# CS3600 — Systems and Networks

## NORTHEASTERN UNIVERSITY

Lecture 21: Internet protocol

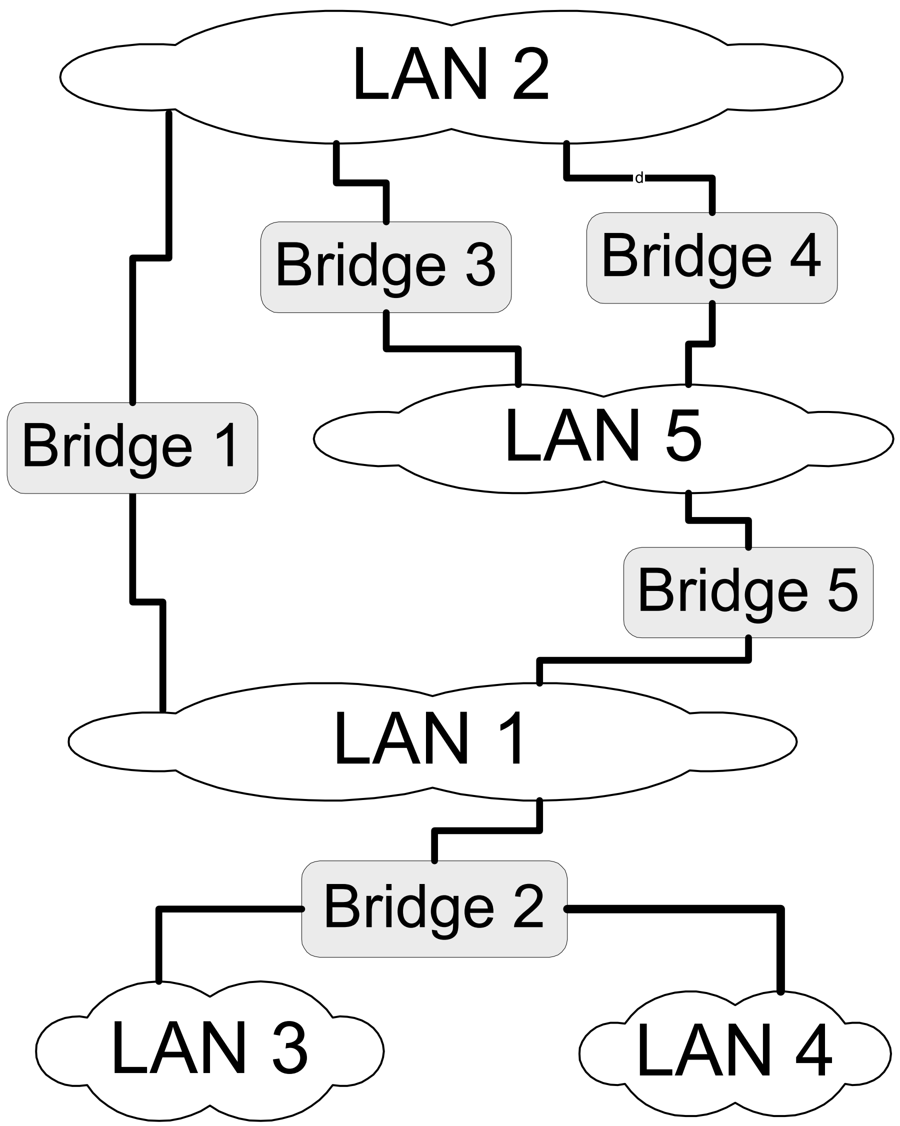
Prof. Alan Mislove (amislove@ccs.neu.edu)

Slides used with permissions from Edward W. Knightly,
T. S. Eugene Ng, Ion Stoica, Hui Zhang

# Recap

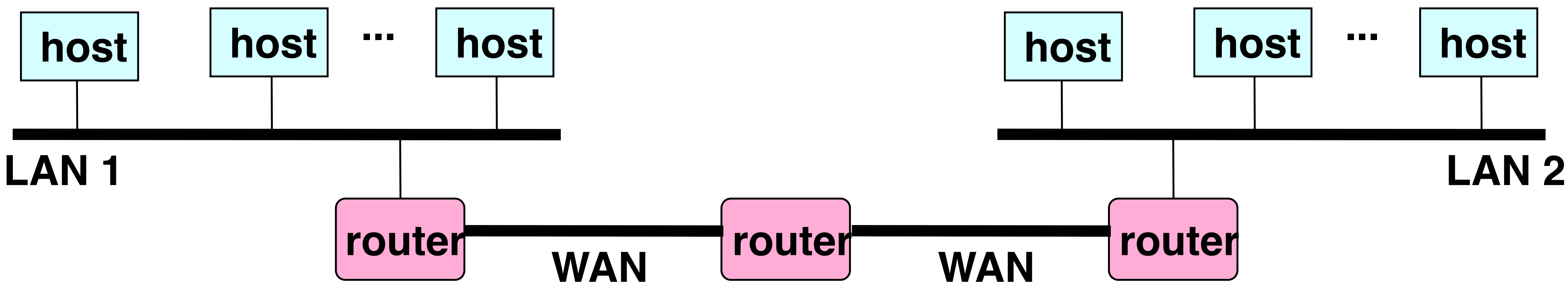
- Cannot build a global network such as the Internet using Ethernet bridges
- Problem 1: Addressing
- Problem 2: Routing

- Additionally, a global network should allow heterogeneous technologies (e.g. ATM, circuit-switched networks, Ethernet, etc)



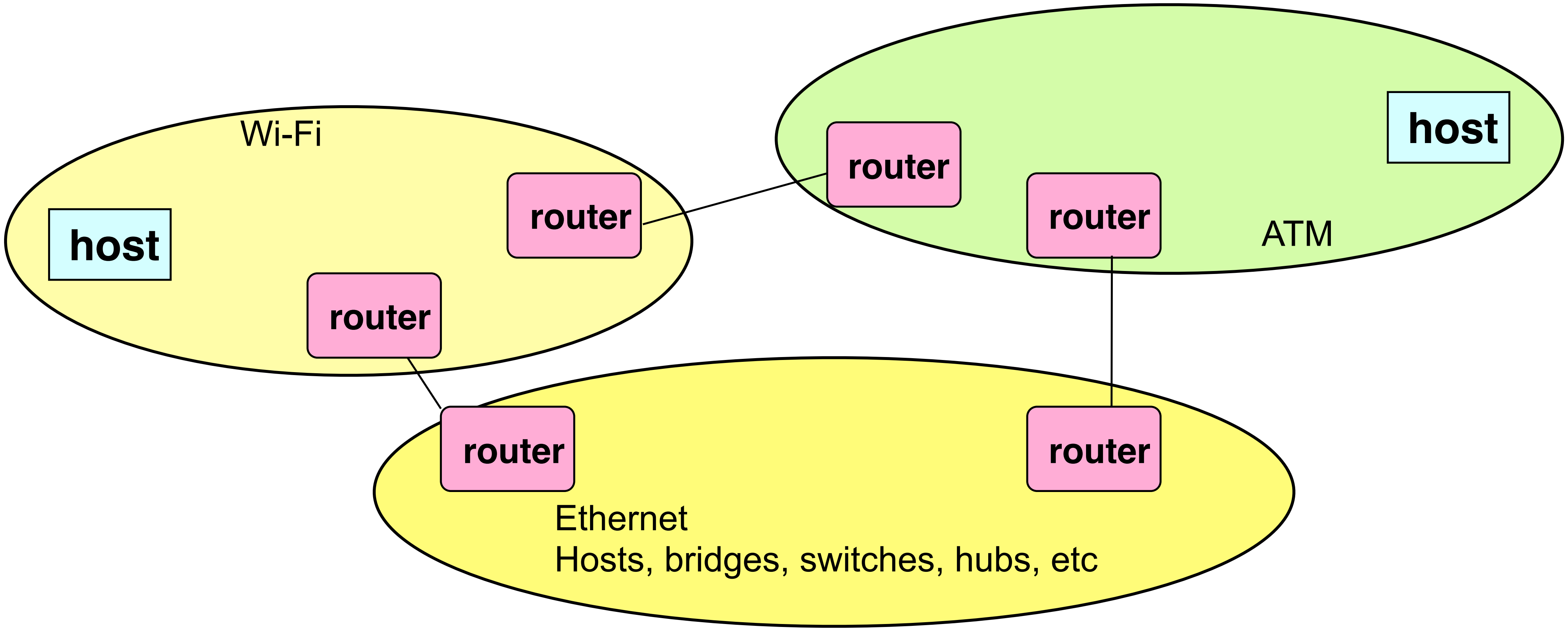
# New Word: Internetwork

- Multiple incompatible LANs can be physically connected by specialized computers called *routers.*
- The connected networks are called an *internetwork*.
  - The “*Internet*” is one (very big & successful) example of an internetwork



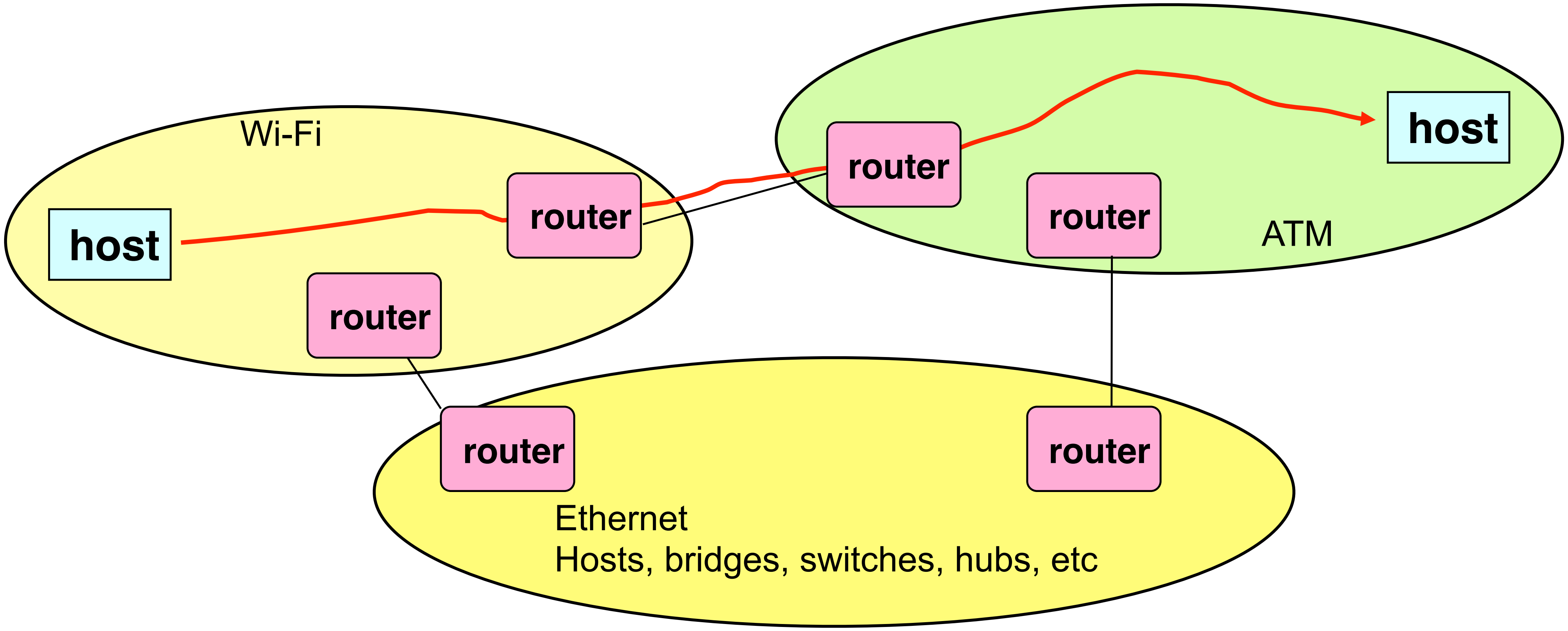

**LAN 1 and LAN 2 might be completely different, totally incompatible LANs (e.g., Ethernet, Wi-Fi, ATM, Circuit-switched)**

# Logical Structure of Internet



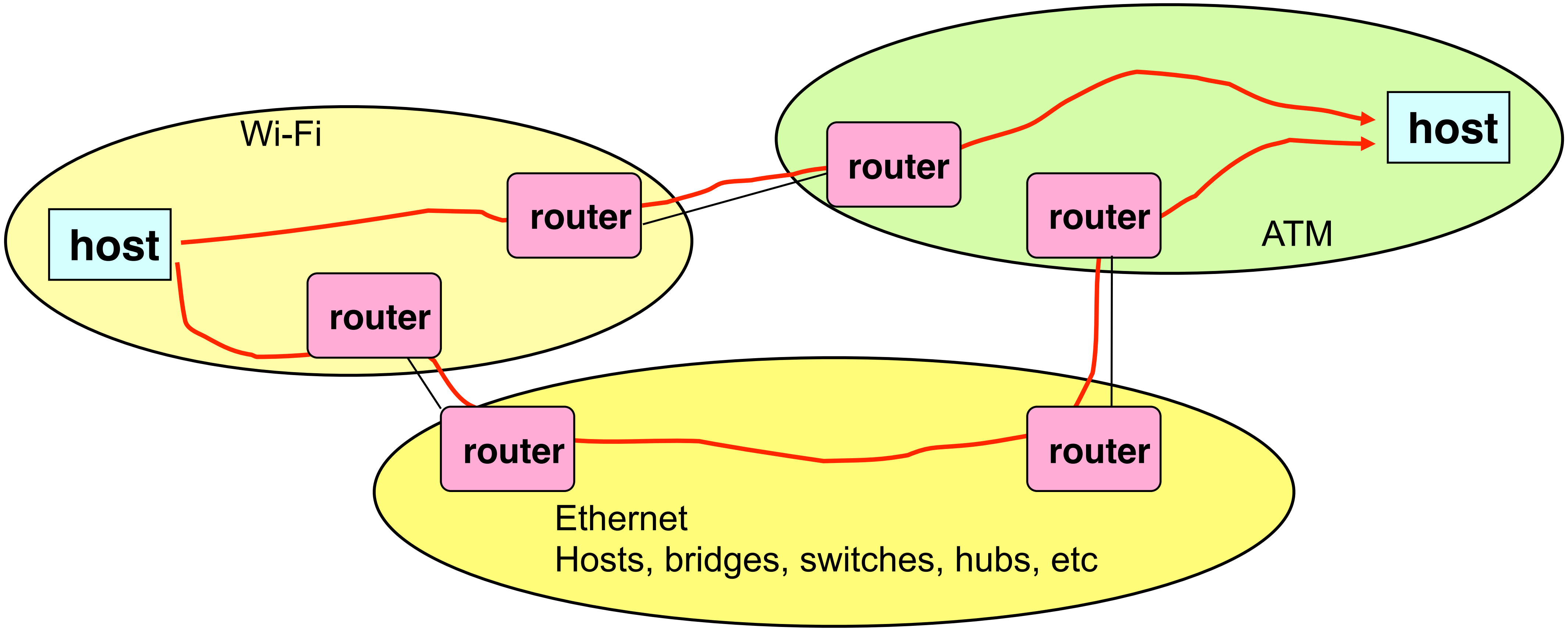

- Ad hoc interconnection of networks
  - No particular topology
  - Vastly different router & link capacities
- Send packets from source to destination by hopping through networks
  - Router connects one network to another
  - Different packets may take different routes

# Logical Structure of Internet



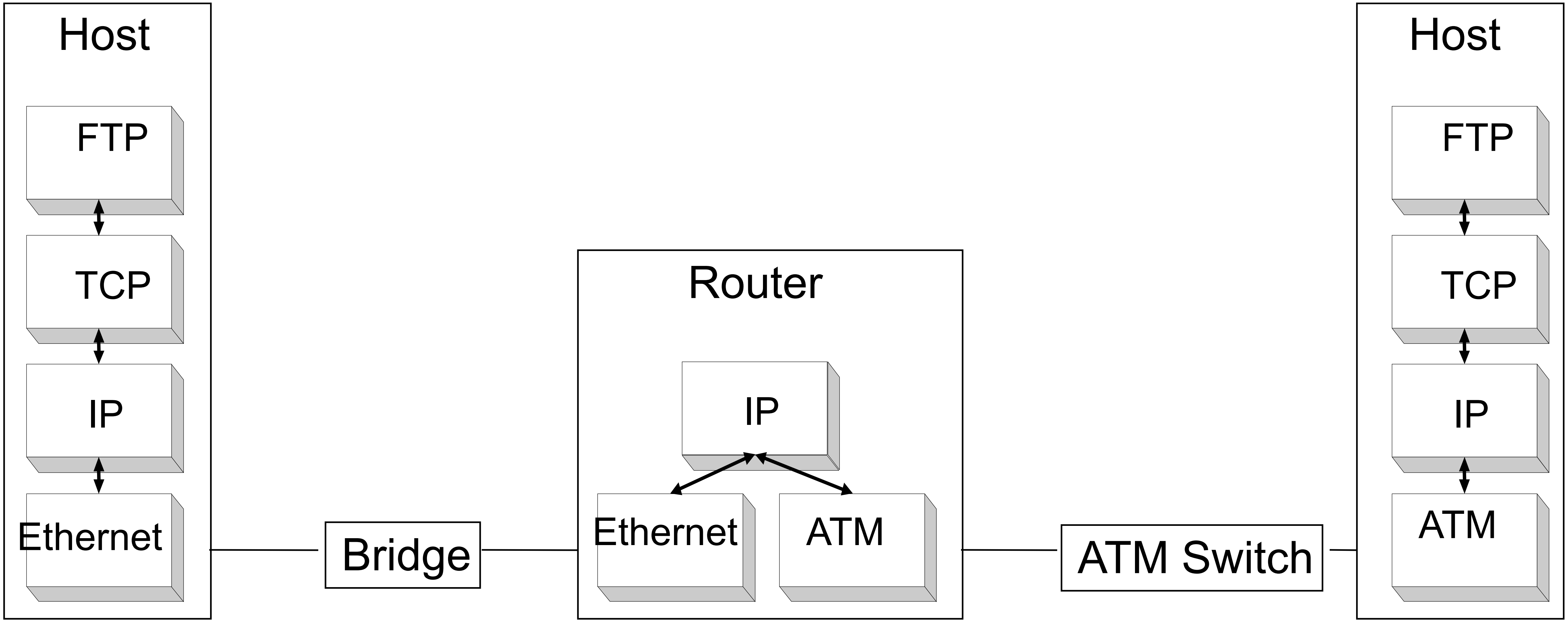

- Ad hoc interconnection of networks
  - No particular topology
  - Vastly different router & link capacities
- Send packets from source to destination by hopping through networks
  - Router connects one network to another
  - Different packets may take different routes

# Logical Structure of Internet




- Ad hoc interconnection of networks
  - No particular topology
  - Vastly different router & link capacities
- Send packets from source to destination by hopping through networks
  - Router connects one network to another
  - Different packets may take different routes

# Adding an Internetwork Layer (IP) for Interoperability



# Issues in Designing an Internetwork

- How do I designate a distant host?
  - Addressing / naming
- How do I send information to a distant host?
  - Underlying service model
    - What gets sent?
    - How fast will it go?
    - What happens if it doesn't get there?
  - Routing
- Challenges
  - Heterogeneity
    - Assembly from variety of different networks
  - Scalability
    - Ensure ability to grow to worldwide scale

Internet: Best-effort, datagram network
A kind of lowest common denominator

# Possible Addressing Schemes

- Flat
  - e.g., every host identified by its 48-bit MAC address
  - Router would need entry for every host in the world
    - Too big (although technology can help this)
    - Too hard to maintain as hosts come & go
- Hierarchy
  - Address broken into segments of increasing specificity
    - 617 (Boston) – 373 (NEU area) – 2000 (Particular phone)
  - Route to general region and then work toward specific destination
  - As people and organizations shift, only update affected routing tables

# IP Addressing

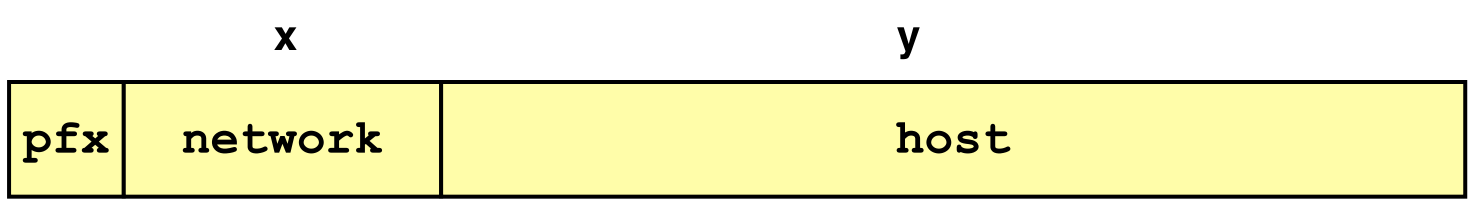
- IPv4: 32-bit addresses
  - Typically, write in dotted decimal format
    - E.g., 128.42.198.135
    - Each number is decimal representation of byte
  - Big-Endian Order

0 8 16 24 31

| | | | | |
|---|---|---|---|---|
| 128 | 42 | 198 | 135 | **Decimal** |
| 80 | 2a | c6 | 87 | **Hexadecimal** |
| 0100 0000 | 0010 1010 | 1100 0110 | 1000 0111 | **Binary** |

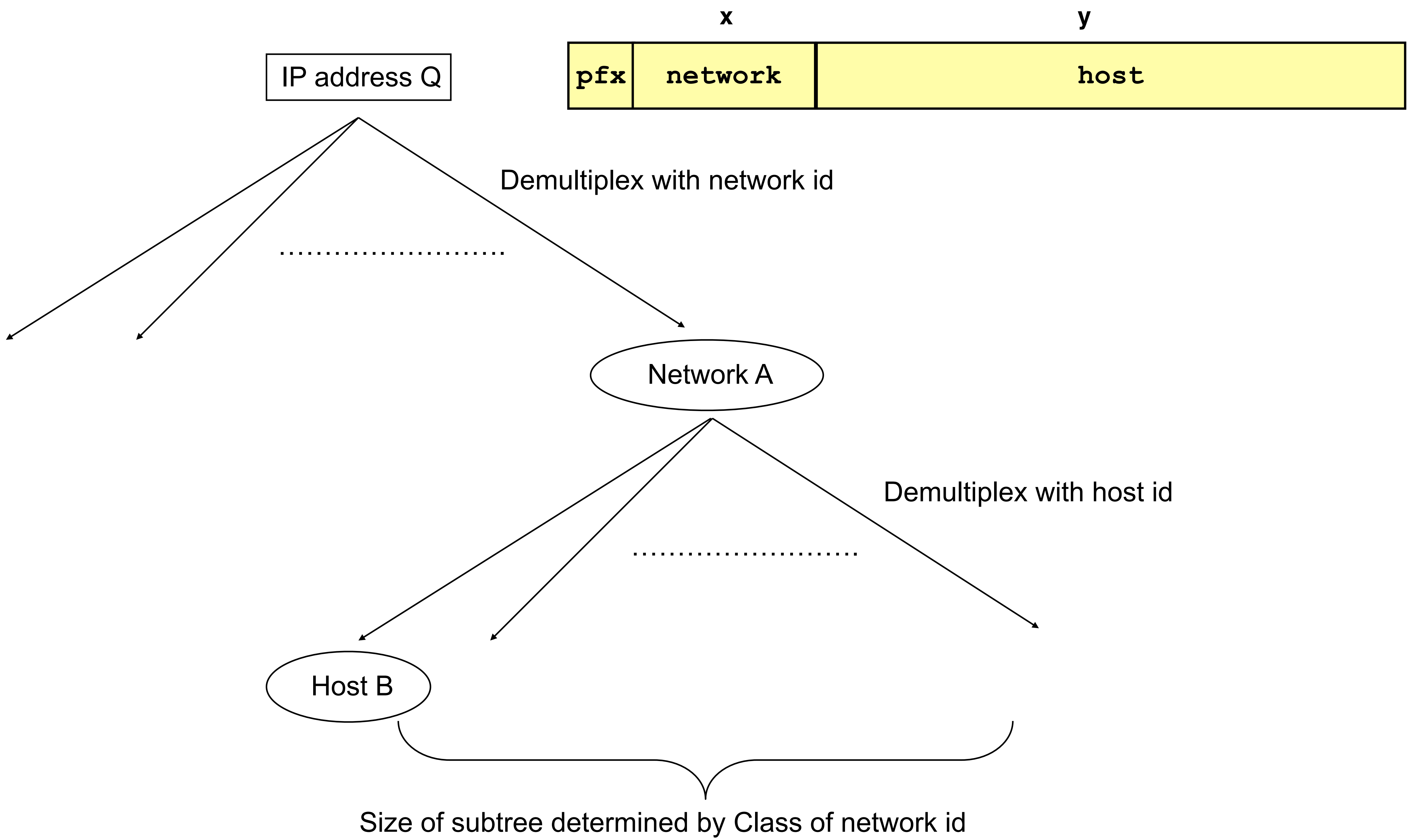
# IP Addressing and Forwarding

- Routing Table Requirement
  - For every possible destination IP address, give next hop
  - Nearly $2^{32}$ ($4.3 \times 10^9$) possibilities!
- Hierarchical Addressing Scheme



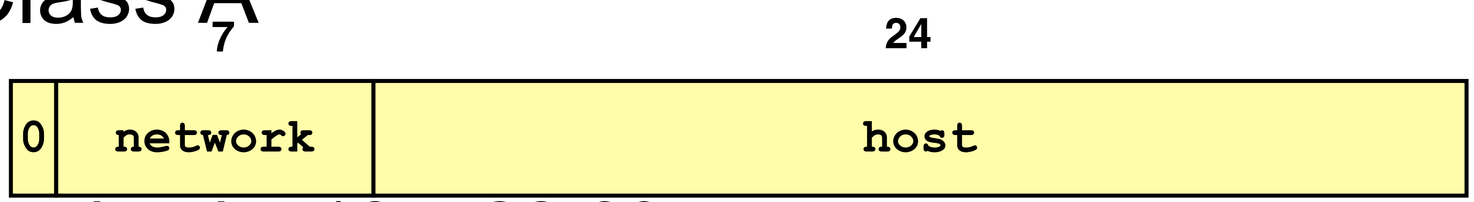

  - Address split into network ID and host ID
  - All packets to given network follow same route
    - Until they reach destination network
  - Fields
    - `pfx` Prefix to specify split between network & host IDs
    - `network` $2^x$ possibilities
    - `host` $2^y$ possibilities

# Two Level Hierarchy of Basic IP addressing



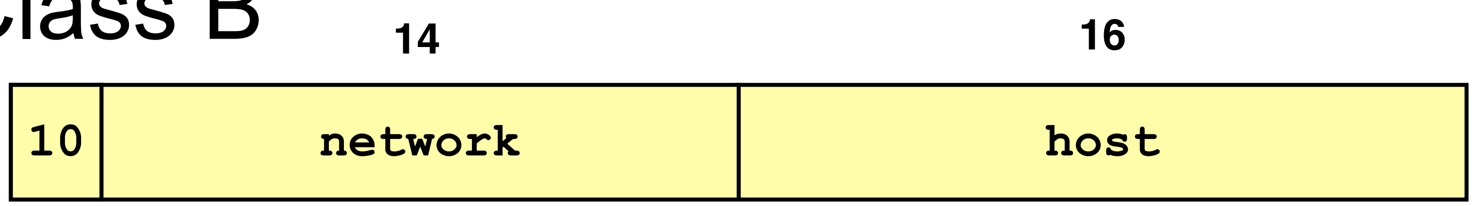
# IP Address Classes

- Class A



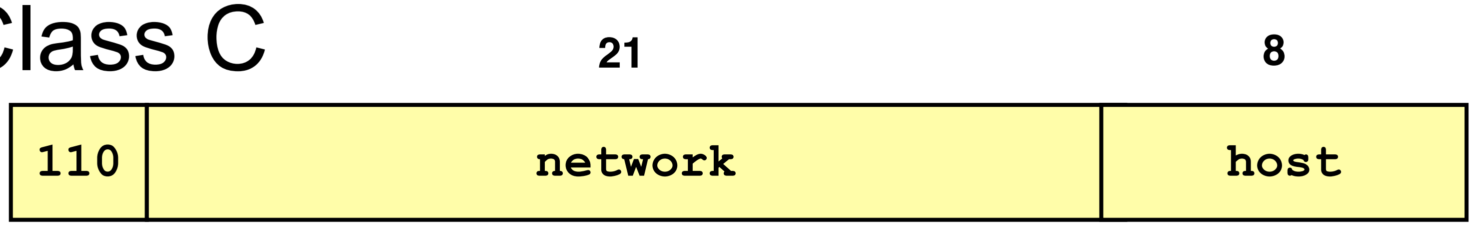

**First octet: 1–126**

  - mit.edu: 18.7.22.69
- Class B




**First octet: 128–191**

  - rice.edu: 128.42.129.23
- Class C




**First octet: 192–223**

  - adsl-216-63-78-18.dsl.hstntx.swbell.net: 216.63.78.18
- Classes D, E, F
  - Not commonly used

# IP Address Classes

| Class | Count | Hosts |
|---|---|---|
| A | $2^7$-2 = 126 (0 & 127 reserved) | $2^{24}$-2 = 16,777,214 (all 0s, all 1s reserved) |
| B | $2^{14}$ = 16,398 | $2^{16}$-2 = 65,534 (all 0s, all 1s reserved) |
| C | $2^{21}$ = 2,097,512 | $2^8$-2 = 254 (all 0s, all 1s reserved) |
| Total | 2,114,036 | |

- Partitioning too Coarse
  - No local organization needs 16.7 million hosts
    - Large organization likely to be geographically distributed
  - Many organizations must make do with multiple class C's
- Too many different Network IDs
  - Routing tables must still have 2.1 million entries

# Within Organization: Subnetting

- Add Another Layer to Hierarchy

| | x | z | y |
|---|---|---|---|
| pfx | network | subnet | host |

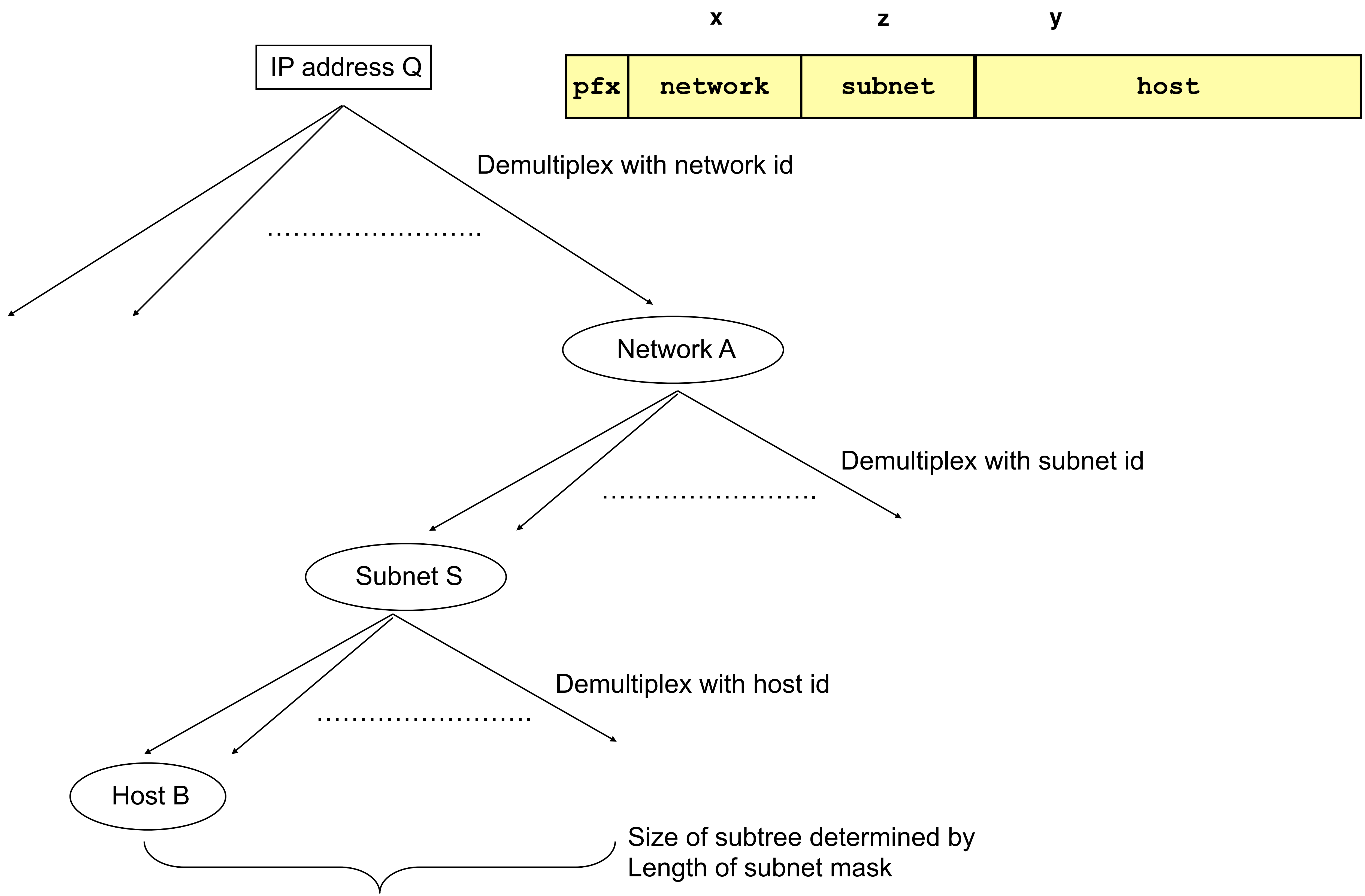
  – From the outside, appears as one monolithic network
    - Single entry in routing table
  – Within network, manage as multiple subnetworks
    - Internal routers must route according to subnet ID
- Subnet Mask
  – Way to specify break between subnet ID and host ID

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

| | | | |
|---|---|---|---|
| pfx | network | subnet | host |

  – Similar masks used in many contexts

# Subnetting



x
z
y
pfx
network
subnet
host
IP address Q
Demultiplex with network id
Network A
Demultiplex with subnet id
Subnet S
Demultiplex with host id
Host B
Size of subtree determined by
Length of subnet mask

# Routing Table

| Address Pattern | Subnet Mask | Next Hop |
|---|---|---|
| 128.42.222.0 | 255.255.255.0 | R1 |
| 128.42.128.0 | 255.255.128.0 | R2 |
| 18.0.0.0 | 255.0.0.0 | R3 |
| 0.0.0.0 | 0.0.0.0 | R4 |
| 128.42.0.0 | 255.255.0.0 | R5 |

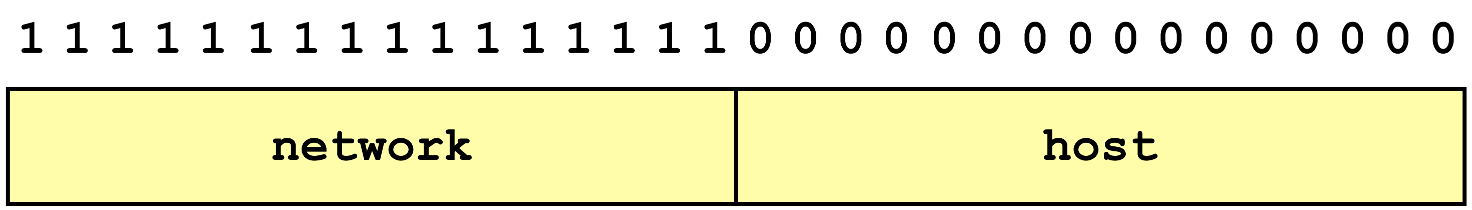
- Address 128.42.222.198 matches 4 entries
- Longest Prefix Match
  - Select entry with longest sequence of 1's in mask
  - Most specific case

# Improving the Hierarchy

- Basic Idea of Hierarchy is Good
  - Organizations of different sizes can be assigned different numbers of IP addresses
- Shortcomings of Class-Based Addressing
  - Class A too coarse; Class C too fine; not enough Class B's
  - When fully deployed would have too many entries in routing table (2.1 million)
- Solution
  - Hierarchy with finer gradation of network/host ID split

# Classless Interdomain Routing

- CIDR, pronounced “cider”
- Arbitrary Split Between Network & Host IDs
  - Specify either by mask or prefix length




  - E.g., NEU can be specified as
    - 129.10.0.0 with netmask 255.255.0.0
    - 129.10.0.0/16

# Aggregation with CIDR

– Original Use: Aggregate Class C Addresses
– One organization assigned contiguous range of class C's
  - e.g., Microsoft given all addresses 207.46.192.X -- 207.46.255.X
  - Specify as CIDR address 207.46.192.0/18

| 0 | 8 | 16 | 24   31 | |
|---|---|---|---|---|
| 207 | 46 | 192 | 0 | **Decimal** |
| cf | 2e | c0 | 00 | **Hexadecimal** |
| 1100 1111 | 0010 1110 | 11xx xxxx | xxxx xxxx | **Binary** |

**Upper 18 bits frozen**    **Lower 14 bits arbitrary**

  - Represents $2^6$ = 64 class C networks

– Use single entry in routing table
  - Just as if were single network address

# Routing Table Entry Examples

- Snapshot From MAE-West Routing Table
  - Probably out of date

| Address | Prefix Length | Third Byte | Byte Range |
|---|---|---|---|
| 207.46.0.0 | 19 | $000xxxxx_2$ | 0 – 31 |
| 207.46.32.0 | 19 | $001xxxxx_2$ | 32 – 63 |
| 207.46.64.0 | 19 | $010xxxxx_2$ | 64 – 95 |
| 207.46.128.0 | 18 | $10xxxxxx_2$ | 128 – 191 |
| 207.46.192.0 | 18 | $11xxxxxx_2$ | 192 – 255 |

**microsoft.com: 207.46.245.214 & 207.46.245.222**

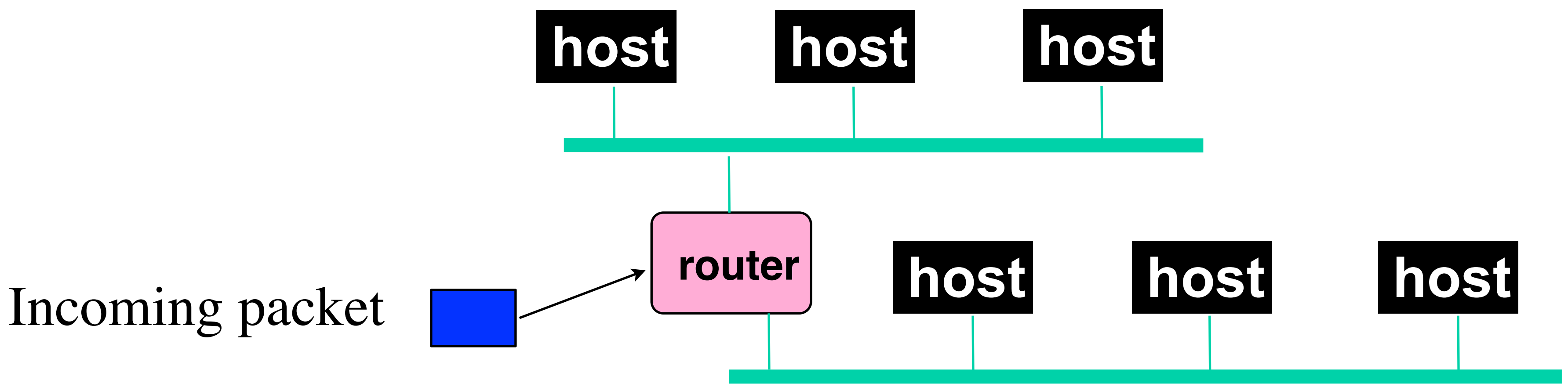
  - Note hole in table: Nothing covers bytes 96 – 127

# Important Concepts

- Hierarchical addressing critical for scalable system
  - Don’t require everyone to know everyone else
  - Reduces amount of updating when something changes
- Non-uniform hierarchy useful for heterogeneous networks
  - Class-based addressing too coarse
  - CIDR helps
  - Move to IPv6 due to limited number of 32-bit addresses
- Implementation Challenge
  - Longest prefix matching much more difficult than when no ambiguity

# Mapping IP to Ethernet

- Each host has IP address **and** Ethernet address




- Incoming packets have IP address of destination
  - Not Ethernet address
- How does a router determine where to send it?

# Address resolution protocol (ARP)

- Simple protocol to map IP addr to Ethernet addr

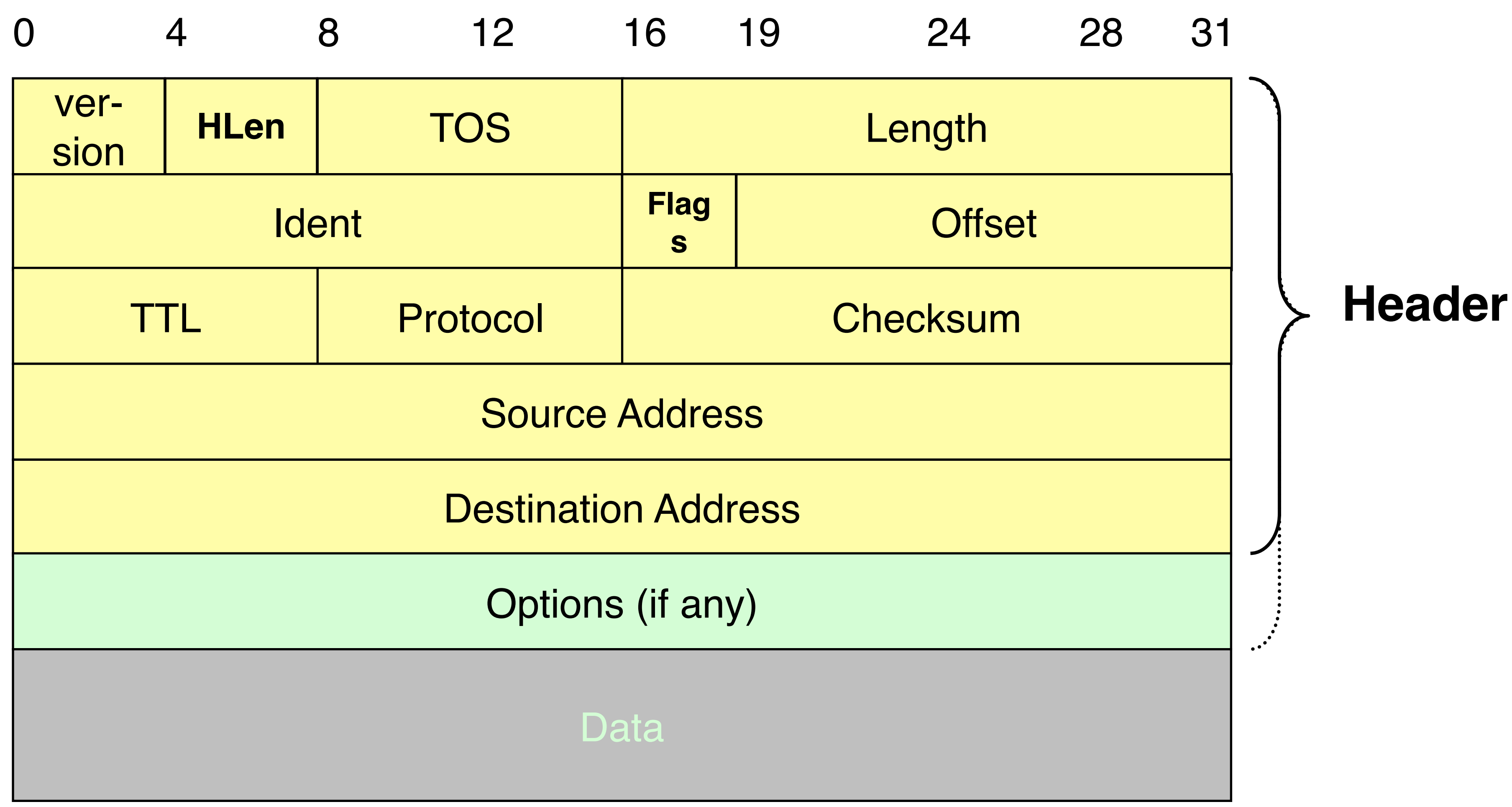
- Format:
  - Query: Who has IP x? Please tell Ethernet xx:xx:xx...
  - Answer: yy:yy:yy... has IP x.

- Allows endpoints to learn IP/Ethernet mapping
  - Can cache results; called ARP cache
  - Entries purged after short time

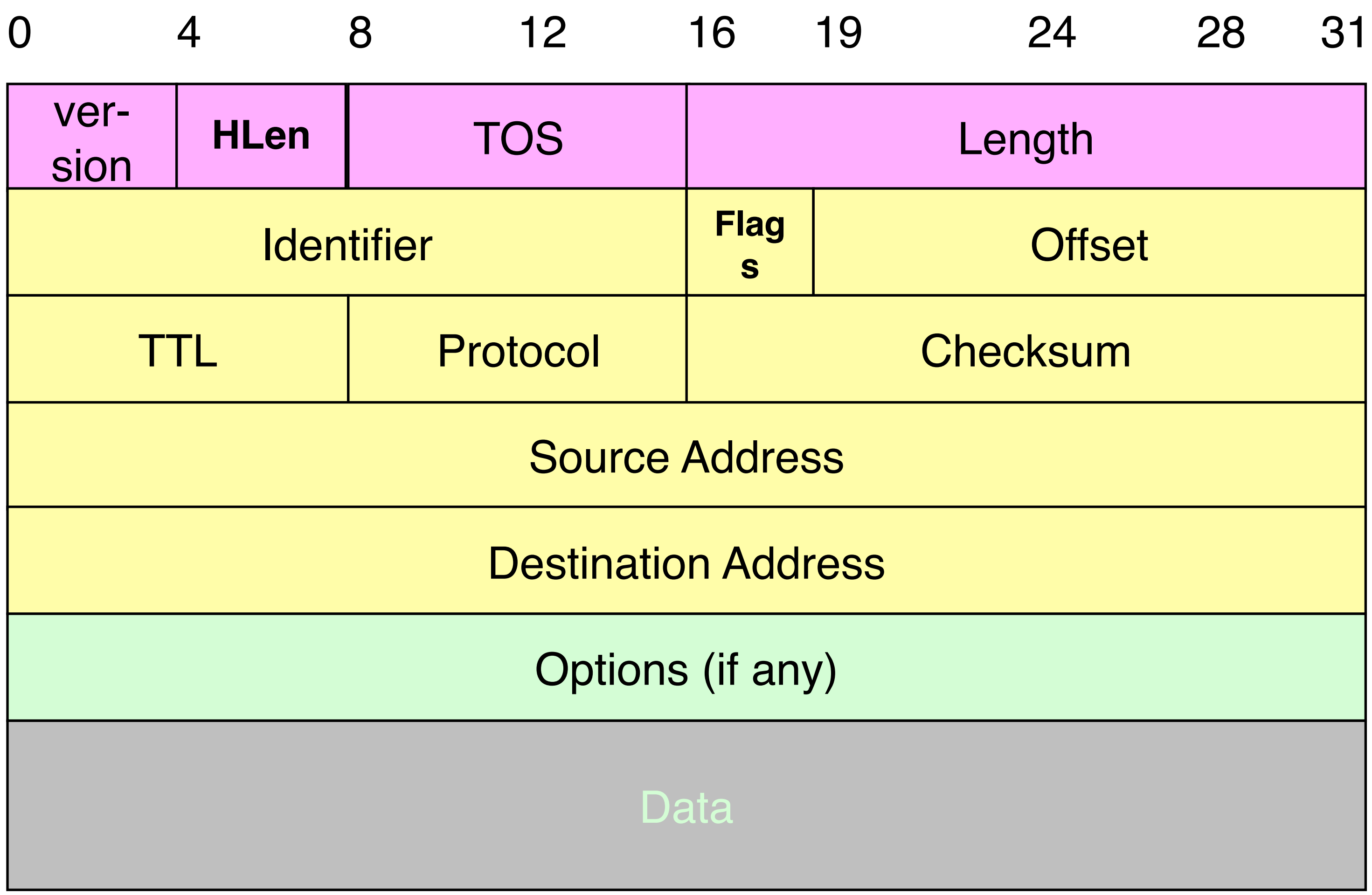
# IP Service Model

- Datagram
  - Each packet self-contained
    - All information needed to get to destination
    - No advance setup or connection maintenance
  - Analogous to letter or telegram

**IPv4 Packet Format**



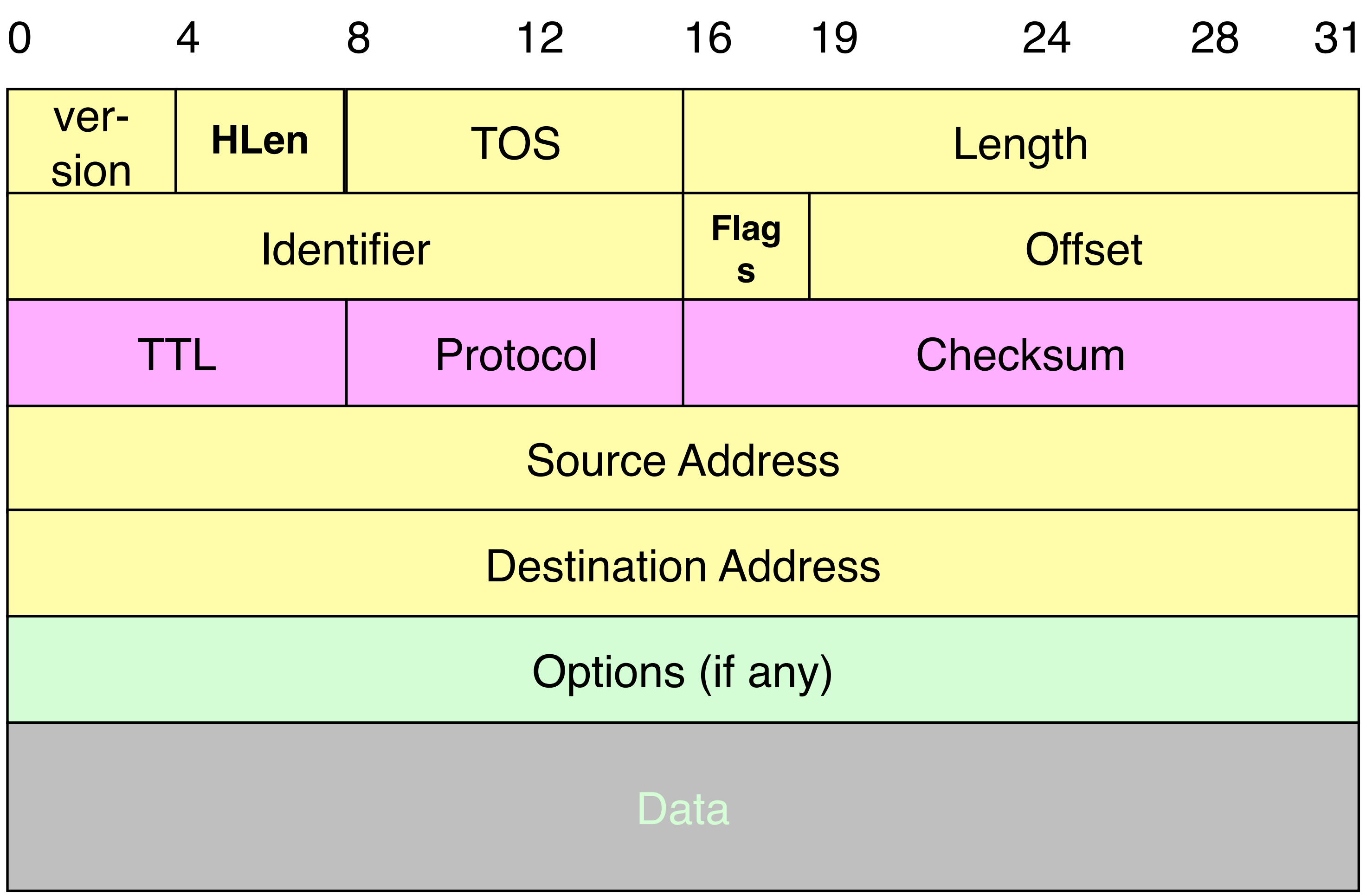
# IP Header Fields: Word 1




- Version: IP Version
  - 4 for IPv4
- HLen: Header Length
  - 32-bit words (typically 5)
- TOS: Type of Service
  - Priority information
- Length: Packet Length
  - Bytes (including header)

- Header format can change with versions
  - First byte identifies version
- Length field limits packets to 65,535 bytes
  - In practice, break into much smaller packets for network performance considerations

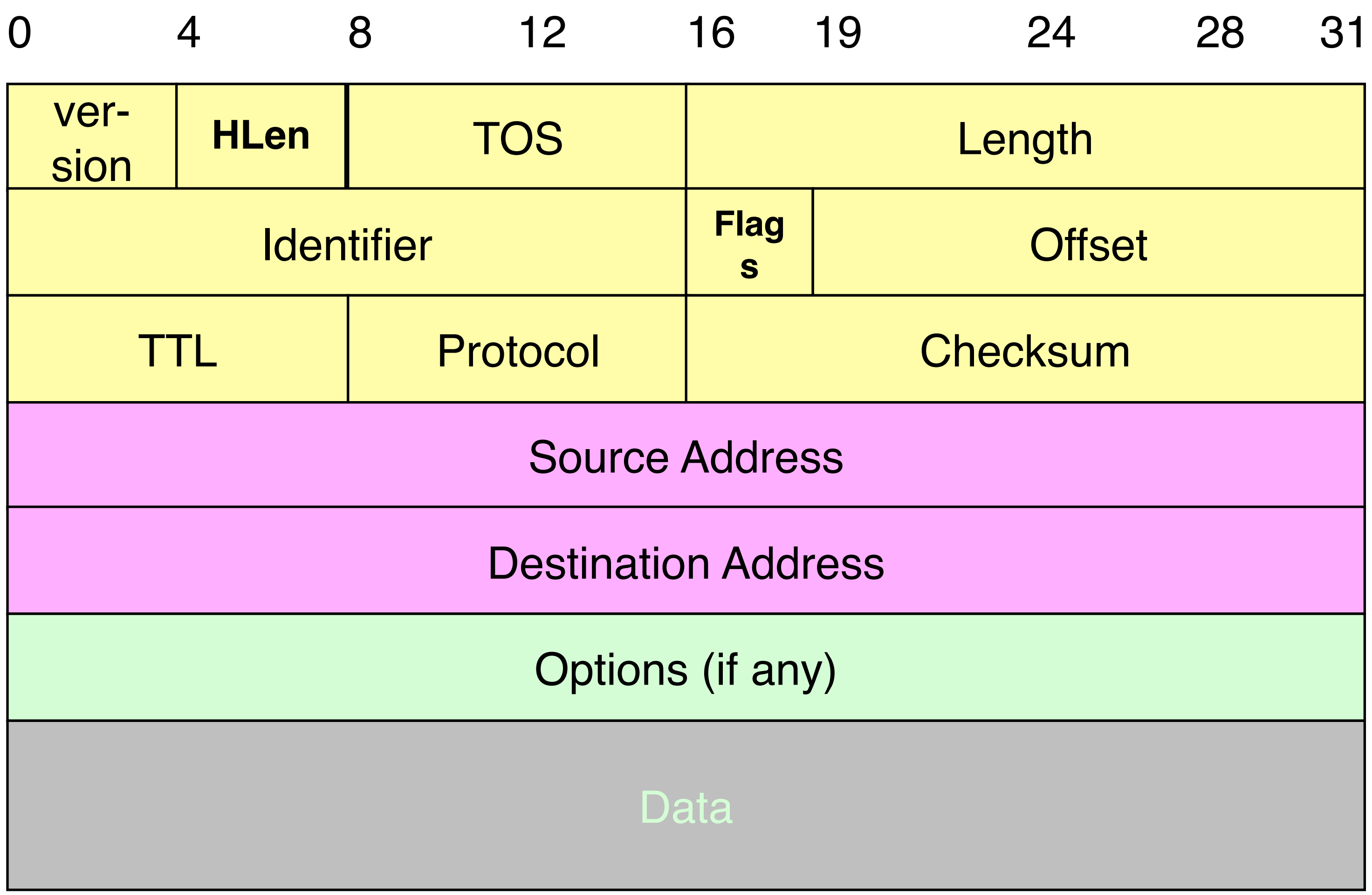
# IP Header Fields: Word 3




- TTL: time to live
  - Decrement by one at each intermediate router
  - Prevent looping forever
- Protocol
  - Protocol of next layer (in “data”)
  - E.g. TCP (6), UDP (17)
- Checksum
  - Of IP header

- Protocol field used for demultiplexing
- Checksum re-computed at each router
  - Why?
- TTL field used to implement traceroute

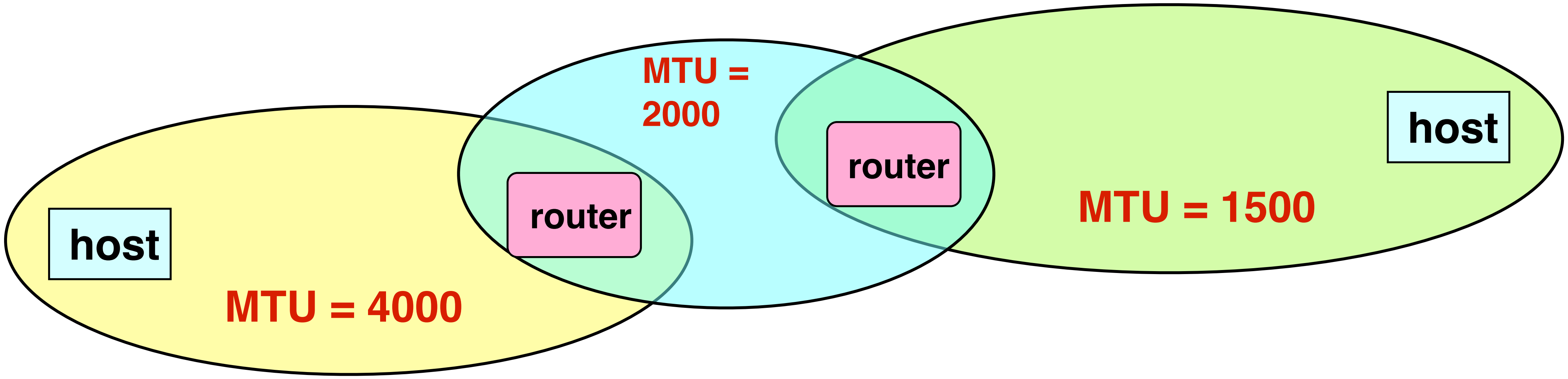
# IP Header Fields: Words 4&5



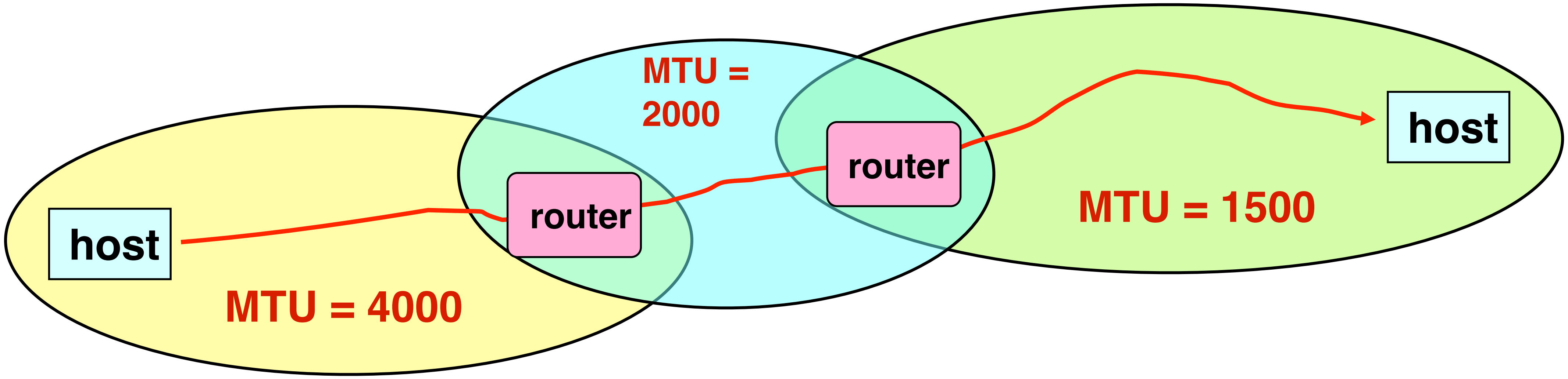

- Source Address
  - 32-bit IP address of sender
- Destination Address
  - 32-bit IP address of destination

- Like the addresses on an envelope
- In principle, globally unique identification of sender & receiver
  - In practice, there are contexts where either source or destination are not the ultimate addressees

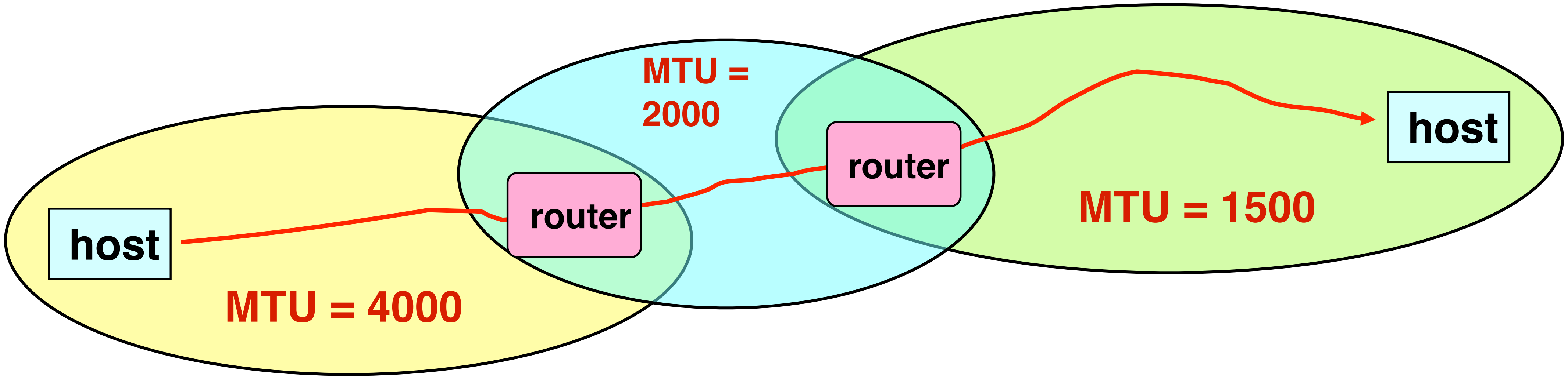
# IP Fragmentation



# IP Fragmentation

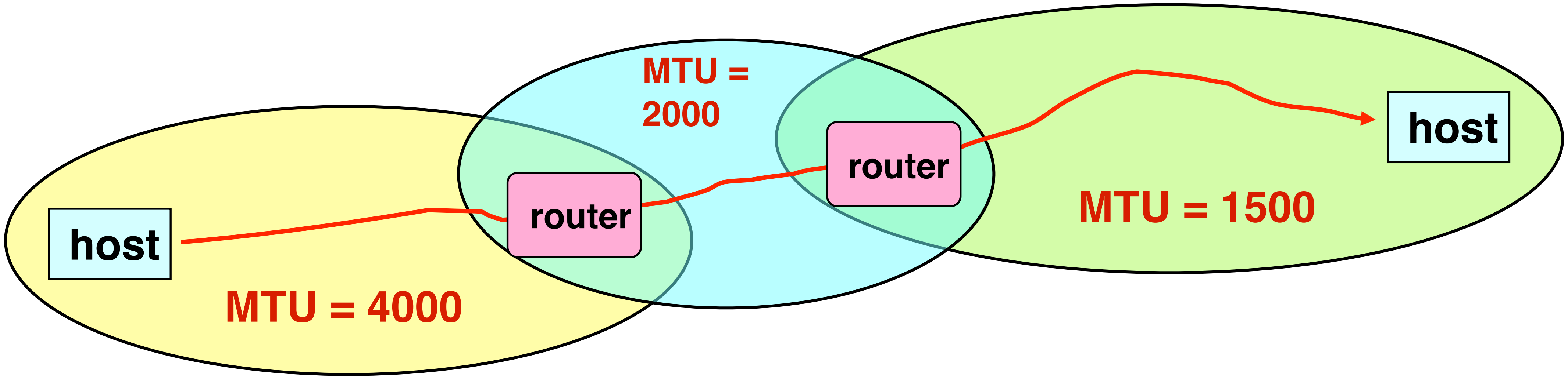


# IP Fragmentation



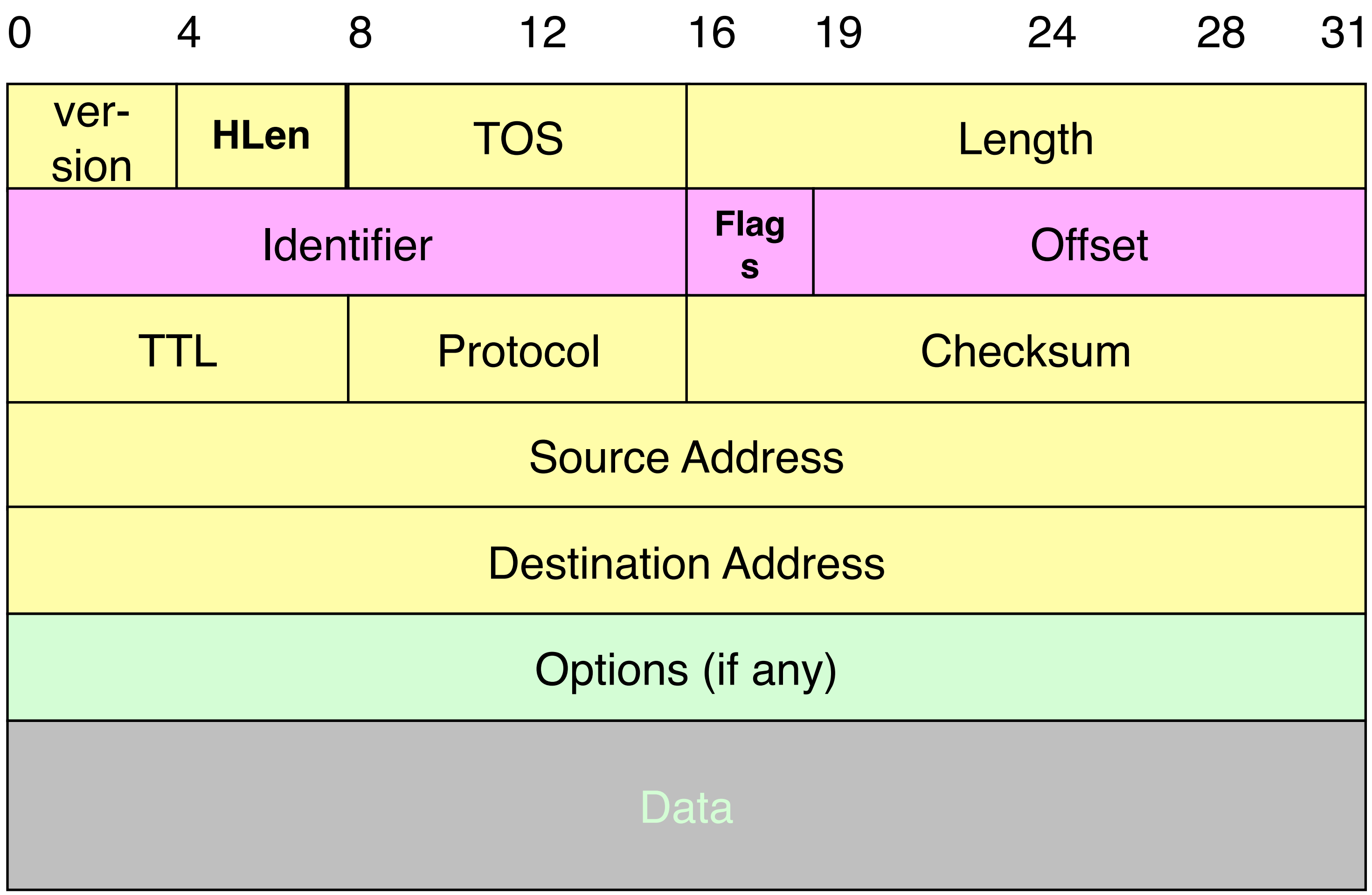

- Every Network has Own Maximum Transmission Unit (MTU)
  - Largest IP datagram it can carry within its own packet frame
    - E.g., Ethernet is 1500 bytes
  - Don't know MTUs of all intermediate networks in advance

# IP Fragmentation




- Every Network has Own Maximum Transmission Unit (MTU)
  - Largest IP datagram it can carry within its own packet frame
    - E.g., Ethernet is 1500 bytes
  - Don't know MTUs of all intermediate networks in advance
- IP Solution
  - When hit network with small MTU, fragment packets
    - Might get further fragmentation as proceed farther
  - Reassemble at the destination
    - If any fragment disappears, delete entire packet

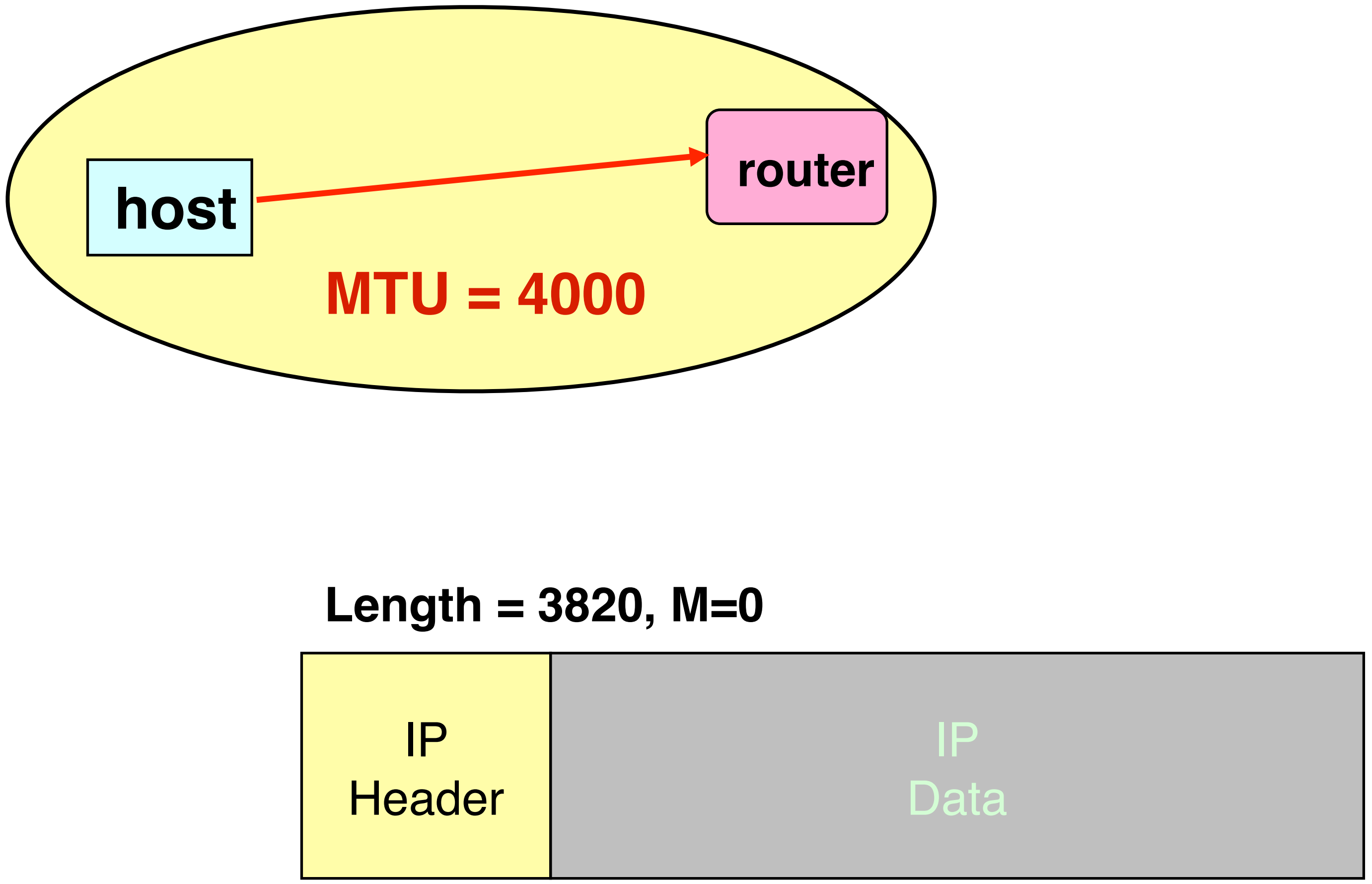
# IP Header Fields: Word 2



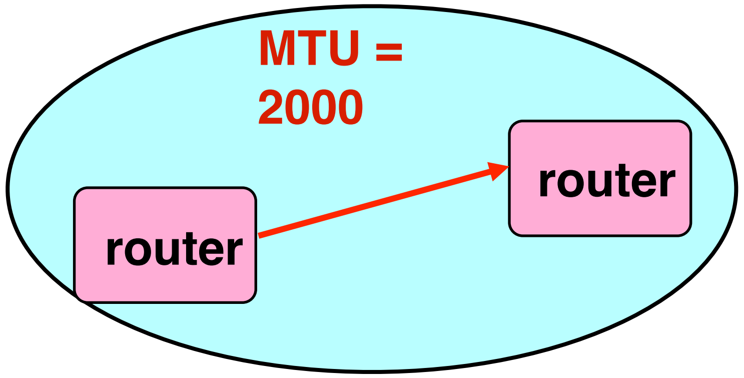

- Identifier
  - Unique identifier for original datagram
    - Typically, source increments counter every time sends packet
- Flags (3 bits)
  - M flag: This is not the last fragment
- Offset
  - Byte position of first byte in fragment ÷ 8
  - Byte position must be multiple of 8

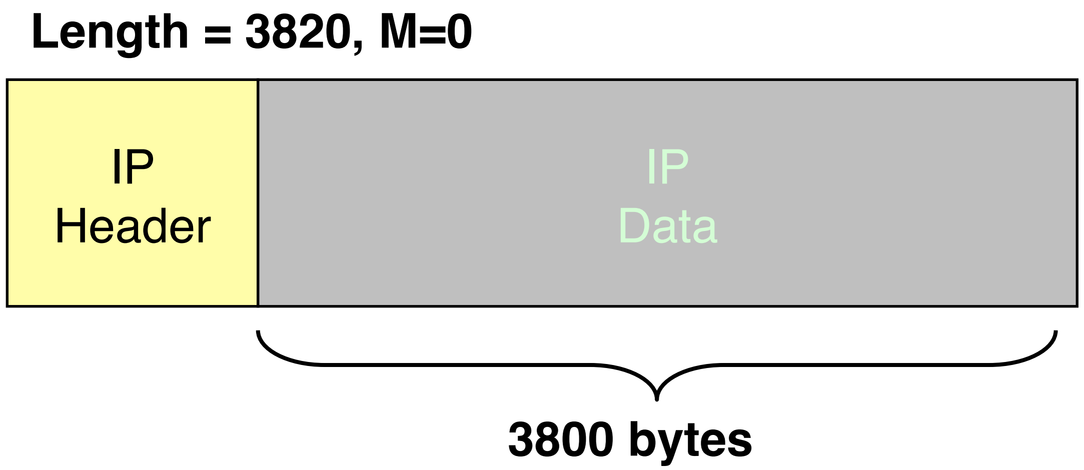
- Each fragment carries copy of IP header
  - All information required for delivery to destination
- All fragments comprising original datagram have same identifier
- Offsets indicate positions within datagram

# IP Fragmentation Example #1



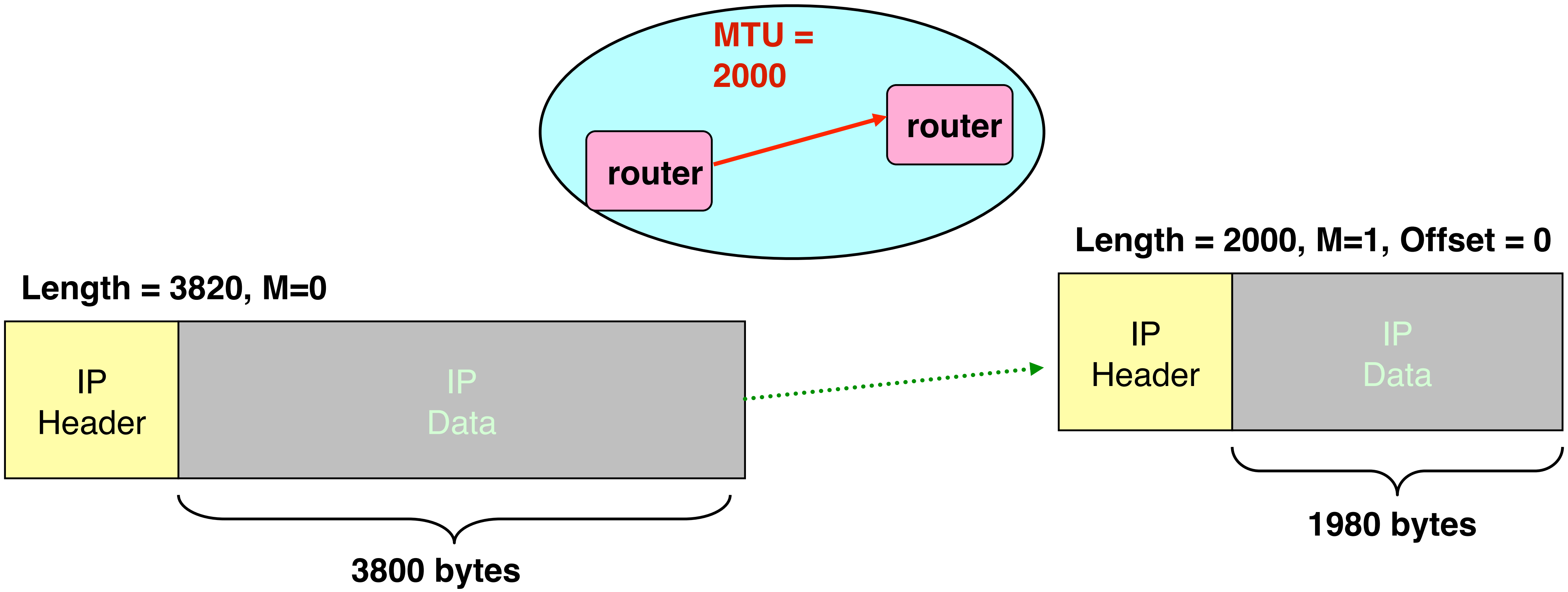
# IP Fragmentation Example #2



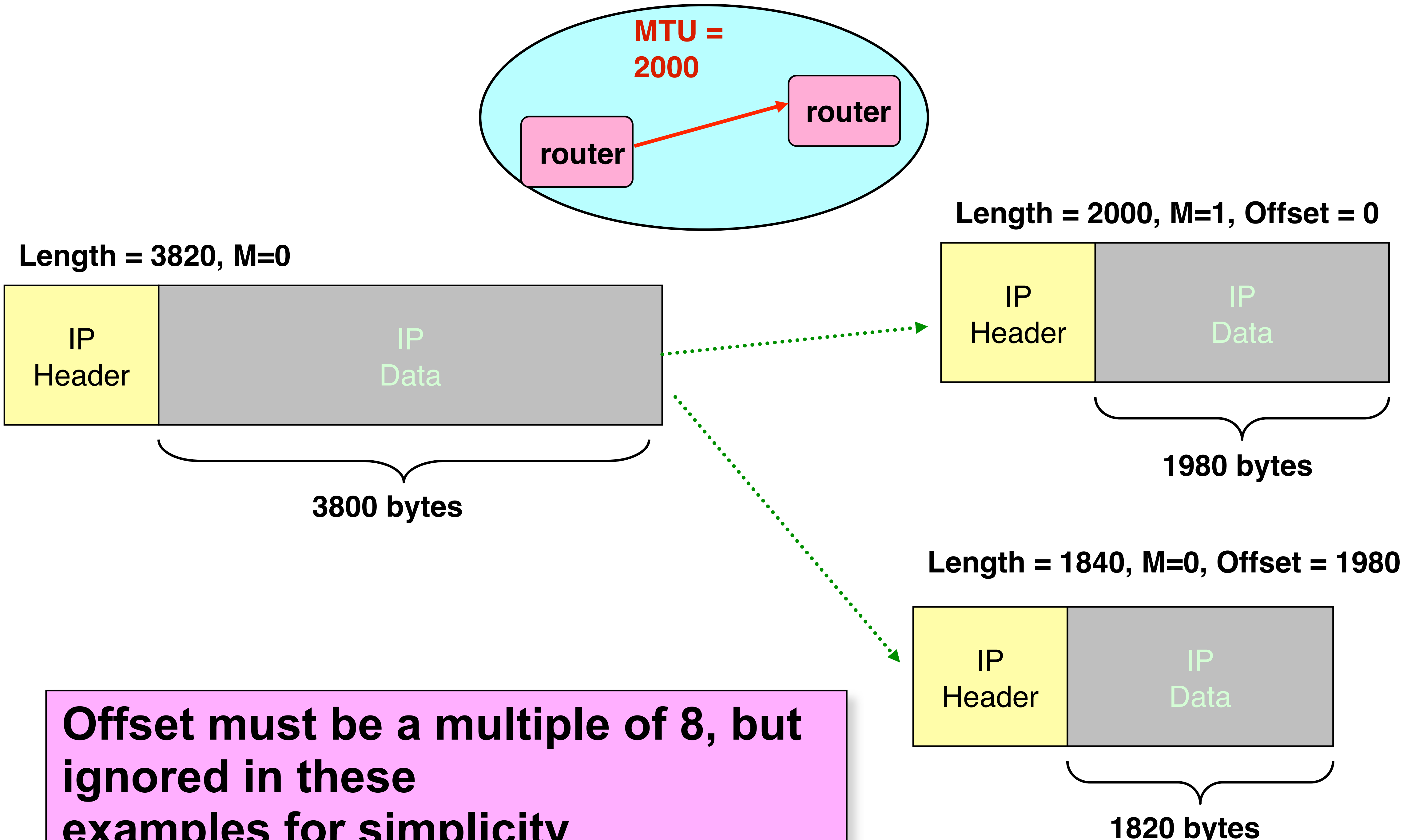

**Offset must be a multiple of 8, but ignored in these examples for simplicity**

# IP Fragmentation Example #2



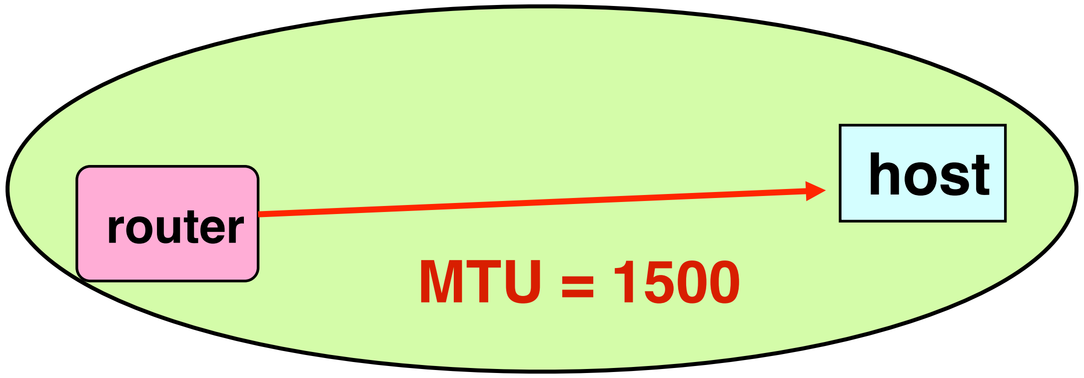

**Offset must be a multiple of 8, but ignored in these examples for simplicity**

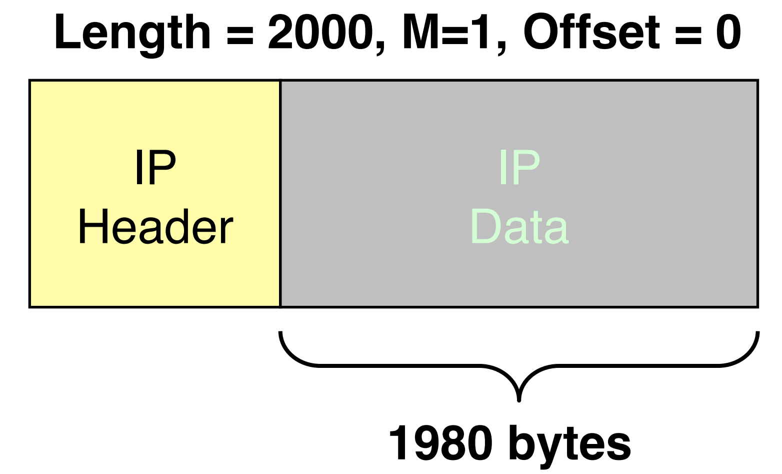
# IP Fragmentation Example #2

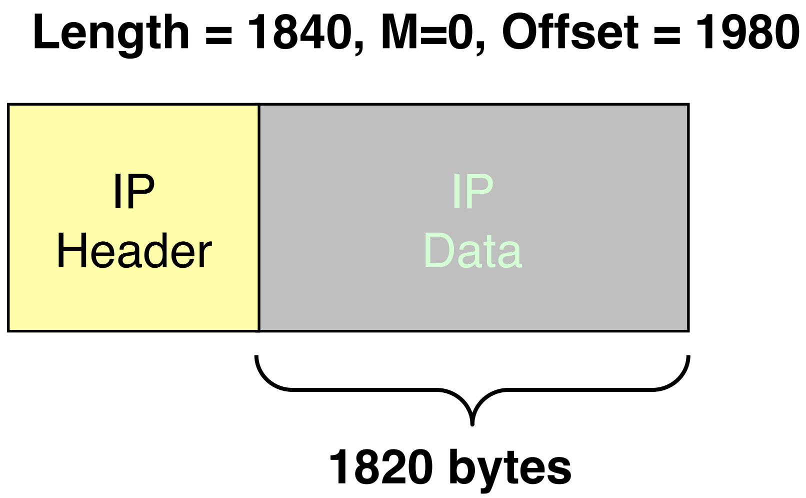



**Offset must be a multiple of 8, but ignored in these examples for simplicity**

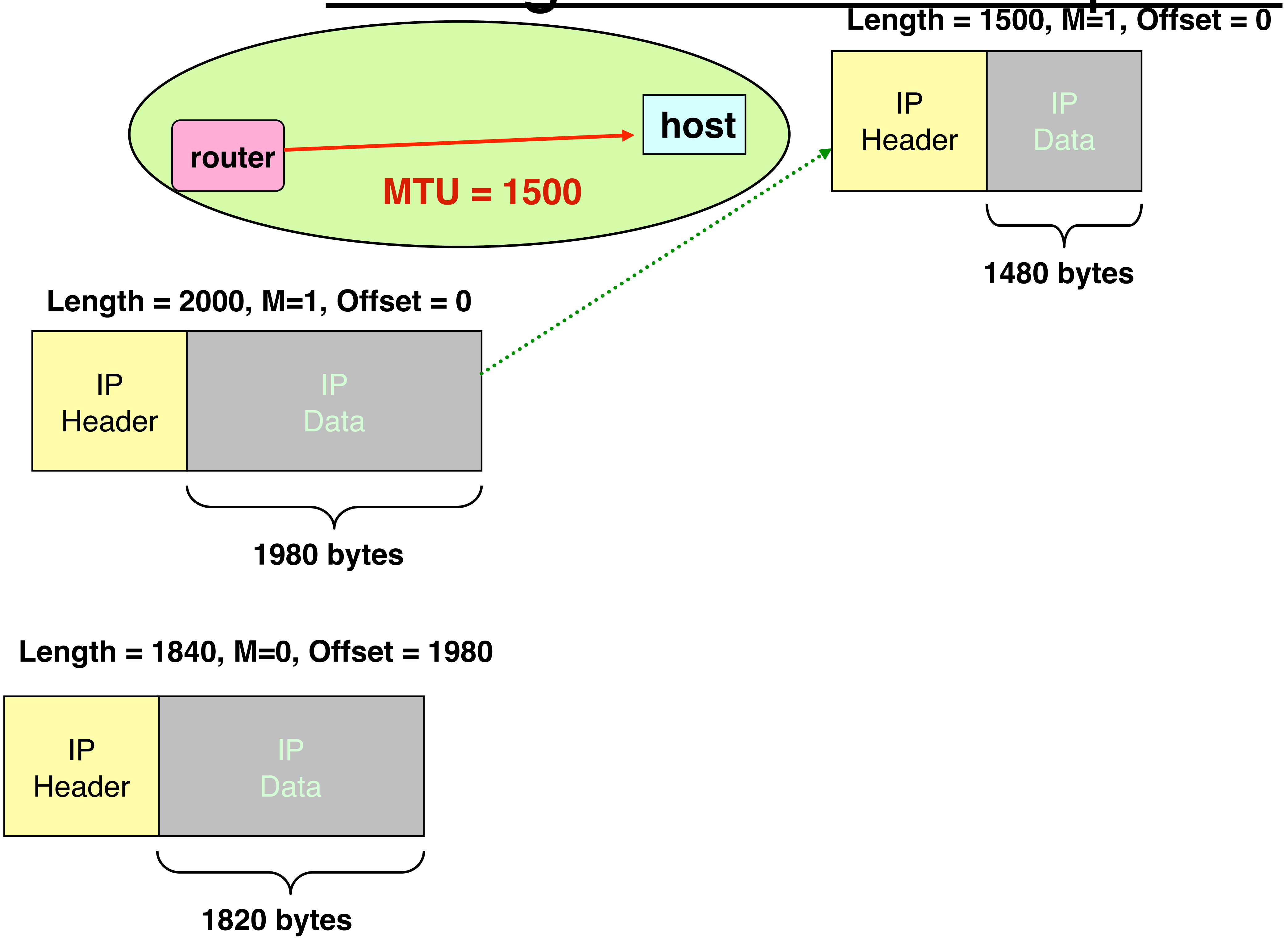
# IP Fragmentation Example #3



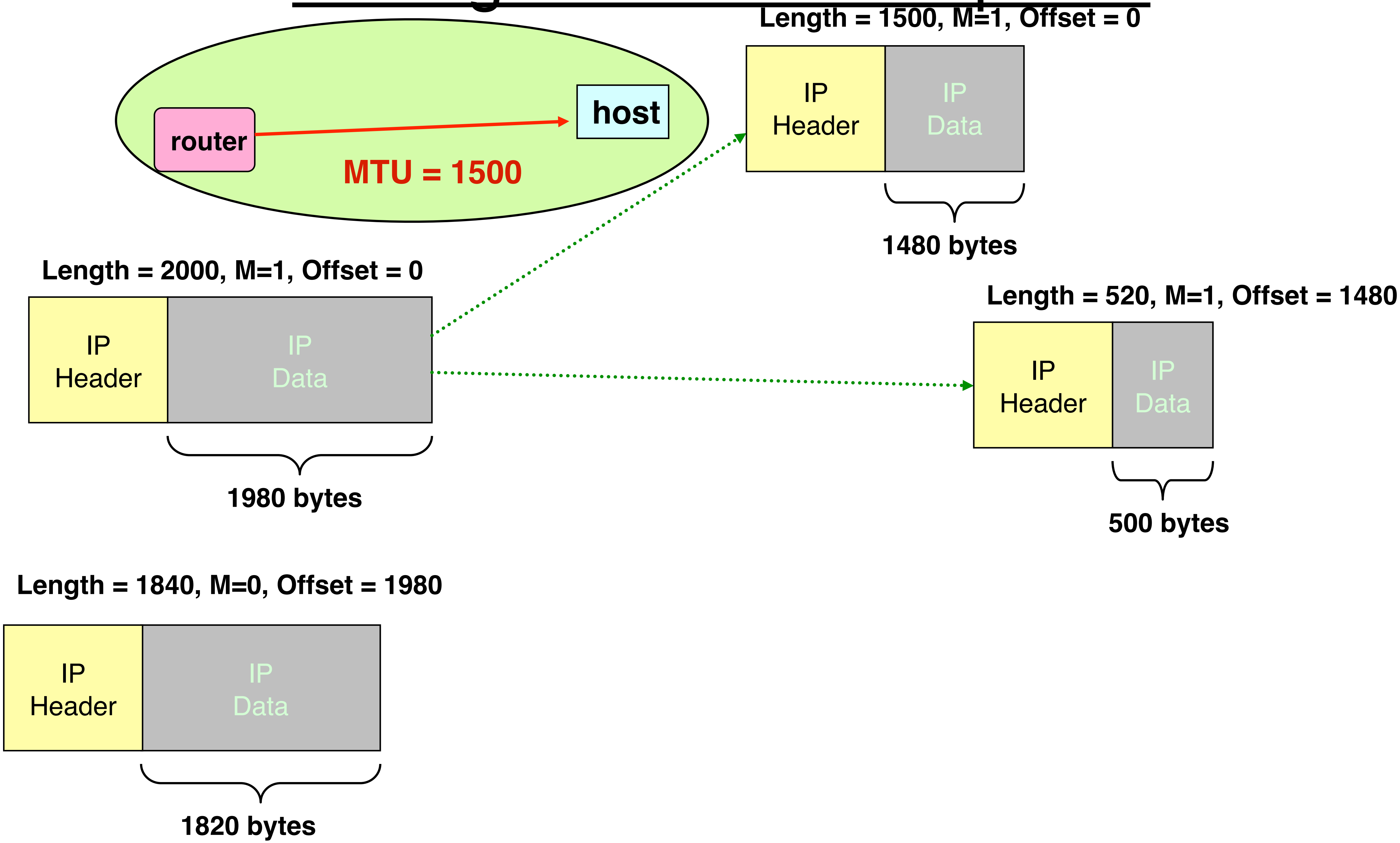



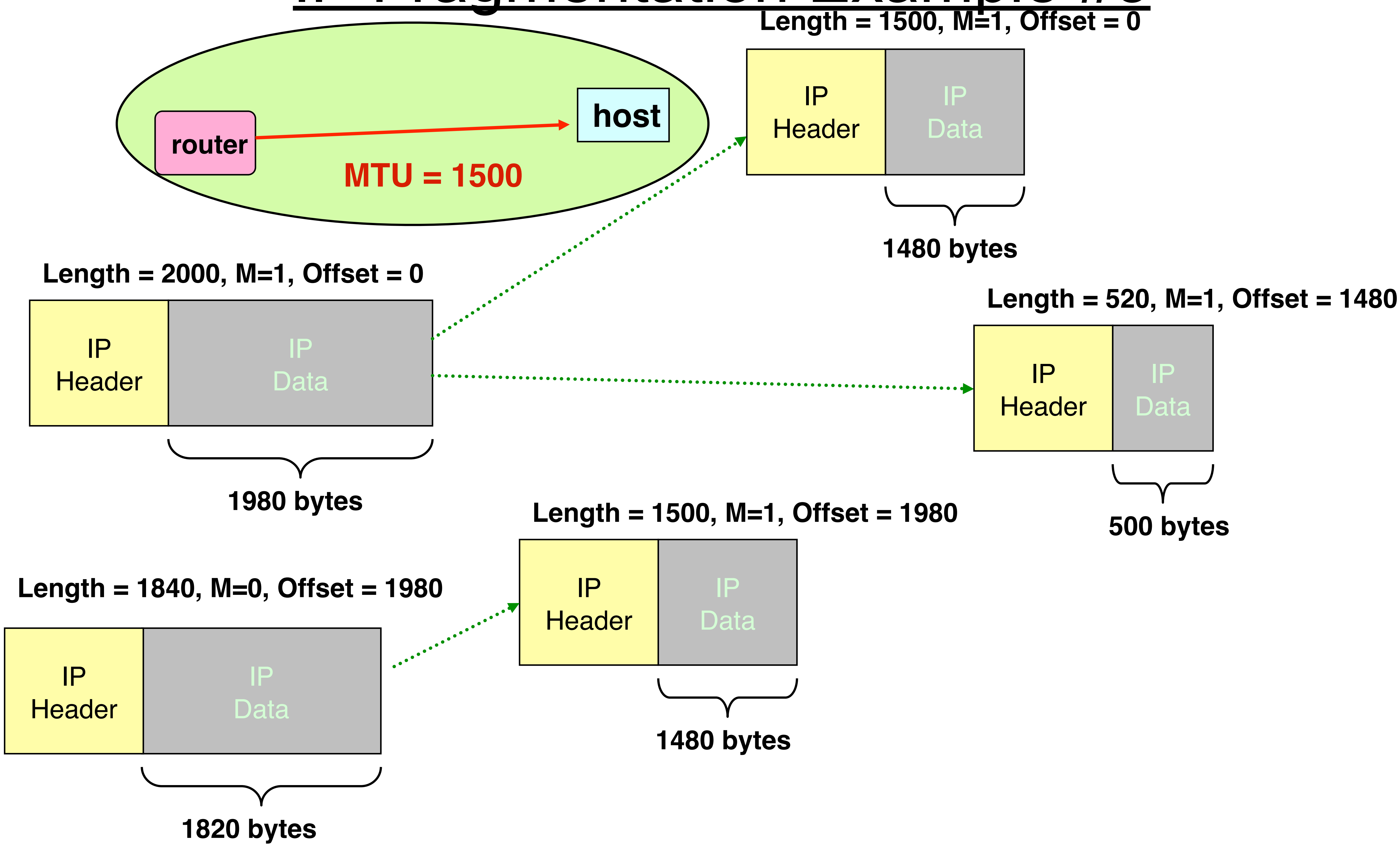
# IP Fragmentation Example #3



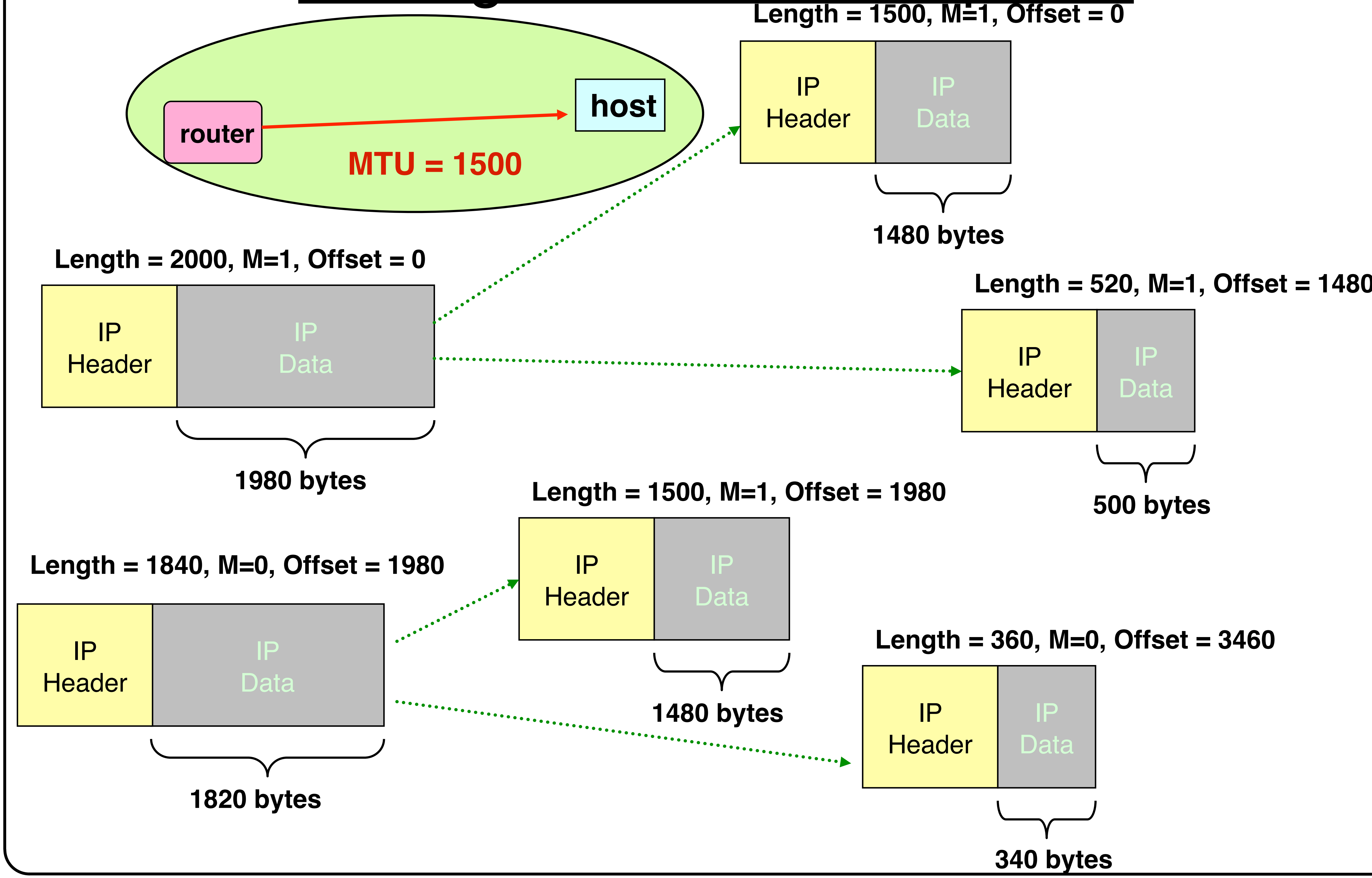
# IP Fragmentation Example #3



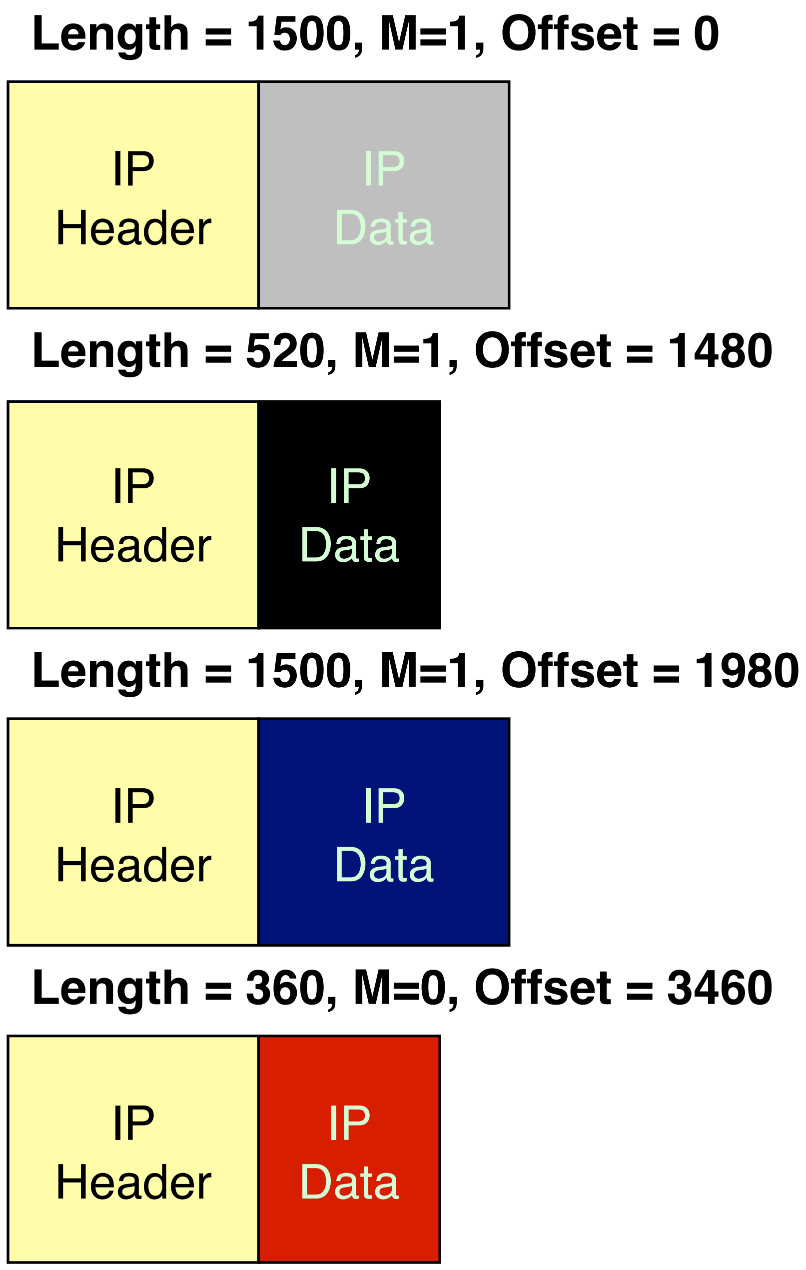
# IP Fragmentation Example #3



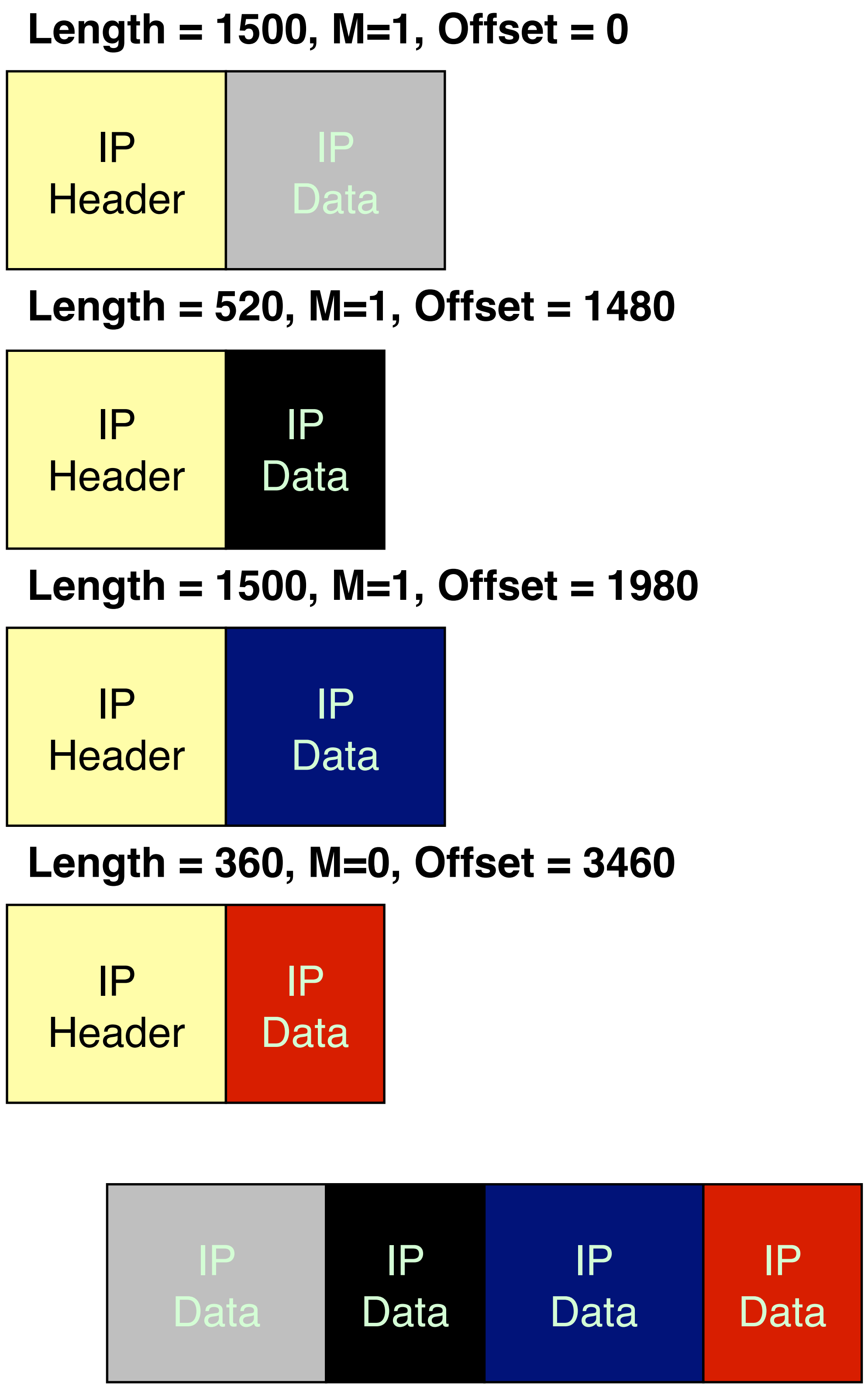
# IP Fragmentation Example #3



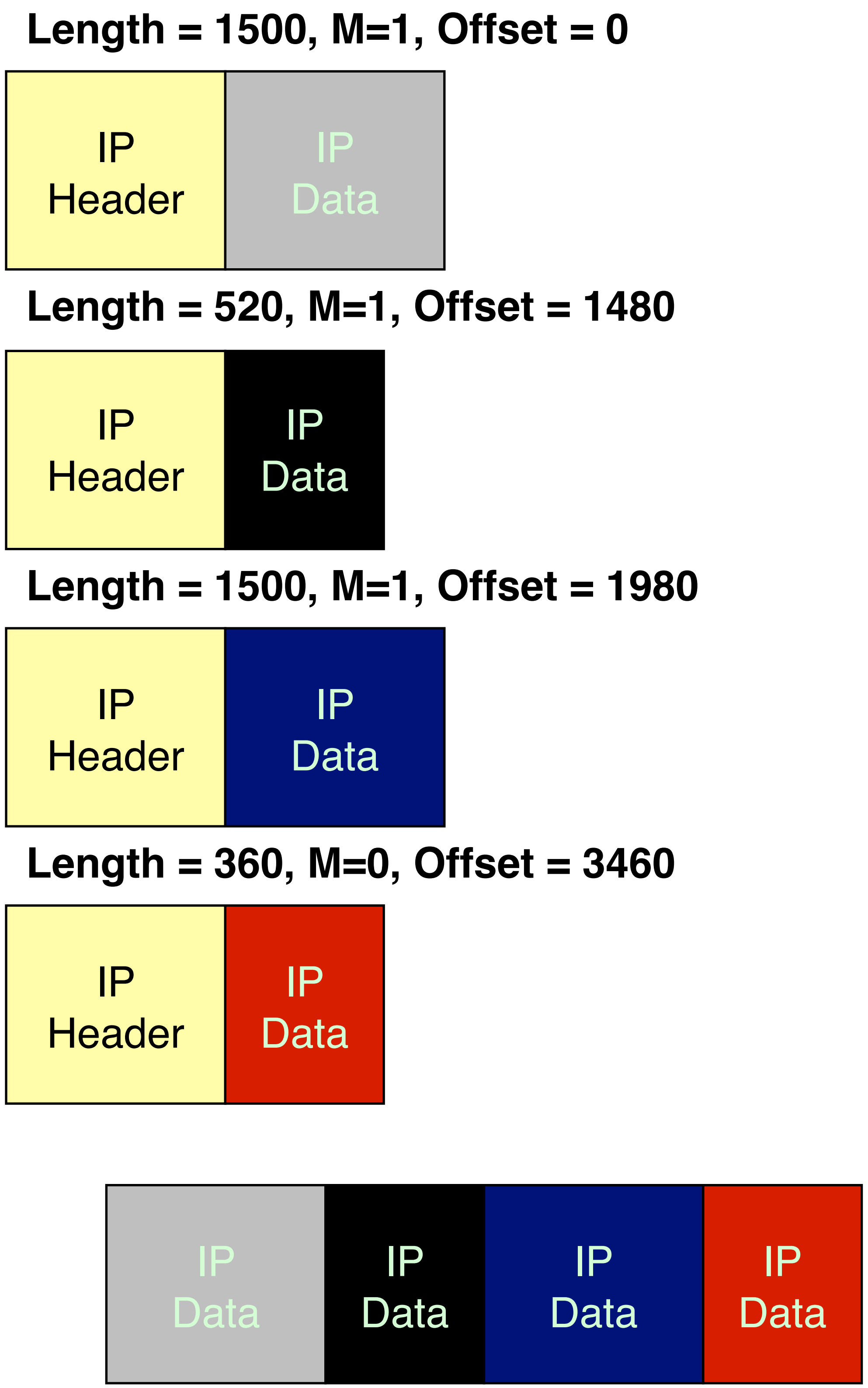
# IP Reassembly



# IP Reassembly

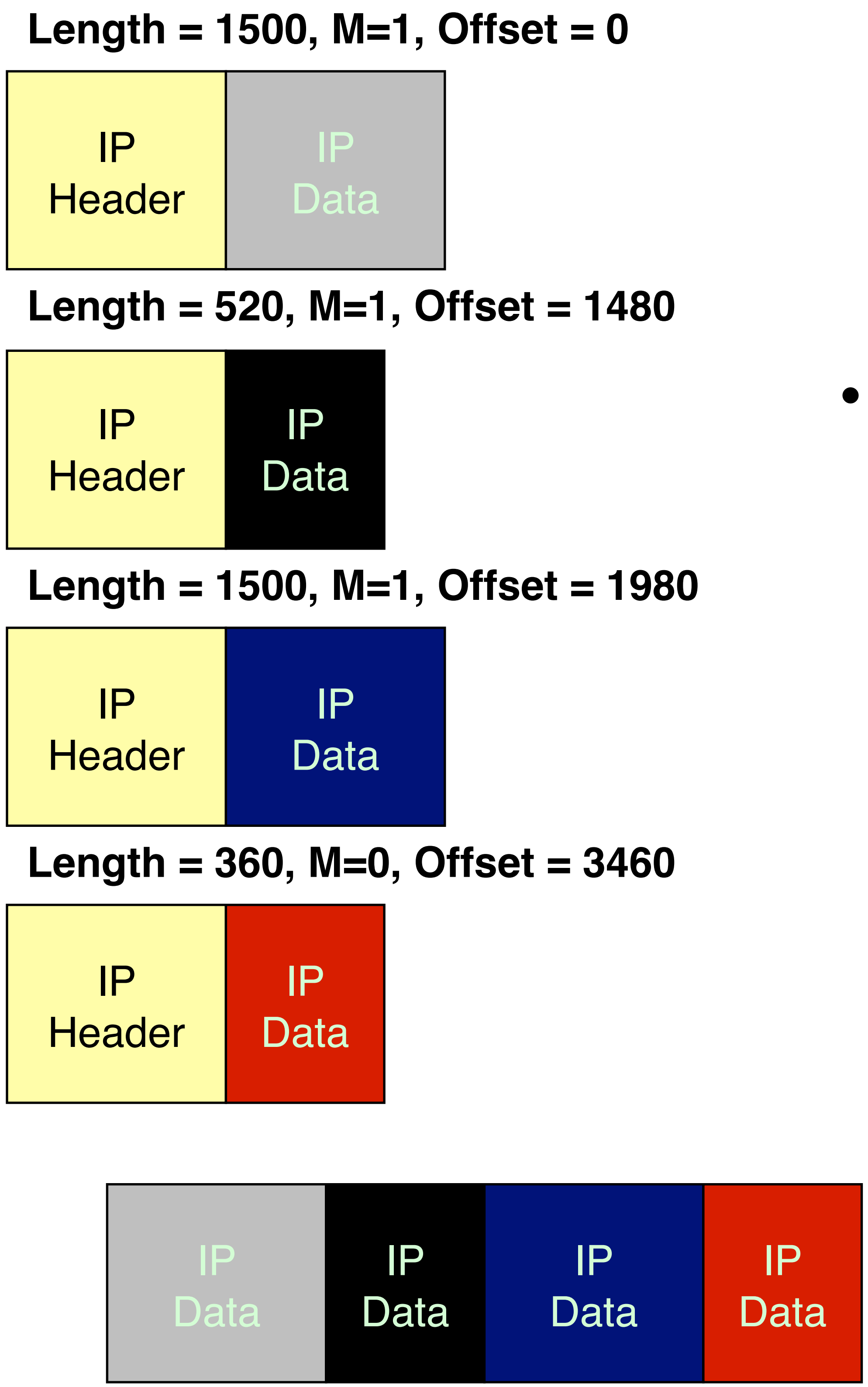


# IP Reassembly




- Performed at final destination
- Fragment with M=0 determines overall length
  - (360-20)+3460

# IP Reassembly




- Performed at final destination
- Fragment with M=0 determines overall length
  - (360-20)+3460
- Challenges
  - Fragments might arrive out-of-order
    - Don’t know how much memory required until receive final fragment
  - Some fragments may be duplicated
    - Keep only one copy
  - Some fragments may never arrive
    - After a while, give up entire process
  - Significant memory management issues

# Frag. & Reassembly Concepts

- Demonstrates Many Internet Concepts

- Decentralized
  - Every network can choose MTU
- Connectionless Datagram Protocol
  - Each (fragment of) packet contains full routing information
  - Fragments can proceed independently and along different routes
- Fail by Dropping Packet
  - Destination can give up on reassembly
  - No need to signal sender that failure occurred
- Keep Most Work at Endpoints
  - Reassembly

# Frag. & Reassembly Reality

- Reassembly Fairly Expensive
  - Copying, memory allocation
  - Want to avoid
- MTU Discovery Protocol
  - Protocol to determine MTU along route
    - Send packets with “don’t fragment” flag set
    - Keep decreasing message lengths until packets get through
    - May get a “can’t fragment error” message from router which contains the correct MTU
  - Assumes every packet will follow same route
    - Routes tend to change slowly over time
- Common Theme in System Design
  - Fragmentation is handled as a special case by slower general processor in router
  - Assure correctness by implementing complete protocol
  - Optimize common cases to avoid full complexity