

EECS 489

Computer Networks

Winter 2025

Mosharaf Chowdhury

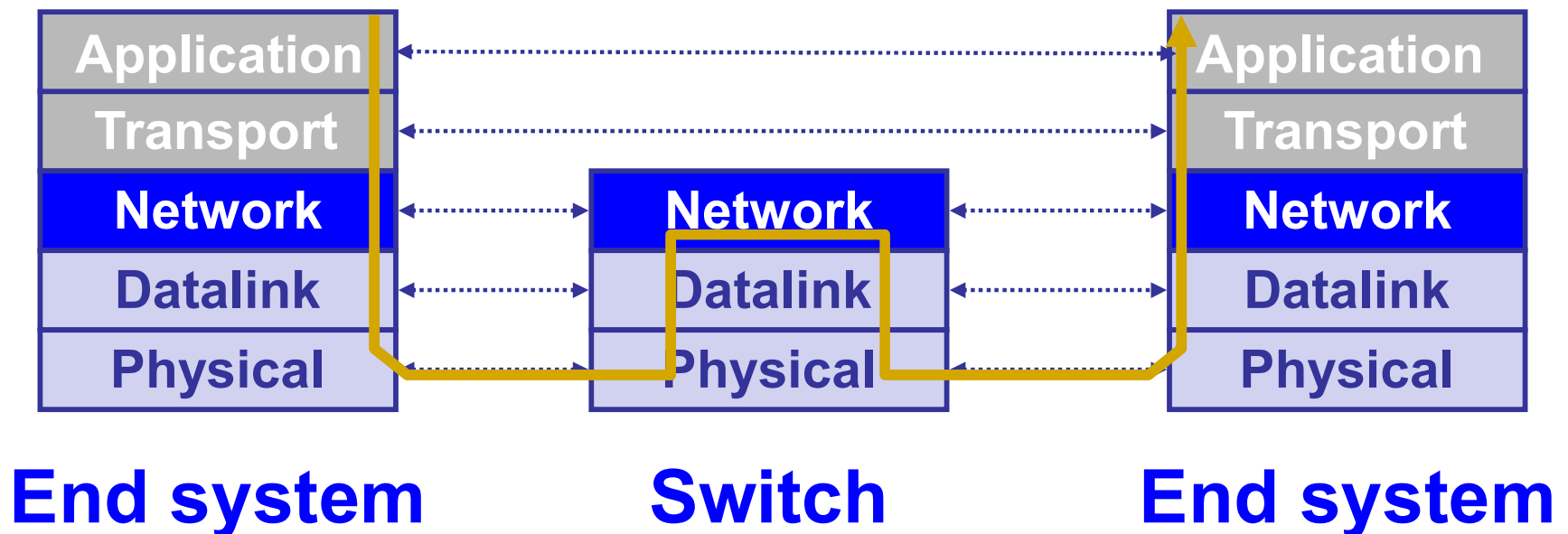
Material with thanks to Aditya Akella, Sugih Jamin, Philip Levis, Sylvia Ratnasamy, Peter Steenkiste, and many other colleagues.

Agenda

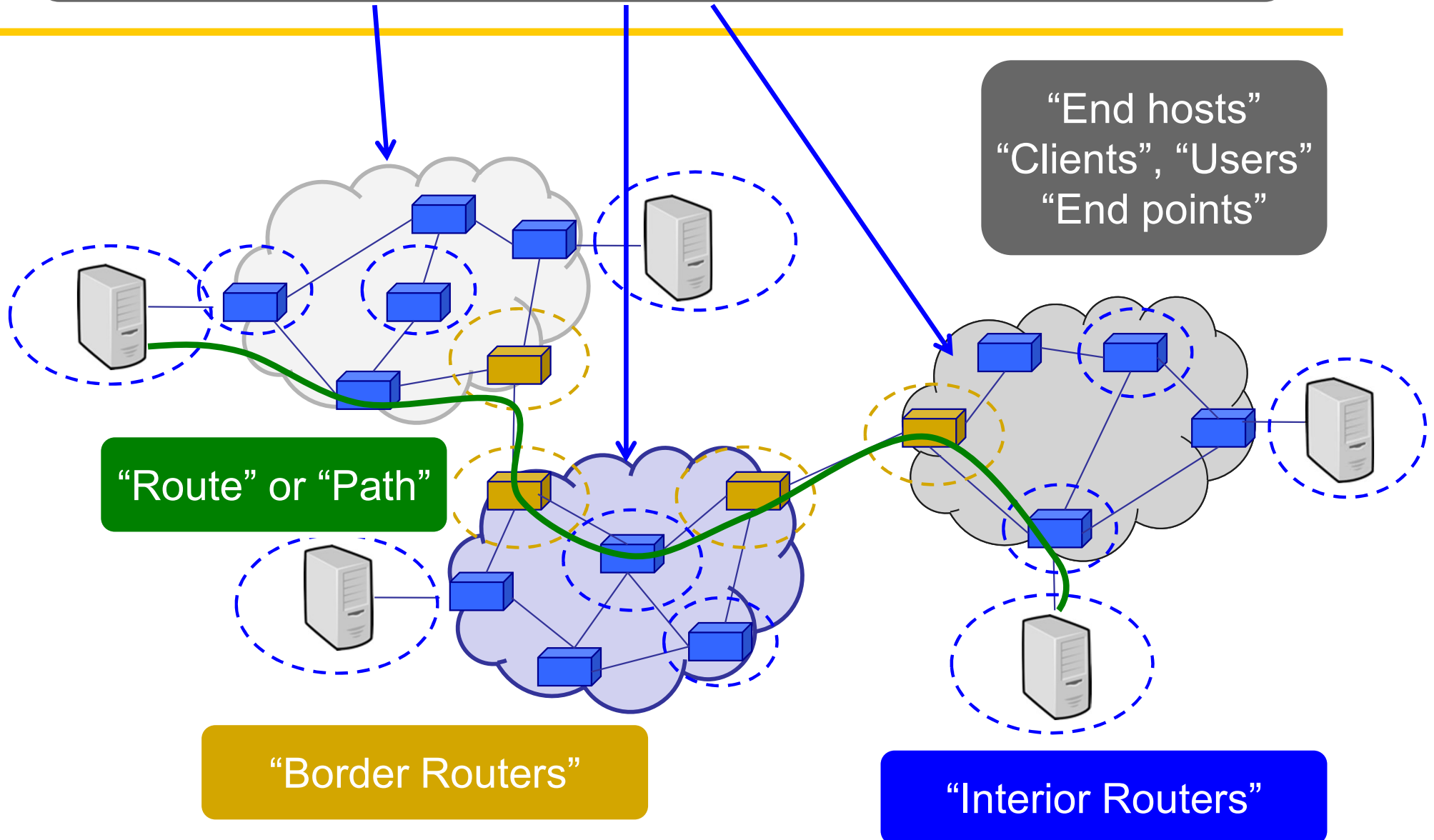
- Network layer basics
- The Internet Protocol (IP)

Network layer

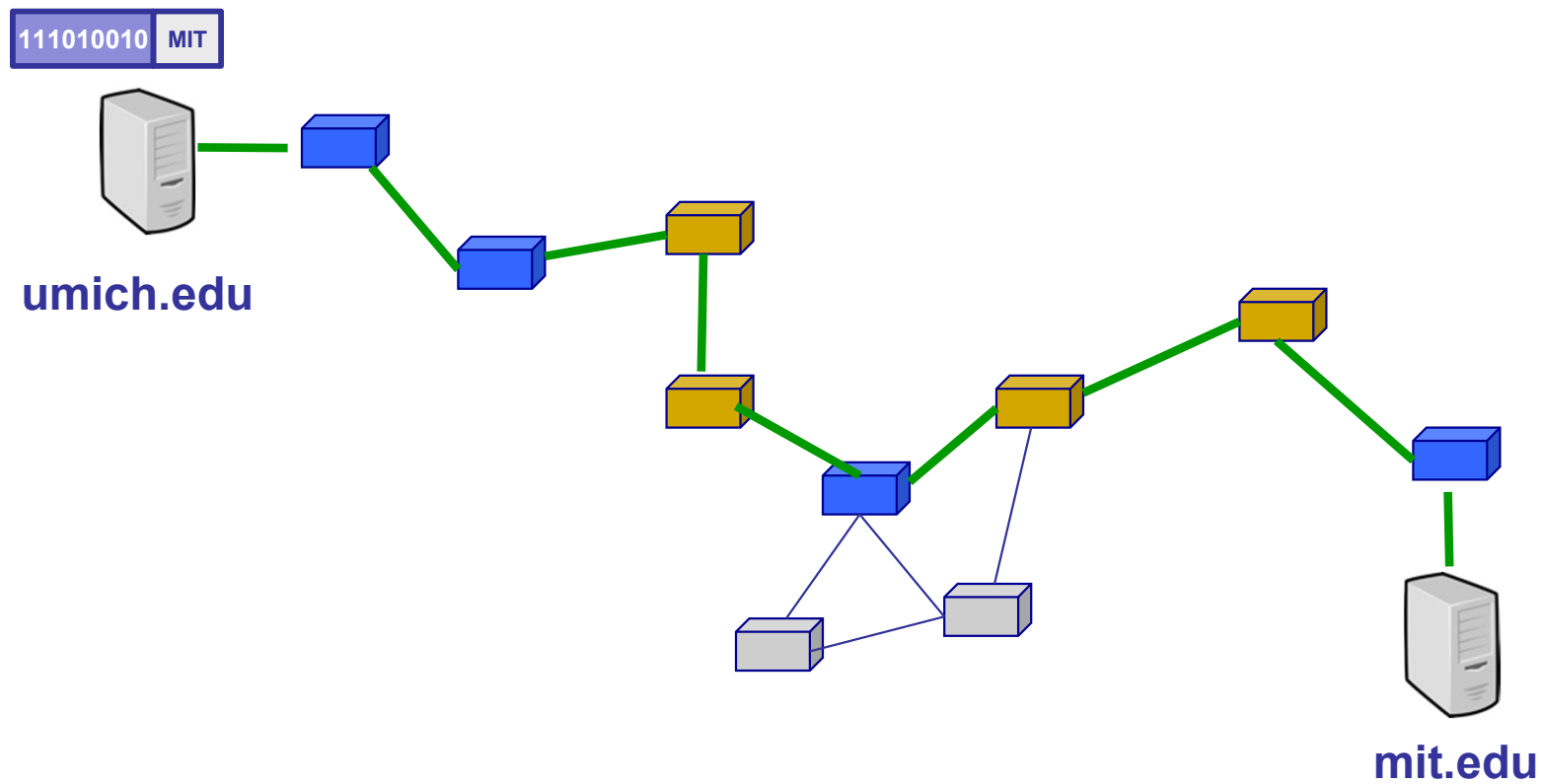
- Present everywhere
- Performs **addressing**, **forwarding**, and **routing**, among other tasks



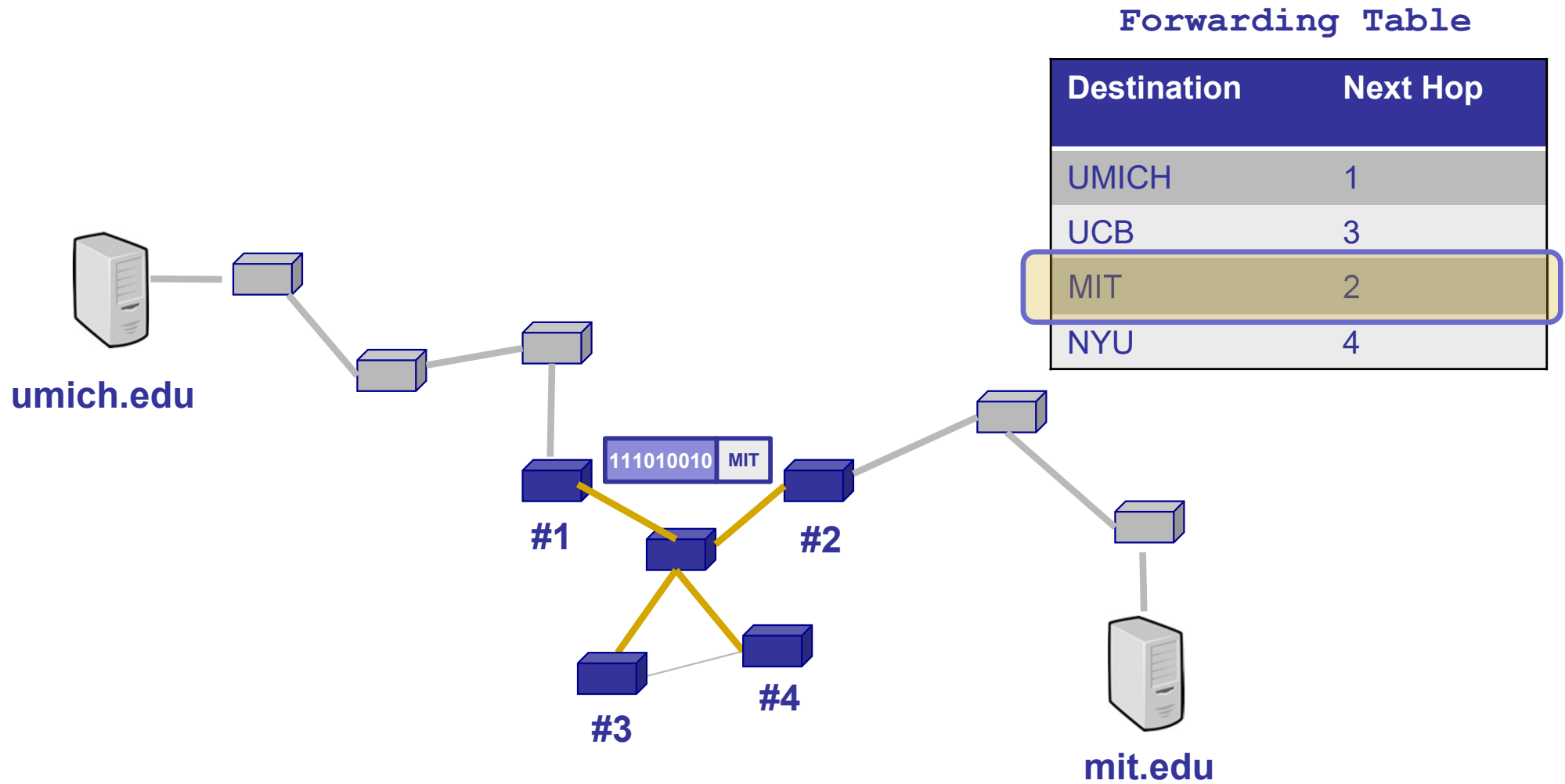
“Autonomous System (AS)” or “Domain” Region of a network under a single administrative entity



Forwarding



Forwarding



Forwarding

- ❑ Directing a packet to the correct interface so that it progresses to its destination
 - Local
- ❑ How?
 - Read address from packet header
 - Search forwarding table

Routing

- Setting up network-wide *forwarding tables* to enable end-to-end communication
 - Global
- How?
 - Using different routing protocols

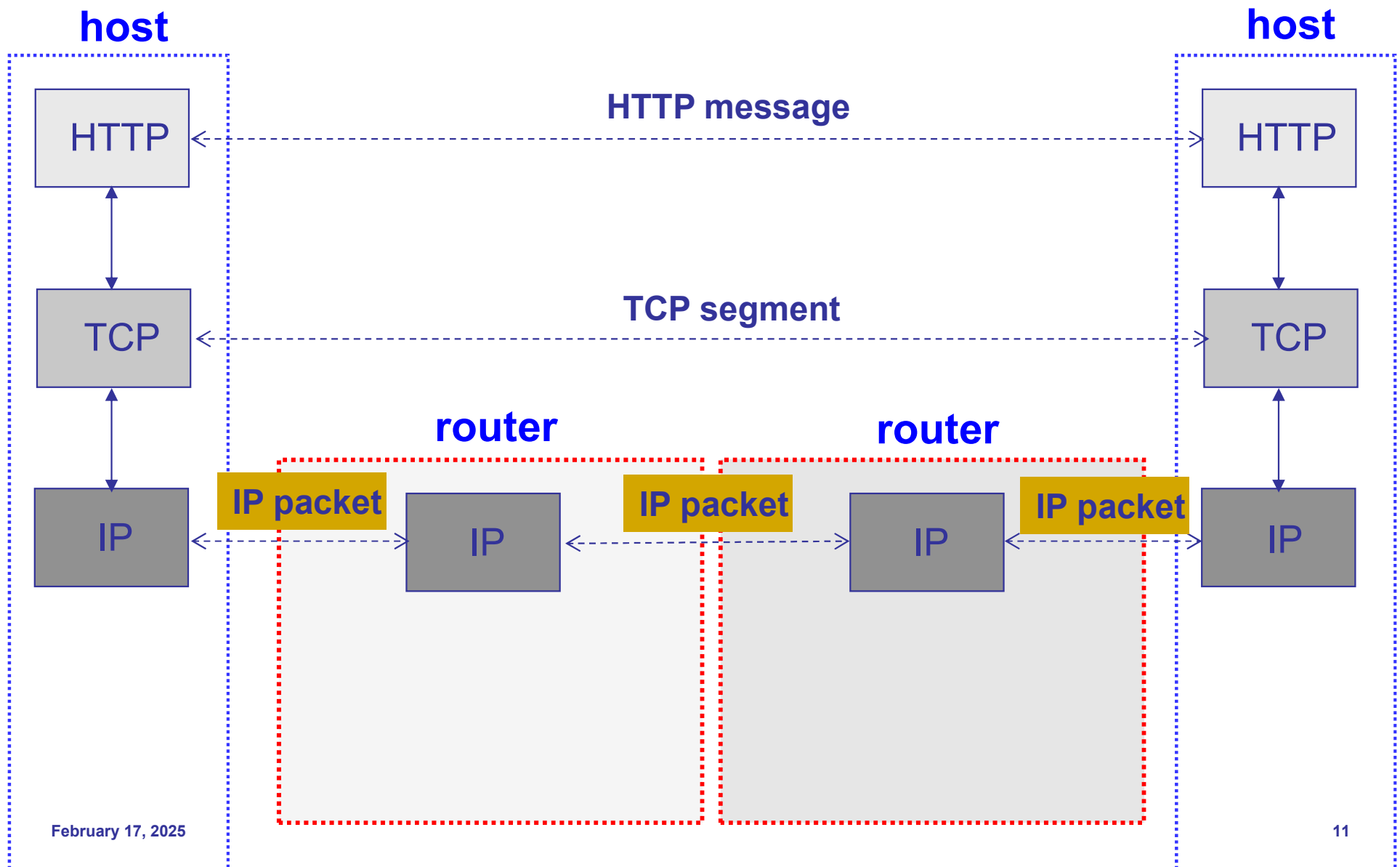
Forwarding vs. routing

- ❑ Forwarding: “data plane”
 - Directing one data packet
 - Each router using local routing state
- ❑ Routing: “control plane”
 - Computing the forwarding tables that guide packets
 - Jointly computed by routers using a distributed algorithm
- ❑ Very different timescales!

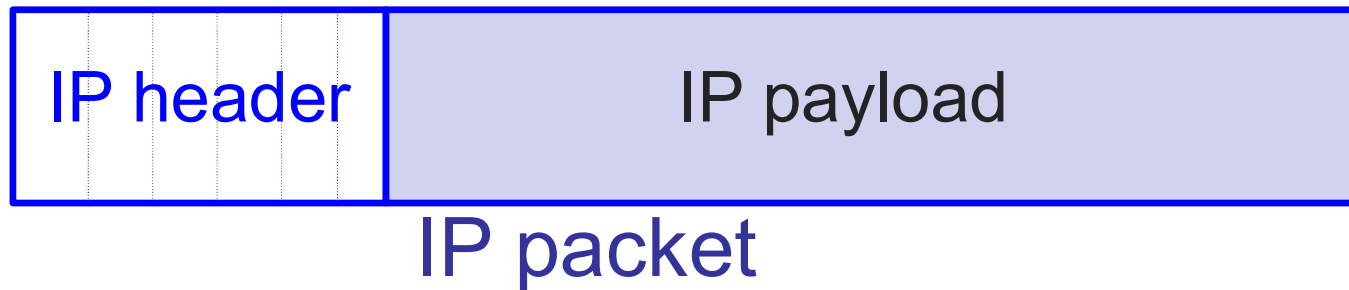
THE IP LAYER

Lecture 2:

Layer encapsulation



Recall: IP packet



- IP packet contains a header and payload
 - Payload is opaque to the network
 - Header is what we care about
 - First end-to-end layer (going bottom-up)

Designing the IP header

- Think of the IP header as an interface
 - Between the source and destination end-systems
 - Between the source and network (routers)
- Designing an interface
 - What task(s) are we trying to accomplish?
 - What information is needed to do it?
- Header reflects information needed for basic tasks

What are these tasks? (in network)

- ▣ Parse packet
- ▣ Carry packet to the destination
- ▣ Deal with problems along the way
 - Loops
 - Corruption
 - Packet too large
- ▣ Accommodate evolution
- ▣ Specify any special handling

What information do we need?

- ▣ Parse packet
- ▣ Carry packet to the destination
- ▣ Deal with problems along the way
 - Loops
 - Corruption
 - Packet too large
- ▣ Accommodate evolution
- ▣ Specify any special handling

What information do we need?

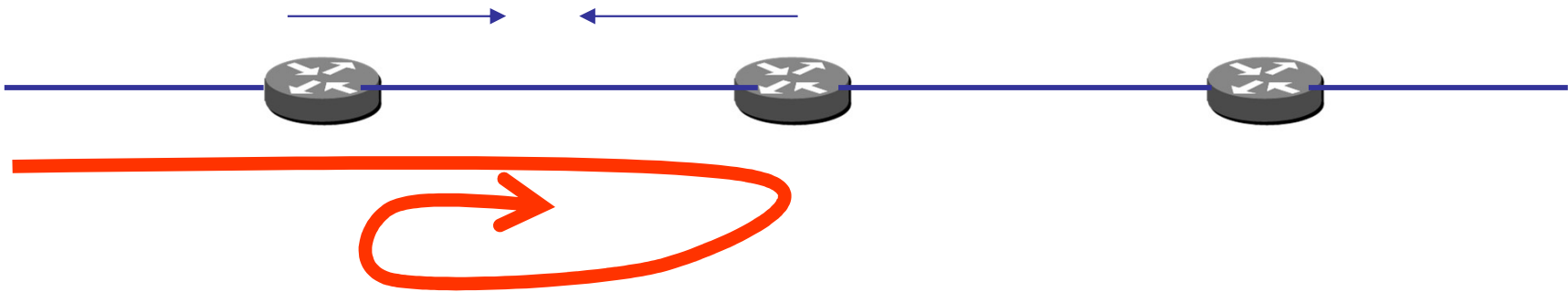
- ▣ Parse packet
 - IP version number (4 bits), packet length (16 bits)
- ▣ Carry packet to the destination
 - Destination's IP address (32 bits)
- ▣ Deal with problems along the way
 - Loops:
 - Corruption:
 - Packet too large:

What information do we need?

- ▣ Parse packet
 - IP version number (4 bits), packet length (16 bits)
- ▣ Carry packet to the destination
 - Destination's IP address (32 bits)
- ▣ Deal with problems along the way
 - Loops: TTL (8 bits)
 - Corruption: checksum (16 bits)
 - Packet too large: fragmentation fields (32 bits)

Preventing loops (TTL)

- ❑ Forwarding loops cause packets to cycle for a long time
 - Left unchecked would accumulate to consume all capacity



- ❑ Time-to-Live (TTL) Field (8 bits)
 - Decrement at each hop; packet discarded if 0
 - » “Time exceeded” message is sent to the source

Header corruption (Checksum)

- ❑ Checksum (16 bits)
 - Particular form of checksum over packet header
- ❑ If not correct, router discards packets
 - So it doesn't act on bogus information
- ❑ Checksum recalculated at every router
 - Why?

Fragmentation

- ❑ Every link has a “Maximum Transmission Unit” (MTU)
 - Largest number of bits it can carry as one unit
- ❑ A router can split a packet into multiple “fragments” if the packet size exceeds the link’s MTU
- ❑ Must reassemble to recover original packet
- ❑ Will return to fragmentation later today...

What information do we need?

- ❑ Parse packet
 - IP version number (4 bits), packet length (16 bits)
- ❑ Carry packet to the destination
 - Destination's IP address (32 bits)
- ❑ Deal with problems along the way
 - TTL (8 bits), checksum (16 bits), frag. (32 bits)
- ❑ Accommodate evolution
 - Version number (4 bits) (+ fields for special handling)
- ❑ Specify any special handling

Special handling

- “Type of Service” (8 bits)
 - Allow packets to be treated differently based on needs
 - » e.g., indicate priority, congestion notification
 - Has been redefined several times
 - Now called “Differentiated Services Code Point (DSCP)”

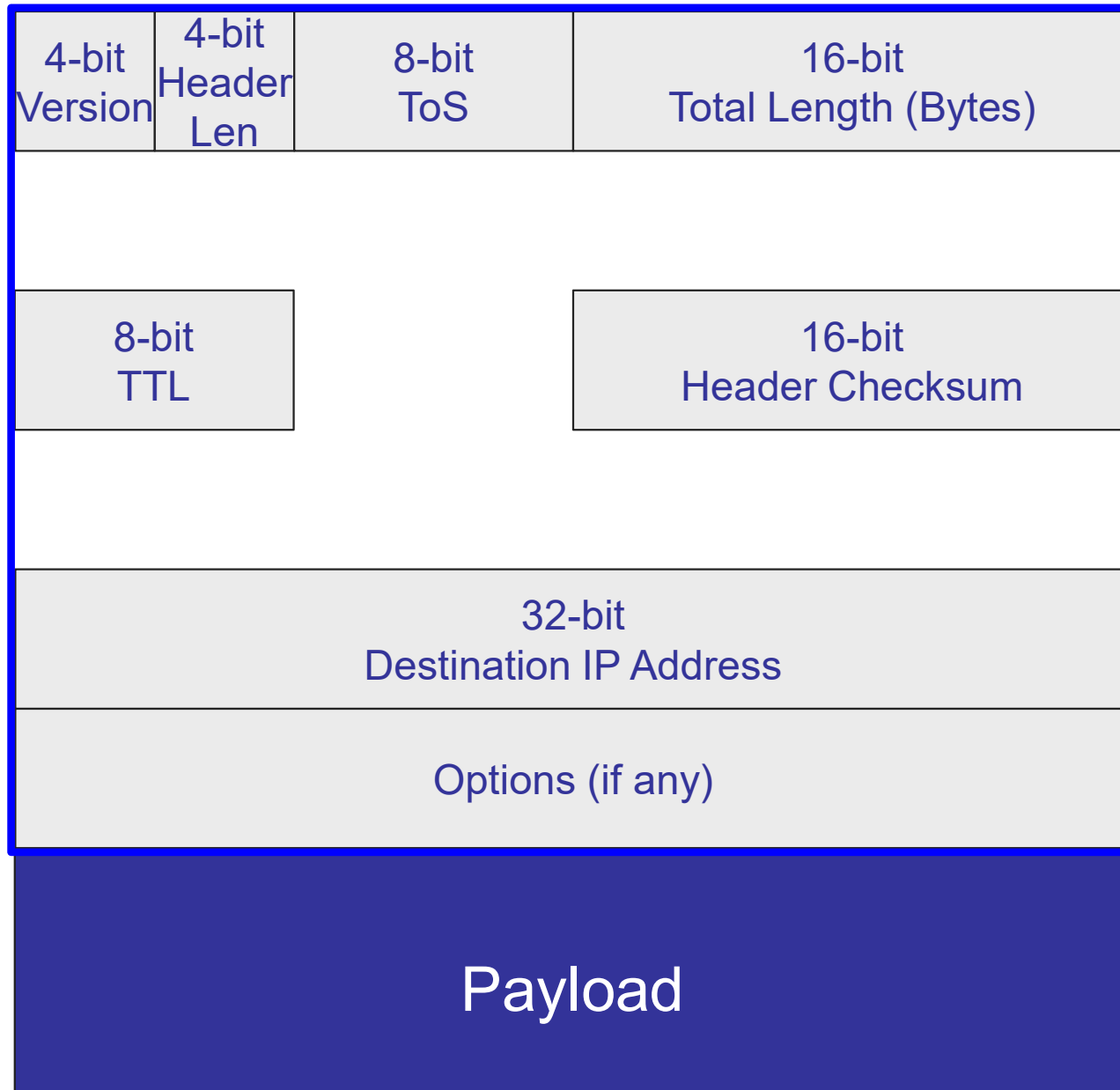
Options

- ❑ Optional directives to the network
 - Not used very often
 - 16 bits of metadata + option-specific data
- ❑ Examples of options
 - Record Route
 - Strict Source Route
 - Loose Source Route
 - Timestamp

What information do we need?

- ❑ Parse packet
 - IP version number (4 bits), packet length (16 bits)
- ❑ Carry packet to the destination
 - Destination's IP address (32 bits)
- ❑ Deal with problems along the way
 - TTL (8 bits), checksum (16 bits), frag. (32 bits)
- ❑ Accommodate evolution
 - Version number (4 bits) (+ fields for special handling)
- ❑ Specify any special handling
 - ToS (8 bits), Options (variable length)

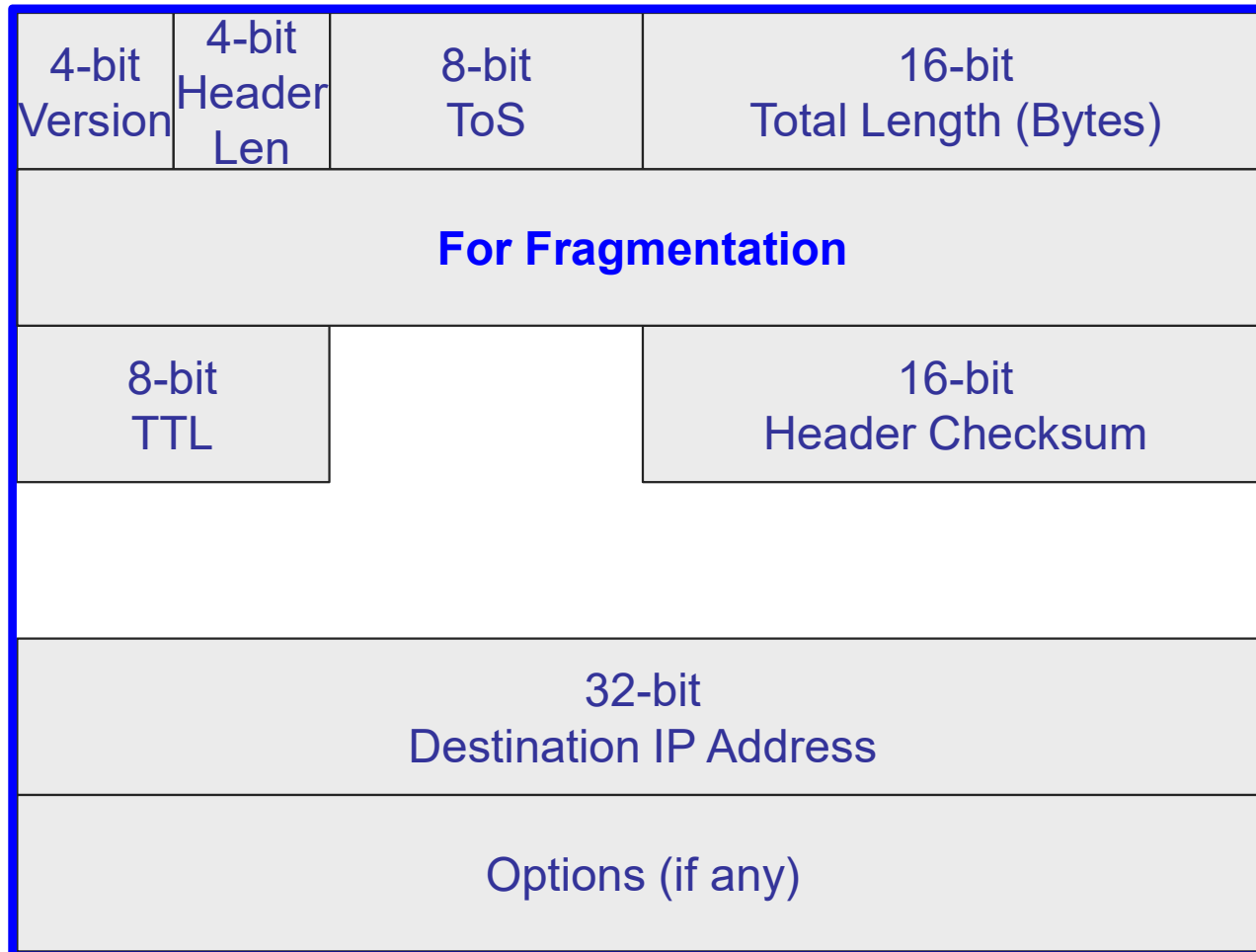
IP packet structure



Parse packet

- Header length (4 bits)
 - Number of 32-bit words in the header
 - Typically “5” (for a 20-byte IPv4 header)
 - Can be more when IP options are used

IP packet structure

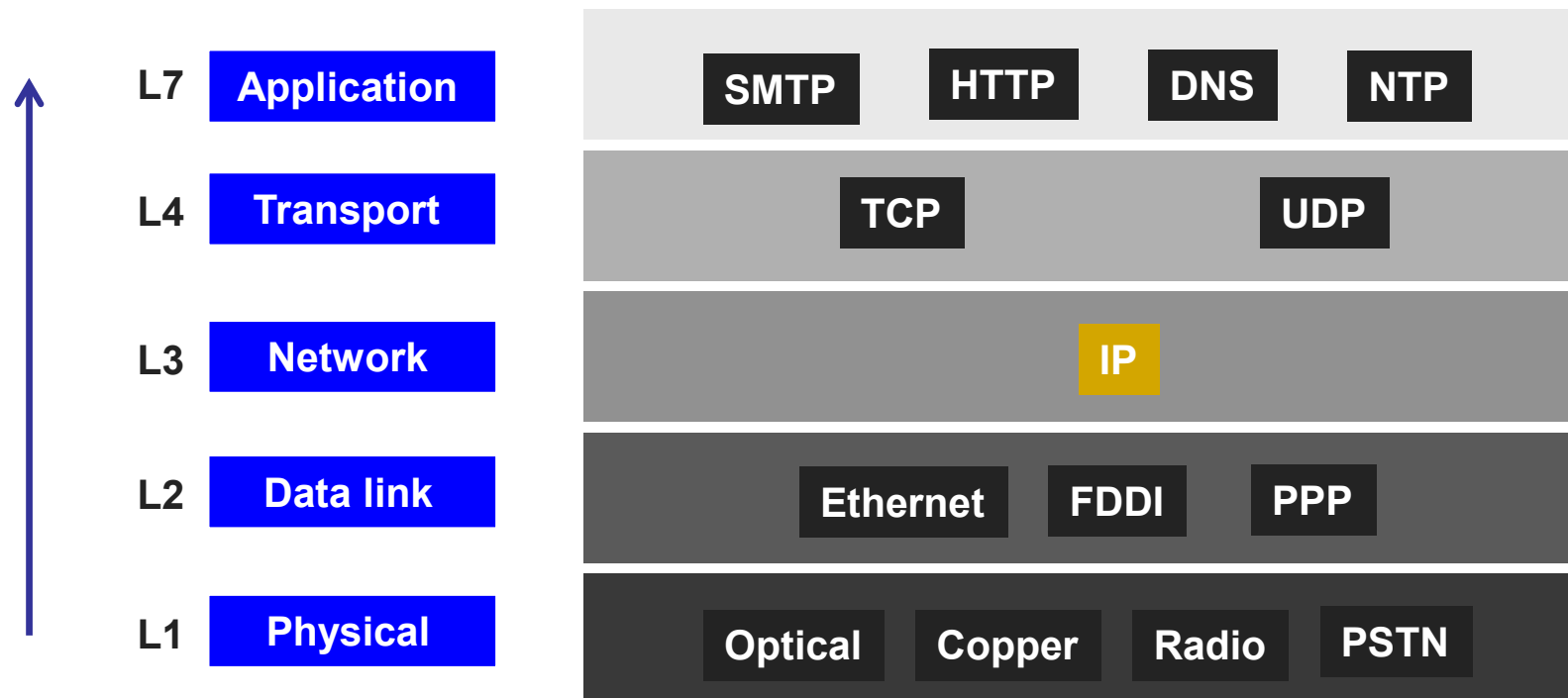


Tasks at the destination end-system

- ▣ Tell destination what to do with the received packet
- ▣ Get responses to the packet back to the source

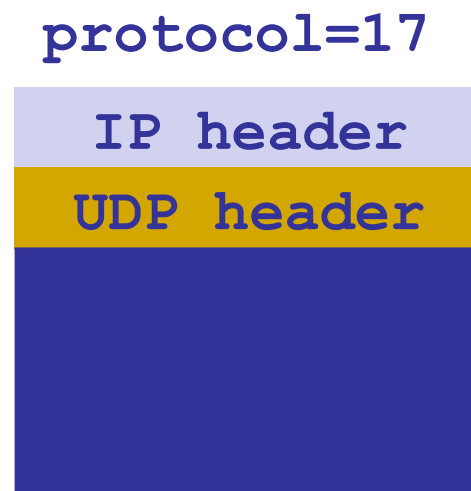
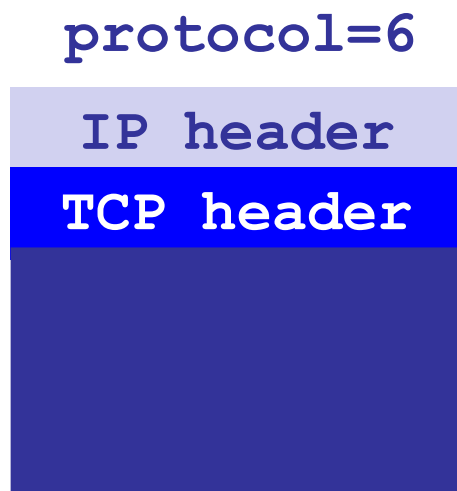
Telling end-host how to handle packet

- Protocol (8 bits)
 - Identifies the higher-level protocol
 - Important for de-multiplexing at receiving host



Telling end-host how to handle packet

- ❑ Protocol (8 bits)
 - Identifies the higher-level protocol
 - Important for de-multiplexing at receiving host
- ❑ Most common examples
 - E.g., “6” for the Transmission Control Protocol (TCP)
 - E.g., “17” for the User Datagram Protocol (UDP)



Tasks at the destination end-system

- Tell destination what to do with the received packet
 - Transport layer protocol (8 bits)
- Get responses to the packet back to the source
 - Source IP address (32 bits)

IP packet structure

4-bit Version	4-bit Header Len	8-bit ToS	16-bit Total Length (Bytes)
For Fragmentation			
8-bit TTL	8-bit Protocol	16-bit Header Checksum	
32-bit Source IP Address			
32-bit Destination IP Address			
Options (if any)			

5-MINUTE BREAK!

Midterm logistics

- ❑ Friday, Feb. 21 from 7PM - 8:30PM
 - Room assignments (by the first two letters of your username) are as follows:
 - » AA - DB: Chrysler 133
 - » DC - ZZ: Chrysler 220
- ❑ SSD accommodations
 - Friday, Feb. 21 from 6PM at DOW 2150
- ❑ Other conflicts
 - You have received separate emails by now

DEALING WITH FRAGMENTATION

A closer look at fragmentation

- ❑ Every link has a “Maximum Transmission Unit” (MTU)
 - Largest number of bits it can carry as one unit
- ❑ A router can split a packet into multiple “fragments” if the packet size exceeds the link’s MTU
- ❑ Must reassemble to recover original packet

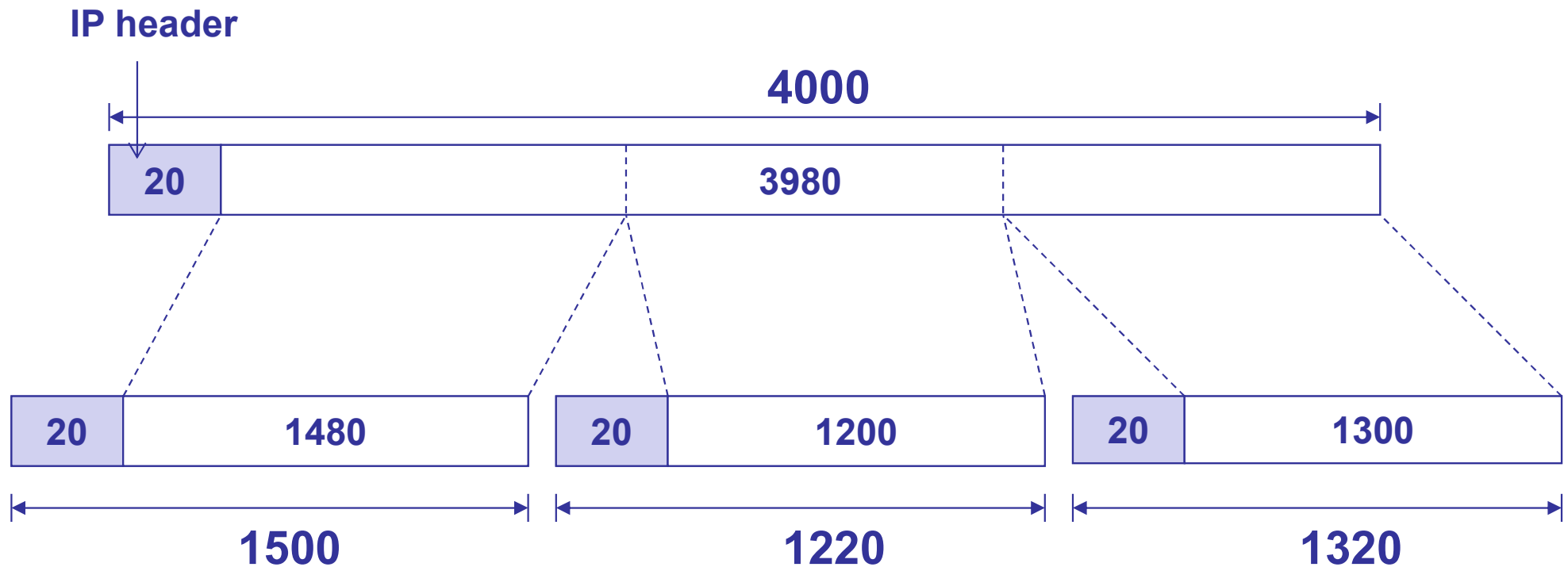
Example of fragmentation

- A 4000 byte packet crosses a link w/ MTU=1500B

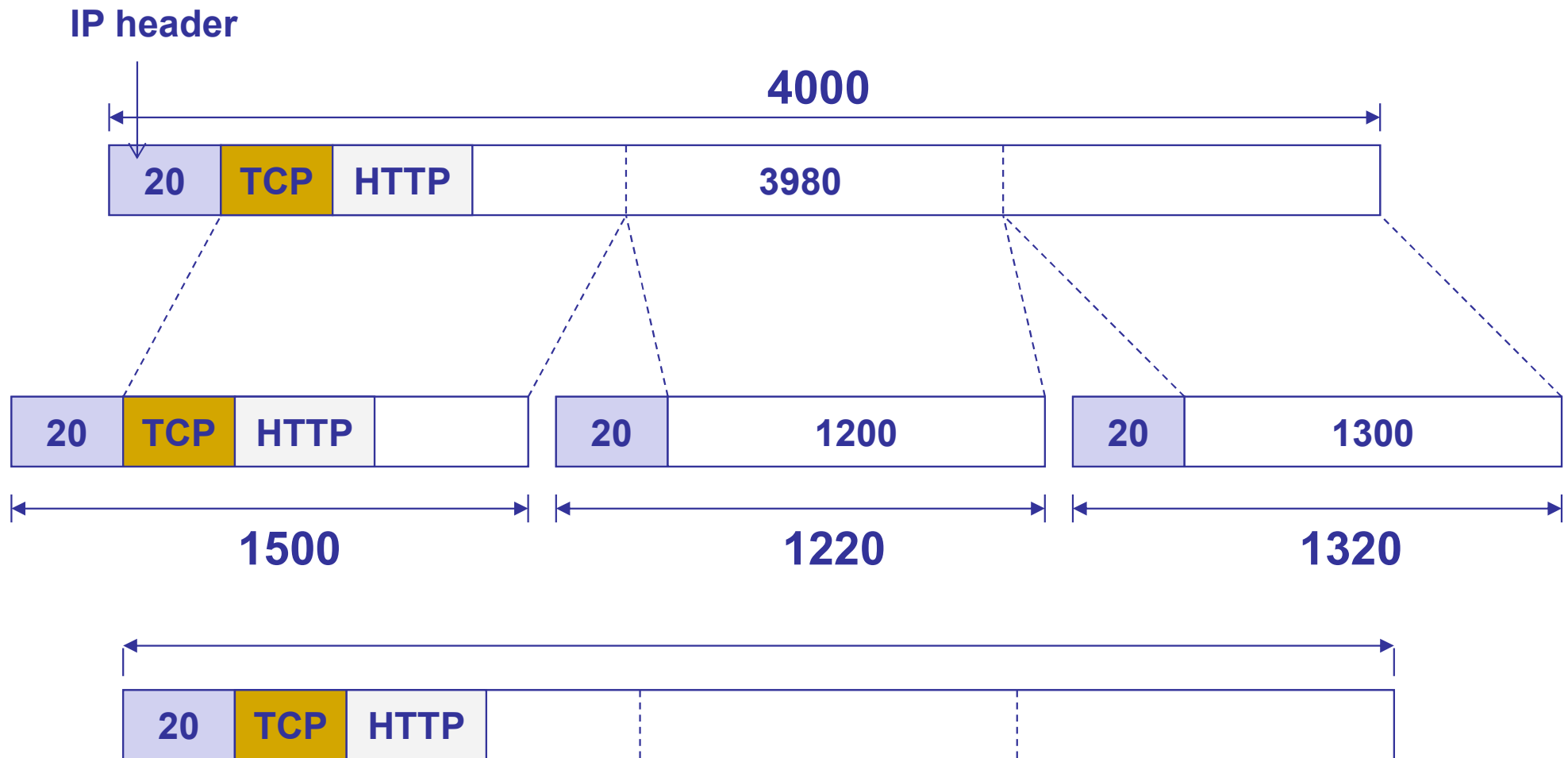


Example of fragmentation

- A 4000 byte packet crosses a link w/ MTU=1500B



Why reassemble?



Must reassemble before sending packet to higher layers!

A few considerations

- ❑ Where to reassemble?
- ❑ Fragments can get lost
- ❑ Fragments can follow different paths
- ❑ Fragments can get fragmented again

Where should reassembly occur?

- ❑ Classic case of E2E principle
- ❑ At next-hop router imposes burden on network
 - Complicated reassembly algorithm
 - Must hold onto fragments/state
- ❑ Any other router may not work
 - Fragments may take different paths
- ❑ Little benefit, large cost for network reassembly
- ❑ Hence, reassembly is done at the destination

Reassembly: What fields?

- ❑ Need a way to identify fragments of the packet
 - Introduce an identifier
- ❑ Fragments can get lost
 - Need some form of sequence number or offset
- ❑ Sequence numbers / offset
 - How do I know when I have them all? (need max seq# / flag)
 - What if a fragment gets re-fragmented?

IPv4's fragmentation fields

- ❑ **Identifier**: which fragments belong together
- ❑ **Flags**:
 - Reserved: ignore
 - DF: don't fragment
 - » May trigger error message back to sender
 - MF: more fragments coming
- ❑ **Offset**: portion of original payload this fragment contains
 - In 8-byte units

IP packet structure

4-bit Version	4-bit Header Len	8-bit ToS	16-bit Total Length (Bytes)
For Fragmentation			
8-bit TTL	8-bit Protocol	16-bit Header Checksum	
32-bit Source IP Address			
32-bit Destination IP Address			
Options (if any)			

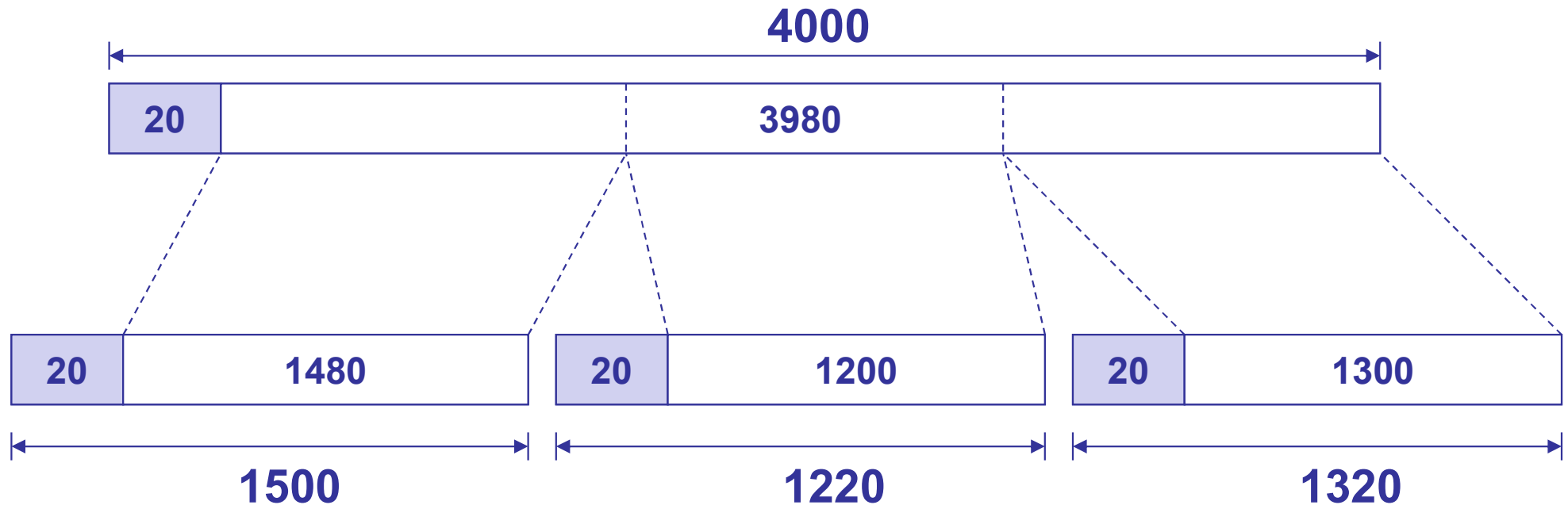
Why this works

- ❑ Fragment without MF set (last fragment)
 - Tells host which are the last bits in original payload
- ❑ All other fragments fill in holes
- ❑ Can tell when holes are filled, regardless of order
 - Use offset field
- ❑ Q: why use a byte-offset for fragments rather than numbering each fragment?
 - Allows further fragmentation of fragments

Example of fragmentation

(contd.)

- Packet split into 3 pieces
- Example:



Example of fragmentation, contd.

- 4000 byte packet from host 1.2.3.4 to 5.6.7.8 traverses a link with MTU 1,500 bytes

Version 4	Header Len 5	ToS 0	Total Length (Bytes) 4000	
Identification 56273			R/D/M 0/0/0	Fragment Offset 0
TTL 127		Protocol 6	Header Checksum 44019	
Source IP Address 1.2.3.4				
Destination IP Address 5.6.7.8				

(3980 more bytes of payload here)

Example of fragmentation, contd.

- Datagram split into 3 pieces. Possible first piece:

Version 4	Header Len 5	ToS 0	Total Length (Bytes) 1500	
Identification 56273			R/D/M 0/0/1	Fragment Offset 0
TTL 126		Protocol 6	Header Checksum xxx	
Source IP Address 1.2.3.4				
Destination IP Address 5.6.7.8				

Example of fragmentation, contd.

- Possible second piece: Frag#1 covered 1480bytes

Version 4	Header Len 5	ToS 0	Total Length (Bytes) 1220	
Identification 56273			R/D/M 0/0/1	Fragment Offset 185 (185 * 8 = 1480)
TTL 126		Protocol 6	Header Checksum yyy	
Source IP Address 1.2.3.4				
Destination IP Address 5.6.7.8				

Example of fragmentation, contd.

- Possible third piece: $1480 + 1200 = 2680$

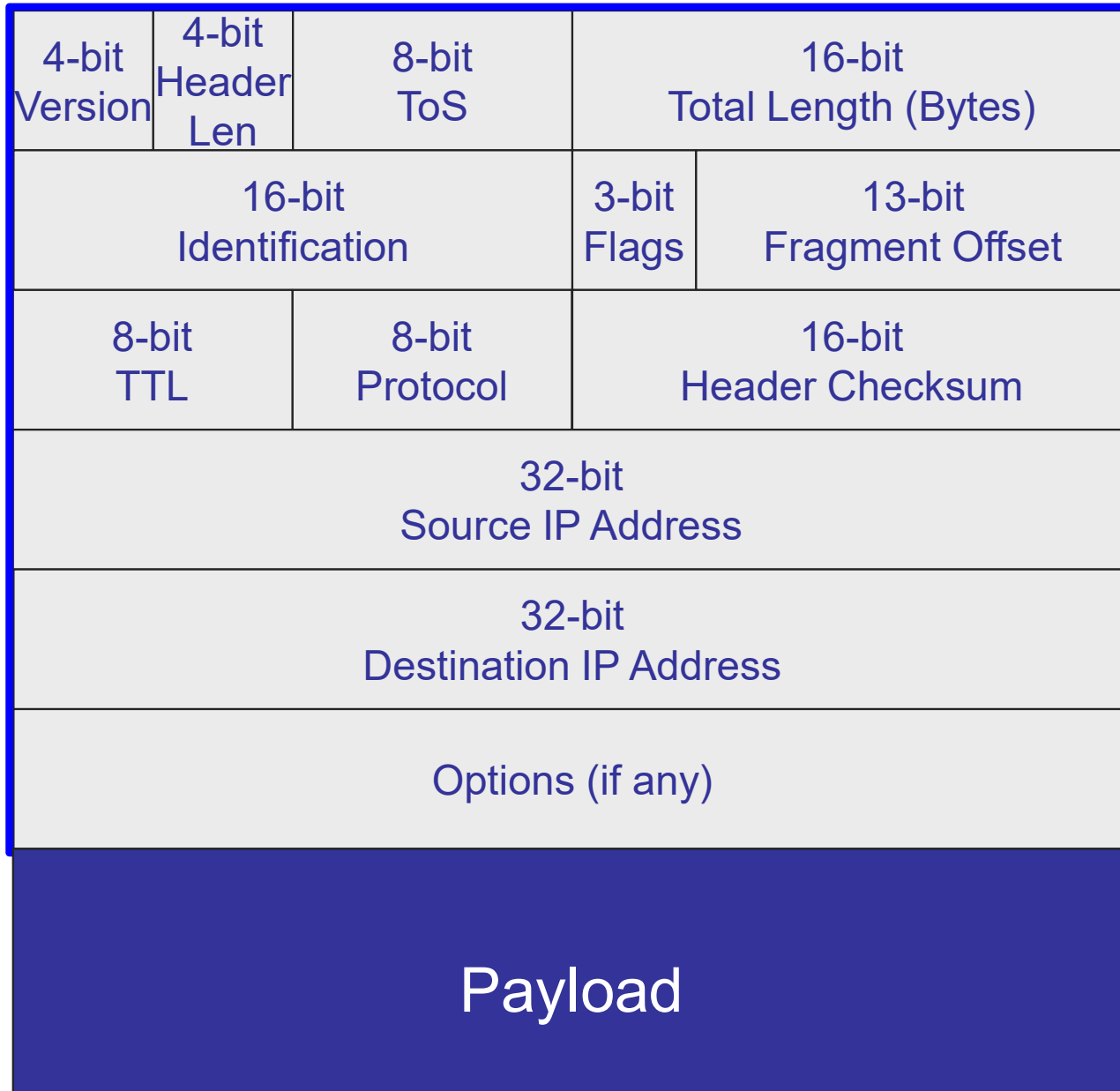
Version 4	Header Len 5	ToS 0	Total Length (Bytes) 1320	
Identification 56273			R/D/M 0/0/0	Fragment Offset 335 (335 * 8 = 2680)
TTL 126		Protocol 6	Header Checksum zzz	
Source IP Address 1.2.3.4				
Destination IP Address 5.6.7.8				

A QUICK LOOK INTO IPV6

IPv6

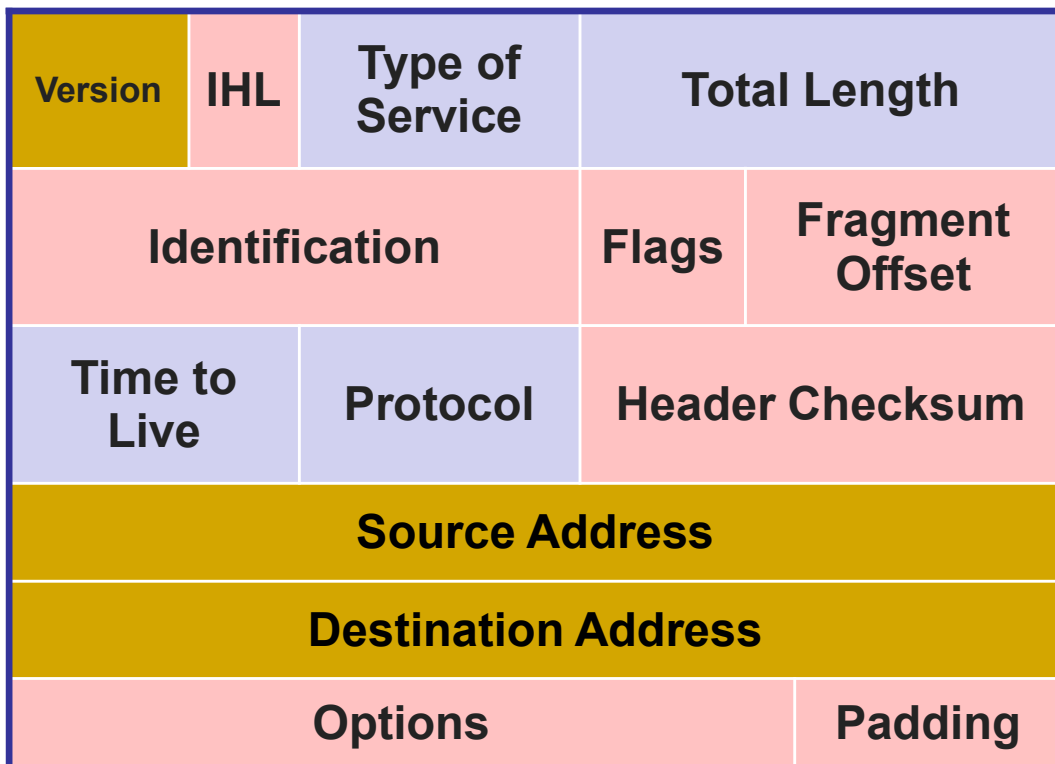
- Motivated (prematurely) by address exhaustion
 - Addresses four times as big (128-bit)
- Focused on simplifying IP
 - Got rid of all fields that were not absolutely necessary
- Result is an elegant, if unambitious, protocol

What “clean up” would you do?

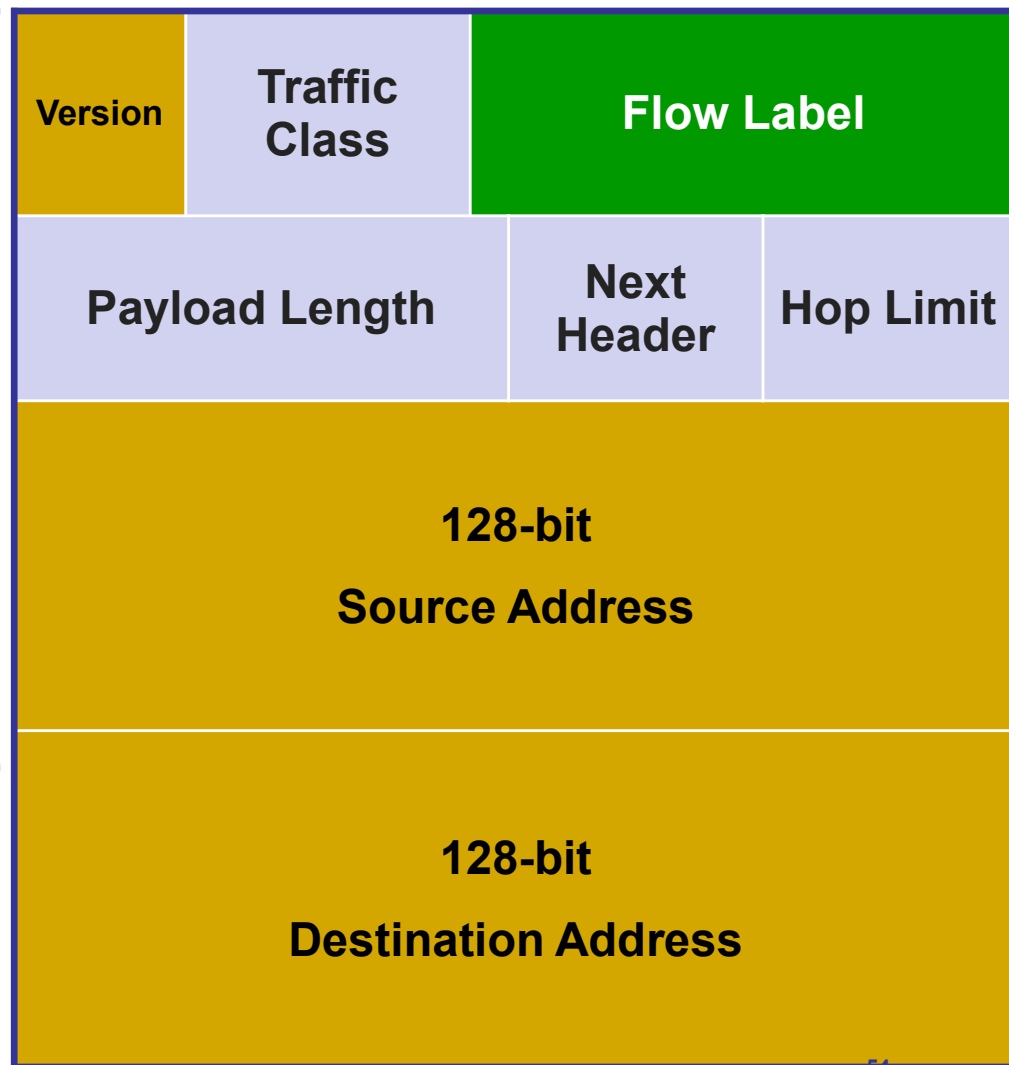


IPv4 and IPv6 header comparison

IPv4



IPv6



- Field name kept from IPv4 to IPv6
- Fields not kept in IPv6
- Name & position changed in IPv6
- New field in IPv6

Summary of changes

- ❑ Eliminated fragmentation (why?)
- ❑ Eliminated checksum (why?)
- ❑ New options mechanism (why?)
- ❑ Eliminated header length (why?)
- ❑ Expanded addresses
- ❑ Added Flow Label

Philosophy of changes

- ❑ Don't deal with problems: leave to ends
 - Eliminated fragmentation and checksum
 - Why retain TTL?
- ❑ Simplify handling:
 - New options mechanism (uses next header)
 - Eliminated header length
 - » Why couldn't IPv4 do this?
- ❑ Provide general flow label for packet
 - Not tied to semantics
 - Provides great flexibility

Summary

- Network layer can be divided into data plane and control plane
 - Data plane deals with “how?”
 - Control plane deals with “what?”
- IP is simple yet nuanced