The IP Data Plane

CS168, Fall 2014

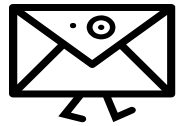
Sylvia Ratnasamy

http://inst.eecs.berkeley.edu/~cs168/fa14/

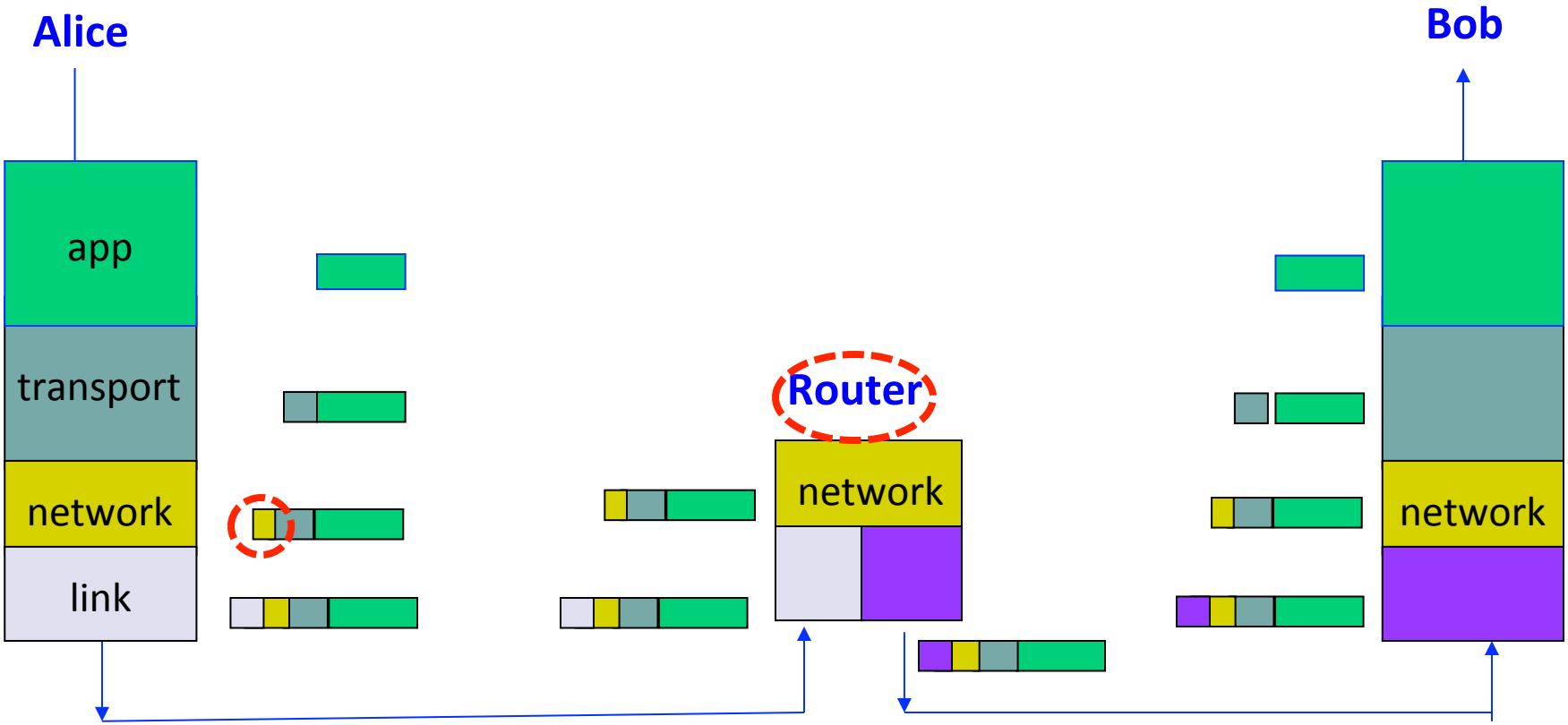
The IP layer

- So far: focused mostly on routing protocols
  - how routers discover and select end-to-end paths
  - part of a network's control plane

- Today: the data plane
  - what data packets look like at the IP layer
  - how routers forward these IP packets



Recall: Layer Encapsulation



Recall: IP packet

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?

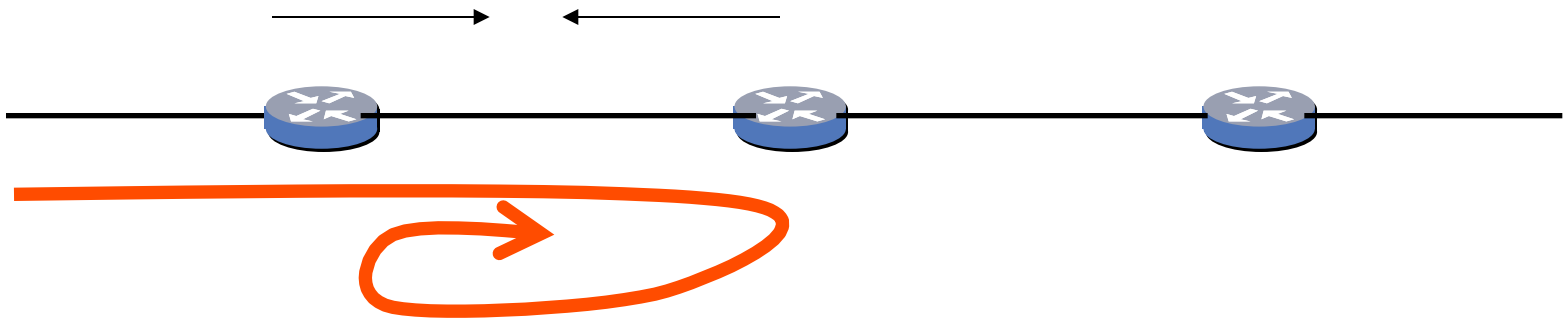
- Parse packet
  - *IP version number (4 bits), packet length (16 bits)*
- Carry packet to the destination
  - *Destination's IP address (32 bits)*
- Deal with problems along the way
  - loops:
  - corruption:
  - packet too large:

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 looong time
  - left unchecked would accumulate to consume all capacity



- Time-to-Live (TTL) Field (8 bits)
  - decremented at each hop, packet discarded if reaches 0
  - …and "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?
  - Why only header?

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 shortly…

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), fragmentation (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

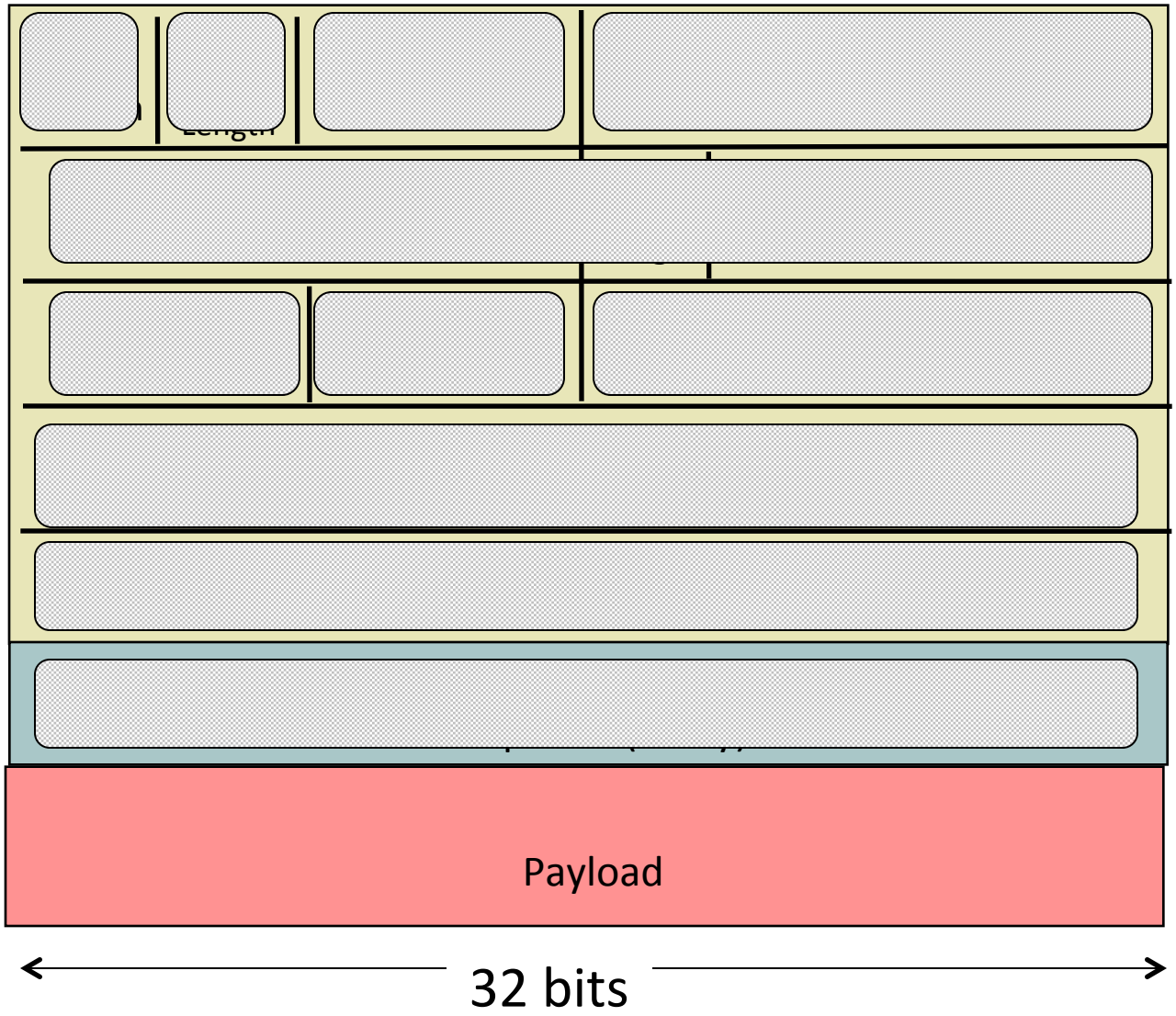
- Optional directives to the network
  - not used very often
  - 16 bits of metadata + option-specific data

- 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), fragmentation (32 bits)*
- Accommodate evolution
  - *version number (4 bits) (+ fields for special handling)*
- Specify any special handling
  - *ToS (8 bits), Options (variable length)*

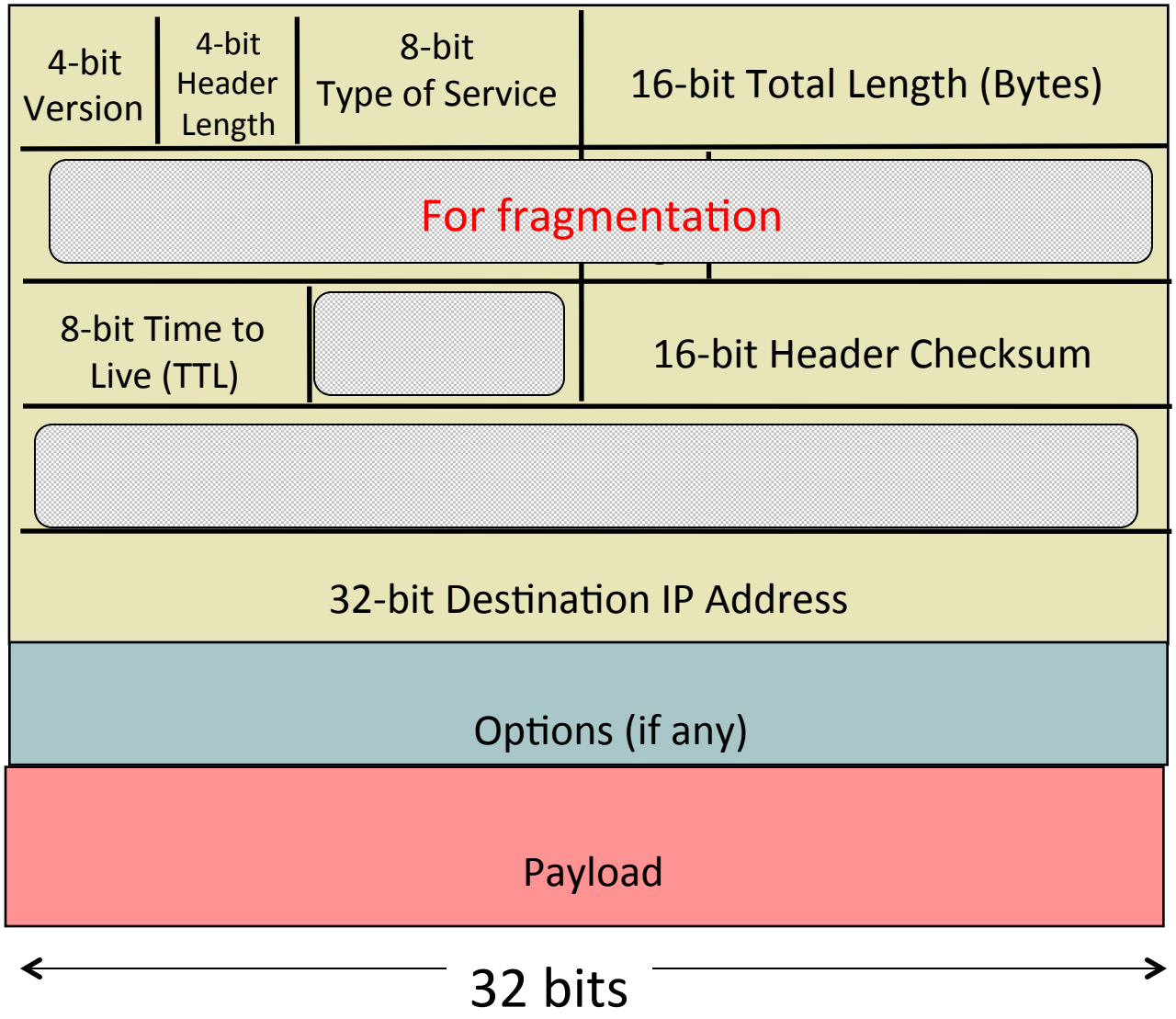
IP Packet Structure



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

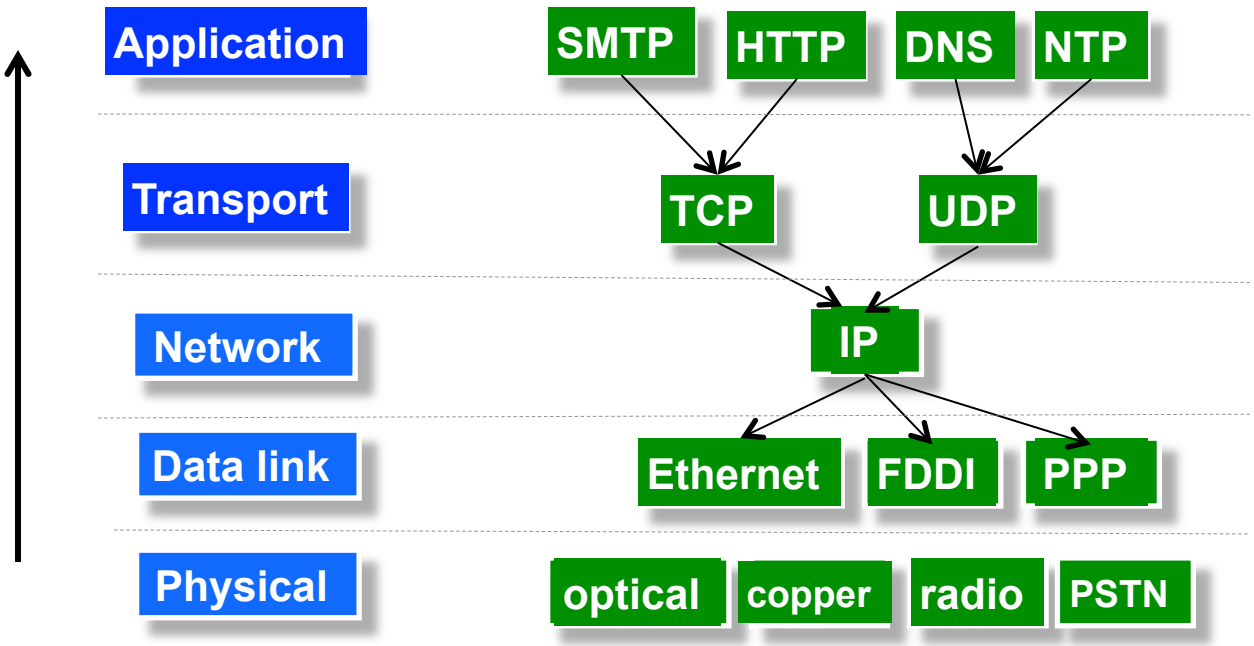


What are these tasks? (at the destination end-system)

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

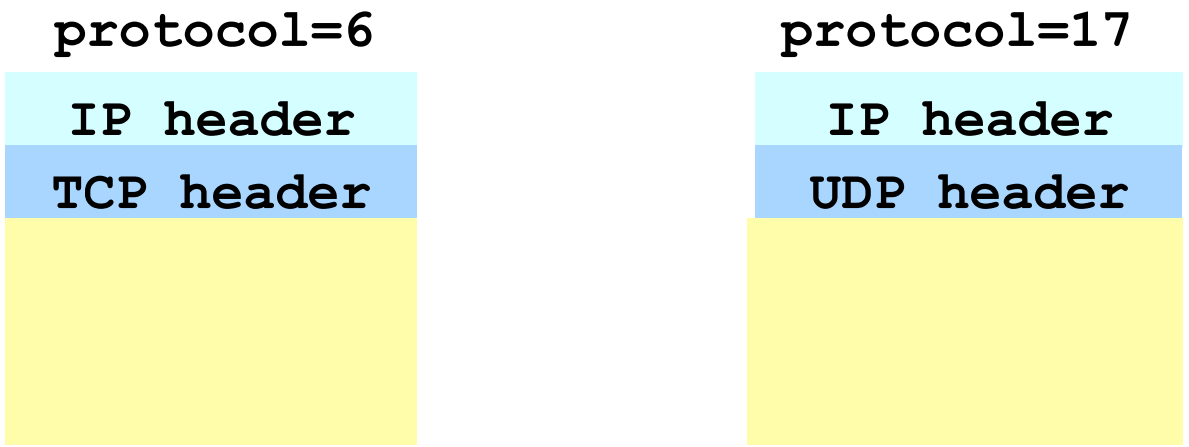
Telling End-Host How to Handle Packet

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



Telling End-Host How to Handle Packet

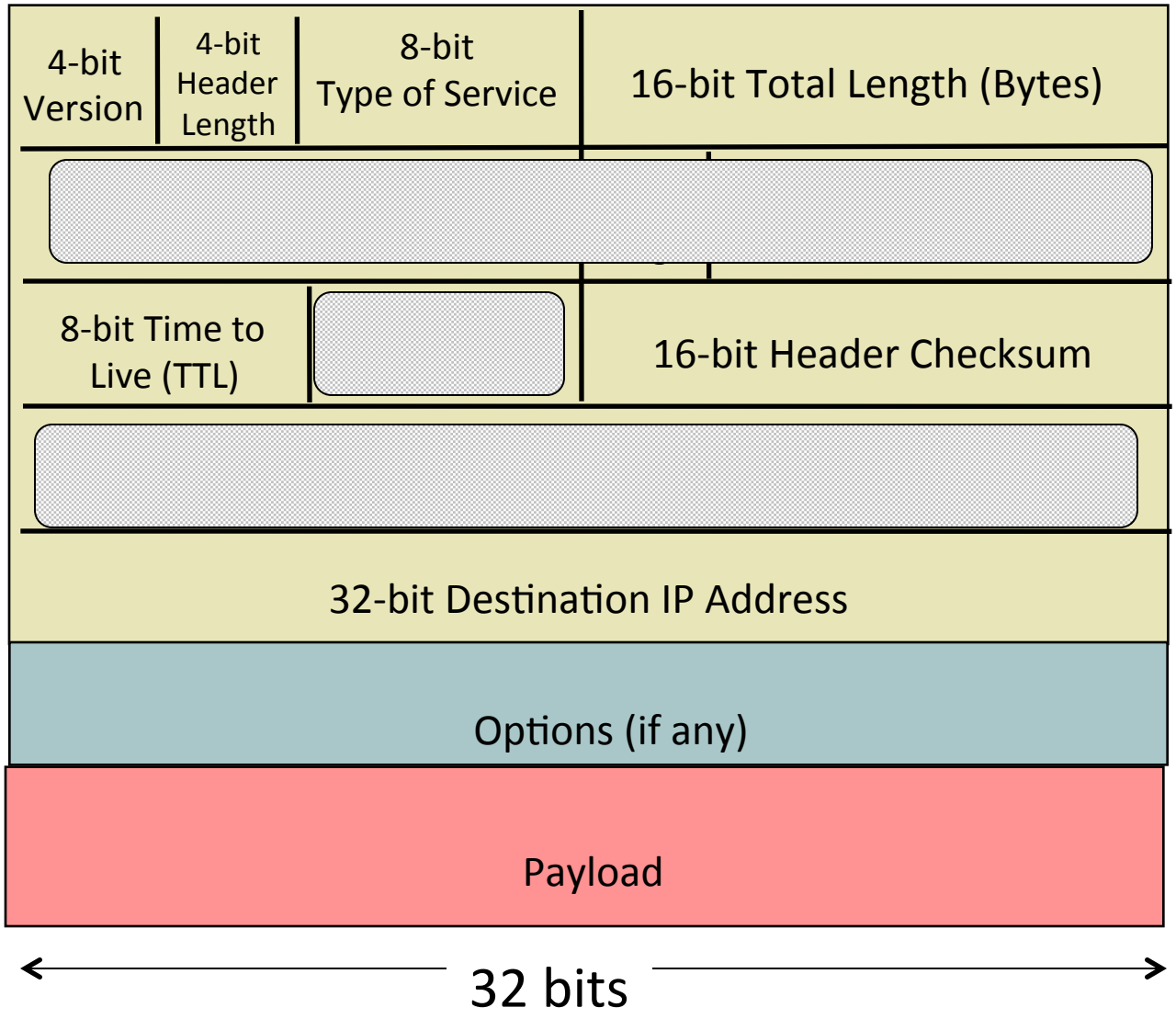
- Protocol (8 bits)
  - Identifies the higher-level protocol
  - Important for de-multiplexing at receiving host
- Most common examples
  - E.g., “6” for the Transmission Control Protocol (TCP)
  - E.g., “17” for the User Datagram Protocol (UDP)



What are these 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

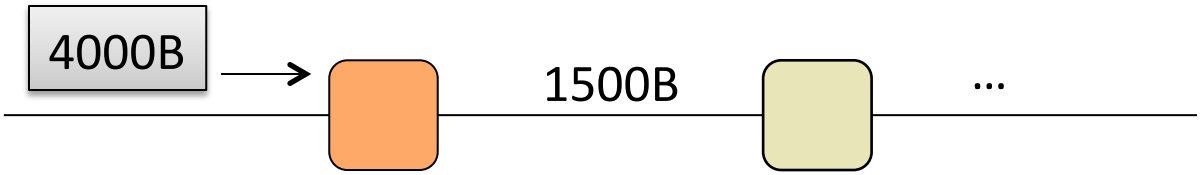


A closer look at fragmentation

- 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

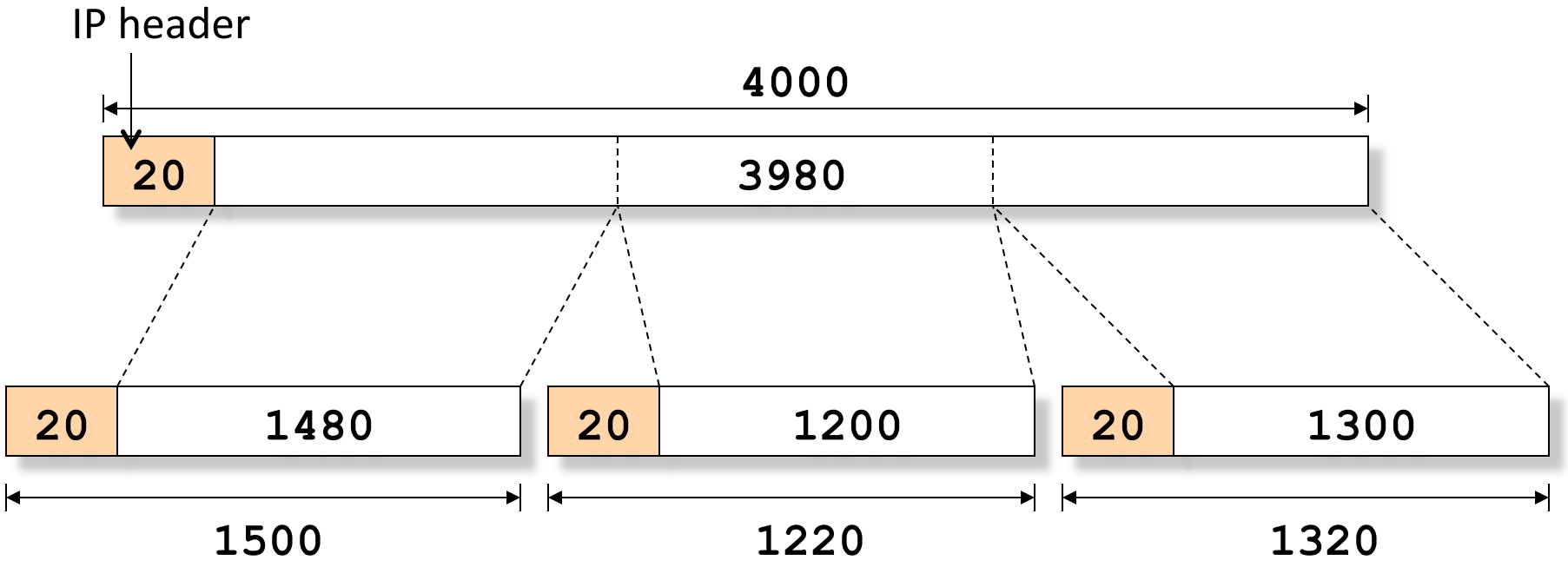
Example of fragmentation

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



Example of fragmentation

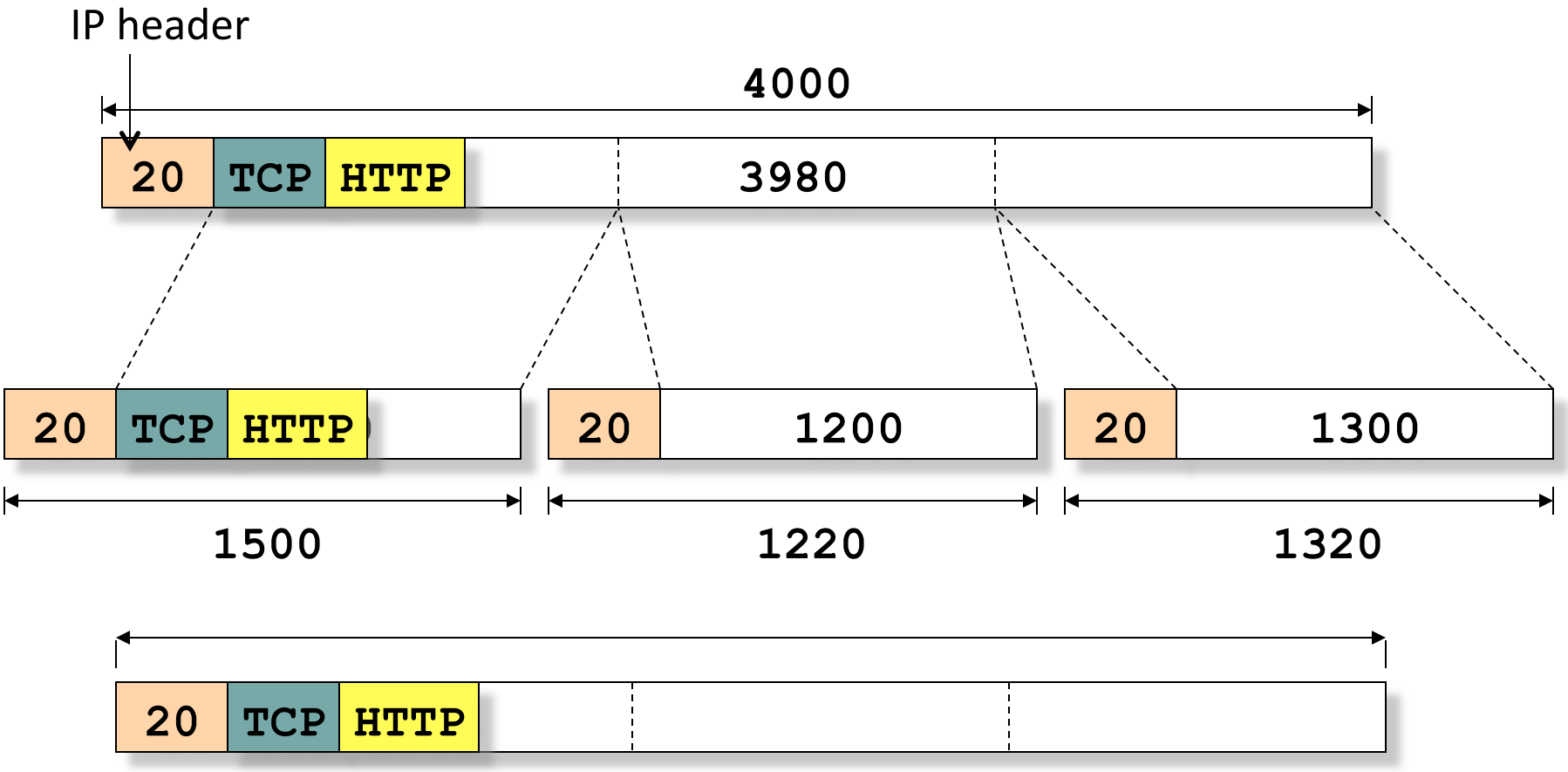
- A 4000 byte packet crosses a link w/ MTU=1500B



DIY exercise in header engineering

- Take 5 minutes to design fragmentation scheme
  - Work with your neighbors
- At the end of 5 minutes, be ready to tell me:
  - How many fields you need?
  - What are they and why?
- Don’t look for the answer elsewhere, the goal is to think about the problem!

Why reassemble?



Must reassemble before sending the packet to the 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 get lost?

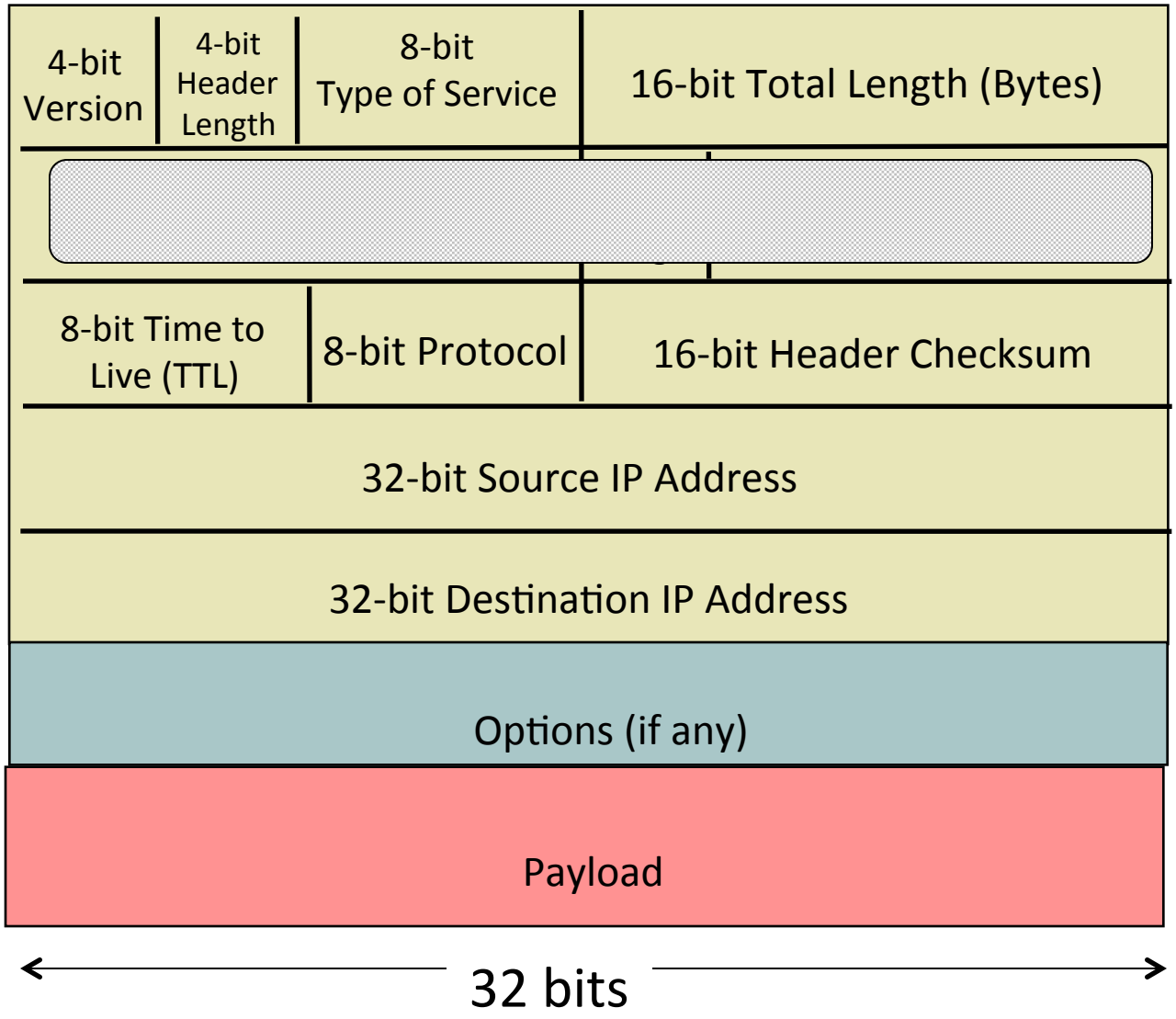
  → need some form of sequence number or offset?

- 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**:
  - **R**eserved: ignore
  - **D**F: don't fragment
    - *may trigger error message back to sender*
  - **M**F: more fragments coming
- **Offset**: portion of original payload this fragment contains
  - in 8-byte units

IP Packet Structure

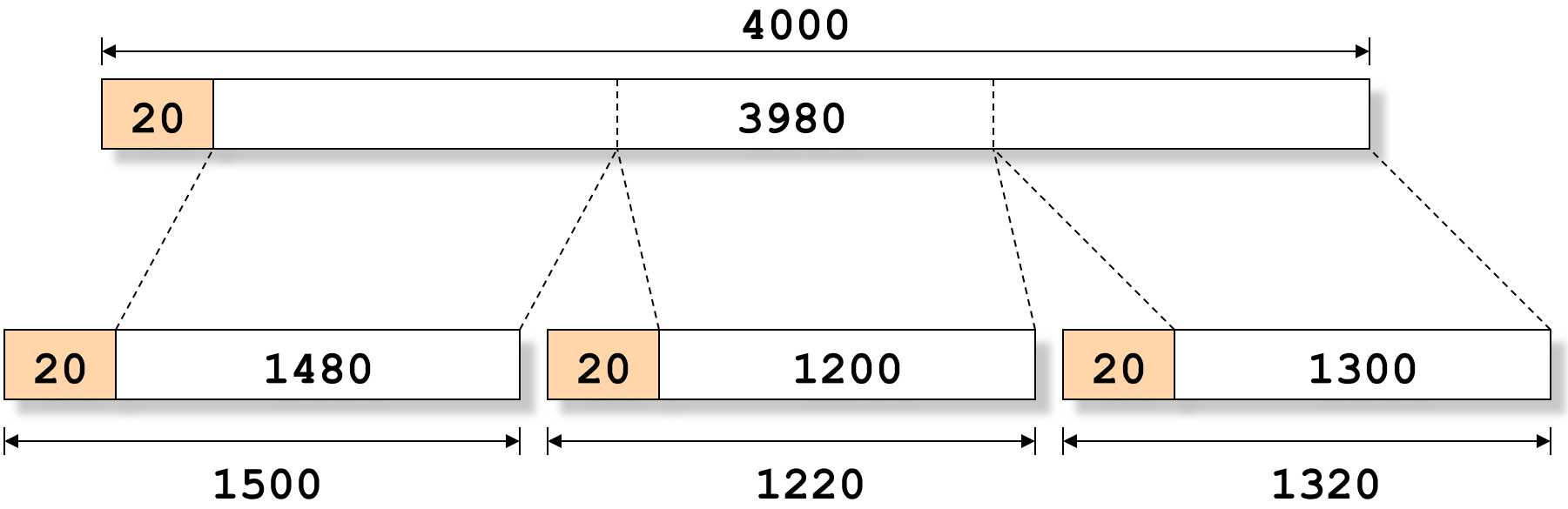


Why This Works

- 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

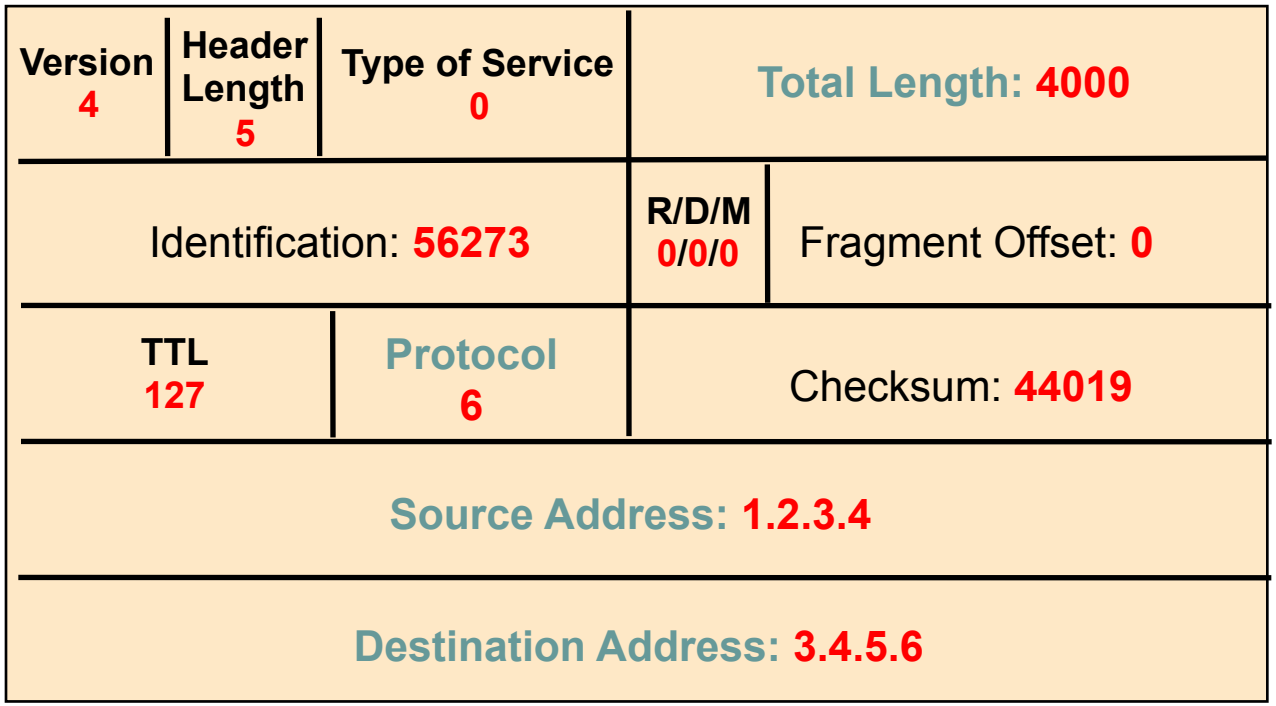
Example of fragmentation (contd.)

- Packet split into 3 pieces
- Example:



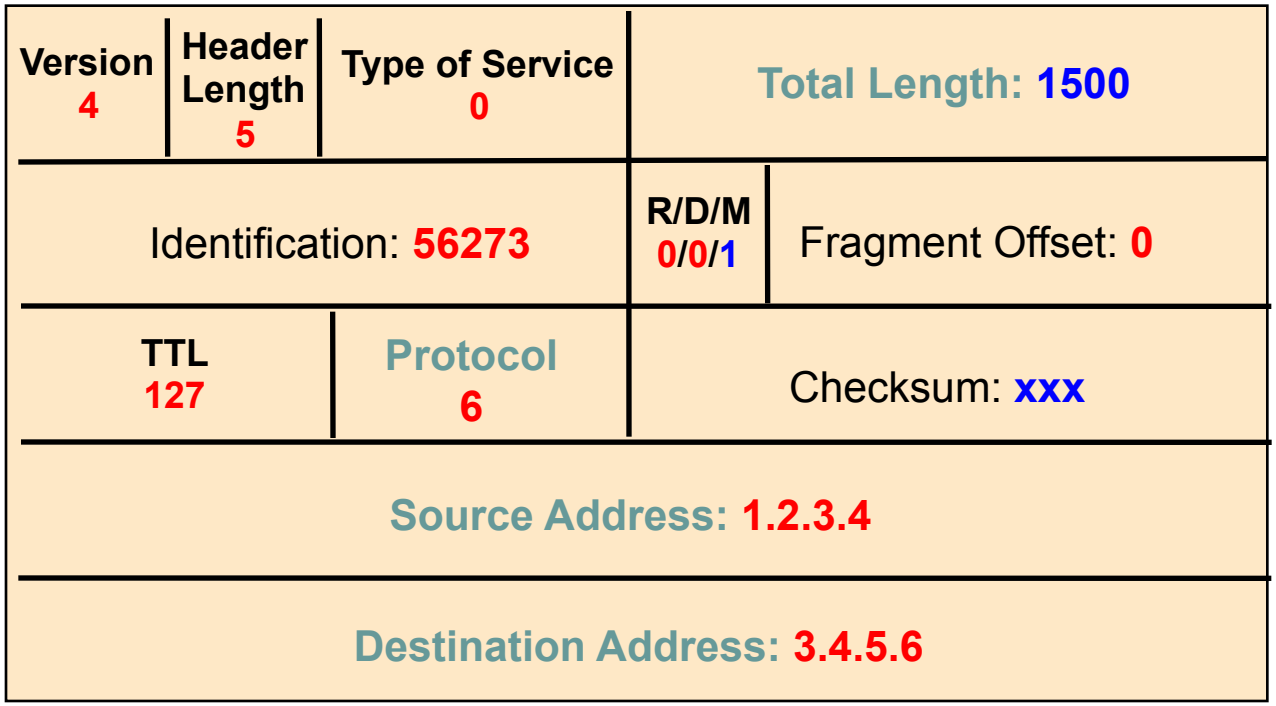
Example of fragmentation, contd.

- 4000 byte packet from host 1.2.3.4 to 3.4.5.6 …
- … traverses a link with MTU 1,500 bytes



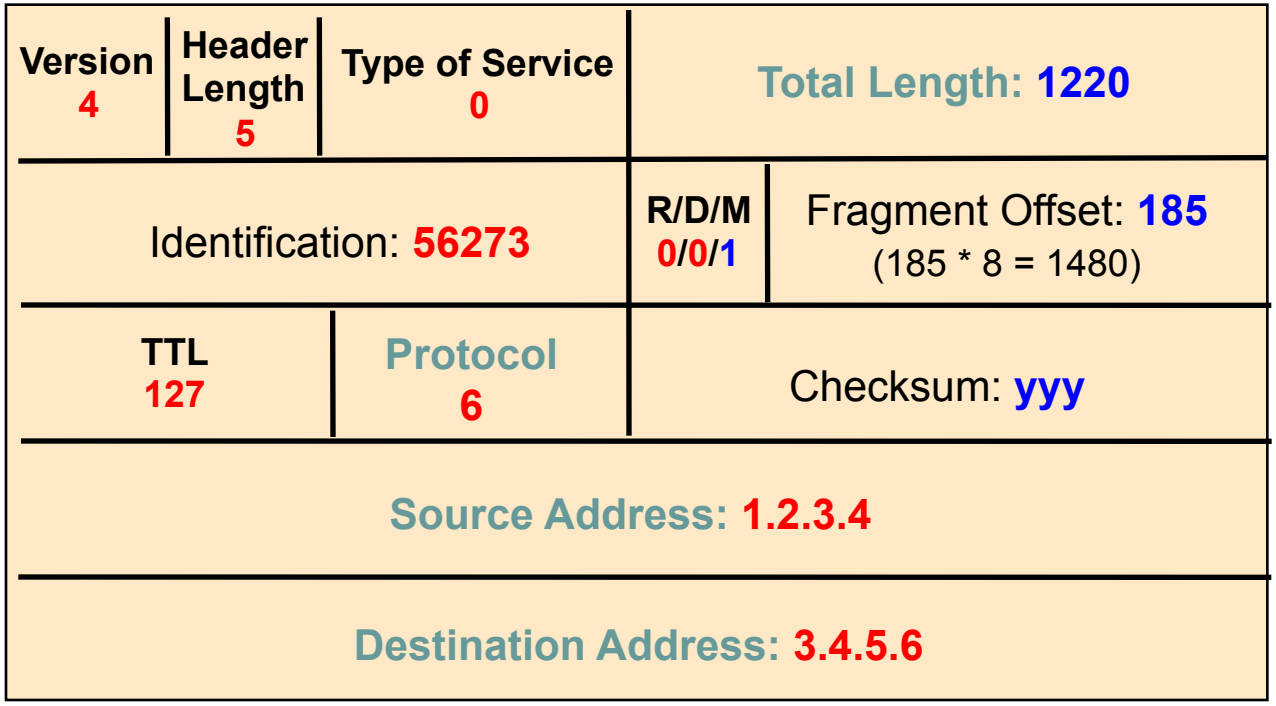
Example of fragmentation, contd.

- Datagram split into 3 pieces. Possible first piece:



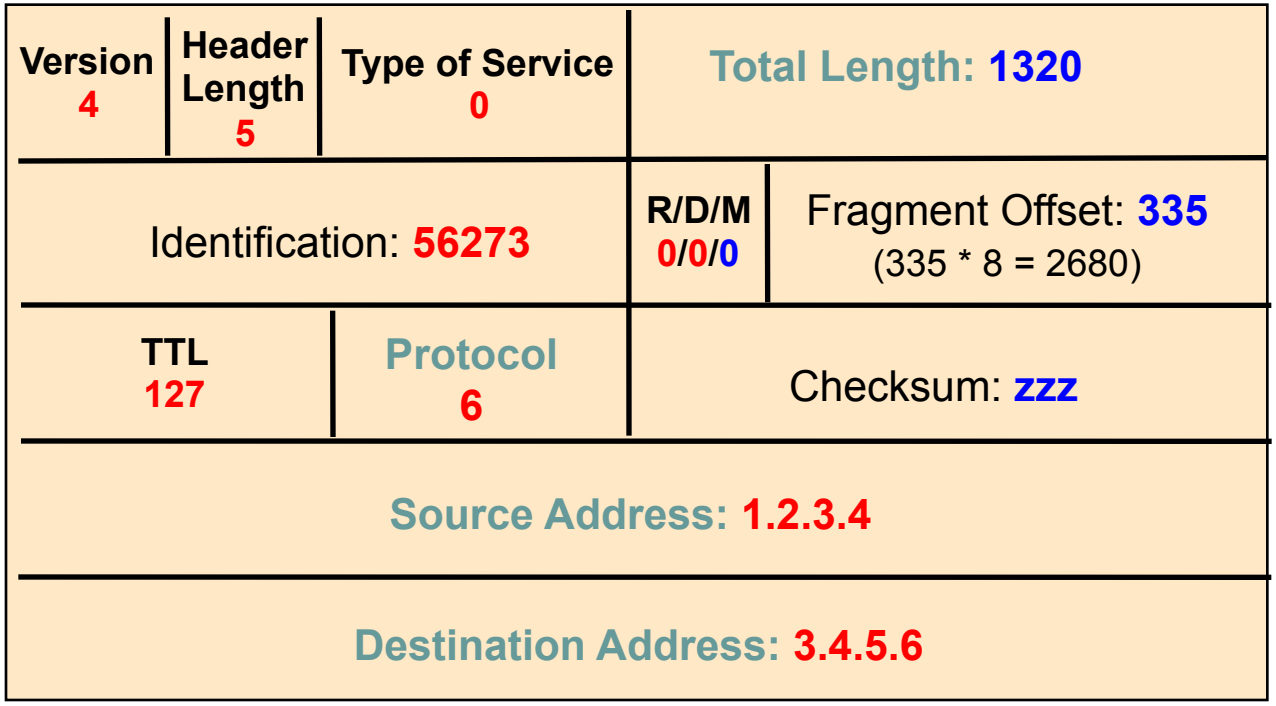
Example of fragmentation, contd.

- Possible second piece: Frag#1 covered 1480bytes

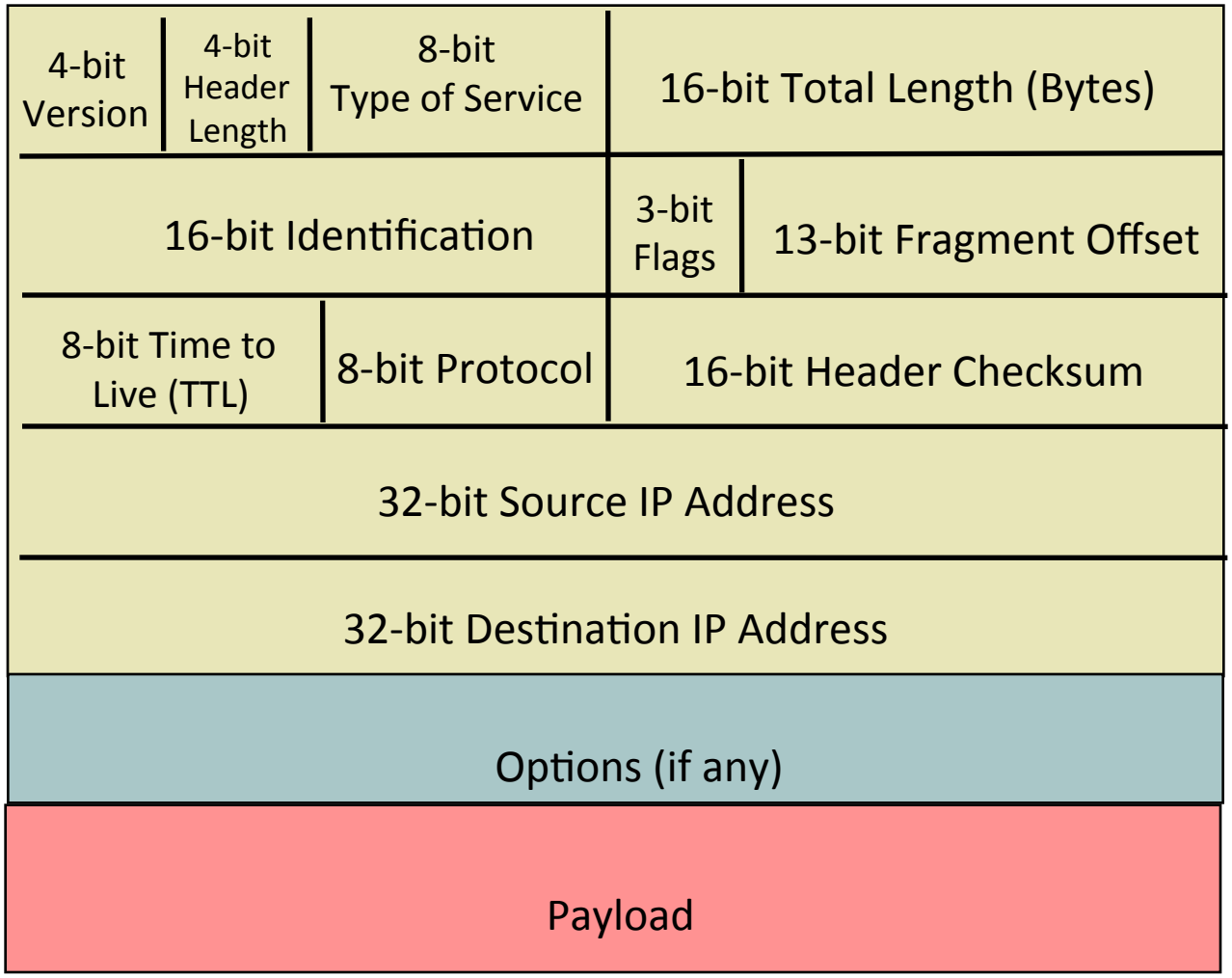


Example of fragmentation, contd.

- Possible third piece: 1480+1200 = 2680



IP Packet Structure

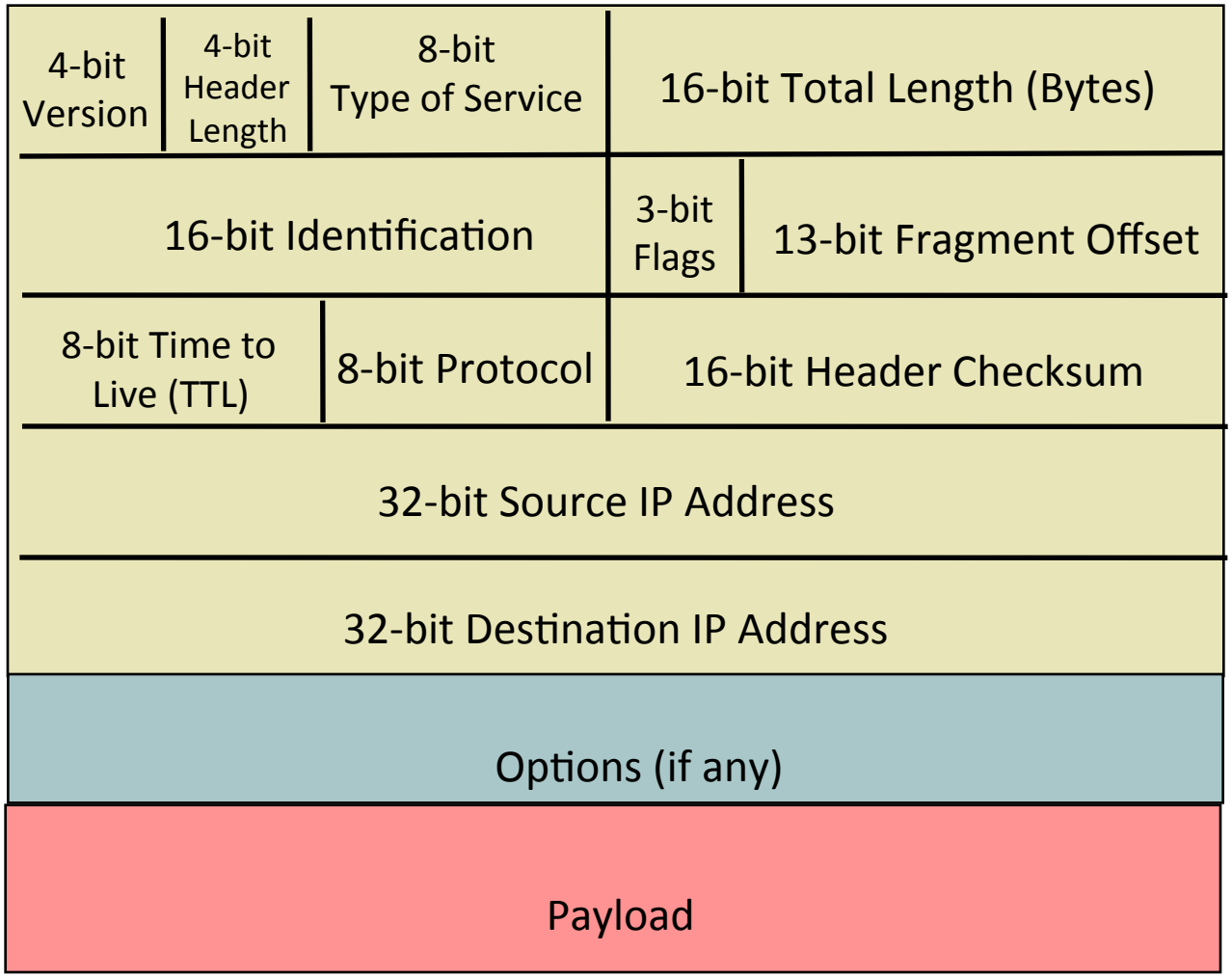


Quick Security Analysis of IP Header

Focus on Sender Attacks

- Vulnerabilities sender can exploit
- Ignore (for now) attacks by others:
  - Traffic analysis
  - Snooping payload
  - Denial of service

IP Packet Structure



IP Address Integrity

- Source address should be the sending host
  - But, who's checking?
  - You could send packets with any source you want
  - Why is checking hard?

Implications of IP Address Integrity

- Why would someone use a bogus source address?
- Launch a **denial-of-service** attack
  - Send excessive packets to the destination
  - … to overload the node, or the links leading to the node
  - But: victim can identify/filter you by the source address
- Evade detection by "spoofing"
  - Put **someone else's** source address in the packets
    - **Or**: use many **different** ones so can't be filtered
- Or: as a way to bother the spoofed host
  - Spoofed host is wrongly blamed
  - Spoofed host may receive return traffic from the receiver

More Security Implications

- IP options
  - Misuse: e.g., **Source Route** lets sender control path taken through network - say, sidestep security monitoring
  - IP options often processed in router's slow path → attacker can try to overload routers
  - Firewalls often configured to **drop** packets with options.

Security Implications of TOS? (8 bits)

- Attacker sets TOS priority for their traffic?
  - If regular traffic does not set TOS, then network prefers the attack traffic, greatly increasing damage
- What if network charges for TOS traffic …
  - … and attacker spoofs the victim's source address?
- Today, network TOS generally **does not work**
  - TOS now redefined for differentiated service
  - Mostly set/used by network operators, not end-systems

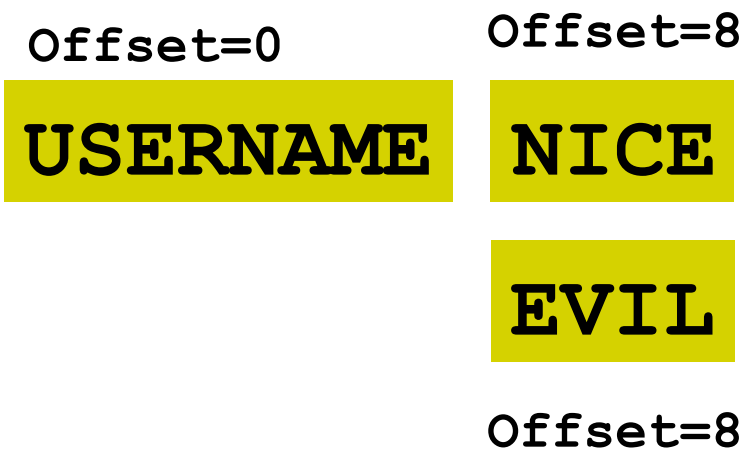
Security Implications of Fragmentation?

- Allows **evasion** of network monitoring/enforcement
- E.g., split an attack across multiple fragments
  - Packet inspection won't match a "signature"

- Monitor must remember previous fragments
  - But that costs **state**, which is another vector of attack

More Fragmentation Attacks

- What if 2 overlapping fragments are inconsistent?



- How does network monitor know whether receiver sees **USERNAME NICE** or **USERNAME EVIL**?

Even More Fragmentation Attacks

- What happens if attacker doesn't send all of the fragments in a packet?
- Receiver (or firewall) winds up holding the ones they receive for a long time
  - **State-holding** attack

Security Implications of TTL? (8 bits)

- Allows discovery of **topology** (a la *traceroute*)
- Can provide a hint that a packet is spoofed
  - It arrives at a router w/ a TTL different than packets from that address usually have
    - Because path from attacker to router has different # hops
  - Though this is *brittle* in the presence of routing changes
- Initial value is somewhat distinctive to sender's operating system. This plus other such initializations allow OS **fingerprinting** …
  - Which allow attacker to infer its likely vulnerabilities

Other Security Implications?

- No apparent problems with protocol field (8 bits)
  - It's just a de-muxing handle
  - If set incorrectly, next layer will find packet ill-formed
- Bad IP **checksum** field (16 bits) will cause packet to be discarded by the network
  - Not an effective attack…

Let’s take a quick look at the IPv6 header…

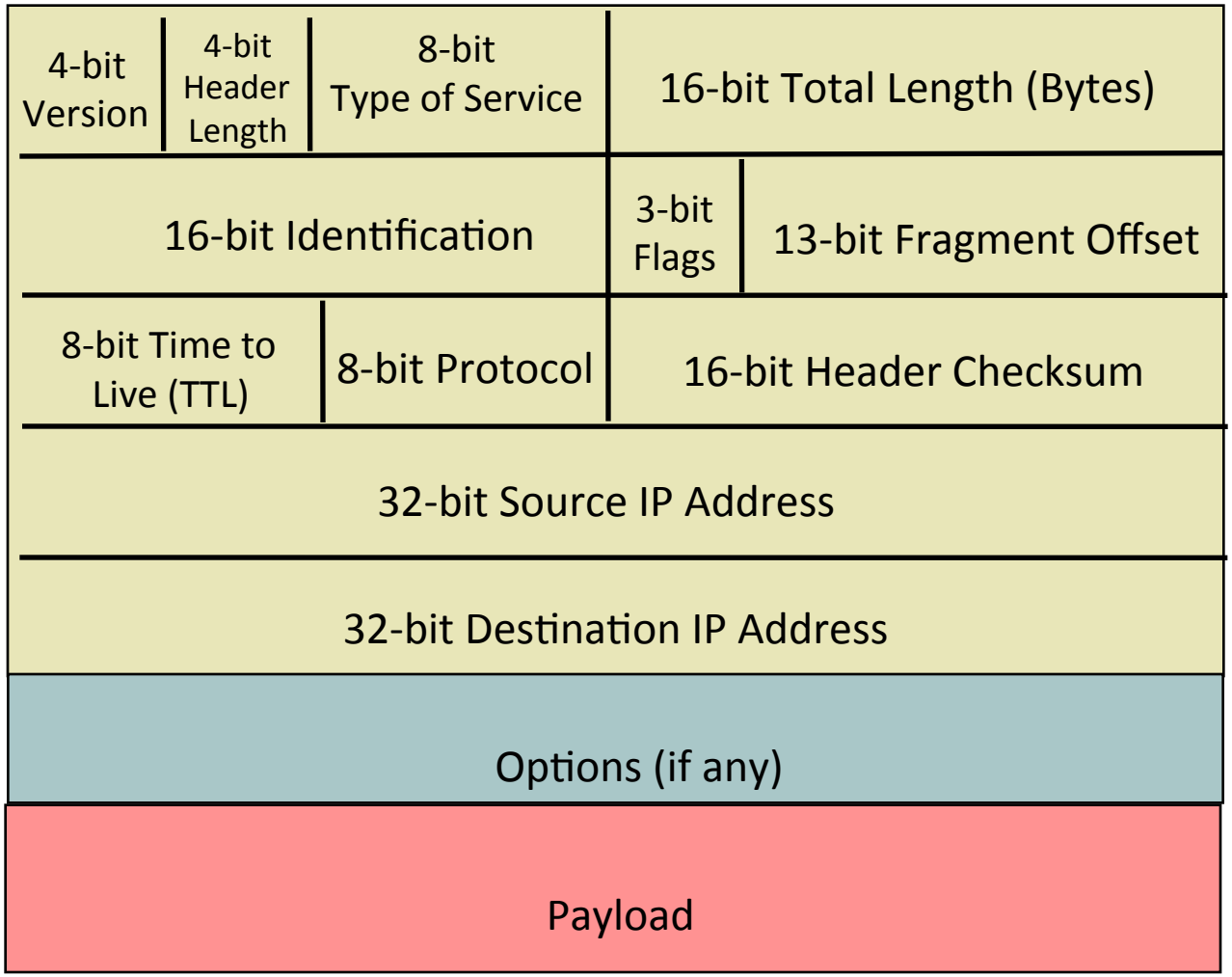
IPv6

- Motivated (prematurely) by address exhaustion
  - Addresses ***four*** times as big

- Steve Deering focused on simplifying IP
  - got rid of all fields that were not absolutely necessary
  - "spring cleaning" for IP

- Result is an elegant, if unambitious, protocol

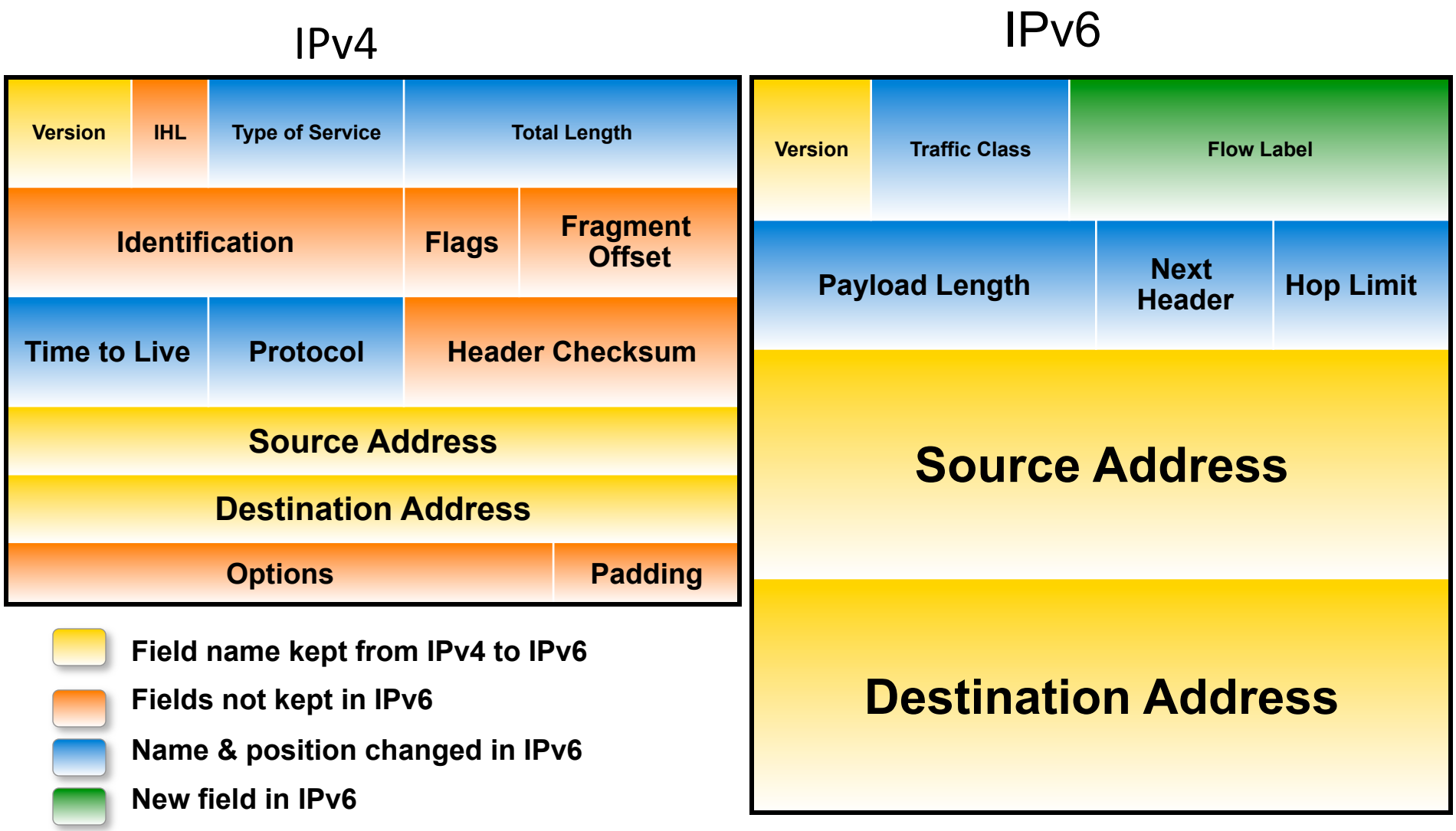
What “clean up” would you do?



Summary of Changes

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

IPv4 and IPv6 Header Comparison



Philosophy of Changes

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

IP header design

- More nuanced than it first seems!
- Must juggle multiple goals
  - Robustness
  - Efficiency
  - Security
  - Completeness
- Plus feature interactions
  - E.g., what happens to IP options when we fragment?
- And future evolution