Internetworking
Chapter 4

The Big Picture

00010001
11001001
00011101

you are here
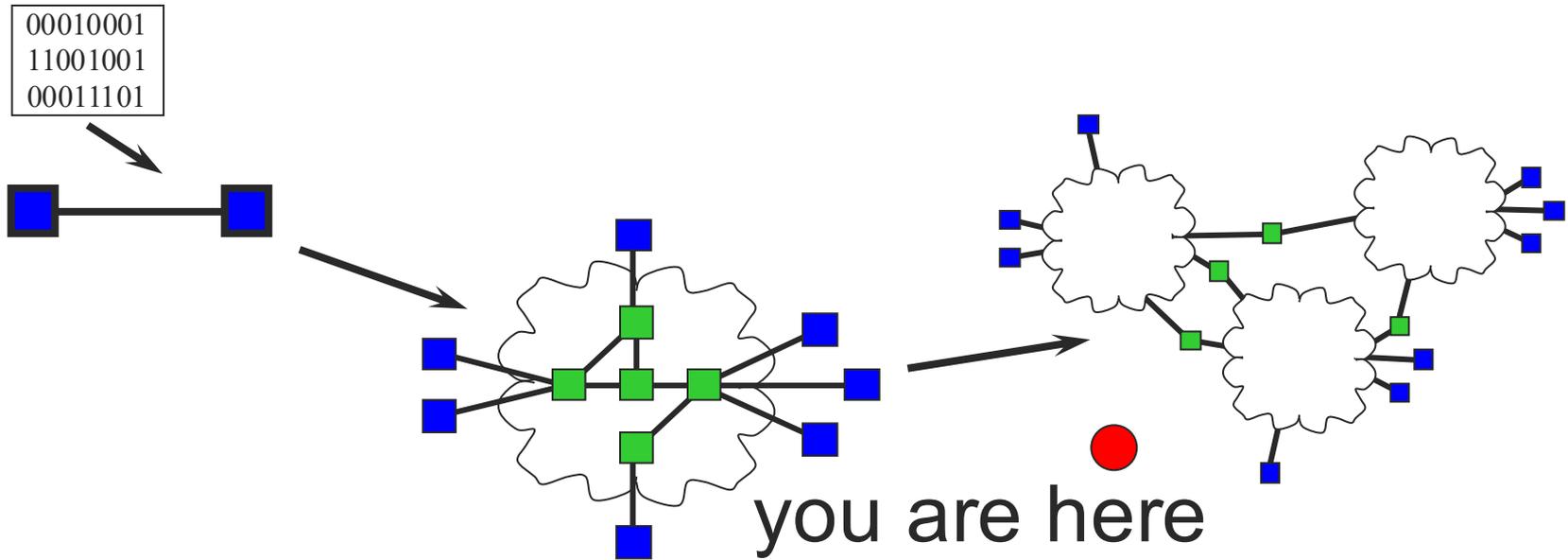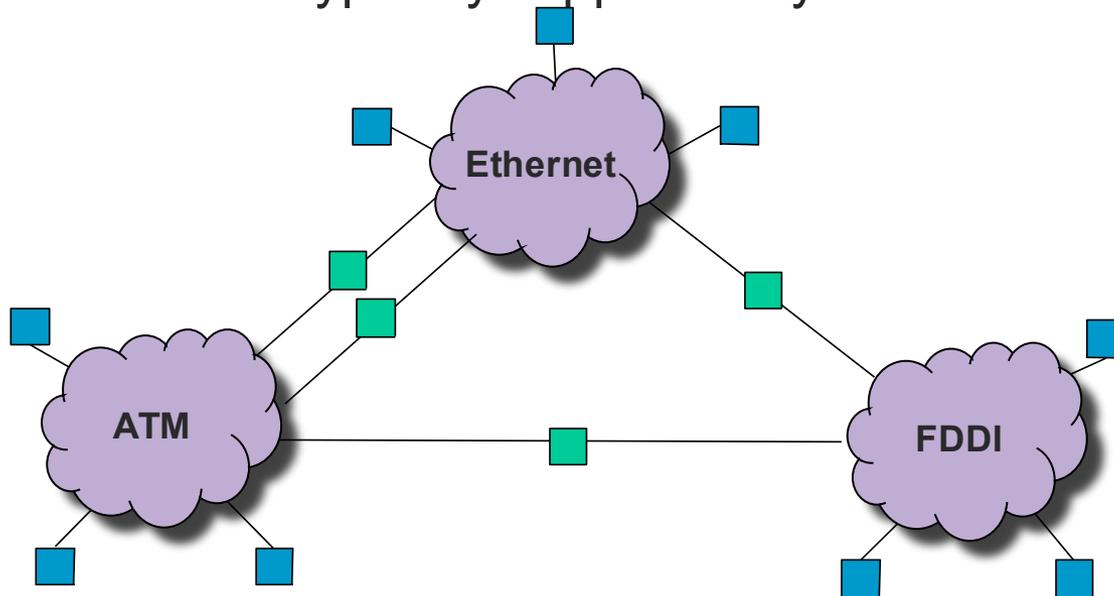
Internetworking

- Dealing with heterogeneity issues
  - Defining a service model
  - Defining a global namespace
  - Structuring the namespace to simplify forwarding
  - Building forwarding information (routing)
  - Translating between global and local (physical) names
  - Hiding variations in frame size limits
- Dealing with global scale
- Moving forward with IP

Basics of Internetworking

- What is an internetwork
  - Illusion of a single (direct link) network
  - Built on a set of distributed heterogeneous networks
  - Abstraction typically supported by software

Basics of Internetworking

- **What is an internetwork**
  - Illusion of a single (direct link) network
  - Built on a set of distributed heterogeneous networks
  - Abstraction typically supported by software
- **Properties**
  - Supports heterogeneity
    - Hardware, OS, network type, and topology independent
  - Scales to global connectivity
- **The Internet is the specific global internetwork that grew out of ARPANET**

What is the Internet?

- A pretty important example of an internetwork!

- It's big and complex
  - ○ ~120,000 ASs, over 2,000,000 routers, ~6B hosts
    - Last I checked …
  - ○ Rich array of systems and protocols

What is the Internet?

- ## How does it work?
  - ○ ISPs compete for business, hide private information, end-hosts misbehave, complex failure modes, cross-dependencies and oscillations

- ## Yet everyone must cooperate to ensure reachability
  - ○ Relies on complex interactions across multiple systems and protocols

Internetworking challenges

- **Heterogeneity**
  - Application service
  - Underlying networking technology
    - physical layer, frame sizes, reliability or not, …
  - Network routing / forwarding protocols
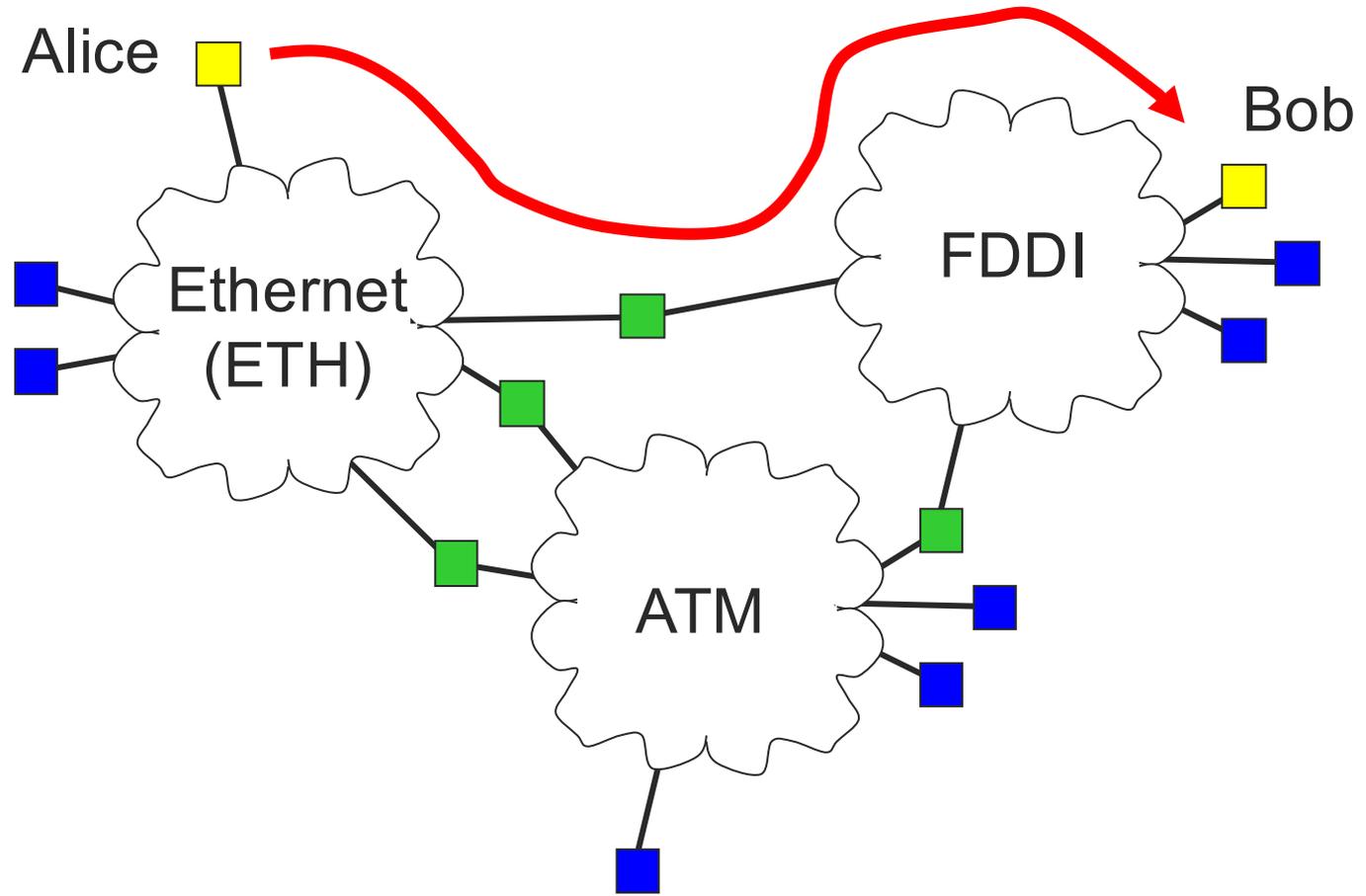 - Intended use

- **Global scale**
  - All of humanity and more …

- **Distributed control**
  - Different parties own different parts of the net
  - Different and conflicting goals for how network is used
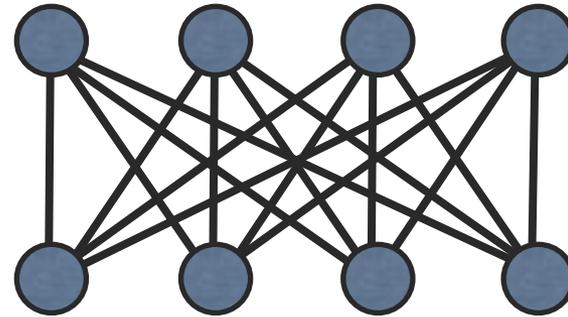
Message Transmission

Alice — Bob

Ethernet (ETH)

FDDI

ATM

Handling heterogeneity badly

Hosts

Networking protocols

n entities: n^2 interactions!

Handling heterogeneity well

Hosts

Networking protocols
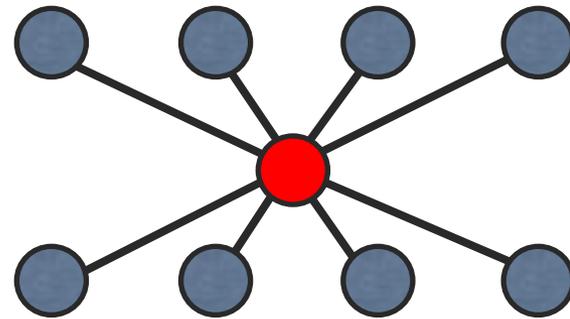
Handling heterogeneity well

- Network-level protocol for the Internet
- Operates on all hosts and routers
  - Routers are nodes connecting distinct networks to the Internet
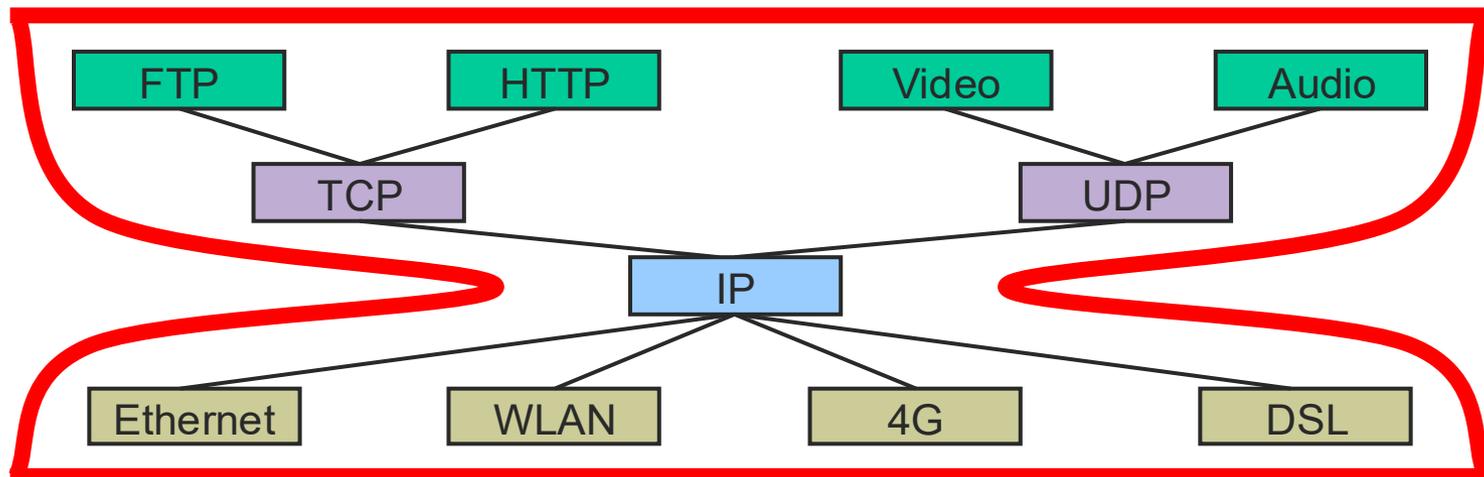
Architectural principle #1: Modularity

- **IP's "narrow waist"**
  - Single simple unifying protocol
  - Many heterogeneous uses above
  - Many heterogeneous implementations below
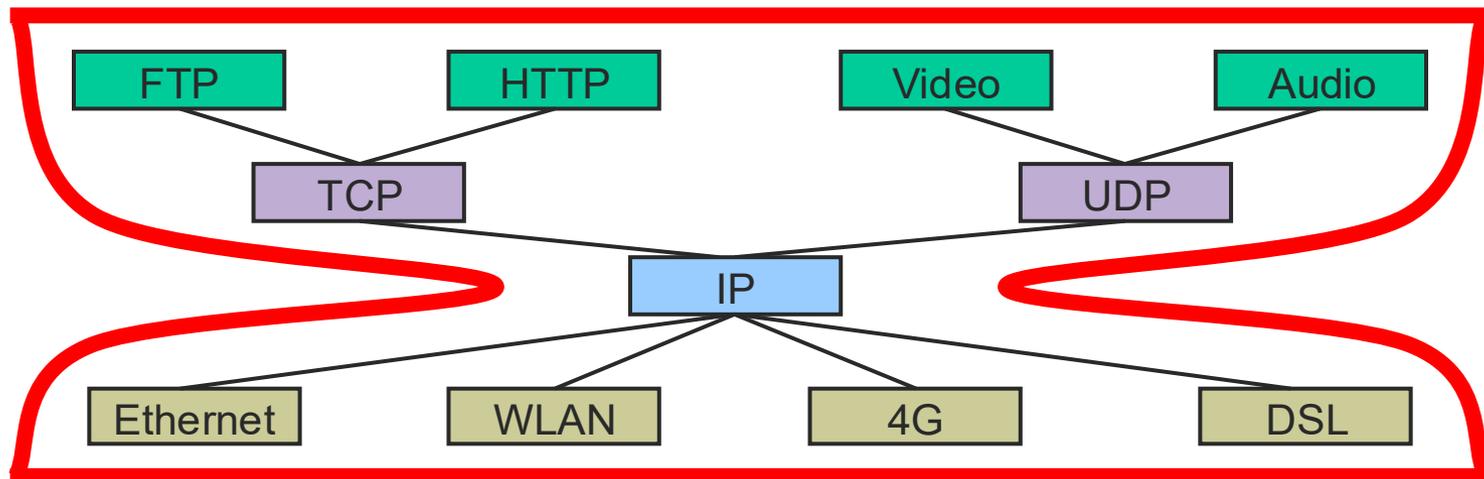
Architectural principle #1: Modularity

- Narrow waist: a special kind of layering
  - Protocols separated into layers
  - Each layer has well-defined interface and service model
  - Layer n needs only to talk to layers n+1 and n-1 (ideal case)

Layering

Alice — TCP / IP / ETH

Rick (a router) — IP / ETH / WiFi

Bob — TCP / IP / WiFi

Message Transmission

| Alice | Rick (a router) | Bob |
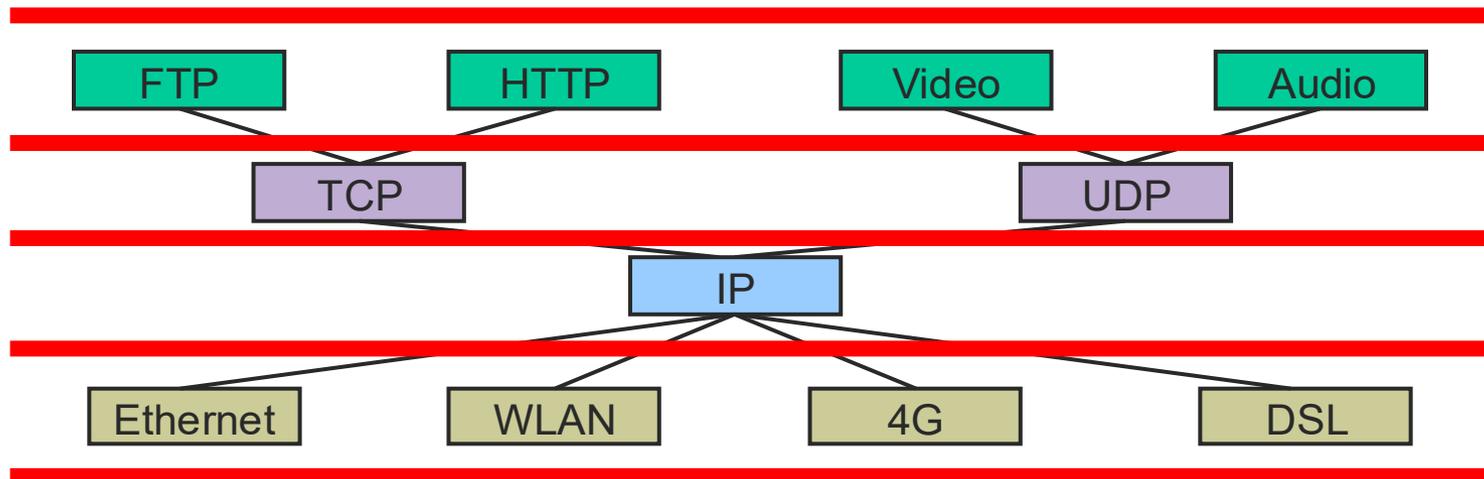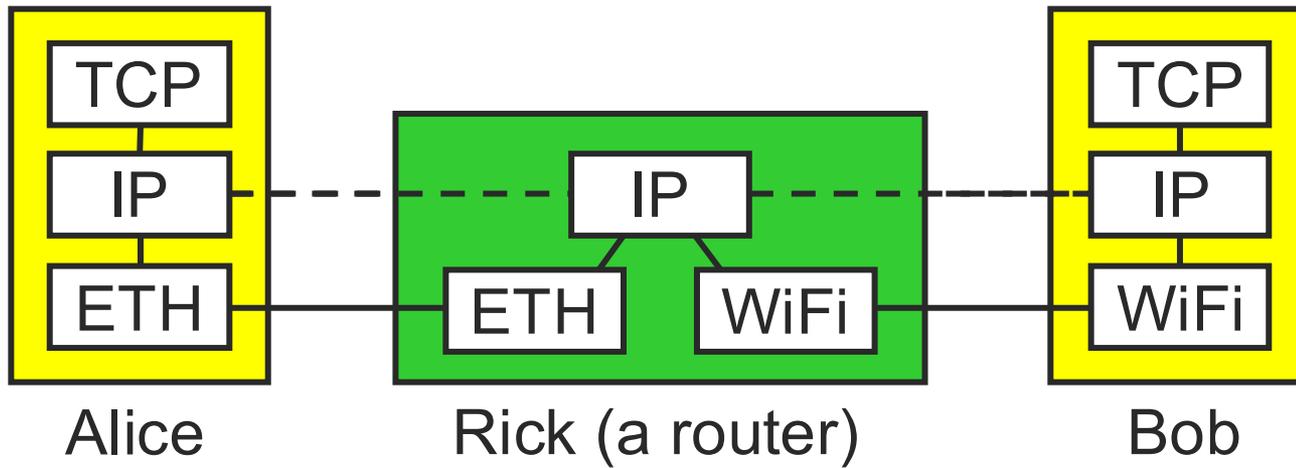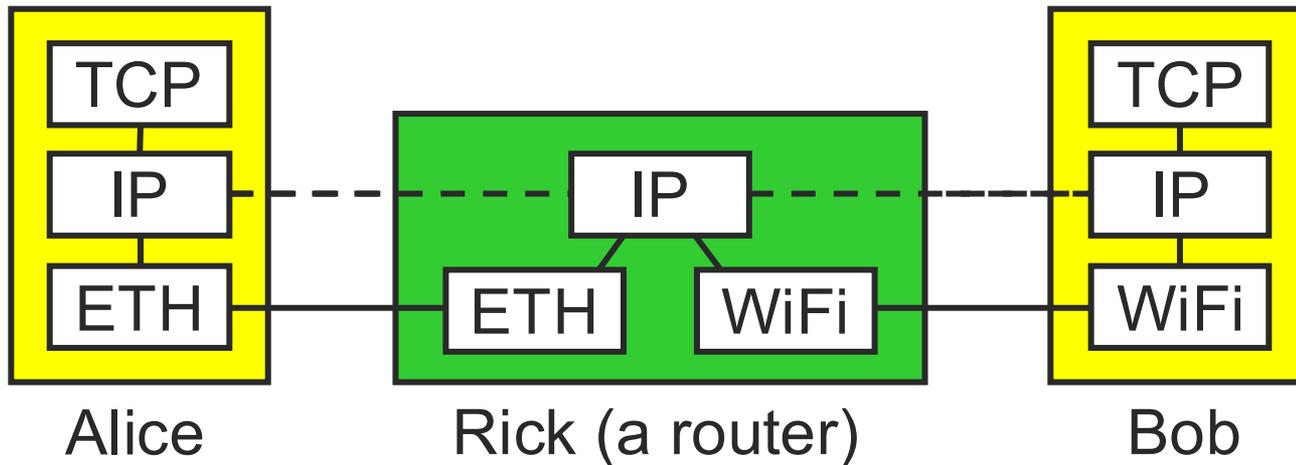
1. Alice/application finds Bob's IP address, sends packet
2. Alice/IP forwards packet to Rick
3. Alice/IP looks up Rick's Ethernet address and sends
4. Rick/IP forwards packet to Bob
5. Rick/IP looks up Bob's WiFi address and sends

Internet Protocol Service Model

- **Service provided to transport layer (TCP, UDP)**
  - Global name space
  - Host-to-host connectivity (connectionless)
  - Best-effort packet delivery
- **Not in IP service model**
  - Delivery guarantees on bandwidth, delay or loss
- **Delivery failure modes**
  - Packet delayed for a very long time
  - Packet loss
  - Packet delivered more than once
  - Packets delivered out of order

Architectural principle #2: End-to-end Principle

If a function can completely and correctly be implemented only with the knowledge and help of the application standing at the endpoints of the communication system, then providing that function as a feature of the communication system itself is not possible.

[Saltzer, Reed, Clark, 1984]

Example: file transfer

- Suppose the link layer is reliable. Does that ensure reliable file transfer?

- Suppose the network layer is reliable. Does that ensure reliable file transfer?

file transfer
application

file transfer
application

network

disk

flow of data

disk

E2E Principle: Interpretation

- If function can only be fully implemented at endpoints of communication, then...
  - End-to-end implementation
    - Correct
    - Simplifies, generalizes lower layers
  - In-network implementation
    - Insufficient
    - May help – or hurt – performance. Examples?
- Be wary to sacrifice generality for performance!

E2E Principle and IP

- **Implications**
  - No need for reliability in network
  - No need to be connection-oriented (virtual circuits): datagram packet switching enough
- **Benefits**
  - Easy to implement IP on top of many different underlying network technologies below IP layer
  - Lightweight yet sufficient platform for many applications on top of the IP layer

Designing IP:
What do we need to know?

- Addresses: source and destination of datagram

- Framing: datagram length

- Dealing with problems
    - Integrity: is the header what it's supposed to be?
    - Loop avoidance: make sure packets don't endlessly circulate
    - Fragmentation: what if the datagram is too large?

- Options and extensibility
    - Ways to give more specific information to routers
    - Ways to extend the protocol with new features

IP Packet Format

0	4	8	16	19	31

Version	HLen	TOS	Length		
Ident			Flags	Offset	
TTL		Protocol	Checksum		
SourceAddr					
DestinationAddr					
Options (variable)				Pad (variable)	

Data

4-bit version
    IPv4 = 4, IPv6 = 6

IP Packet Format

0	4	8	16	19	31

Version	HLen	TOS	Length		
Ident			Flags	Offset	
TTL		Protocol	Checksum		
SourceAddr					
DestinationAddr					
Options (variable)				Pad (variable)	
Data					

4-bit IP header length
    Counted in 32-bit words, minimum value is 5

IP Packet Format

| 0 | 4 | 8 | 16 | 19 | 31 |

Version	HLen	TOS	Length		
Ident			Flags	Offset	
TTL		Protocol	Checksum		
SourceAddr					
DestinationAddr					
Options (variable)				Pad (variable)	

8-bit type of service field (TOS)
    Specifies priorities: priority level,
    minimize delay, maximize throughput,
    maximize reliability, minimize
    monetary cost, etc
    Mostly unused

© CS 423 Staff, University of Illinois

IP Packet Format

0	4	8	16	19	31

Version	HLen	TOS	Length		
Ident			Flags	Offset	
TTL	Protocol		Checksum		
SourceAddr					
DestinationAddr					
Options (variable)				Pad (variable)	
Data					
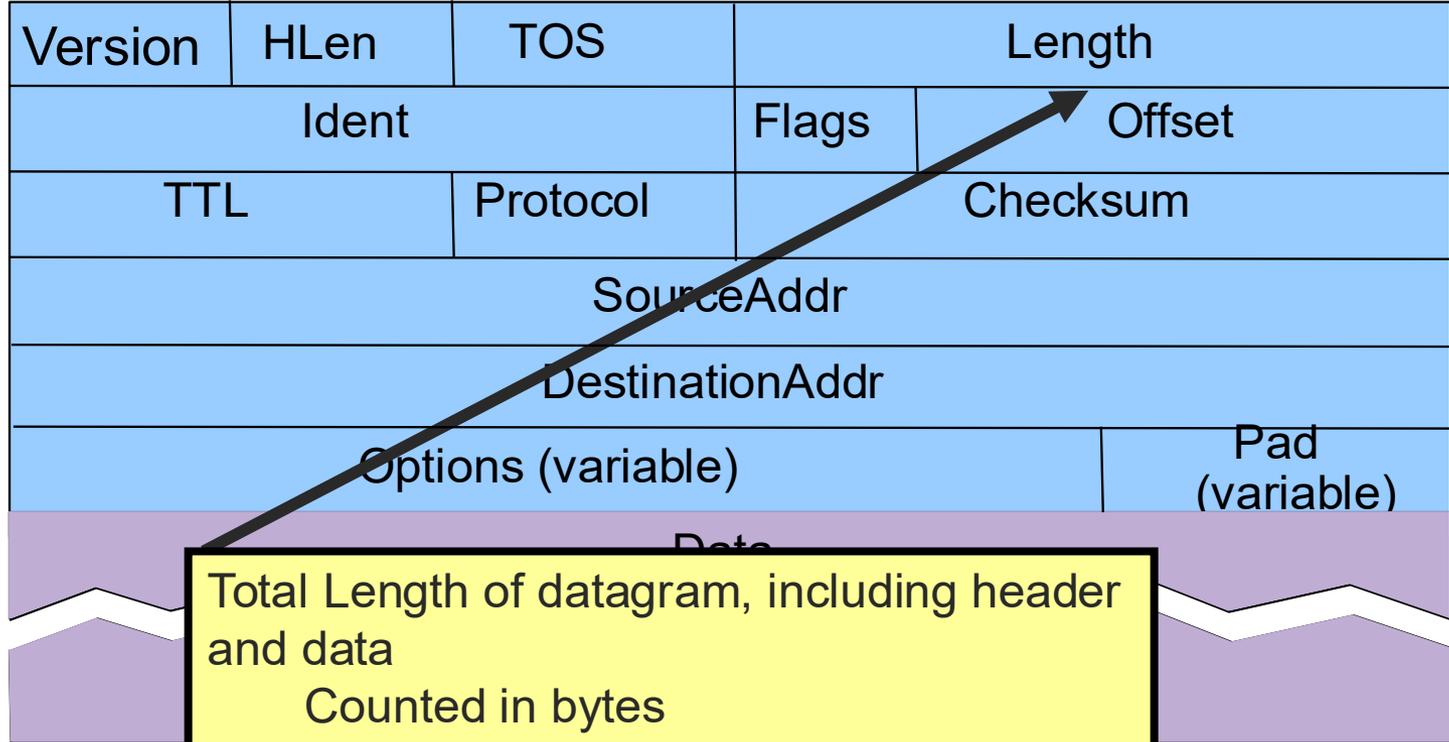
Total Length of datagram, including header and data
    Counted in bytes
    Maximum size is 65536 bytes

IP Packet Format

0	4	8	16	19	31

Version	HLen	TOS	Length		
Ident			Flags	Offset	
TTL		Protocol	Checksum		
SourceAddr					
DestinationAddr					
Options (variable)				Pad (variable)	
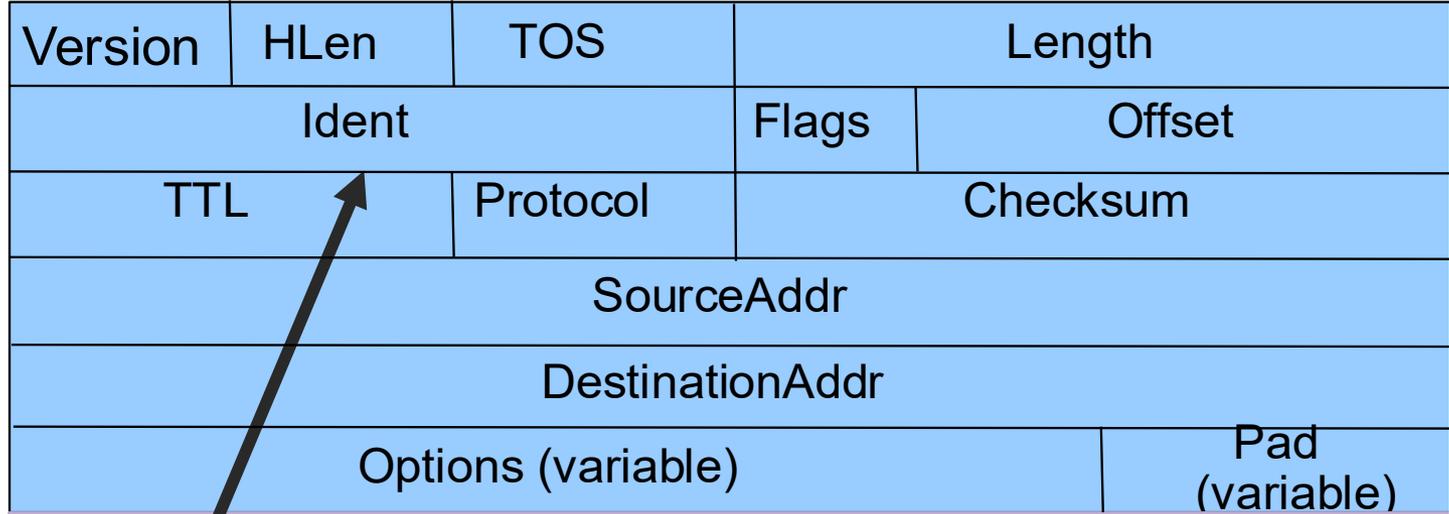
16-bit packet ID (Fragmentation support)
All fragments from the same packet have the same ID
Sender must make sure ID is unique for that (src,dest) pair, for the time the datagram will be active in the Internet

IP Packet Format

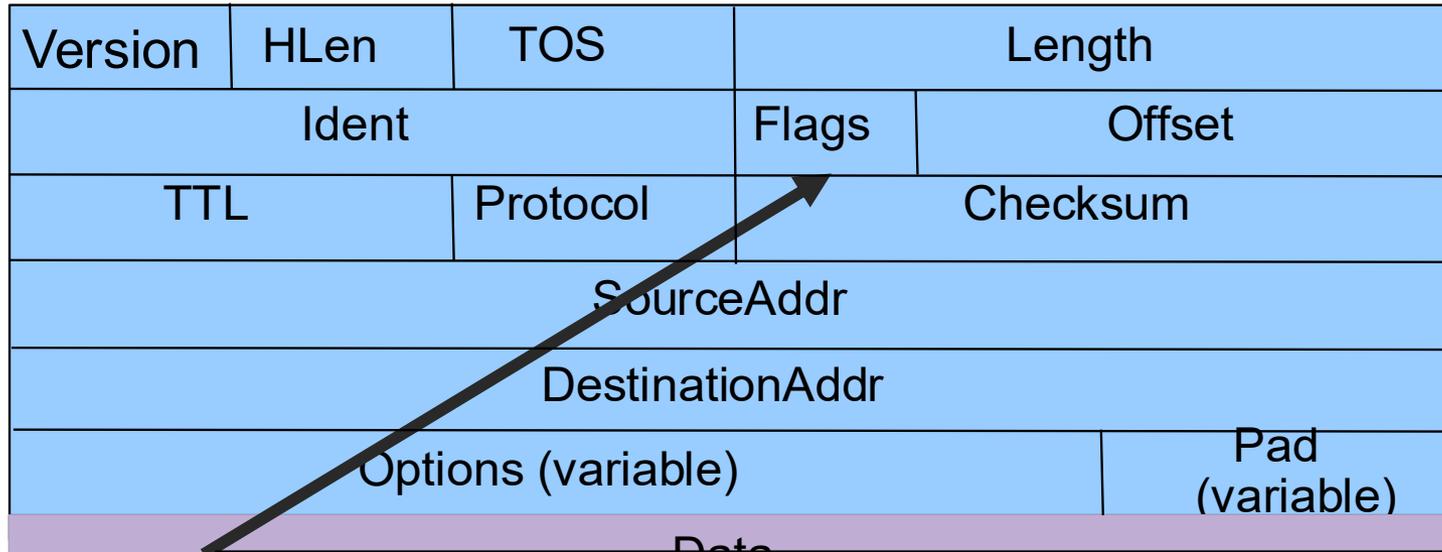| 0 | 4 | 8 | 16 | 19 | 31 |

Version	HLen	TOS	Length		
Ident			Flags	Offset	
TTL		Protocol		Checksum	
SourceAddr					
DestinationAddr					
Options (variable)				Pad (variable)	
Data					

3-bit flags
R = Reserved (unused)
DF = Don't fragment (if set, router returns
      ICMP if MTU too small)
MF = More fragments (1 indicates datagram has
additional fragments on the way, 0 indicates this is the
last fragment)

IP Packet Format

0	4	8	16	19	31

Version	HLen	TOS	Length		
Ident			Flags	Offset	
TTL		Protocol	Checksum		
SourceAddr					
DestinationAddr					
Options (variable)				Pad (variable)	
Data					

13-bit fragment offset into packet (Fragmentation) Indicates where in packet this fragment belongs (first fragment has offset zero, fragment starting at byte X has offset of X/8) Counted in 8-byte words

IP Packet Format

0	4	8	16	19	31

Version	HLen	TOS	Length		
Ident			Flags	Offset	
TTL		Protocol	Checksum		
SourceAddr					
DestinationAddr					
Options (variable)				Pad (variable)	

Data

8-bit time-to-live field (TTL)
  Initial value: 128 (Windows), 64 (Linux)
  Hop count decremented at each router
  Packet is discarded if TTL = 0

IP Packet Format

0	4	8		16	19	31

Version	HLen	TOS	Length		
Ident			Flags	Offset	
TTL		Protocol	Checksum		
SourceAddr					
DestinationAddr					
Options (variable)				Pad (variable)	

Data

8-bit protocol field (specifies encapsulated protocol)
TCP = 6, UDP = 17

IP Packet Format

0	4	8	16	19	31

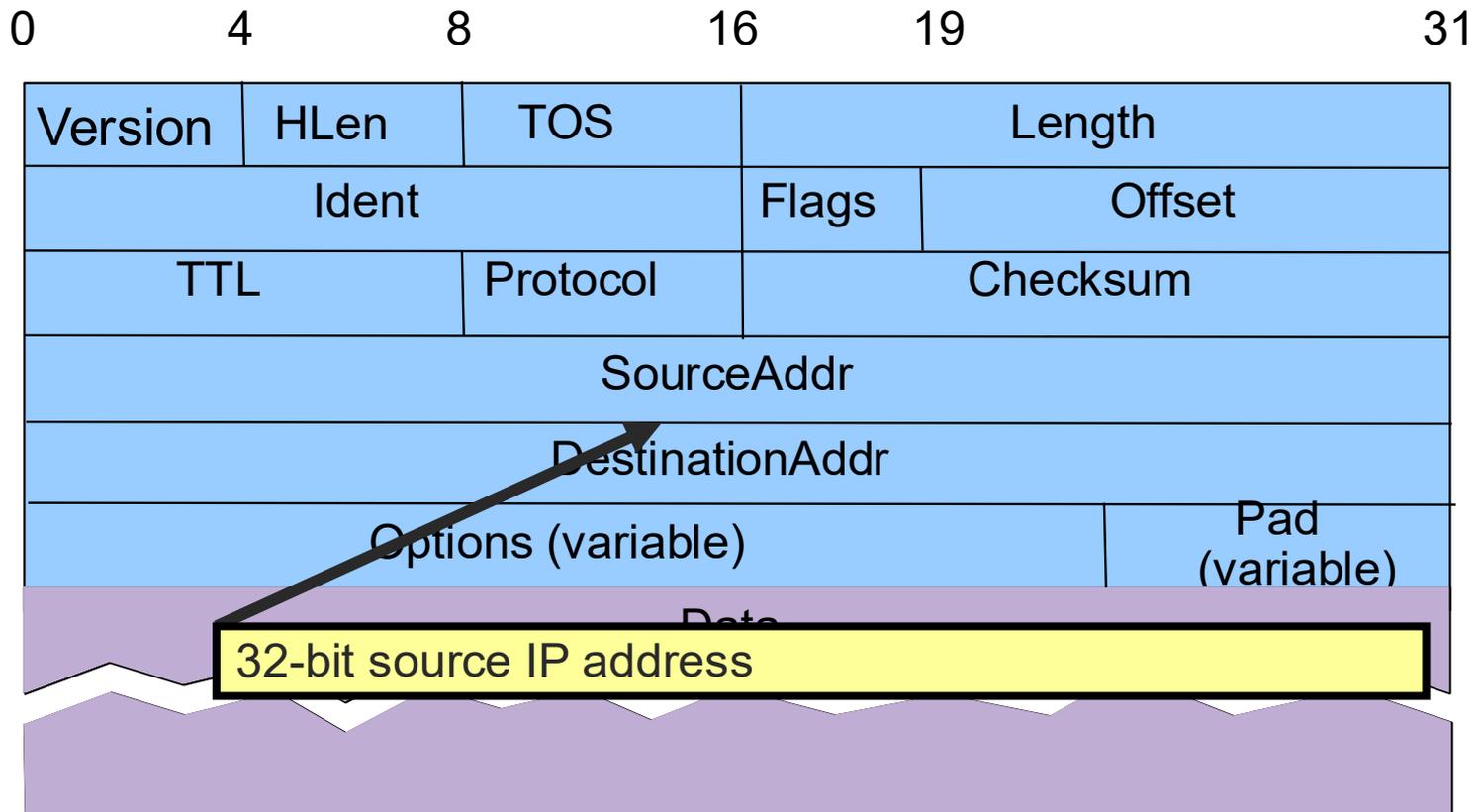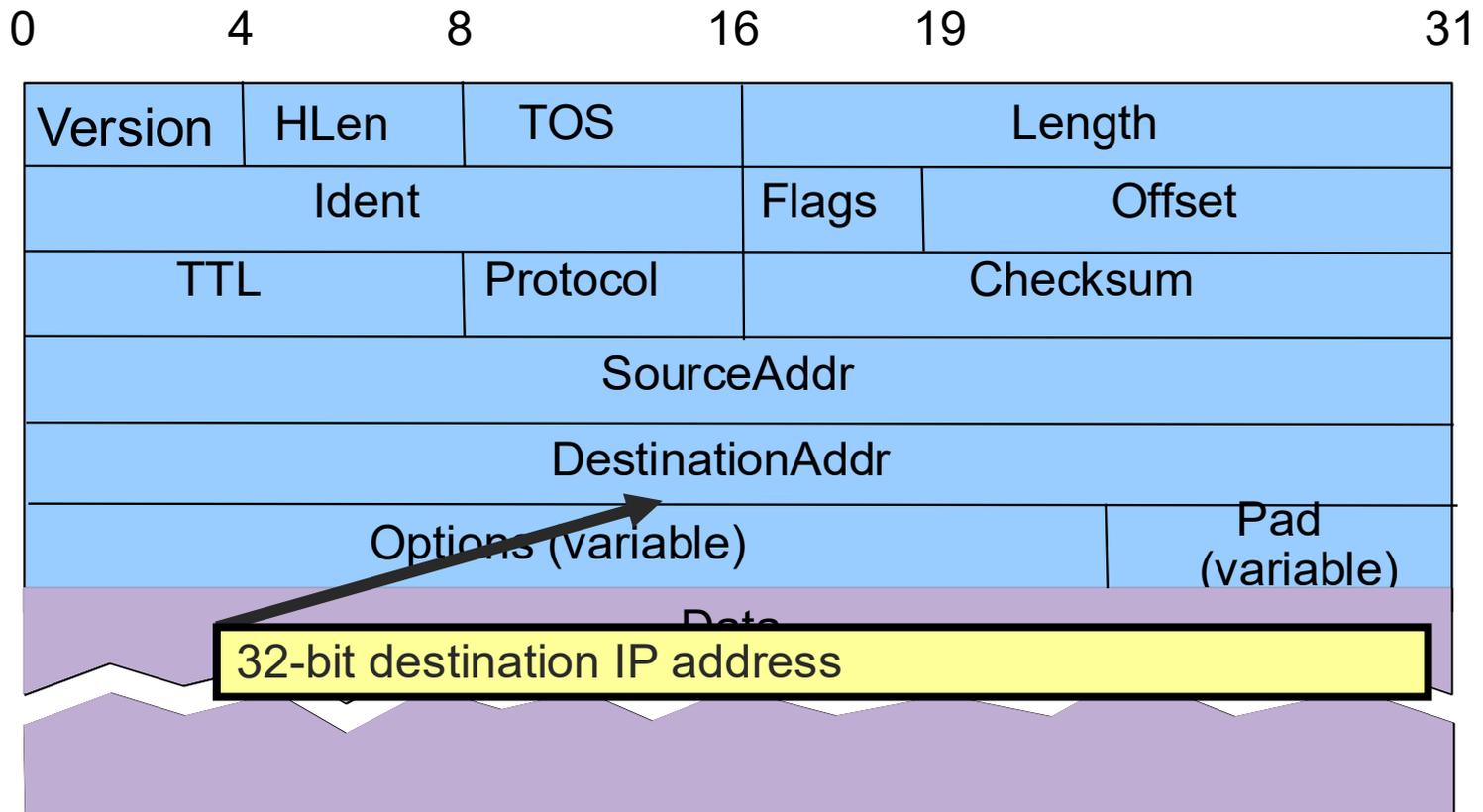| Version | HLen | TOS | Length | | |
Ident			Flags	Offset	
TTL		Protocol	Checksum		
SourceAddr					
DestinationAddr					
Options (variable)				Pad (variable)	
Data					

16-bit IP checksum on header

Since some header fields change at each router, this is recomputed at each router
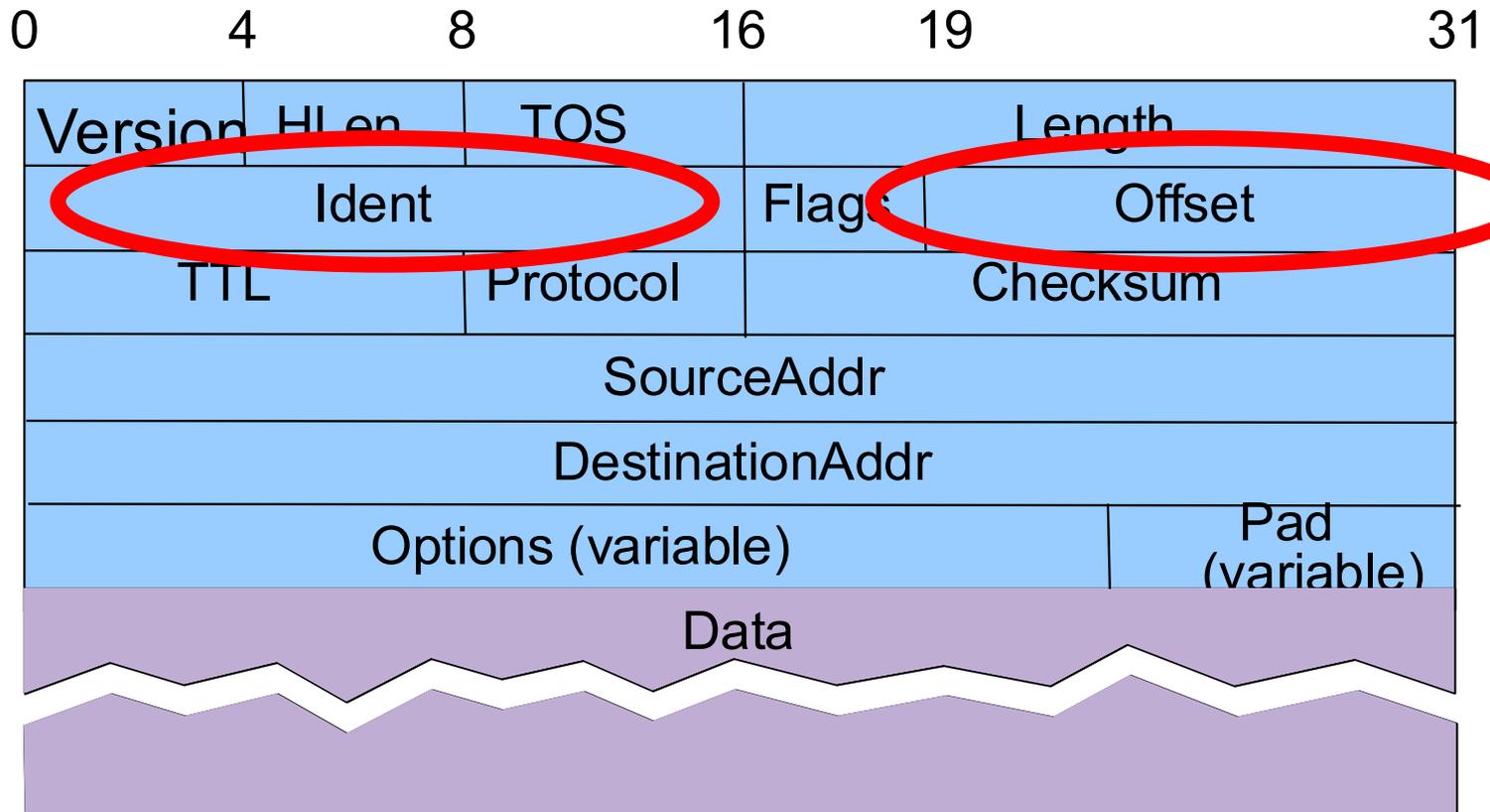
IP Packet Format

| 0 | 4 | 8 | 16 | 19 | 31 |

Version	HLen	TOS	Length		
Ident			Flags	Offset	
TTL		Protocol	Checksum		
SourceAddr					
DestinationAddr					
Options (variable)				Pad (variable)	
Data					

32-bit source IP address

IP Packet Format

0	4	8	16	19	31

Version	HLen	TOS	Length		
Ident			Flags	Offset	
TTL		Protocol	Checksum		
SourceAddr					
DestinationAddr					
Options (variable)				Pad (variable)	
Data					

32-bit destination IP address

IP Packet Format

- **Options**
  - Variable size
  - Examples:
    - Source-based routing
    - Record route
- **Padding**
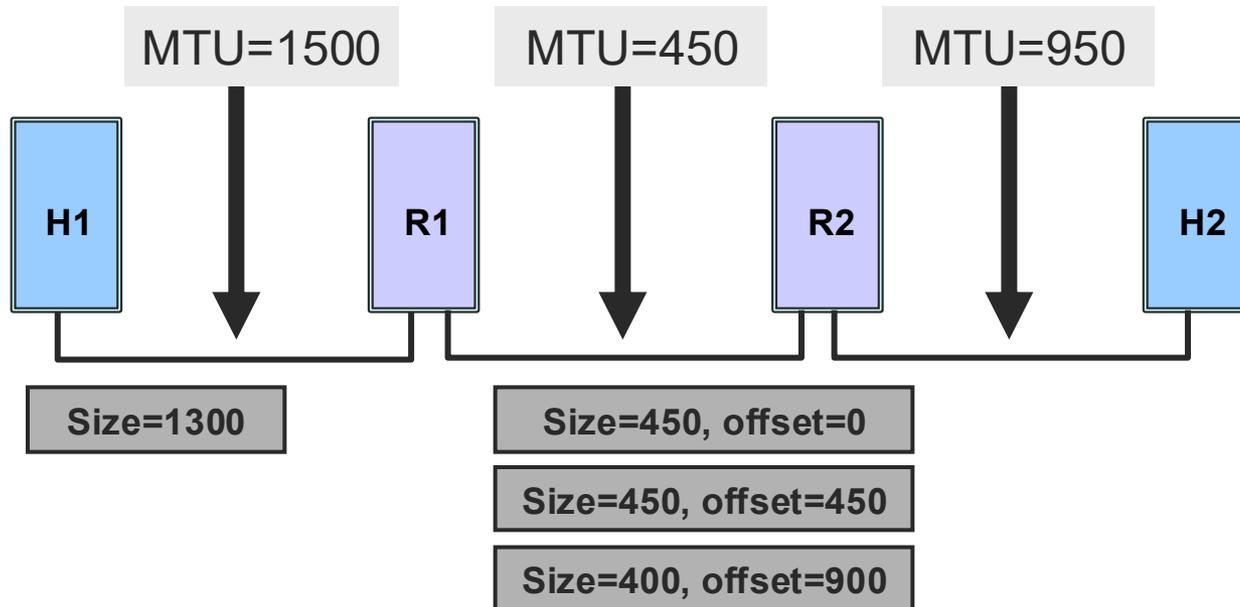 - Fill to 32-bit boundaries

IP Packet Format

0	4	8	16	19	31
Version	HLen	TOS		Length	
	Ident		Flags	Offset	
TTL		Protocol		Checksum	
SourceAddr					
DestinationAddr					
Options (variable)				Pad (variable)	
Data					

IP Packet Size

- **Problem**
  - ○ Different physical layers provide different limits on frame length
    - Maximum transmission unit (MTU)
  - ○ Source host does not know minimum value
    - Especially along dynamic routes
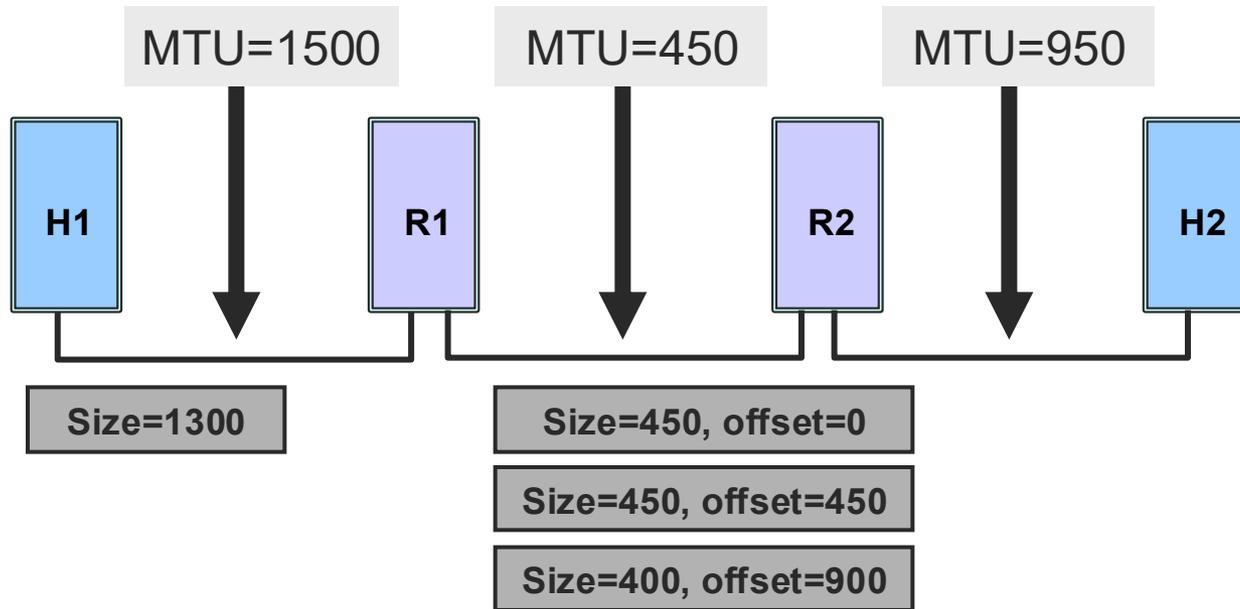
IP Fragmentation and Reassembly

MTU=1500 MTU=450 MTU=950

H1 R1 R2 H2

Size=1300

Size=450, offset=0

Size=450, offset=450

Size=400, offset=900

- Solution
  - When necessary, split IP packet into acceptably sized packets prior to sending over physical link

IP Fragmentation and Reassembly

MTU=1500   MTU=450   MTU=950

H1   R1   R2   H2

Size=1300

Size=450, offset=0

Size=450, offset=450

Size=400, offset=900

- Questions
  - Where should reassembly occur?
  - What happens when a fragment is damaged/lost?

IP Fragmentation and Reassembly

- **Fragments**
  - self-contained IP datagrams
- **Reassemble at destination**
  - Minimizes refragmentation
- **If one or more fragments are lost**
  - Drop all fragments in packet
- **Avoid fragmentation at source host**
  - Transport layer should send packets small enough to fit into one MTU of local physical network
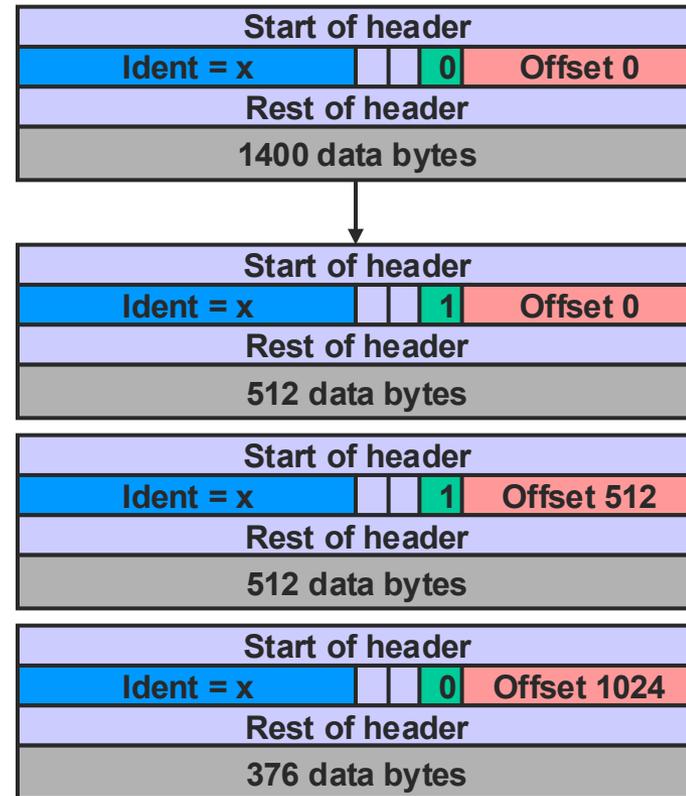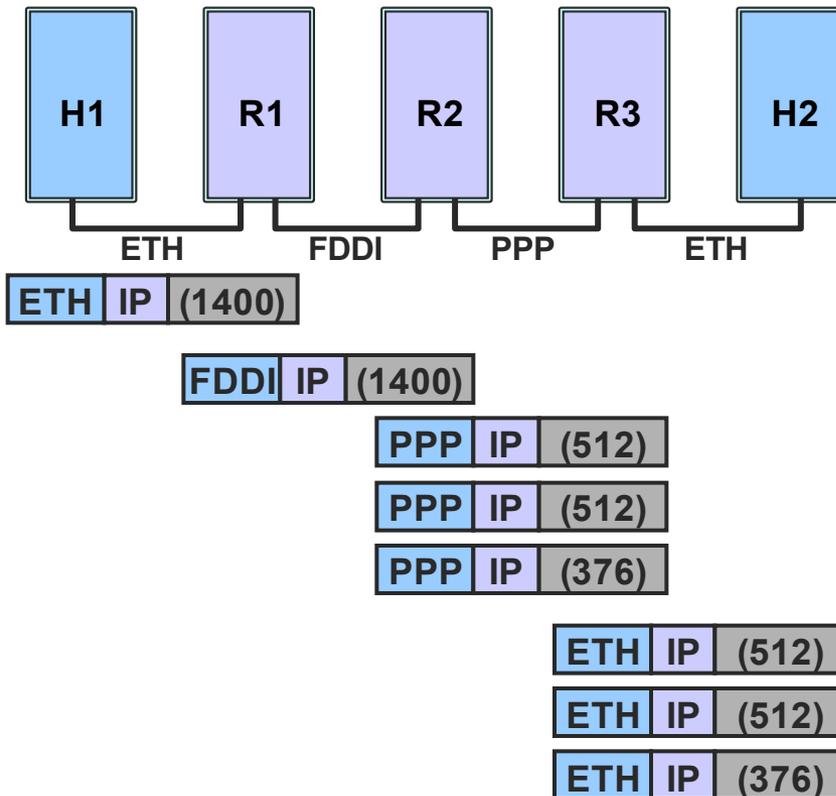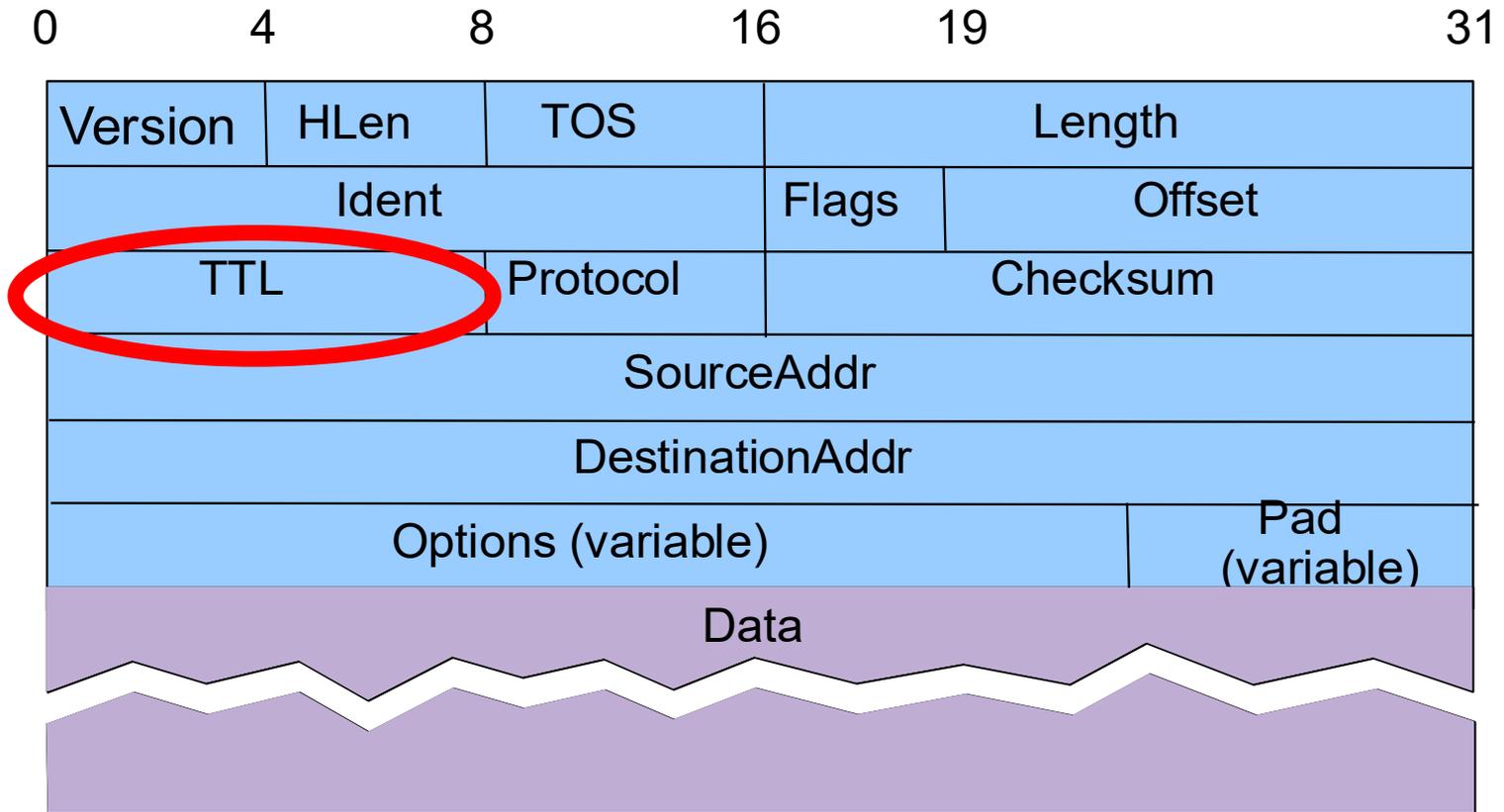 - Must consider IP header

Path MTU discovery

- Set "don't fragment" bit in IP header
  - Size is MTU of first hop
- Interface with too-small MTU
  - Responds back with "ICMP" message
  - Unfortunately, many networks drop ICMP traffic
- Reduce packet size
  - Repeat until smallest MTU on path discovered
- Binary search
  - Better yet: note there are small number of MTUs in the Internet

IP Fragmentation and Reassembly

IP Packet Format

0	4	8	16	19	31
Version	HLen	TOS	Length		
Ident			Flags	Offset	
TTL		Protocol	Checksum		
SourceAddr					
DestinationAddr					
Options (variable)				Pad (variable)	
Data					

Time to Live (TTL) field

- Problem: forwarding loop
  - My FIB says to forward to you, yours says to forward to me
    - Or much more complicated scenarios with longer loops
  - Packets circle forever
 - After a few packets, 100% of capacity taken up by cycling packets

Time to Live (TTL) field

- Solution: TTL field
  - Hope we never have to use it…
  - Begins at source at some value (e.g., 255)
  - Decremented at each hop
  - Dropped at any hop where TTL = 0
    - "Time exceeded" message returned to source via ICMP
    - Traceroute is a trick based on this

Internet Control Message Protocol (ICMP)

- **IP companion protocol**
  - Handles error and control messages
  - Used for troubleshooting and measurement
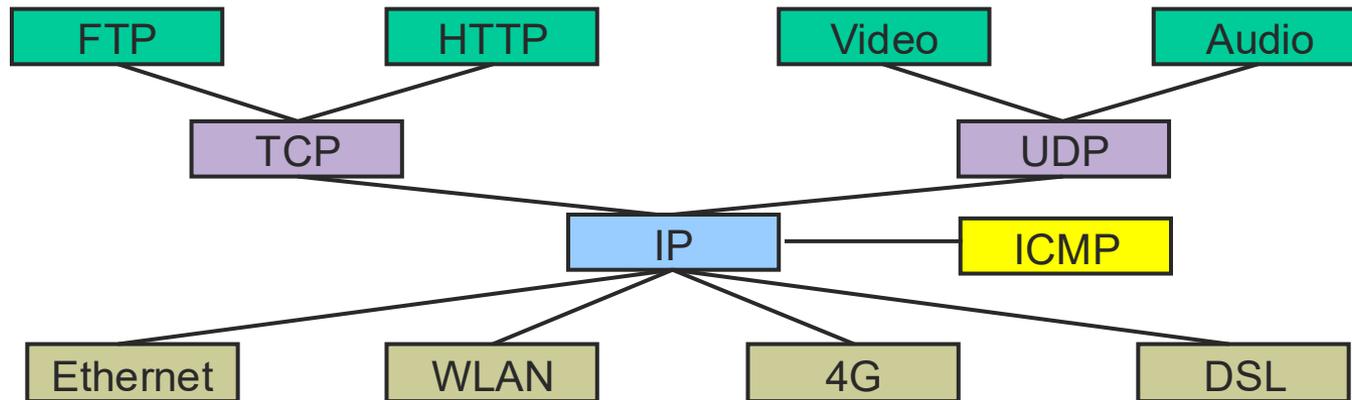
ICMP

- **Use**
  - Pings (probing remote hosts)
  - Traceroutes (learning set of routers along a path)
  - etc.

- **Control Messages**
  - Echo/ping request and reply
  - Echo/ping request and reply with timestamps
 - Route redirect

- **Error Messages**
  - Host unreachable
  - Reassembly failed
  - IP checksum failed
  - TTL exceeded (packet dropped)
  - Invalid header

IPv4 Address Translation support

- **IP addresses** to **LAN physical addresses**
- Problem
  - An IP route can pass through many physical networks
  - Data must be delivered to destination's physical network
  - Hosts only listen for packets marked with physical interface names
    - Each hop along route
    - Destination host
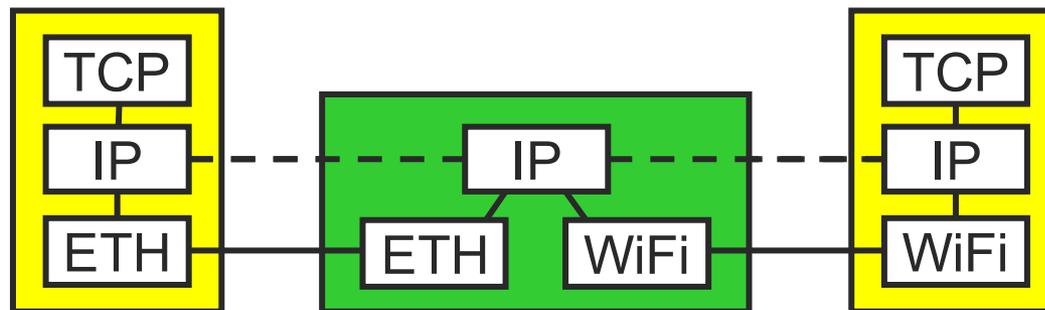
IP to Physical Address Translation

- **Hard-coded**
  - Encode physical address in IP address
  - Ex: Map Ethernet addresses to IP addresses
    - Makes it impossible to associate address with topology
- **Fixed table**
  - Maintain a central repository and distribute to hosts
    - Bottleneck for queries and updates
- **Automatically generated table**
  - Use ARP to build table at each host
  - Use timeouts to clean up table

ARP: Address Resolution Protocol

- Check ARP table for physical address
- If address not present
  - Broadcast an ARP query, include querying host's translation
  - Wait for an ARP response
- Upon receipt of ARP query/response
  - Targeted host responds with address translation
  - If address already present
    - Refresh entry and reset timeout
  - If address not present
    - Add entry for requesting host
    - Ignore for other hosts
- Timeout and discard entries after O(10) minutes

ARP Packet

0	8	16	31
Hardware type = 1		ProtocolType = 0x0800	
HLEN = 48	PLEN = 32	Operation	
SourceHardwareAddr (bytes 0 – 3)			
SourceHardwareAddr (bytes 4 – 5)		SourceProtocolAddr (bytes 0 – 1)	
SourceProtocolAddr (bytes 2 – 3)		TargetHardwareAddr (bytes 0 – 1)	
TargetHardwareAddr (bytes 2 – 5)			
TargetProtocolAddr (bytes 0 – 3)			

ARP

IP	MAC
4.4.4.4	CC:CC:CC:CC:CC

Broadcast ARP request:
"Who owns IP address 4.4.4.4?"
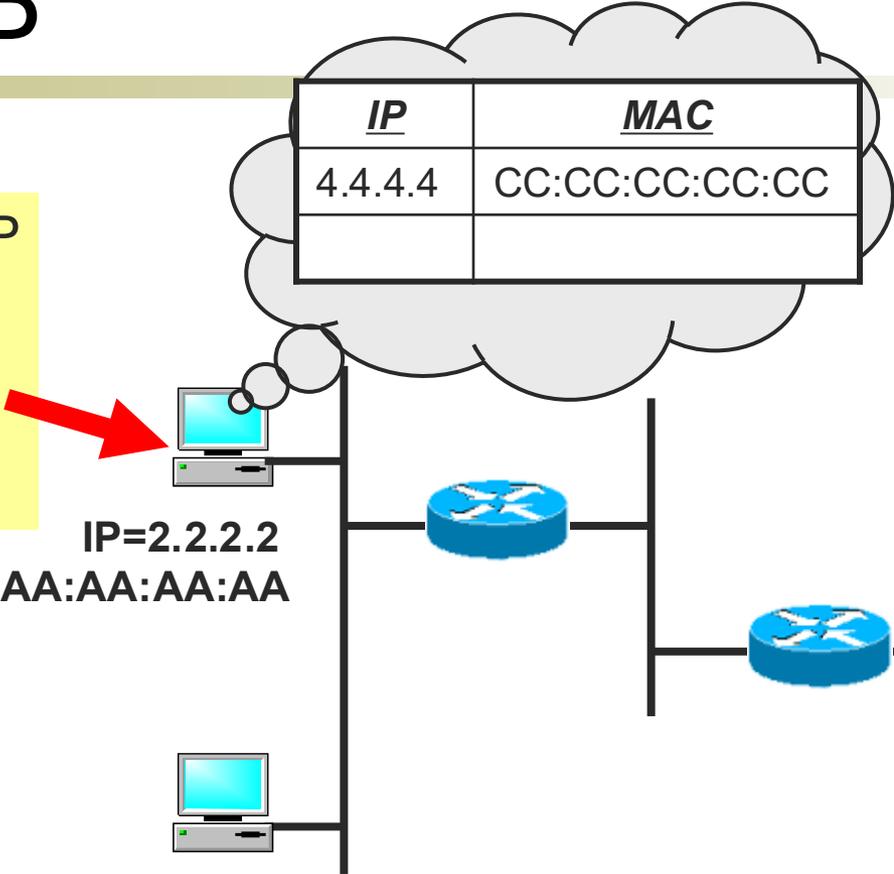
Broadcast ARP reply: "I own 4.4.4.4, and my MAC address is CC:CC:CC:CC:CC"

IP=2.2.2.2
MAC=AA:AA:AA:AA:AA

IP=4.4.4.4
MAC=CC:CC:CC:CC:CC

IP=3.3.3.3
MAC=BB:BB:BB:BB:BB

- ARP: determines mapping from IP to MAC address

ARP

- What if IP address is not on subnet?
  - Each host configured with "default gateway"
  - Use ARP to resolve its IP address
- Gratuitous ARP: tell network your IP to MAC mapping
  - Used to detect IP conflicts, IP address changes; update other machines' ARP tables, update bridges' learned information

ARP

IP	*MAC*
4.4.4.4	CC:CC:CC:CC:CC
5.5.5.5	DD:DD:DD:DD:DD

IP=2.2.2.2
MAC=AA:AA:AA:AA:AA

IP=4.4.4.4
MAC=CC:CC:CC:CC:CC

Broadcast *Gratuitous* ARP reply:
"I own 5.5.5.5, and my MAC address is DD:DD:DD:DD:DD"

IP=3.3.3.3
MAC=BB:BB:BB:BB:BB

IP=5.5.5.5
MAC=DD:DD:DD:DD:DD

Proxy ARP

IP	MAC
4.4.4.4	DD:DD:DD:DD:DD

IP=2.2.2.2
MAC=AA:AA:AA:AA:AA

IP=4.4.4.4
MAC=CC:CC:CC:CC:CC

Broadcast *proxy* ARP reply:
"I own 4.4.4.4, and my MAC address is DD:DD:DD:DD:DD"

IP=3.3.3.3
MAC=BB:BB:BB:BB:BB

IP=5.5.5.5
MAC=DD:DD:DD:DD:DD
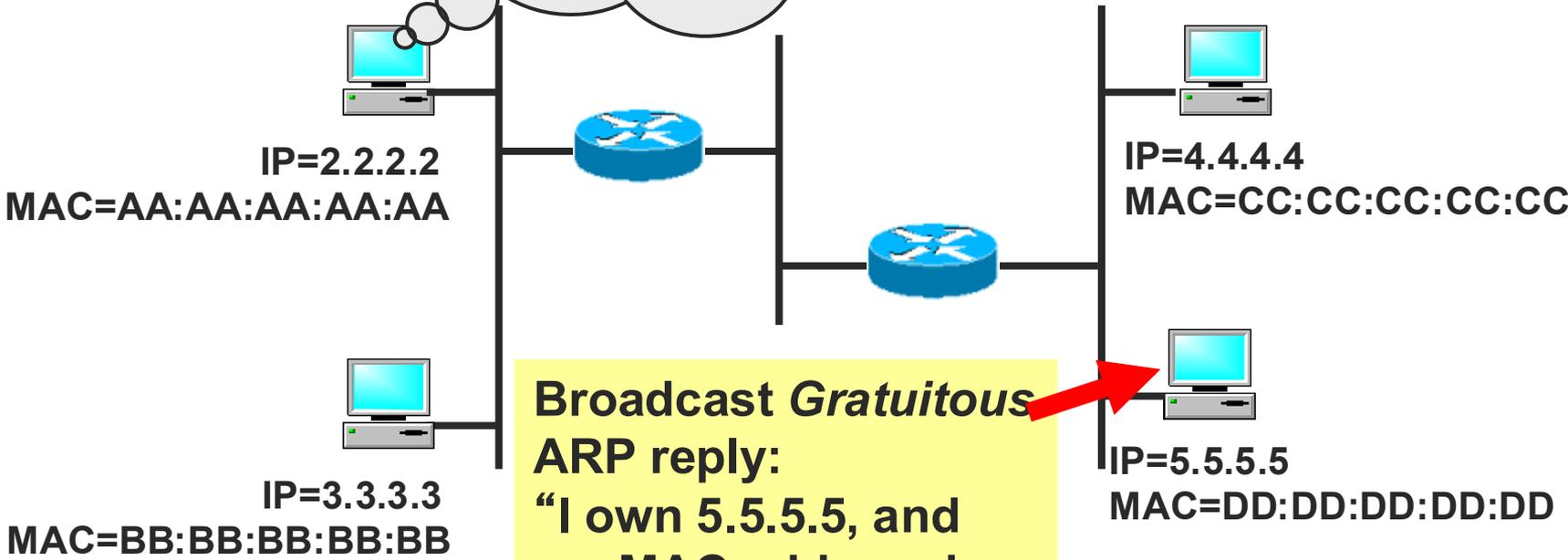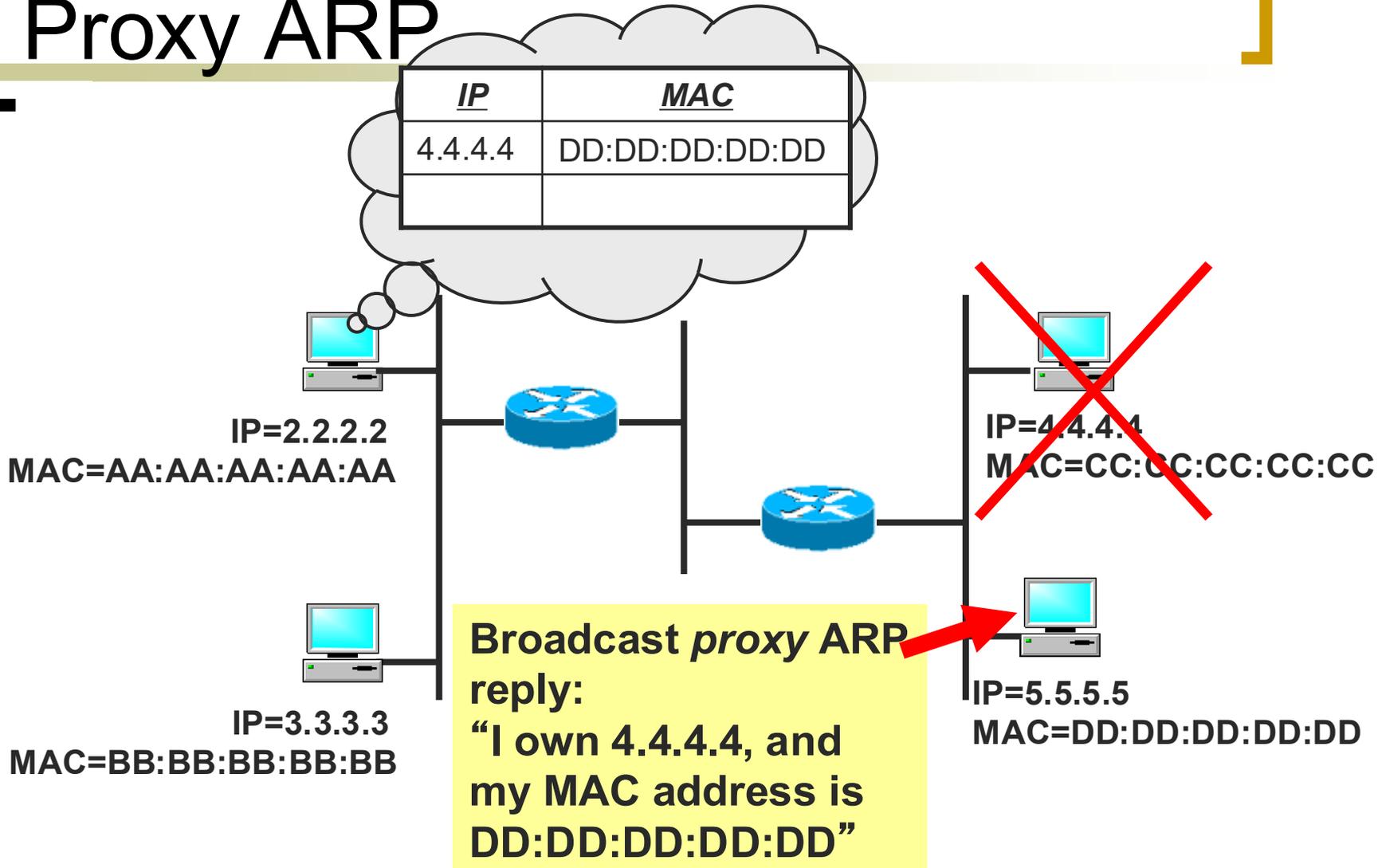
Host Configuration

- **Plug new host into network**
  - How much information must be known?
  - What new information must be assigned?
  - How can process be automated?
- **Some answers**
  - Host needs an IP address (must know it)
  - Host must also
    - Send packets out of physical (direct) network
    - Thus needs physical address of router

Host Configuration

- **Reverse Address Resolution Protocol (RARP)**
  - Translate physical address to IP address
  - Used to boot diskless hosts
  - Host broadcasts request to boot
  - RARP server tells host the host's own IP address
- **Boot protocol (BOOTP)**
  - Use UDP packets for same purpose as RARP
  - Allows boot requests to traverse routers
  - IP address of BOOTP server must be known
  - Also returns file server IP, subnet mask, and default router for host

Dynamic Host Configuration Protocol (DHCP)

- A simple way to automate configuration information
  - Network administrator does not need to enter host IP address by hand
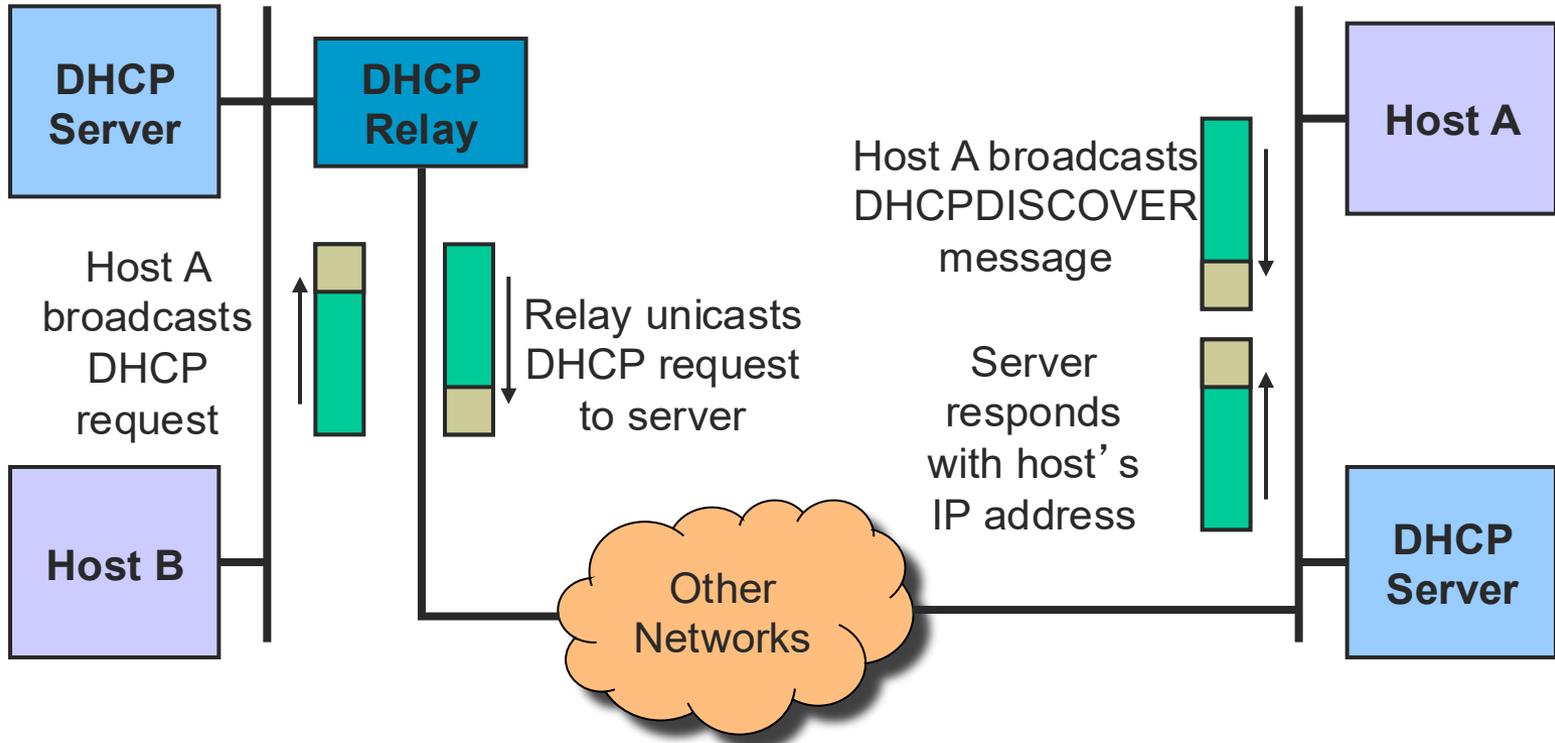 - Good for large and/or dynamic networks

Dynamic Host Configuration Protocol (DHCP)

- **New machine sends request to DHCP server for assignment and information**
- **Server receives**
  - Directly
    - If new machine given server's IP address
  - Through broadcast
    - If on same physical network
    - Via DHCP relay nodes
      - Forward requests onto the server's physical network
- **Server assigns IP address and provides other info**
- **Can be made secure**
  - Present signed request or just a "valid" physical address

DHCP

DHCP Server

DHCP Relay

Host A

Host A broadcasts DHCP request

Relay unicasts DHCP request to server

Host A broadcasts DHCPDISCOVER message

Server responds with host's IP address

Host B

Other Networks

DHCP Server

DHCP

- Remaining challenge: configuring DHCP servers
  - Need to ensure consistency across servers, between servers and network, address assignment across routers
  - But simpler than directly managing end hosts

Virtual Private Networks

- Goal
  - Controlled connectivity
    - Restrict forwarding to authorized hosts
  - Controlled capacity
    - Change router drop and priority policies
    - provide guarantees on bandwidth, delay, etc.
- Virtual Private Network
  - A group of connected subnets
  - Connections may be over shared network
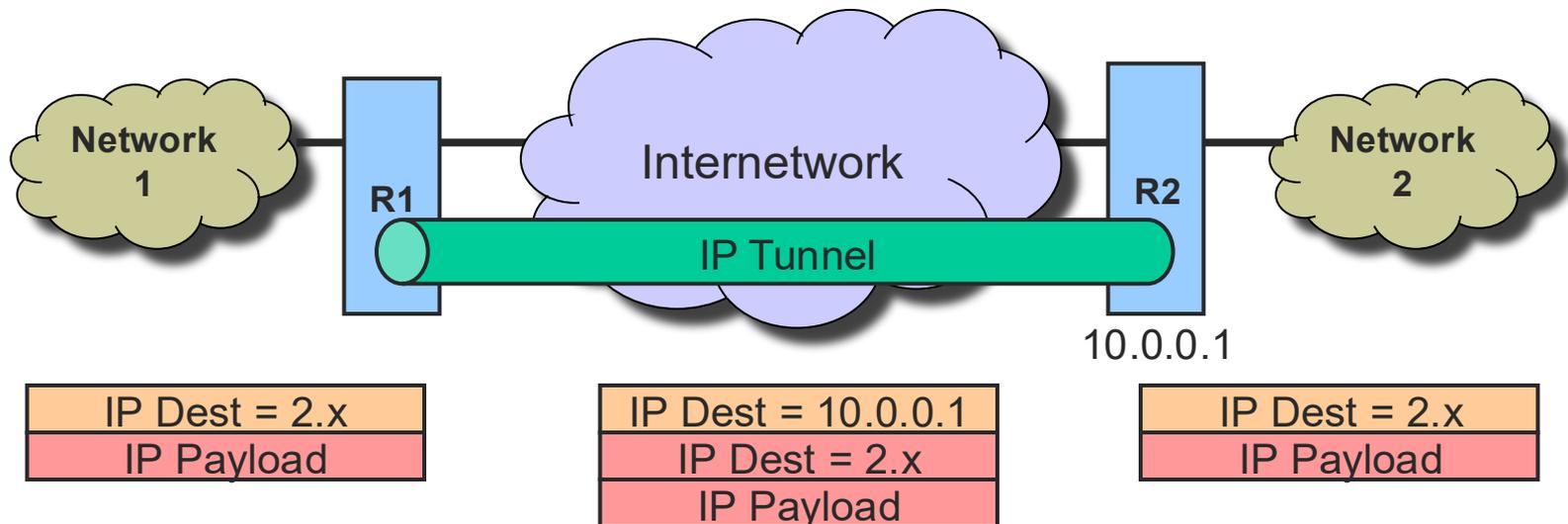 - Similar to LANE, but over IP allowing the use of heterogeneous internets

Virtual Private Networks
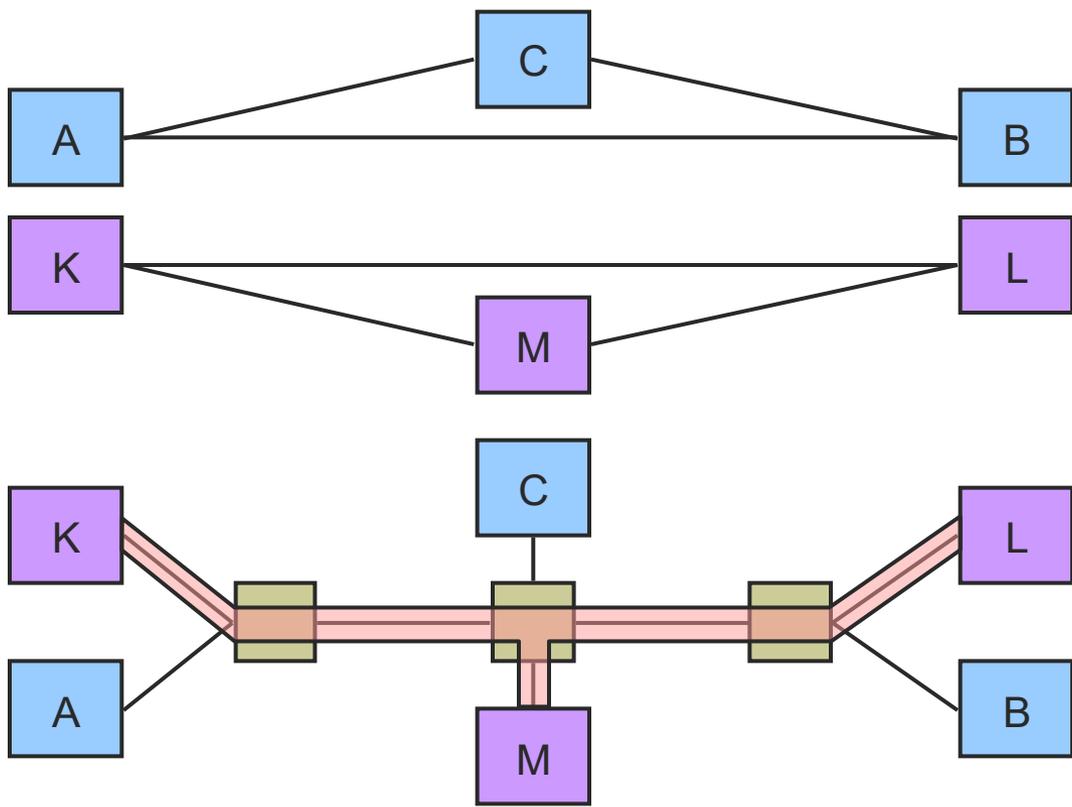
Tunneling

- **IP Tunnel**
  - Virtual point-to-point link between an arbitrarily connected pair of nodes

IP Dest = 2.x		IP Dest = 10.0.0.1		IP Dest = 2.x
IP Payload		IP Dest = 2.x		IP Payload
		IP Payload		

Tunneling

- **Advantages**
  - Transparent transmission of packets over a heterogeneous network
  - Only need to change relevant routers

- **Disadvantages**
  - Increases packet size
  - Processing time needed to encapsulate and unencapsulate packets
  - Management at tunnel-aware routers