed

    ARPANet, Tymnet, Telenet, DataPAC
  - Needed: Common global address space
- IP Specification: RFC 791, September 1981
- "Flag Day" cutover to TCP/IP: January 1, 1981
  - Note: Originally (late 70's) TCP and IP were one protocol

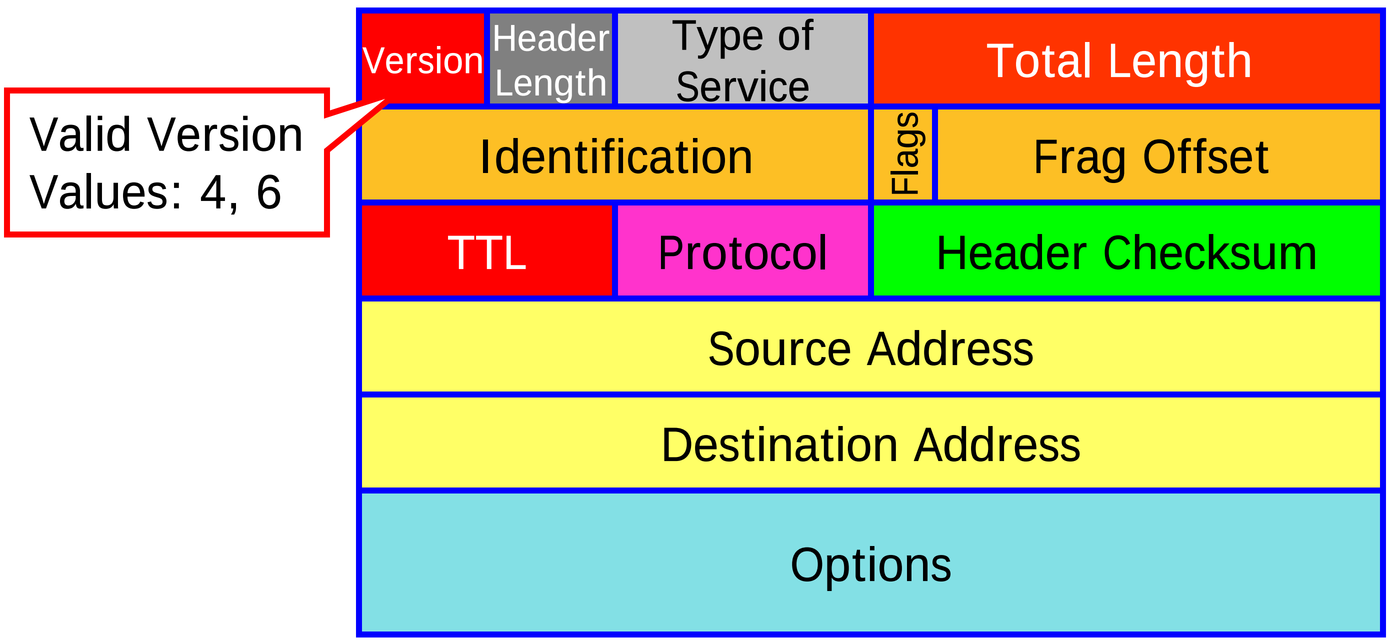
# What IPv4 Provides

- 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)

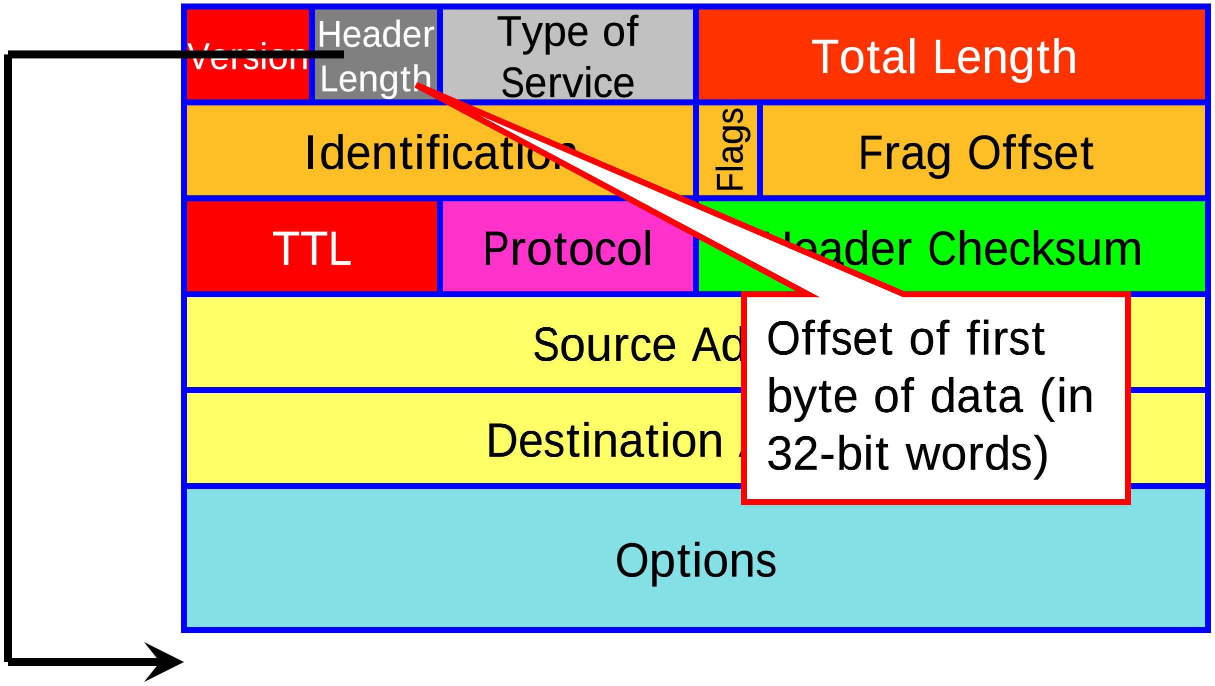
# IP Version 4 Header



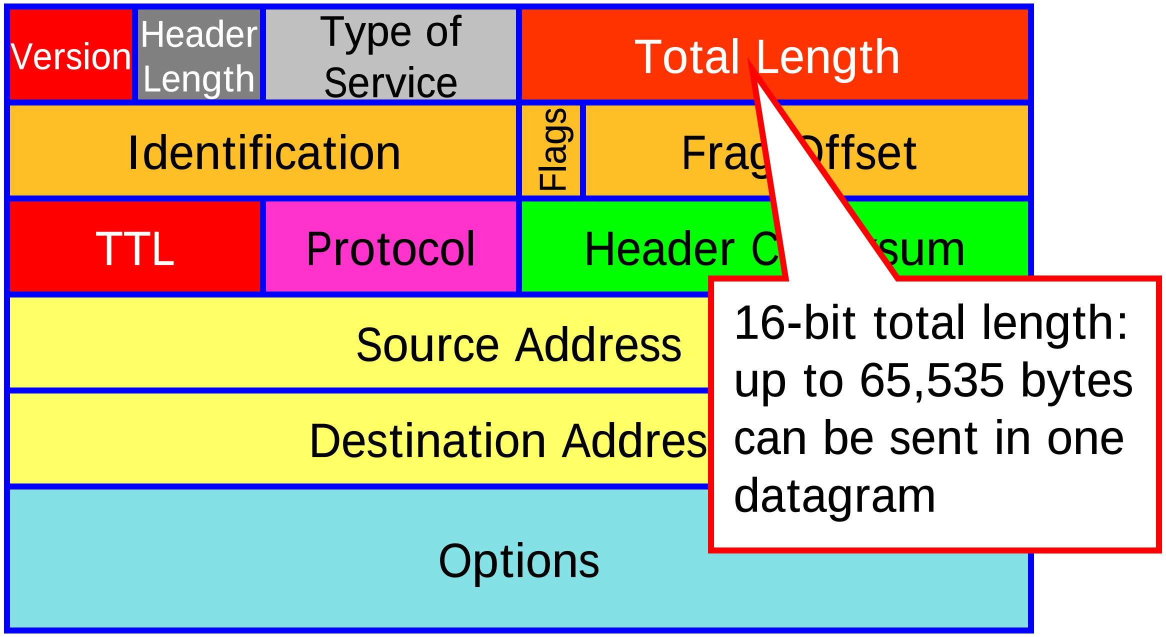
# IP Version 4 Header



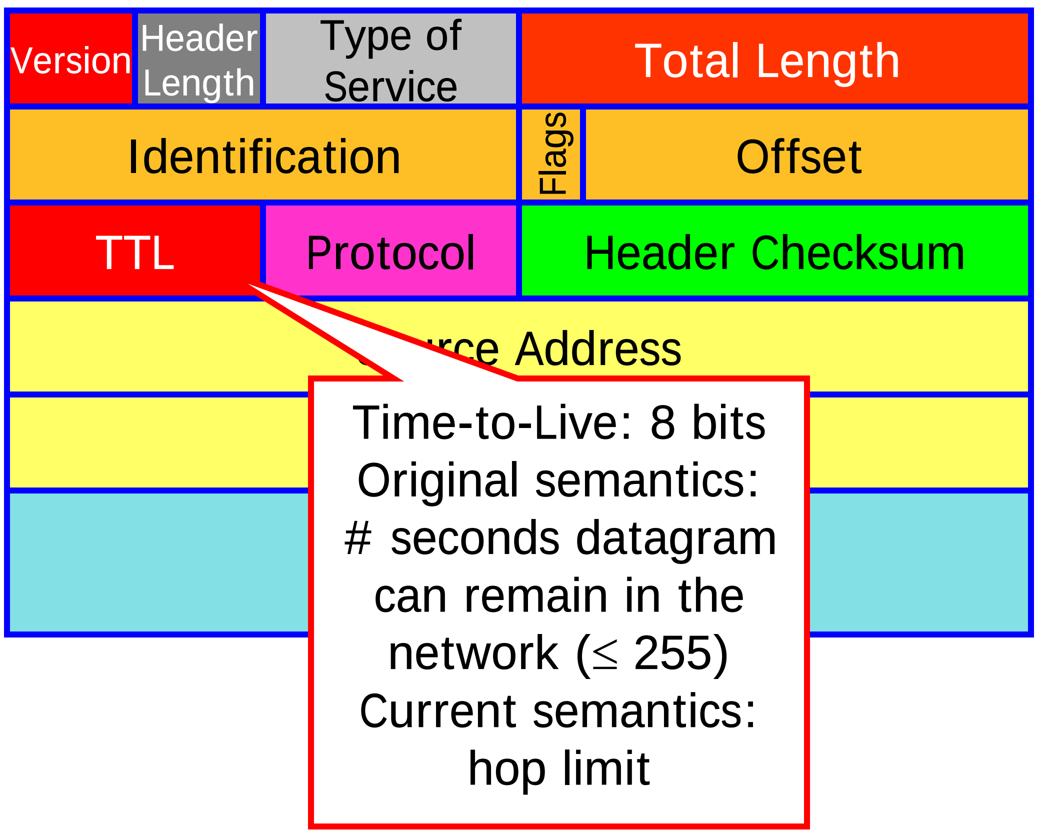
# IP Version 4 Header



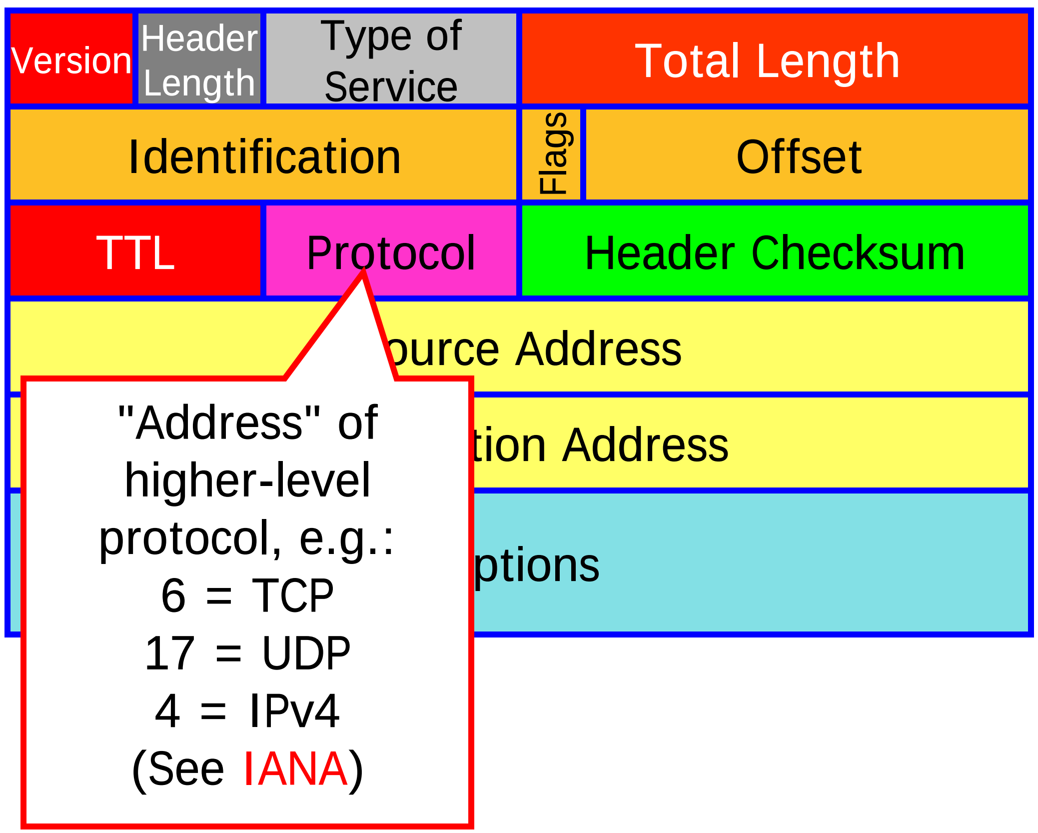
# IP Version 4 Header



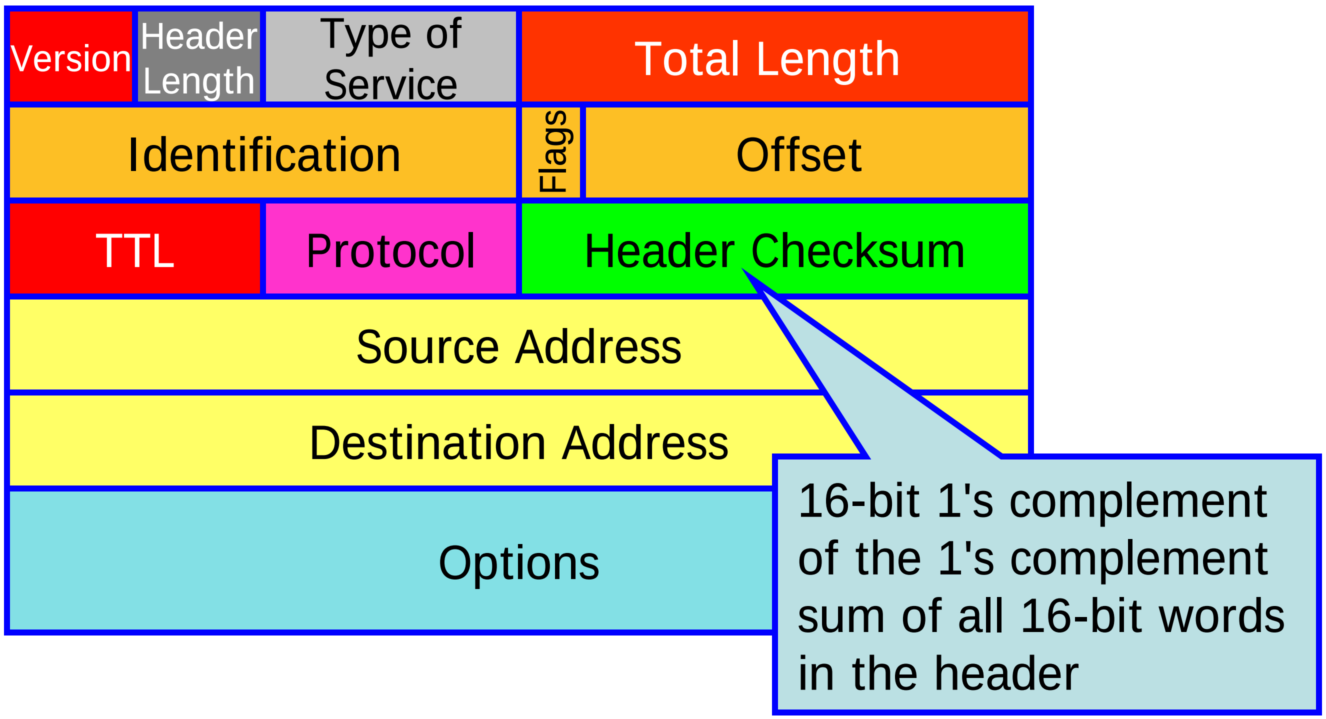
# IP Version 4 Header



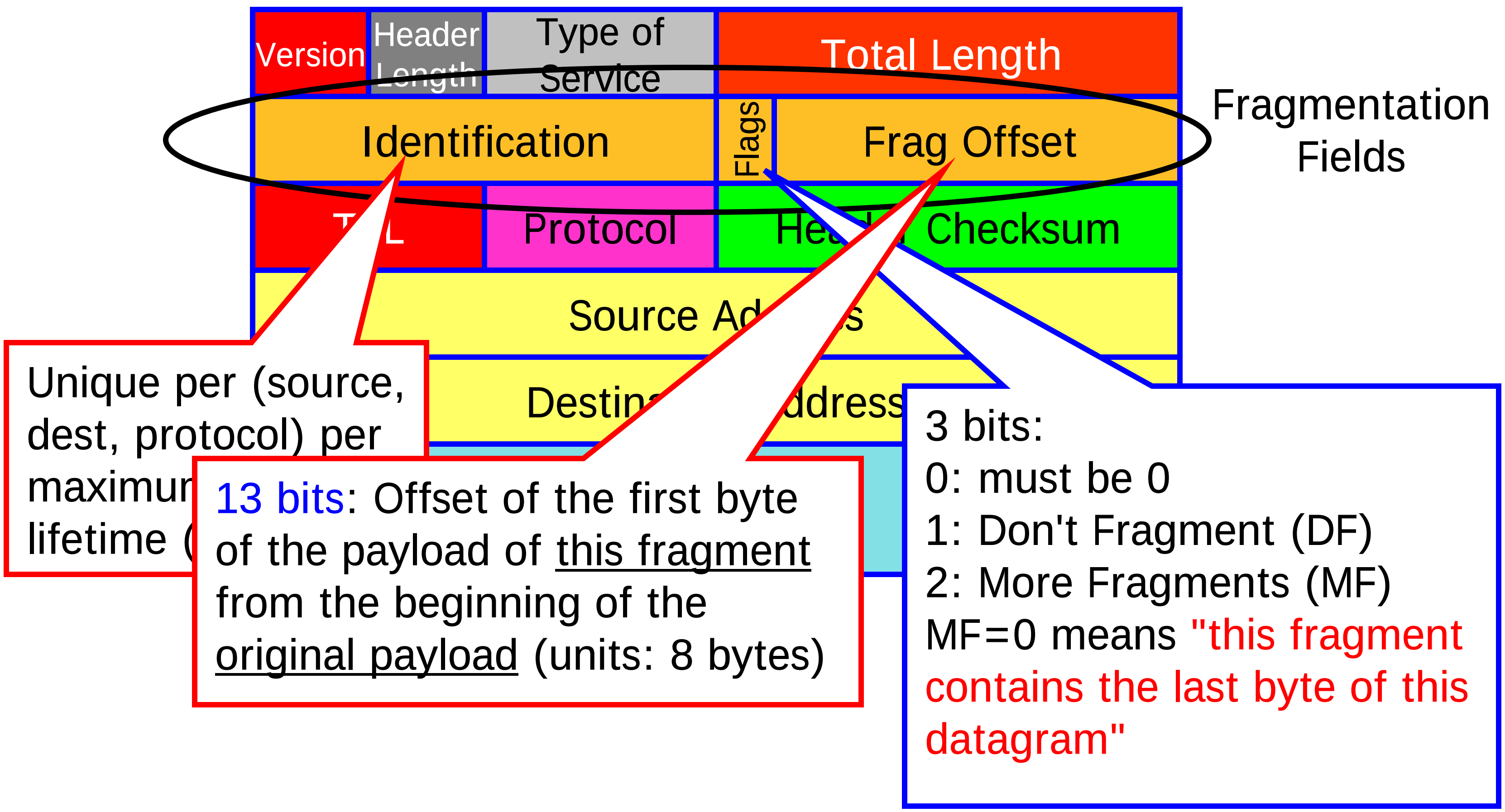
# IP Version 4 Header



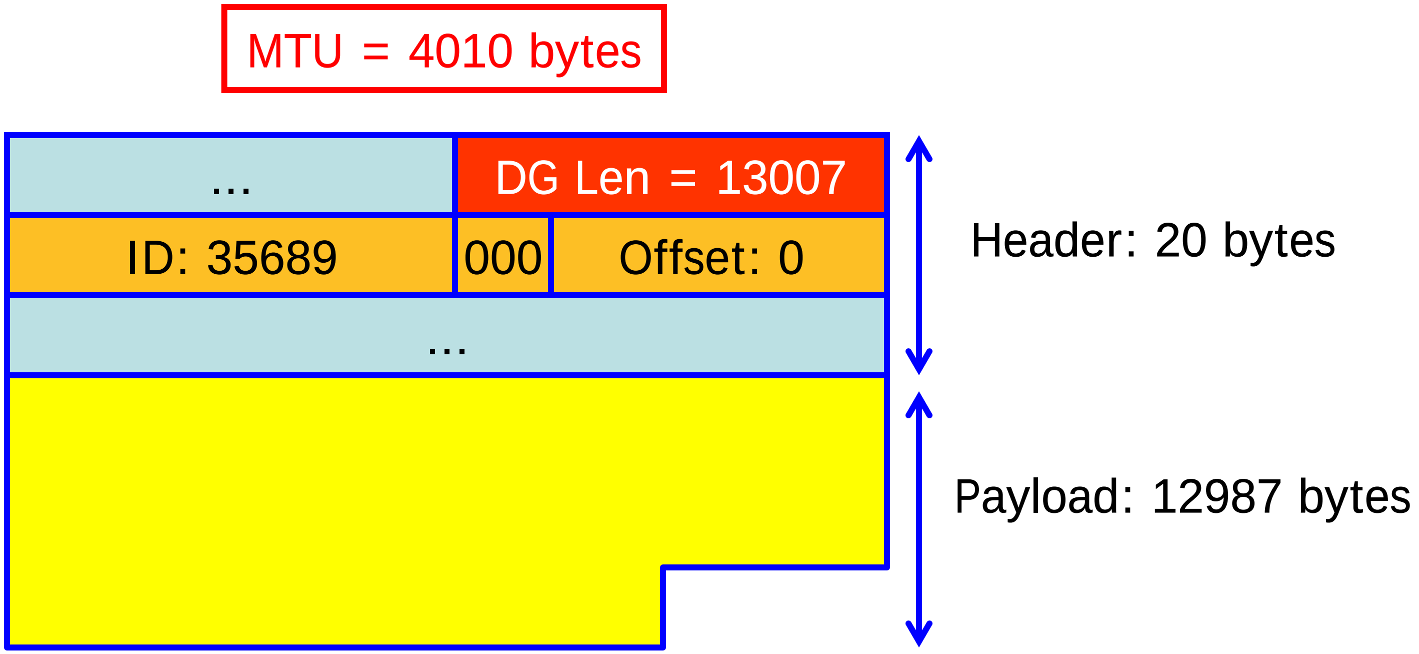
# IP Version 4 Header



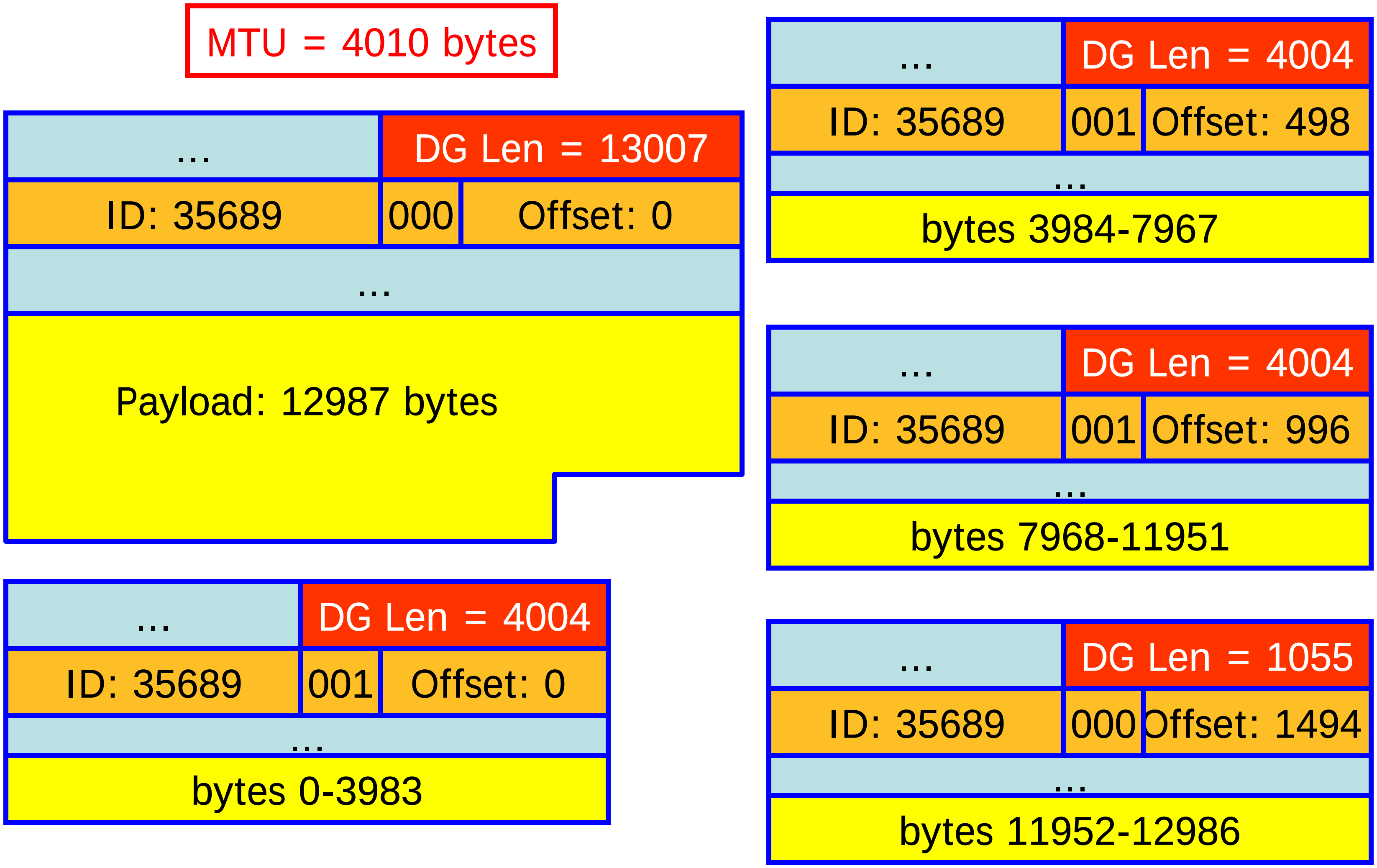
# IP Version 4 Header



# Fragmentation Example



# Fragmentation Example



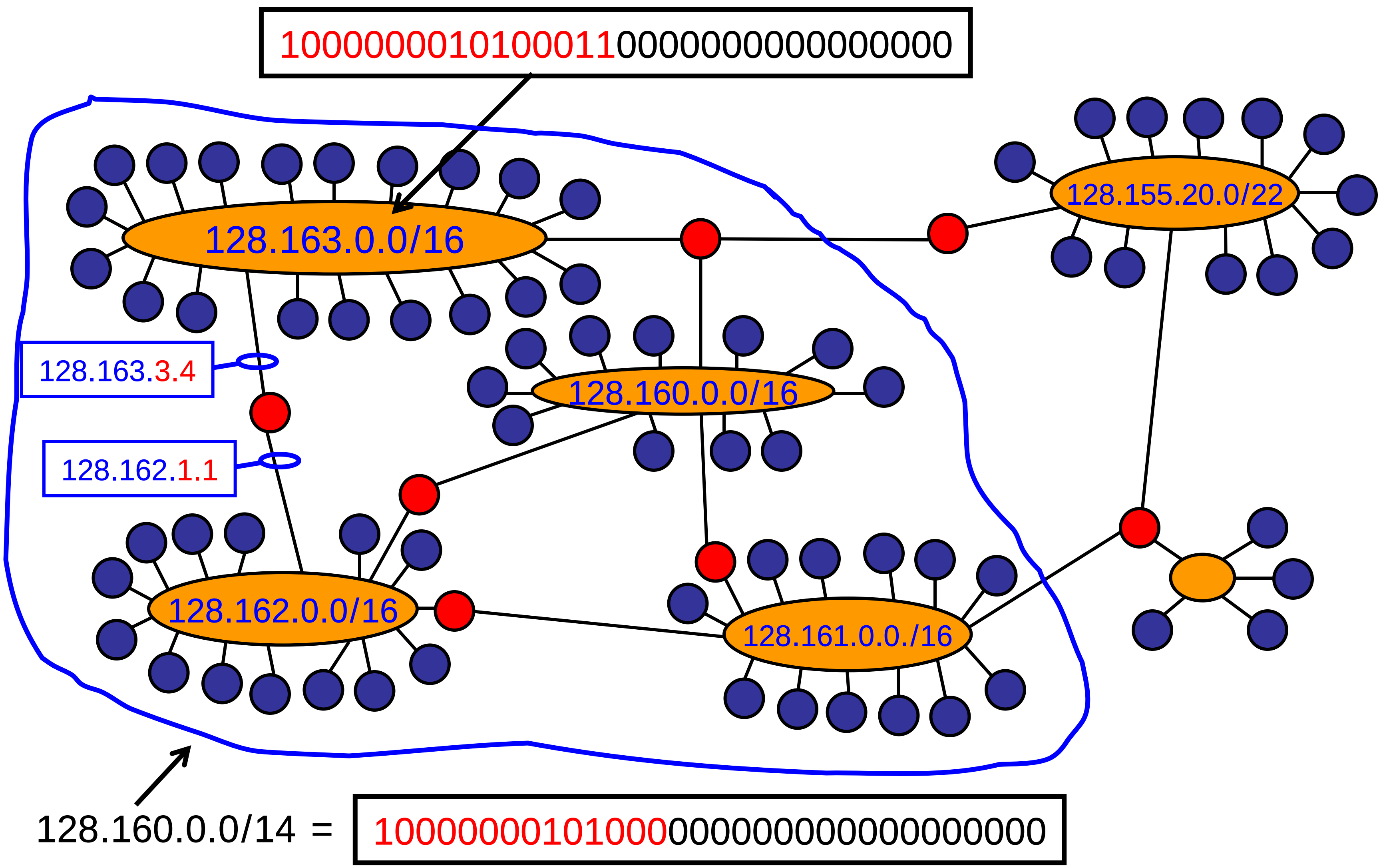
# Internet Addresses

- 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



# 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

# 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

| Prefix | Mask |
| --- | --- |
| 204.198.76.0 | 255.255.255.0 |
| 128.163.0.0 | 255.255.192.0 |
| 128.163.0.0 | 255.255.0.0 |

# Longest-Prefix Matching

Boolean Match(IPAddr dest, IPAddr prefix, IPAddr mask)

{ return ((dest & mask) == prefix); }

| | |
|---|---|
| 11001100110001100100110000000000 | 11111111111111111111111100000000 |
| 10000000101000110000000000000000 | 11111111111111111000000000000000 |
| 10000000101000110000000000000000 | 11111111111111110000000000000000 |

dest addr = 128.163.13.1

10000000101000110000110100000001

10000000101000110000110100000000

# Longest-Prefix Matching

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

| | |
|---|---|
| 11001100110001100100110000000000 | 11111111111111111111111100000000 |
| 10000000101000110000000000000000 | 11111111111111110000000000000000 |
| 10000000101000110000000000000000 | 11111111111111110000000000000000 |

dest addr = 128.163.13.1

```
10000000101000110000110100000001
10000000101000110000000000000000
```

Match!

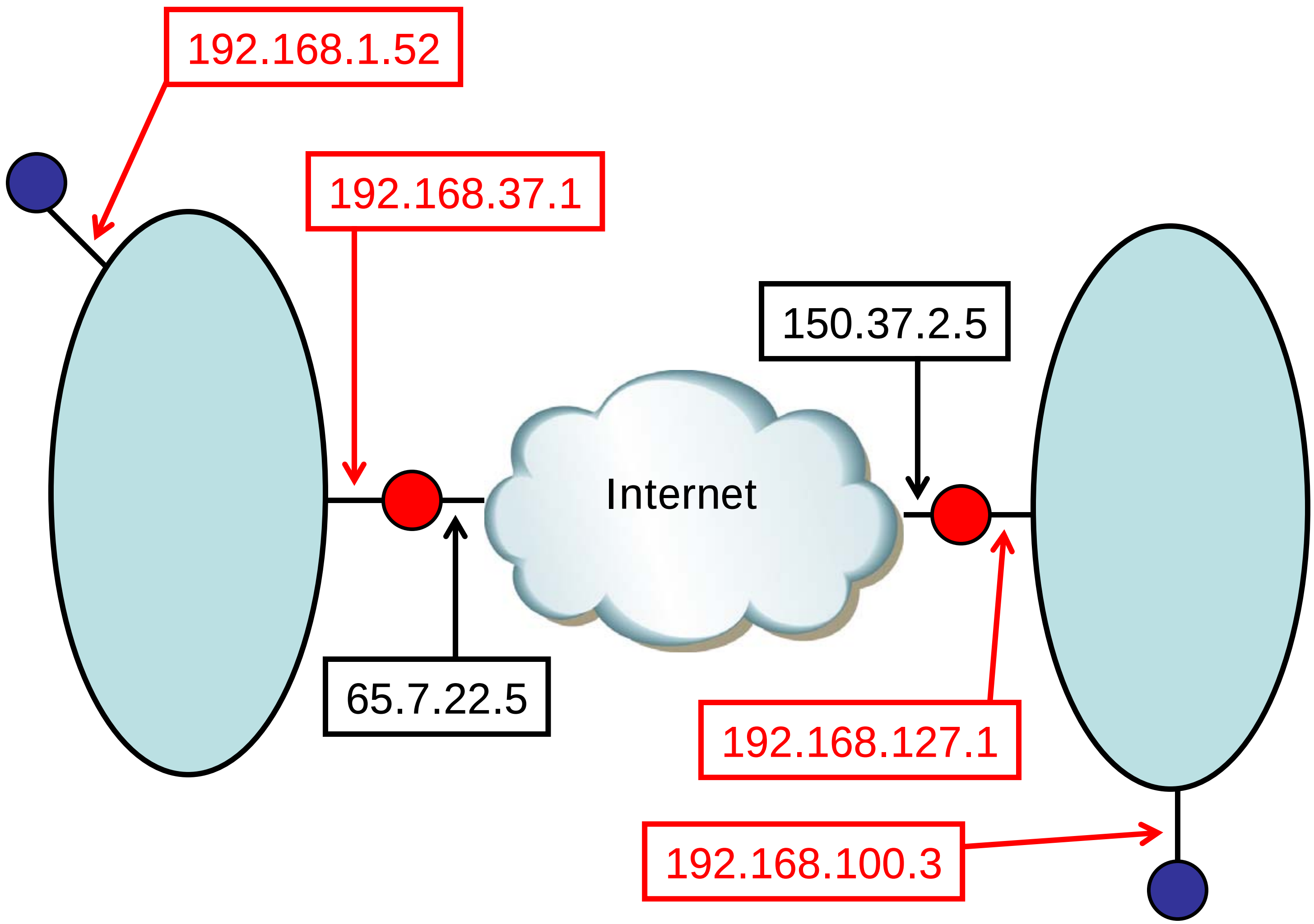
# Where does my address come from?

- 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

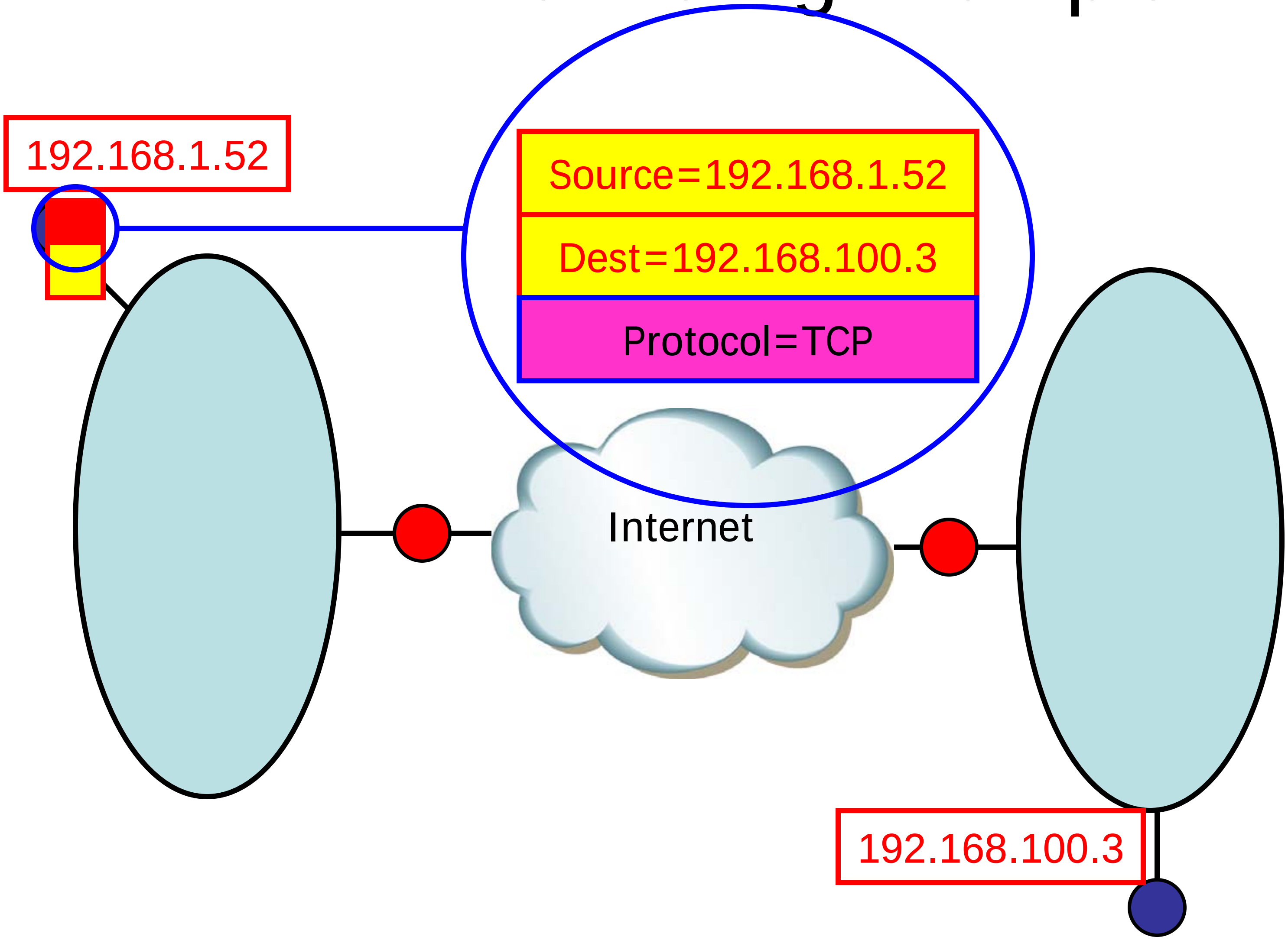
# Private Address Space

- 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

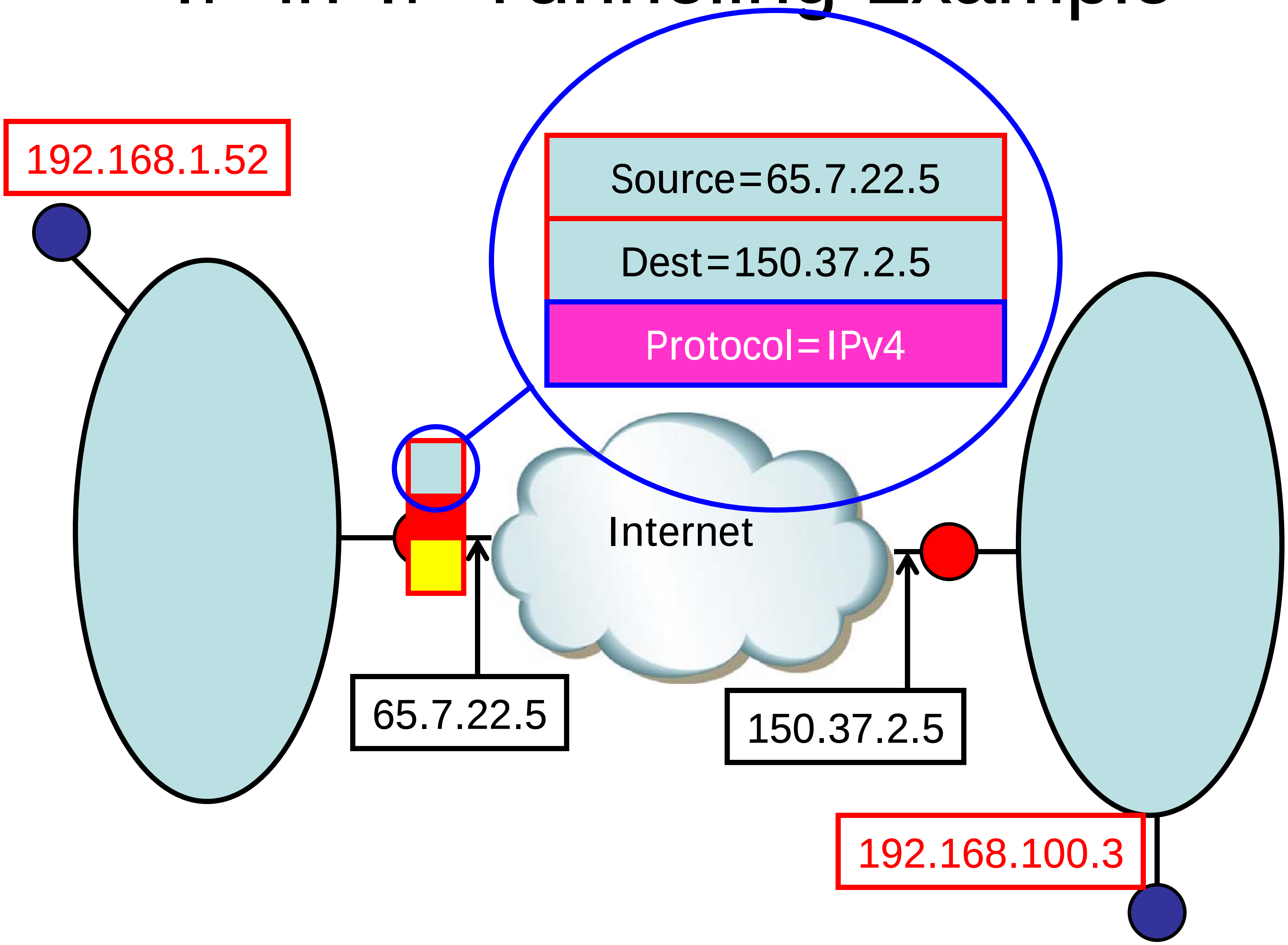
# IP-in-IP Tunneling Example



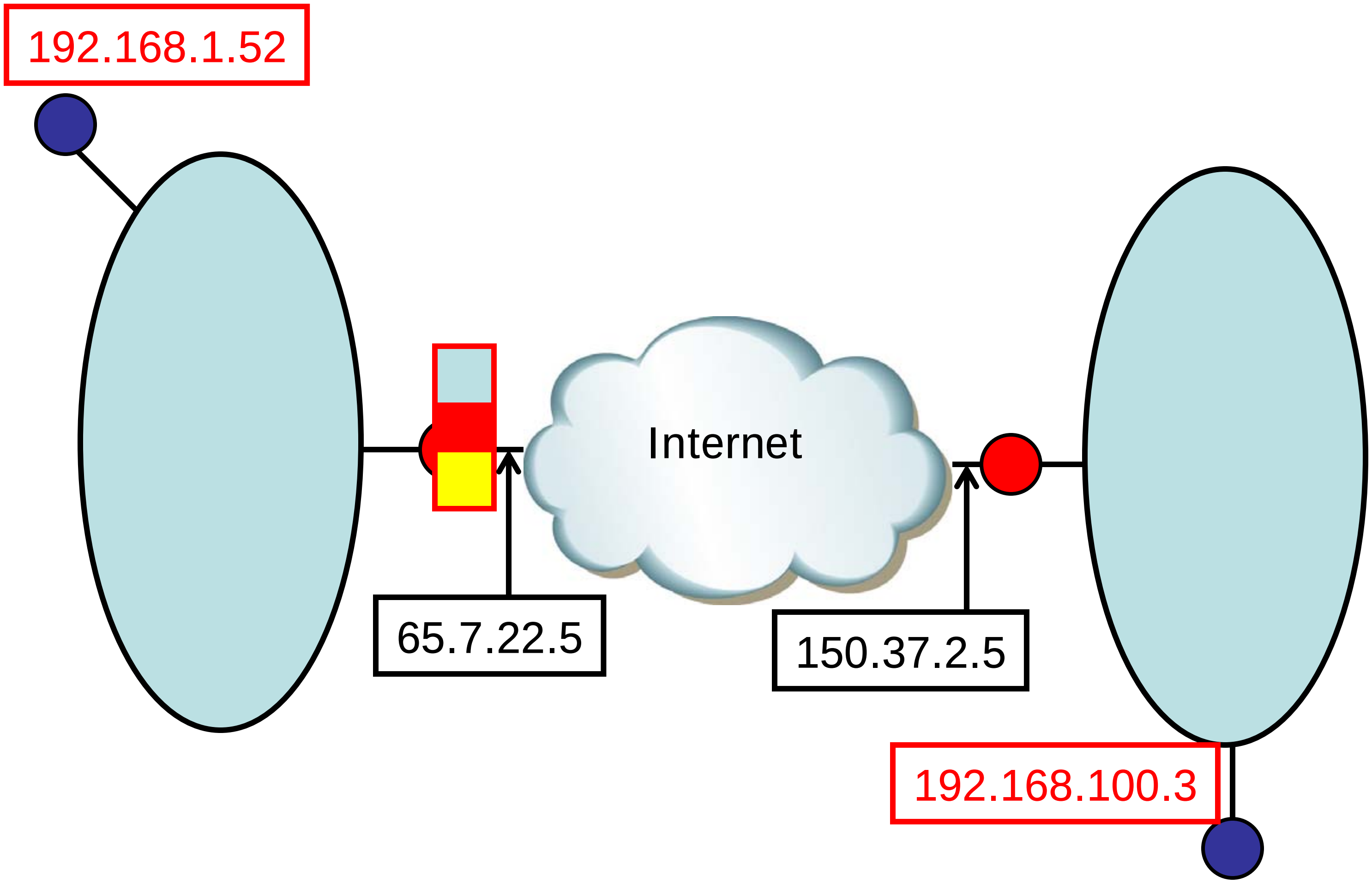
# IP-in-IP Tunneling Example



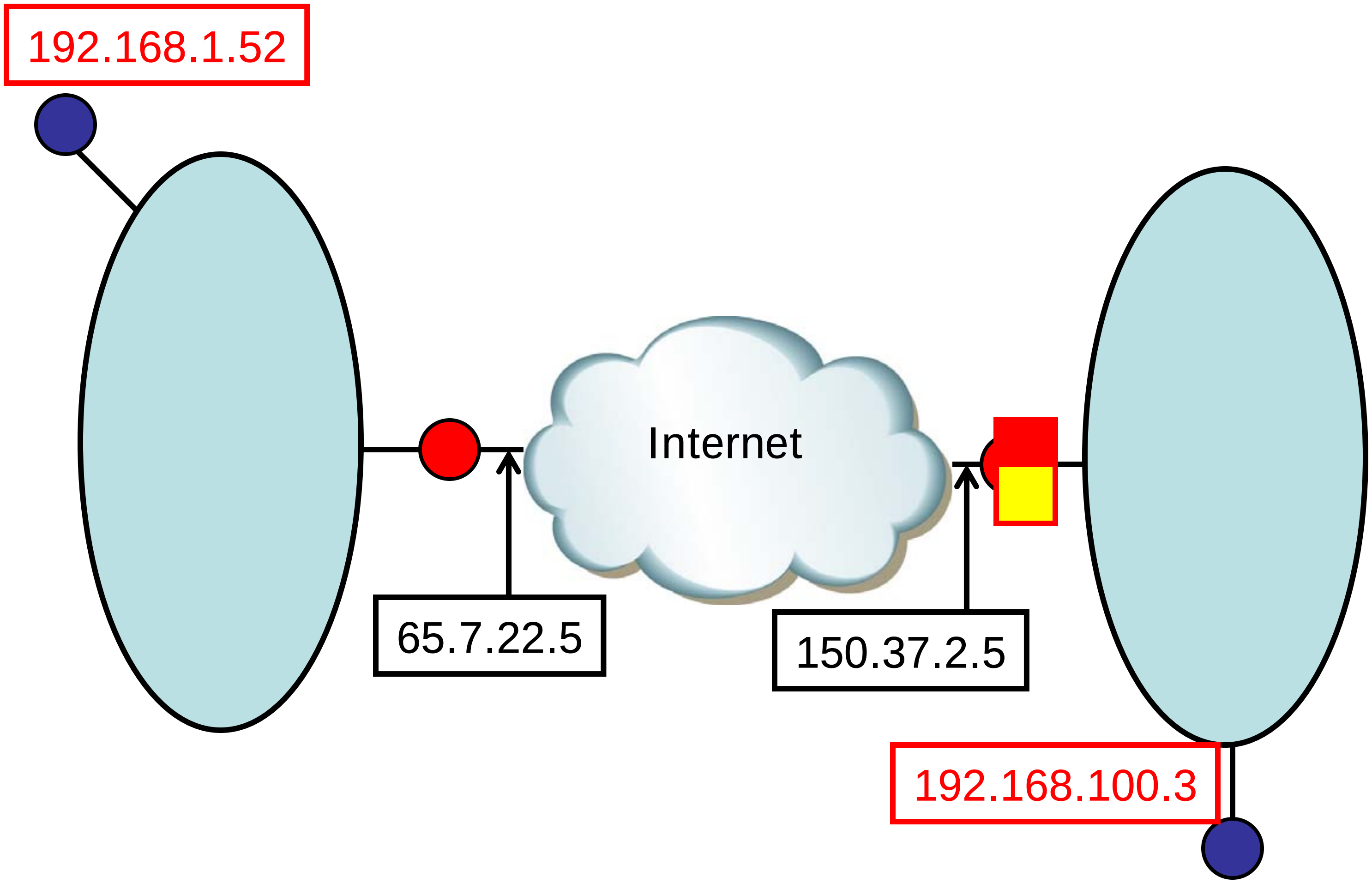
# IP-in-IP Tunneling Example



# IP-in-IP Tunneling Example



# IP-in-IP Tunneling Example



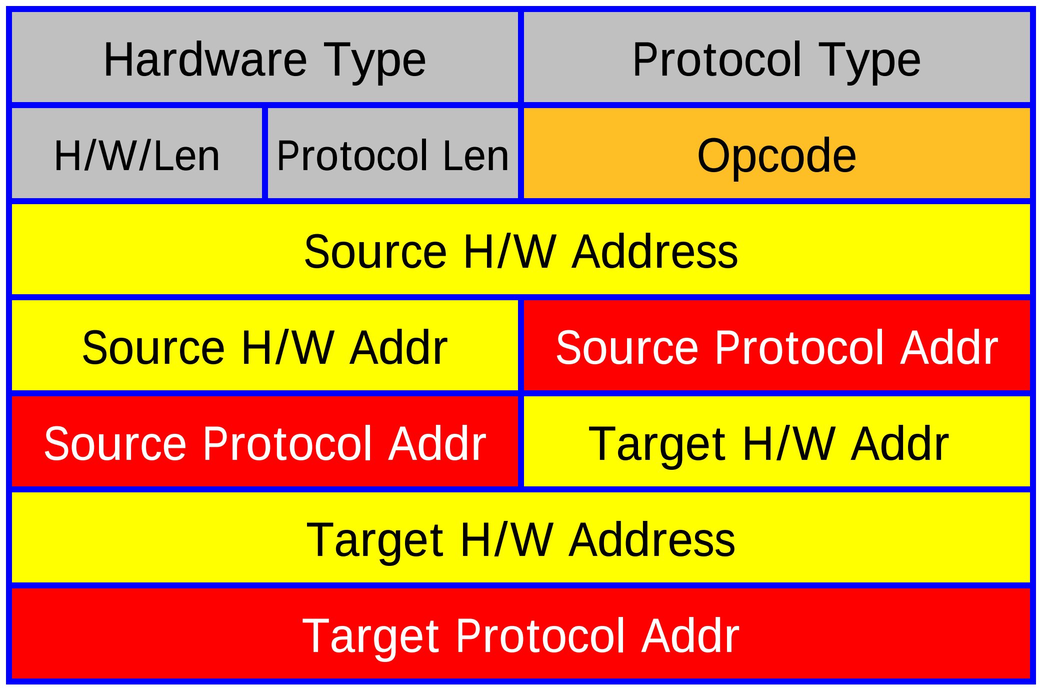
# Mapping IP to Lower-level

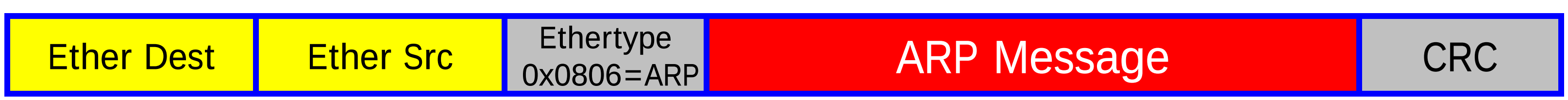
- Routing protocols (therefore forwarding tables) identify next-hop with an IP address
- This address must be mapped to a lower-level address in order to actually forward a datagram!
- For point-to-point channels, this mapping may be statically configured
  - Lower-level address doesn't matter much
    - 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

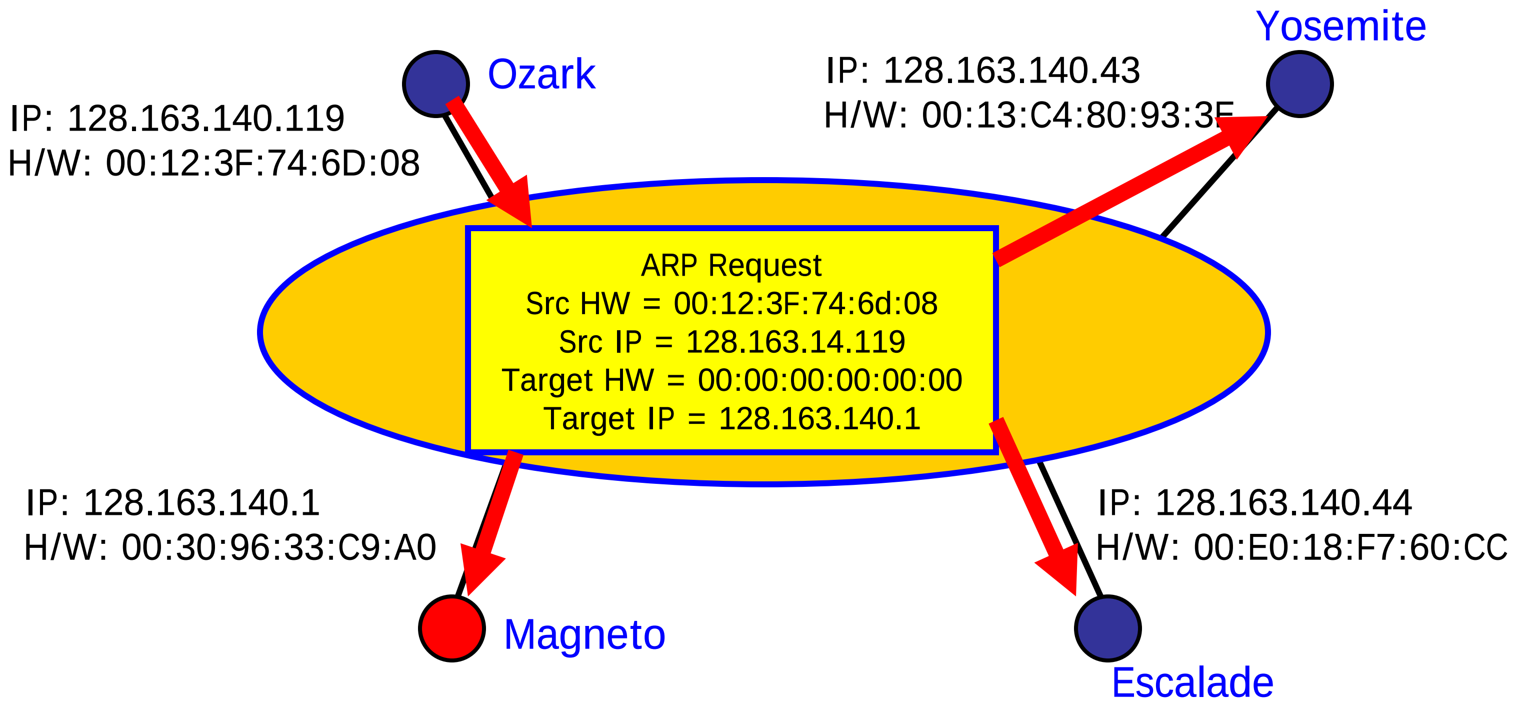


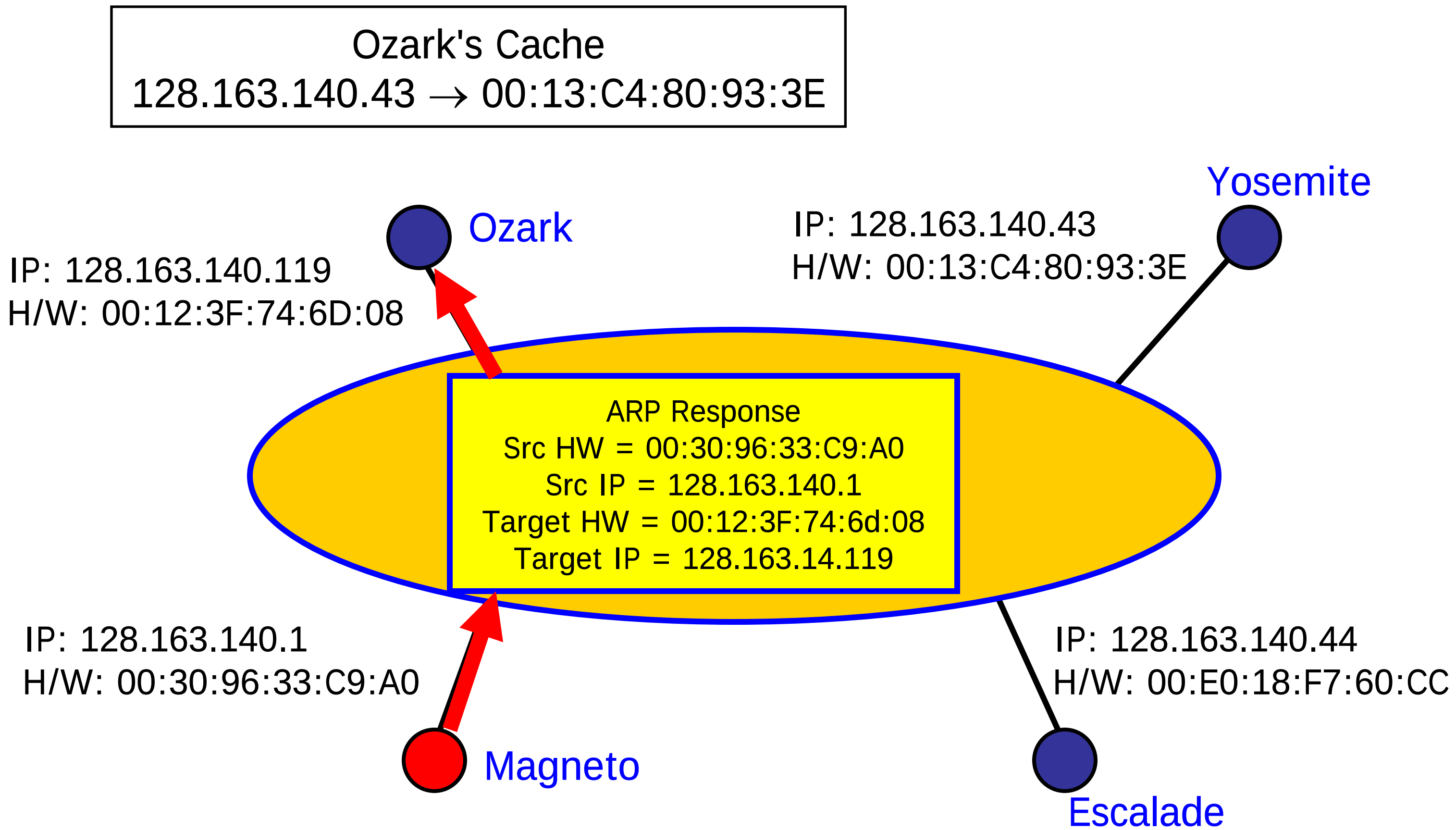

Example Ethernet Frame

# ARP Operation

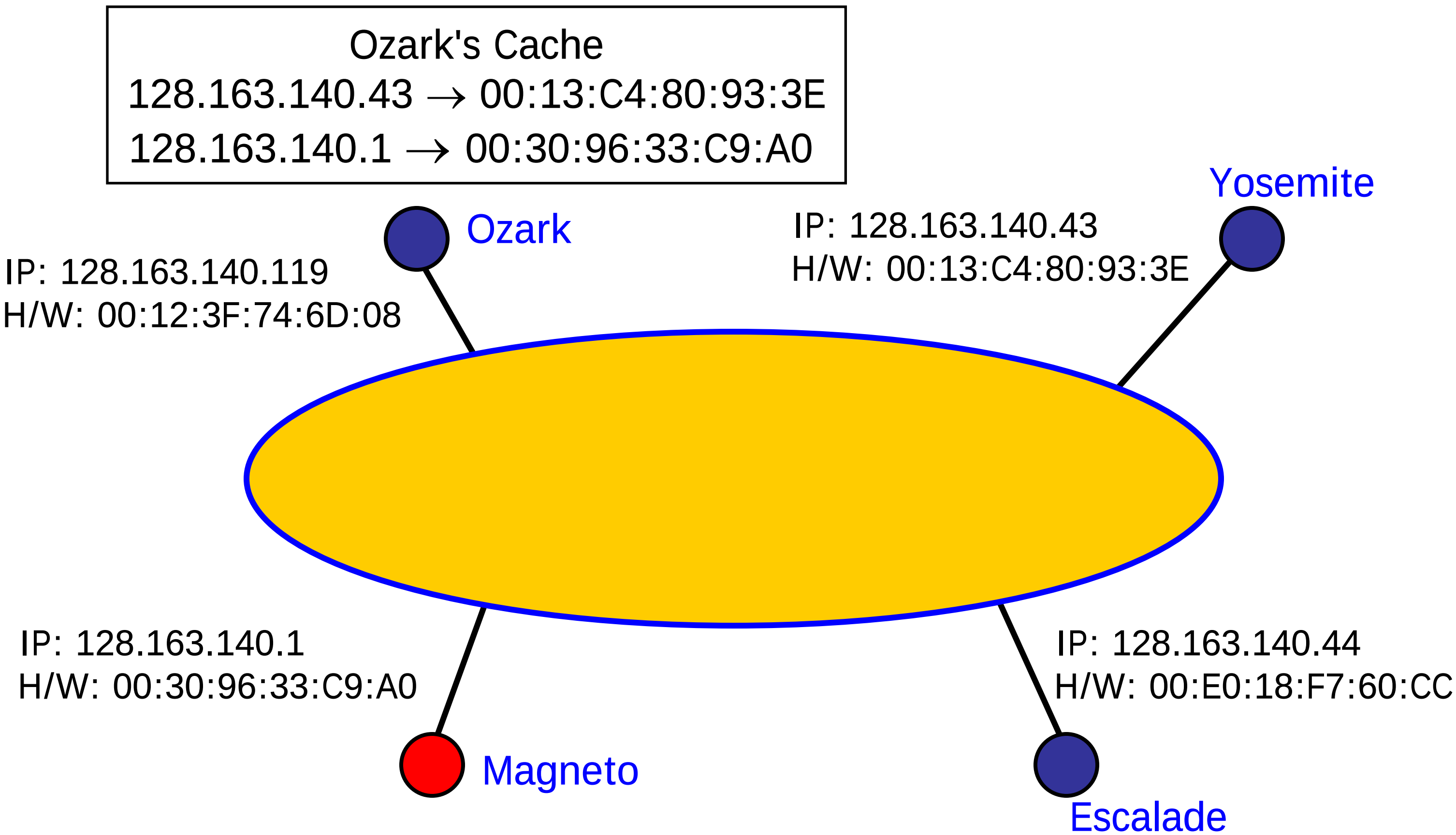
Ozark's Cache
128.163.140.43 → 00:13:C4:80:93:3E



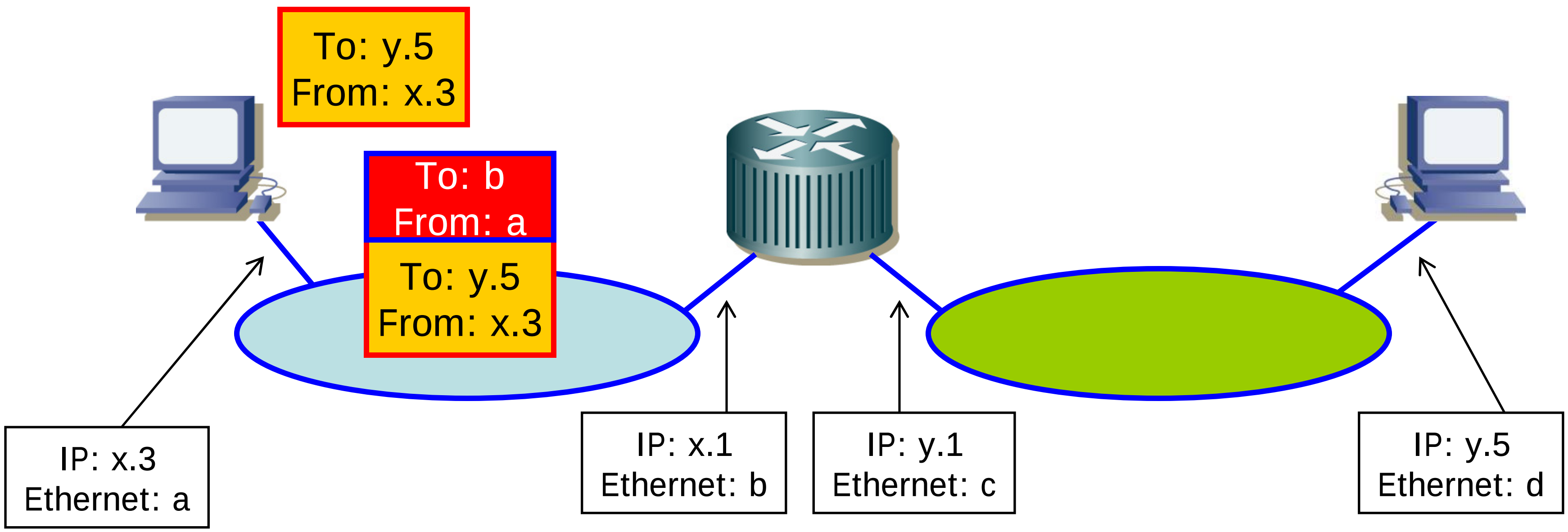
# ARP Operation



# ARP Operation

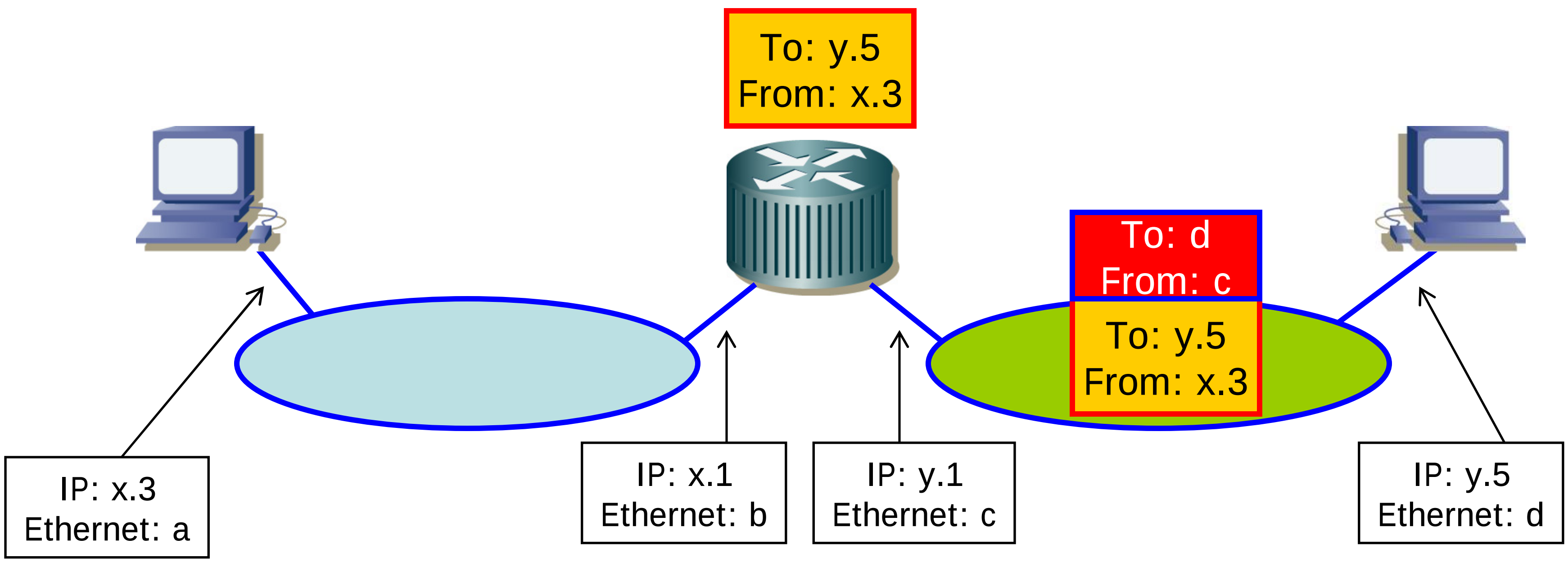


# IP Operation



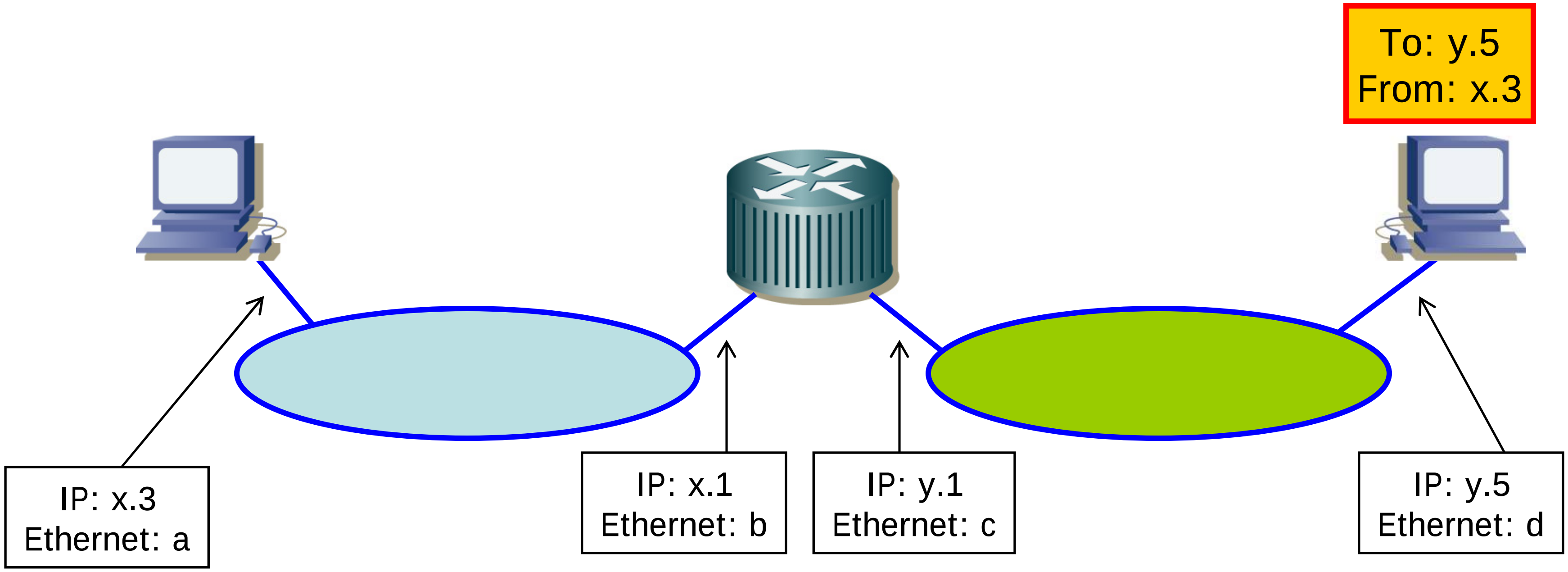

1. x.3 Looks up y.5 in forwarding table, finds next hop is x.1
2. x.3 resolves x.1 to Ethernet address b
3. x.3 transmits datagram as payload of Ethernet frame from a to b

# IP Operation




4. Router receives Ethernet frame, strips header, passes payload to IP
5. Router looks up y.5 in fwding 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

# IP Operation




8. y.5 receives frame, strips header, passes to IP