IP Data Plane

CPSC 433/533, Spring 2021

Anurag Khandelwal

BGP: Errata

- **Rules for route selection in priority order**

Priority	Rule	Remarks
1	LOCAL PREF	Pick highest LOCAL PREF
2	ASPATH	Pick shortest ASPATH length
3	eBGP > iBGP	Did AS learn route via eBGP (preferred) or iBGP?
4	Hot Potato	Lowest IGP cost to next hop (egress router)
5	MED	Lowest MED Preferred
6	Router ID	Smallest next-hop router's IP address as tie-breaker

Today: The IP Layer

Today: 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

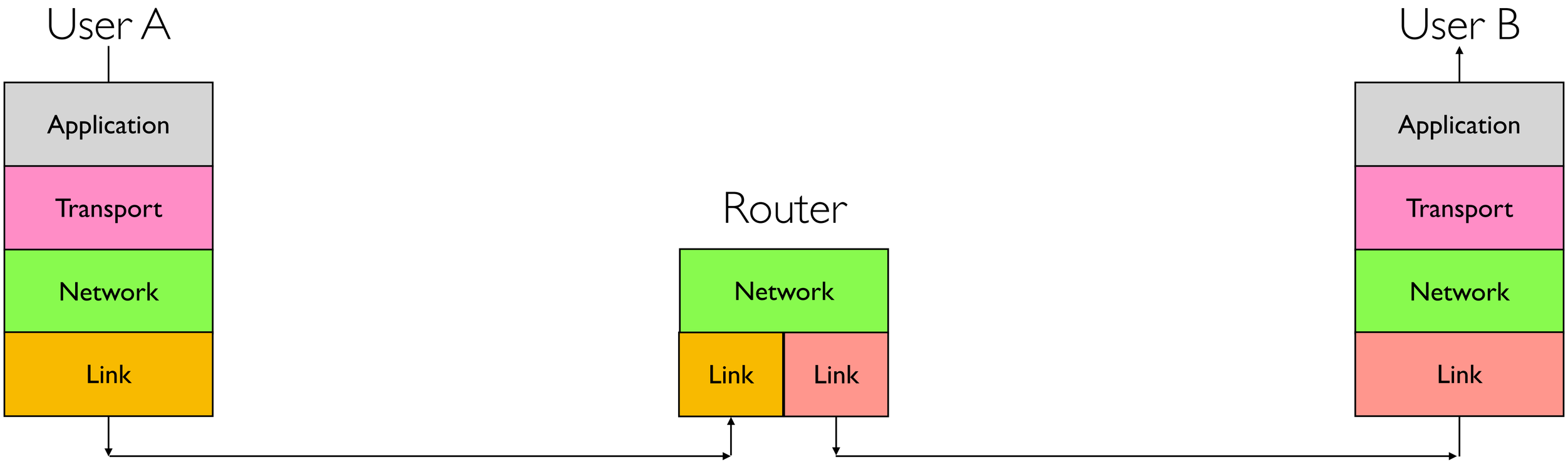
Today: The IP Layer

- **So far: focused mostly on routing protocols**
  - How routers discover and select end-to-end paths
  - Part of a network's *control* plane
- **Today & next class: the data plane**
  - What data packets look like at the IP layer
  - How routers forward these IP packets

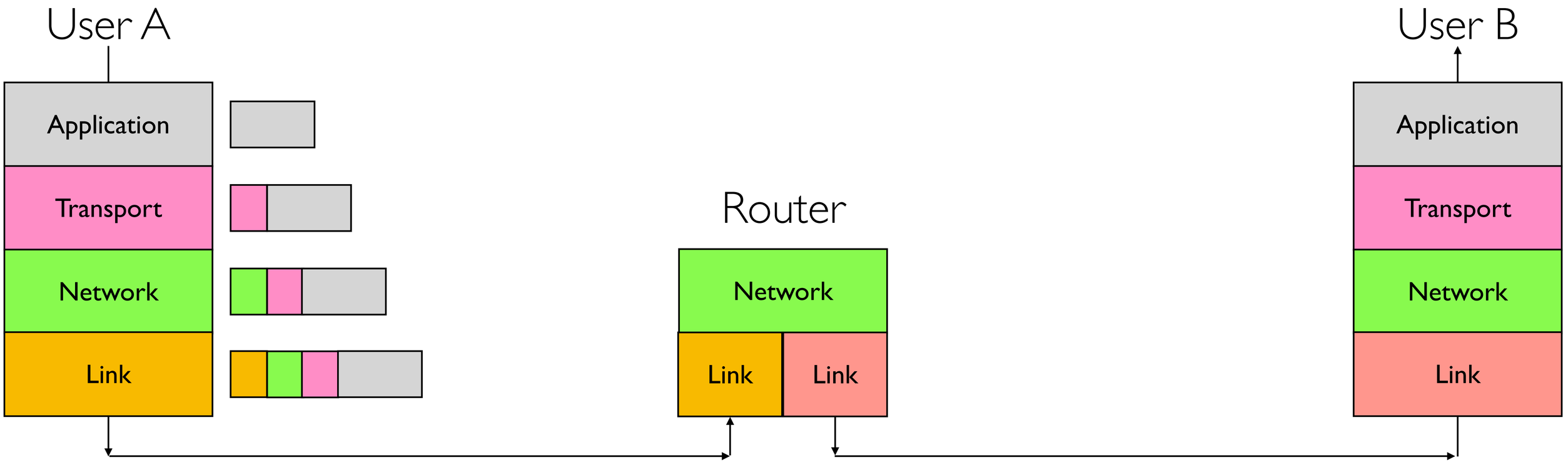
Recall: Layer Encapsulation & Protocol Headers

Recall: Layer Encapsulation & Protocol Headers

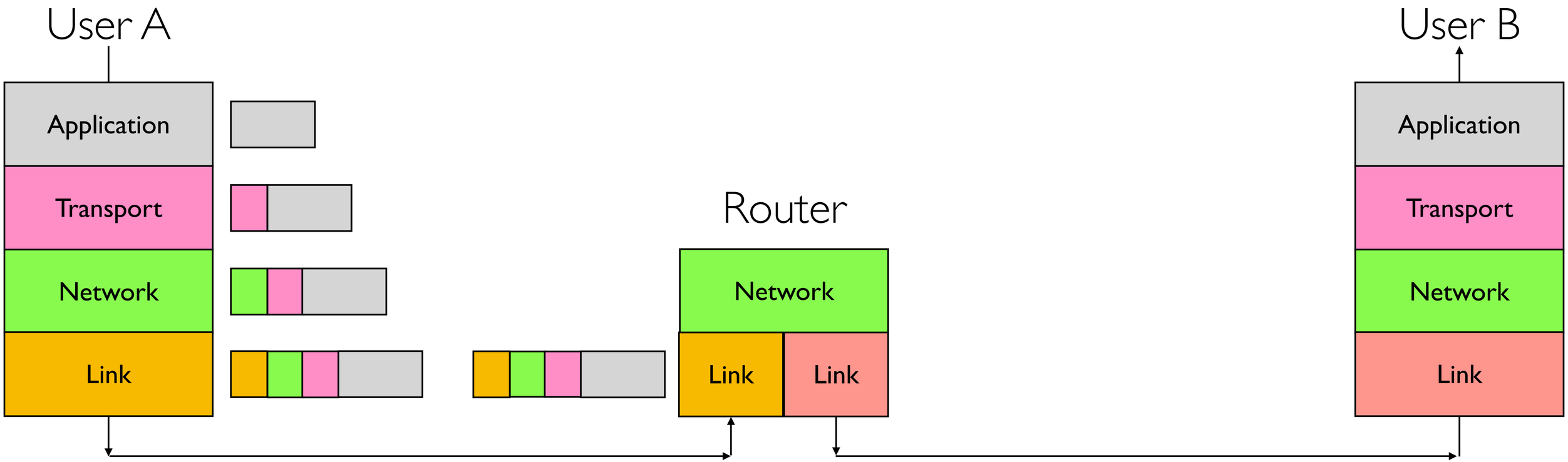
Recall: Layer Encapsulation & Protocol Headers



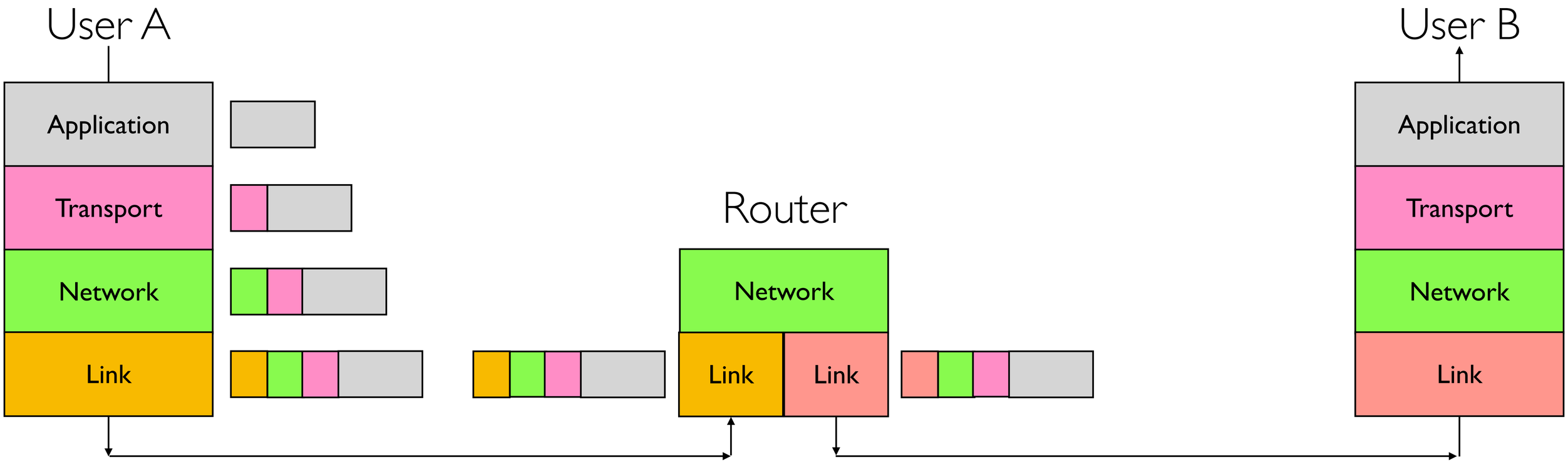
Recall: Layer Encapsulation & Protocol Headers



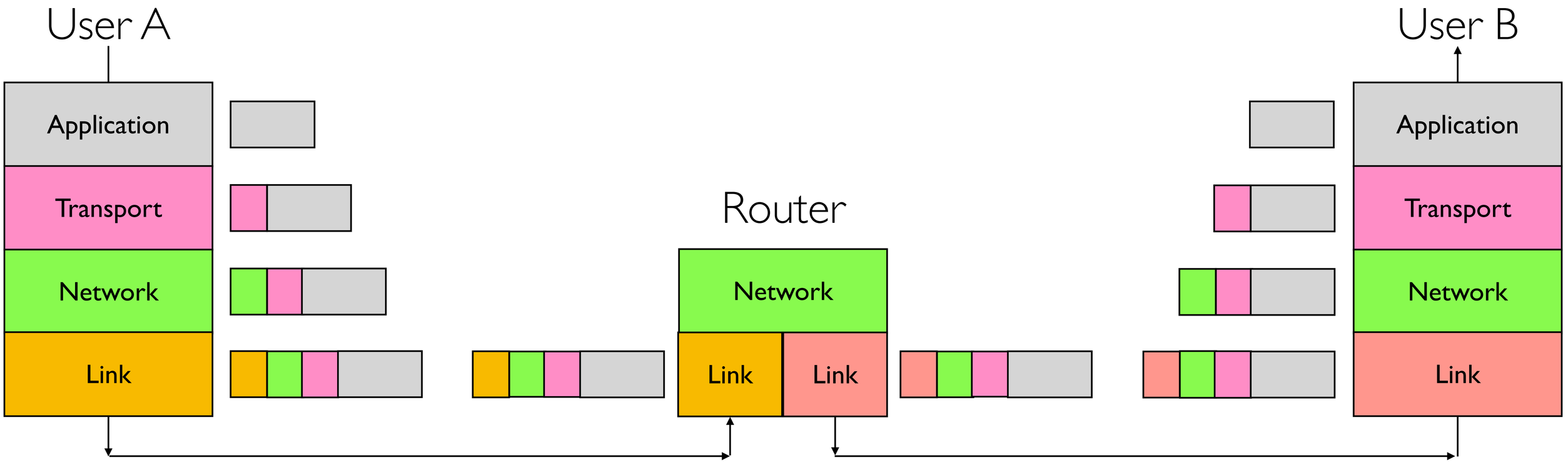
Recall: Layer Encapsulation & Protocol Headers



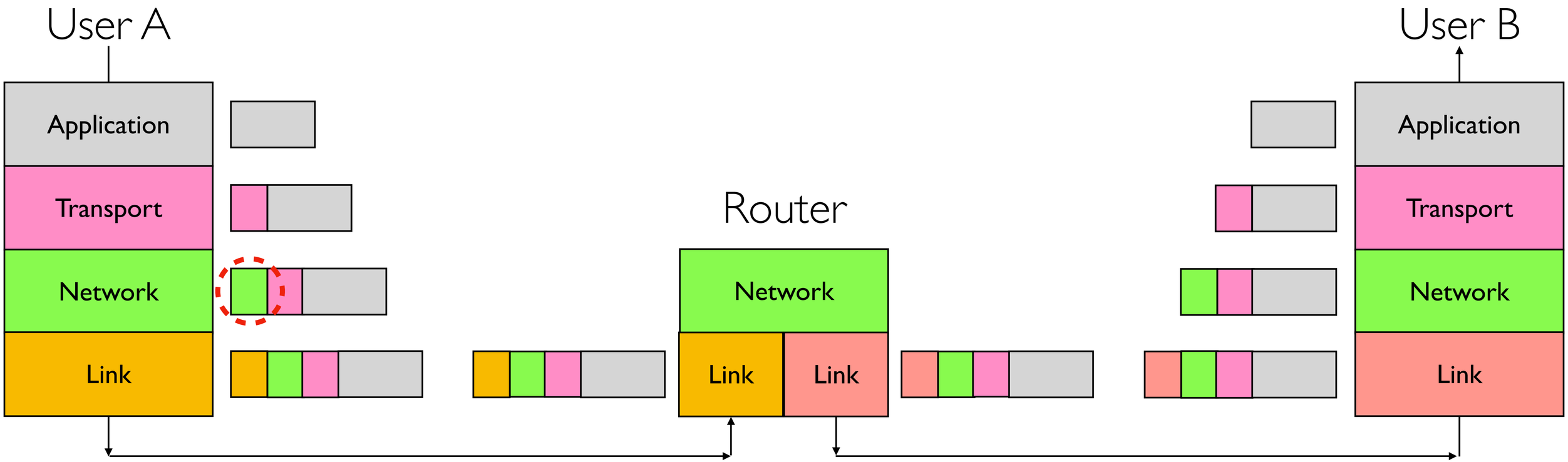
Recall: Layer Encapsulation & Protocol Headers



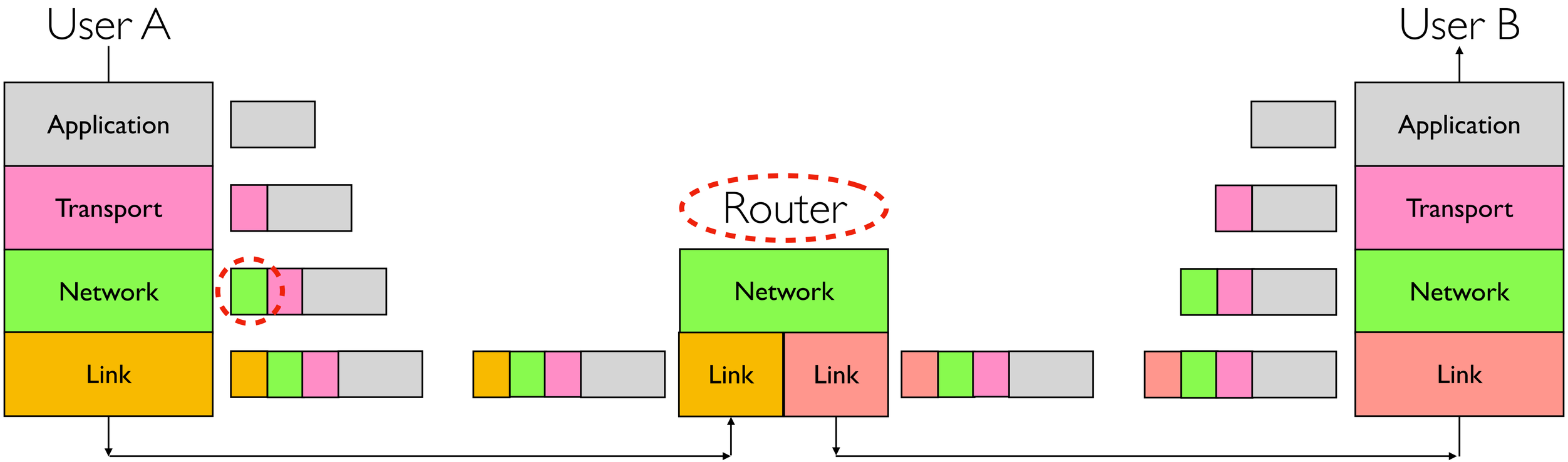
Recall: Layer Encapsulation & Protocol Headers



Recall: Layer Encapsulation & Protocol Headers



Recall: Layer Encapsulation & Protocol Headers



IP Packet

IP Packet

- **IP packet contains a header and payload**
  - Payload is opaque to the network
  - Header is what we care about

Designing the IP Header

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 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?

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?

What are these tasks?

- Parse packet

What are these tasks?

- Parse packet
- Carry packet to the destination

What are these tasks?

- Parse packet
- Carry packet to the destination
- Deal with problems along the way

What are these tasks?

- Parse packet
- Carry packet to the destination
- Deal with problems along the way
- Specify any special handling

What are these tasks?

- Parse packet
- Carry packet to the destination
- Deal with problems along the way
- Specify any special handling

In-network (routers)

What are these tasks?

- Parse packet
- Carry packet to the destination
- Deal with problems along the way
- Specify any special handling
- Tell the destination how to handle the packet

In-network (routers)

What are these tasks?

- Parse packet
- Carry packet to the destination
- Deal with problems along the way
- Specify any special handling
- Tell the destination how to handle the packet
- Tell the destination how to respond

In-network (routers)

What are these tasks?

- Parse packet
- Carry packet to the destination
- Deal with problems along the way
- Specify any special handling

In-network (routers)

- Tell the destination how to handle the packet
- Tell the destination how to respond

At end-host

What information do we need?

- Parse packet
- Carry packet to the destination
- Deal with problems along the way
- Specify any special handling
- Tell the destination how to handle the packet
- Tell the destination how to respond

What information do we need?

- Parse packet

What information do we need?

- Parse packet
  - *IP version number (4 bits), packet length (16 bits), header length (4 bits)*

Parse packet

Parse packet

- **IP Version Number**
  - Indicates version of IP protocol
  - Need this to know what other fields to expect
  - Typically 4 (for IPv4), and sometimes 6 (for IPv6)

Parse packet

- **IP Version Number**
  - Indicates version of IP protocol
  - Need this to know what other fields to expect
  - Typically 4 (for IPv4), and sometimes 6 (for IPv6)
- **Header Length (4 bits)**
  - Number if 32-bit words in the header
  - Typically "5" (for a 20-byte IPv4 header)
  - Can be more when IP *options* are used

Parse packet

- **IP Version Number**
  - Indicates version of IP protocol
  - Need this to know what other fields to expect
  - Typically 4 (for IPv4), and sometimes 6 (for IPv6)
- **Header Length (4 bits)**
  - Number if 32-bit words in the header
  - Typically "5" (for a 20-byte IPv4 header)
  - Can be more when IP *options* are used
- **Total Length (16 bits)**
  - Number of bytes in the packet
  - Maximum size is 65536 (2^16 - 1)
  - … though underlying links may impose smaller limits

What information do we need?

- Parse packet
  - *IP version number (4 bits), packet length (16 bits), header length (4 bits)*
- Carry packet to the destination

What information do we need?

- Parse packet
  - *IP version number (4 bits), packet length (16 bits), header length (4 bits)*
- Carry packet to the destination
  - *Destination's IP address (32 bits)*

What information do we need?

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

What information do we need?

- Parse packet
  - *IP version number (4 bits), packet length (16 bits), header length (4 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), header length (4 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)*

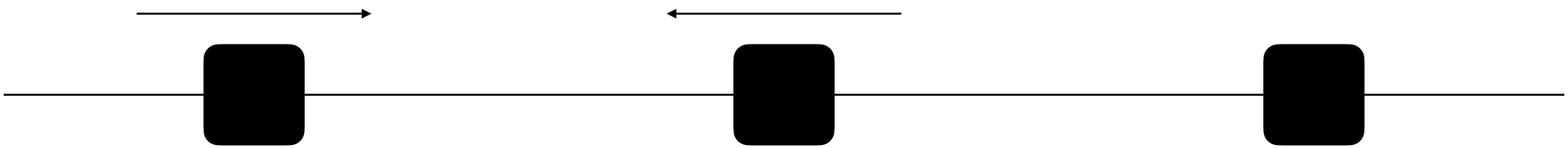
Preventing Loops (TTL)

Preventing Loops (TTL)

- **Forwarding loops cause packets to cycle for a looooong time**
  - Left unchecked, would accumulate to consume all capacity

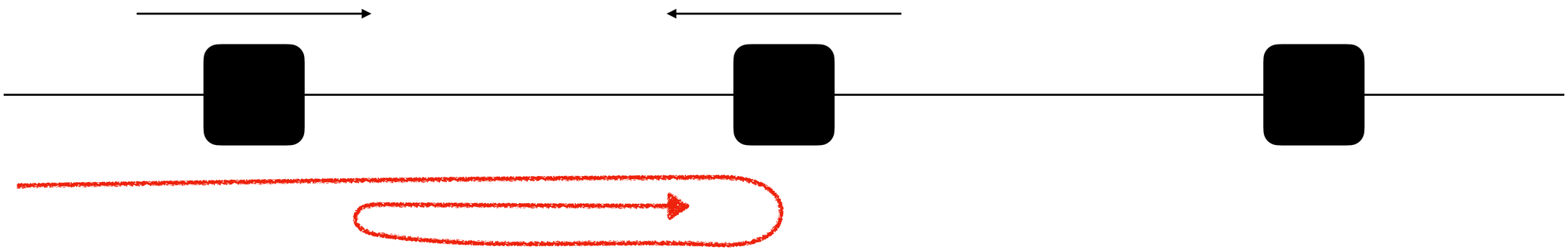
Preventing Loops (TTL)

- **Forwarding loops cause packets to cycle for a looooong time**
  - Left unchecked, would accumulate to consume all capacity



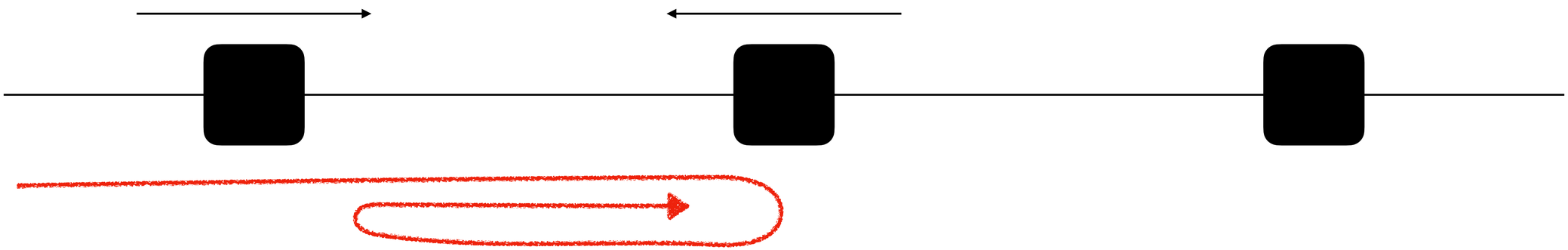
Preventing Loops (TTL)

- **Forwarding loops cause packets to cycle for a looooong time**
  - Left unchecked, would accumulate to consume all capacity



Preventing Loops (TTL)

- **Forwarding loops cause packets to cycle for a looooong time**
  - Left unchecked, would accumulate to consume all capacity



- **Time-To-Live (TTL) field (8 bits)**
  - Decremented at each hop, packet discarded if TTL reaches 0
  - … and "time exceeded" message sent to the source
    - *Basis for traceroute!*

Header Corruption (Checksum)

Header Corruption (Checksum)

- **Checksum (16 bits)**
  - Particular form of checksum over packet header

Header Corruption (Checksum)

- **Checksum (16 bits)**
  - Particular form of checksum over packet header
- **If not correct, router discards packet**
  - So it doesn't act on bogus information

Header Corruption (Checksum)

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

Fragmentation

Fragmentation

- **Every link has a "Maximum Transmission Unit" (MTU)**
  - Largest number of bits it can carry as one unit

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**

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**

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), header length (4 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 fields (32 bits)*
- Specify any special handling

What information do we need?

- Parse packet
  - *IP version number (4 bits), packet length (16 bits), header length (4 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 fields (32 bits)*
- Specify any special handling
  - *ToS (8 bits), Options (variable length)*

Special Handling

Special Handling

- **"Type-of-Service" (ToS) (8 bits)**
  - Allow packets to be treated differently based on needs
    - e.g., low delay audio, high bandwidth data transfer, etc.
  - Has been redefined several times
  - Now called "Differentiated Services Code Point (DSCP)"
  - No general use…

Special Handling

Special Handling

- **Options (Variable Length)**

Special Handling

- **Options (Variable Length)**
- **Optional Directives to the network**
  - Not used very often
  - 16 bits of metadata + option-specific data

Special Handling

- **Options (Variable Length)**
- **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
  - …

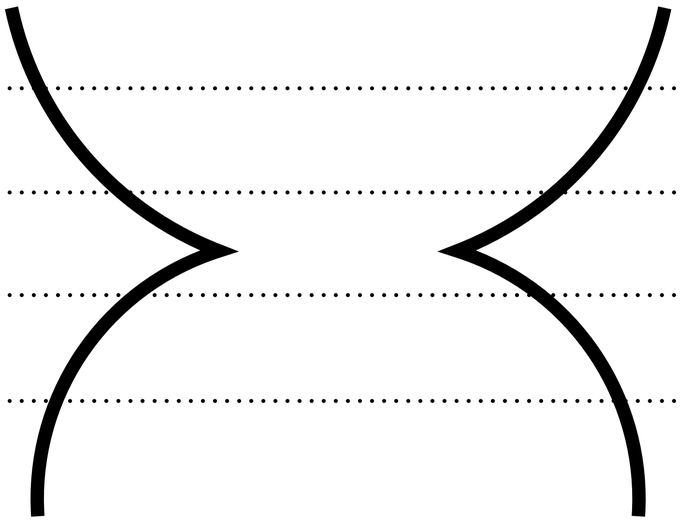
What information do we need?

- Parse packet
  - *IP version number (4 bits), packet length (16 bits), header length (4 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 fields (32 bits)*
- Specify any special handling
  - *ToS (8 bits), Options (variable length)*
- Tell the destination how to handle the packet

What information do we need?

- Parse packet
  - *IP version number (4 bits), packet length (16 bits), header length (4 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 fields (32 bits)*
- Specify any special handling
  - *ToS (8 bits), Options (variable length)*
- Tell the destination how to handle the packet
  - *Protocol (8 bits)*

Telling End-System How to Handle Packet



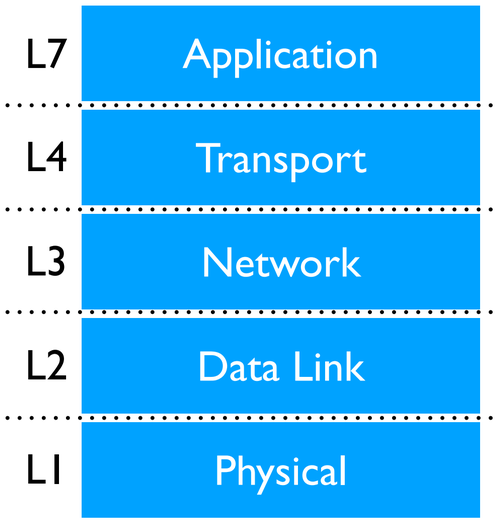
Telling End-System How to Handle Packet

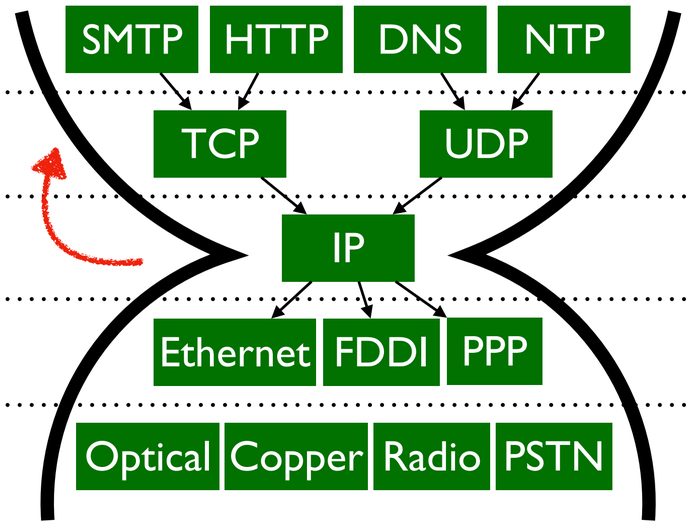
Telling End-System How to Handle Packet

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

Telling End-System How to Handle Packet

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



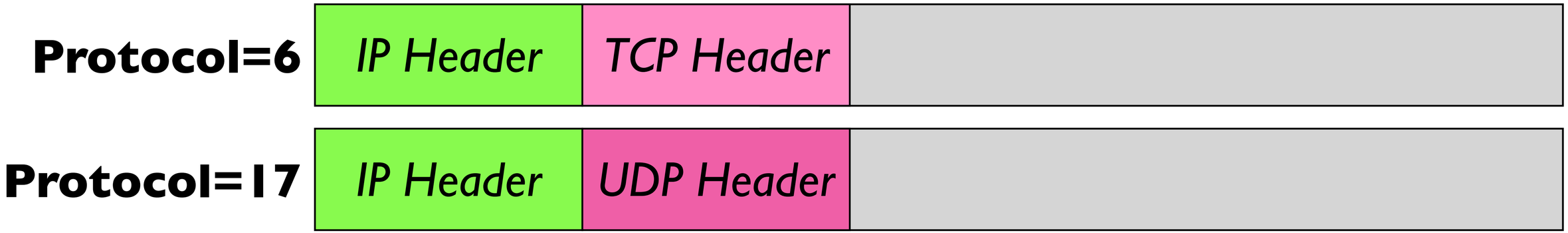


Telling End-System How to Handle Packet

- **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)

Telling End-System 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)



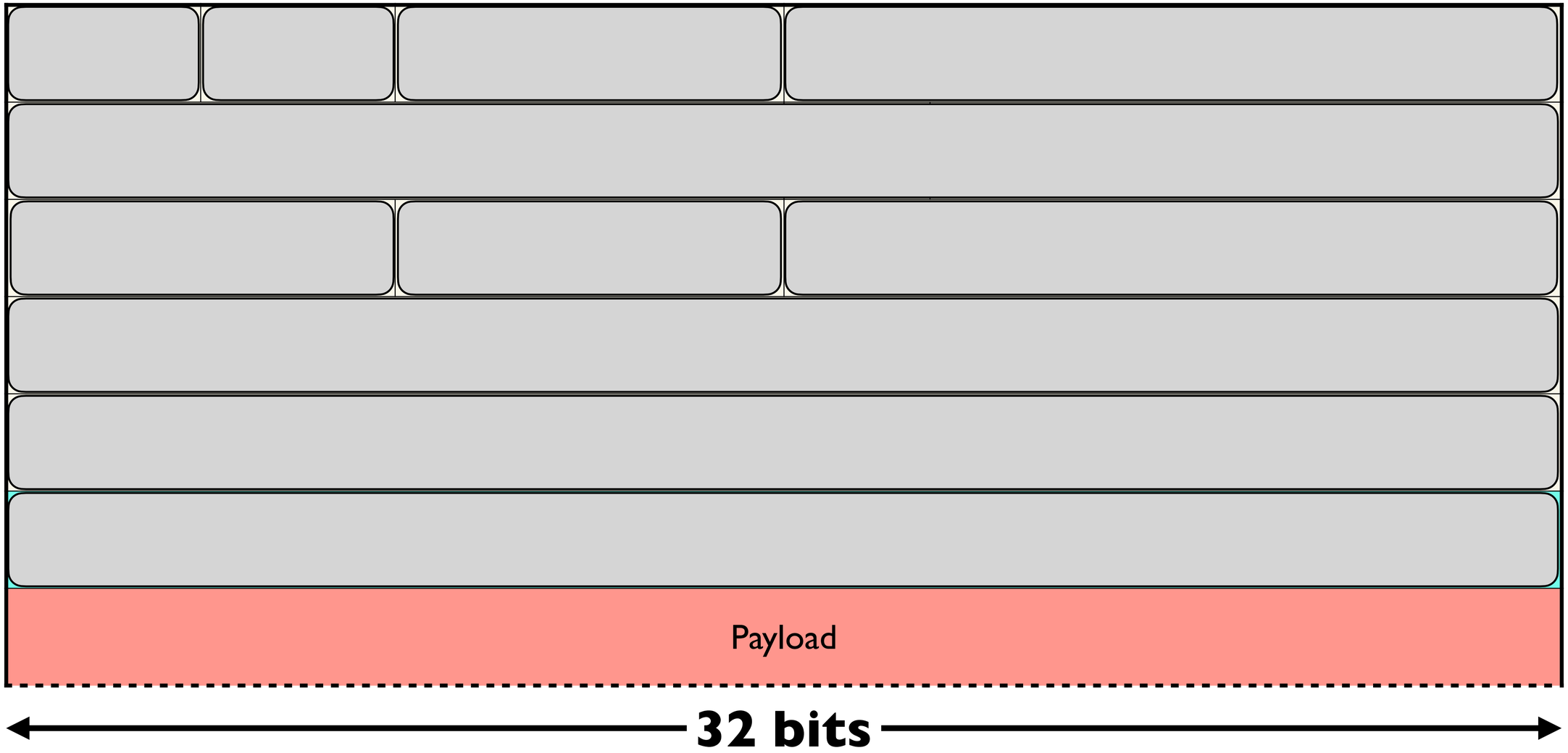
What information do we need?

- Parse packet
  - *IP version number (4 bits), packet length (16 bits), header length (4 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 fields (32 bits)*
- Specify any special handling
  - *ToS (8 bits), Options (variable length)*
- Tell the destination how to handle the packet
  - *Protocol (8 bits)*
- Tell the destination how to respond

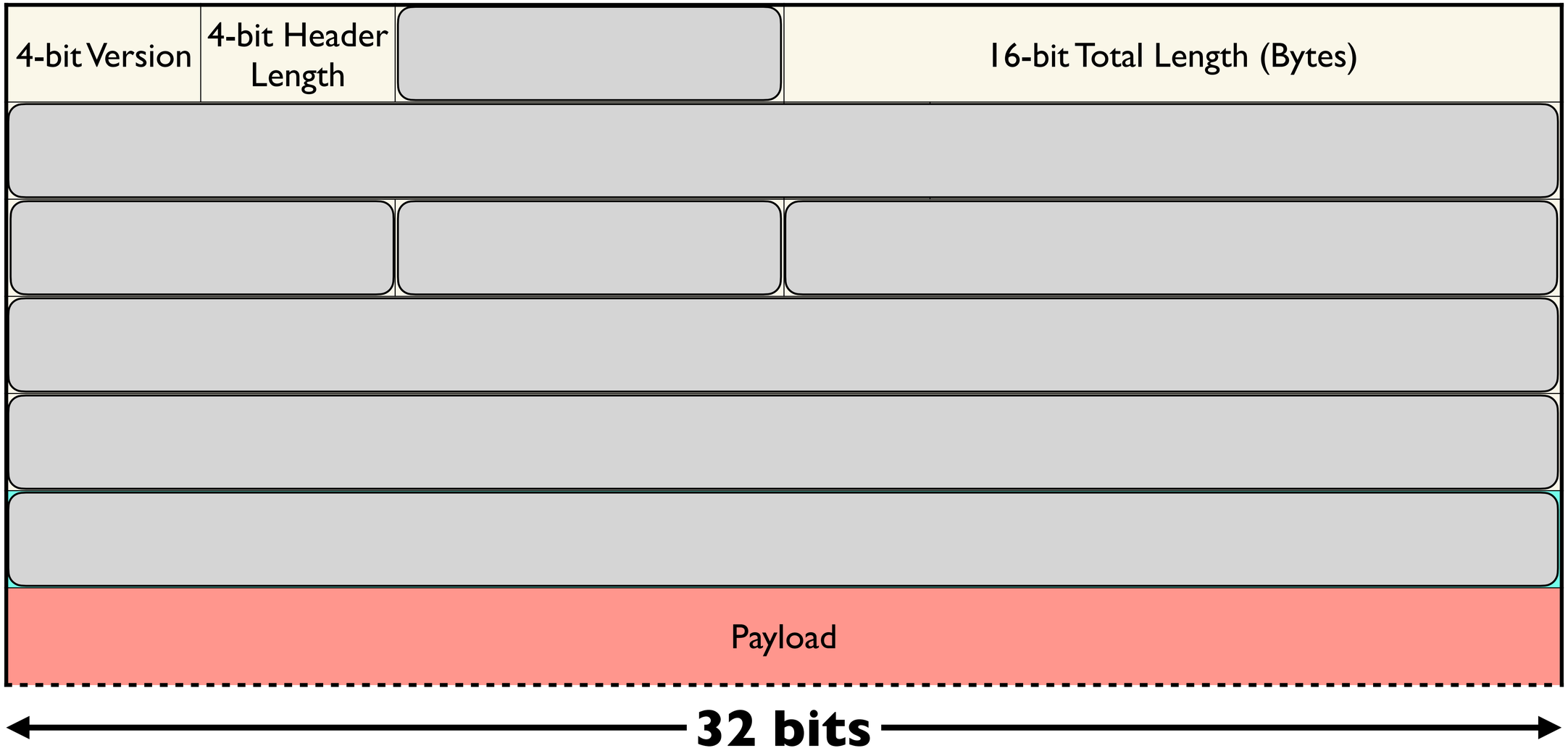
What information do we need?

- Parse packet
  - *IP version number (4 bits), packet length (16 bits), header length (4 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 fields (32 bits)*
- Specify any special handling
  - *ToS (8 bits), Options (variable length)*
- Tell the destination how to handle the packet
  - *Protocol (8 bits)*
- Tell the destination how to respond
  - *Source's IP address (32 bits)*

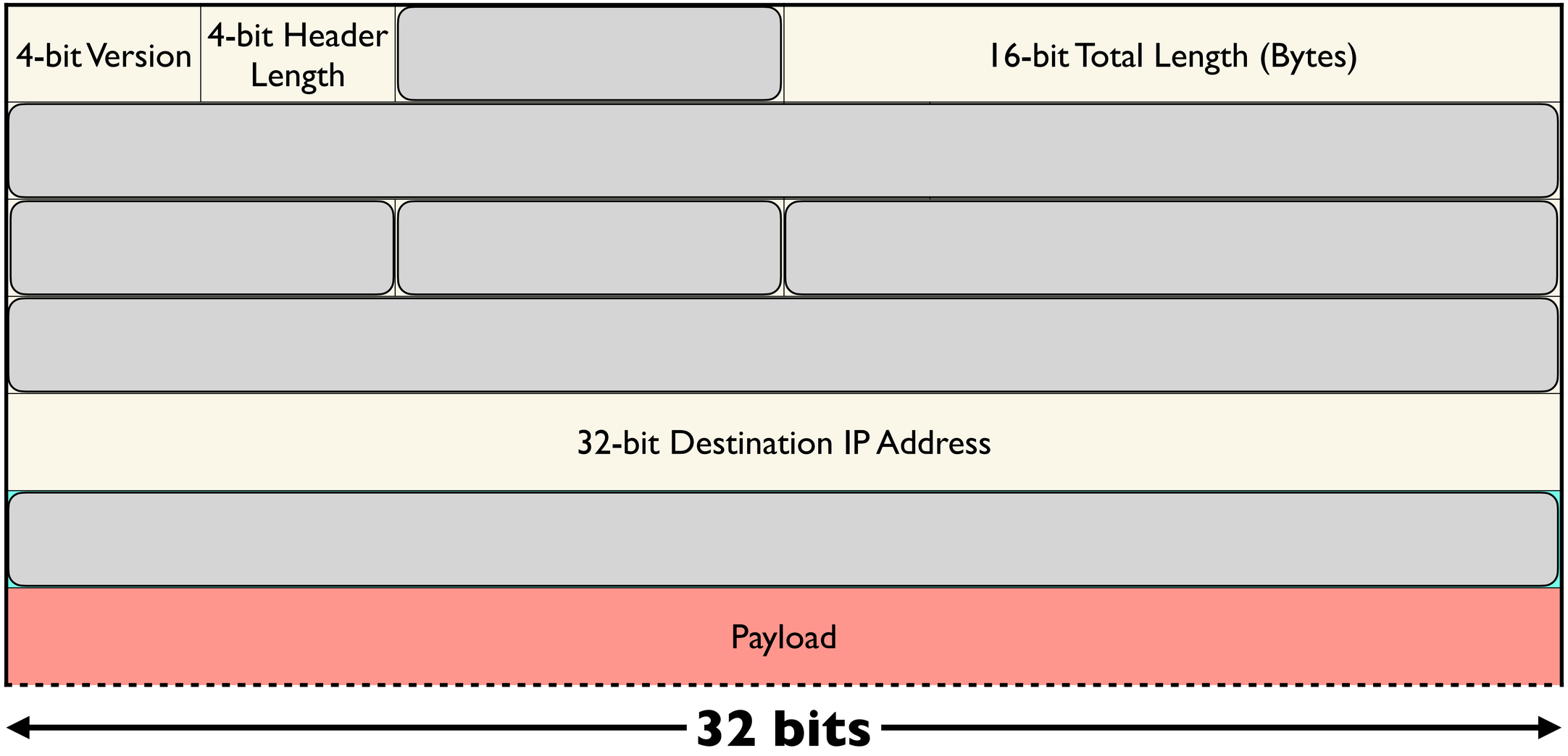
IP Packet Structure



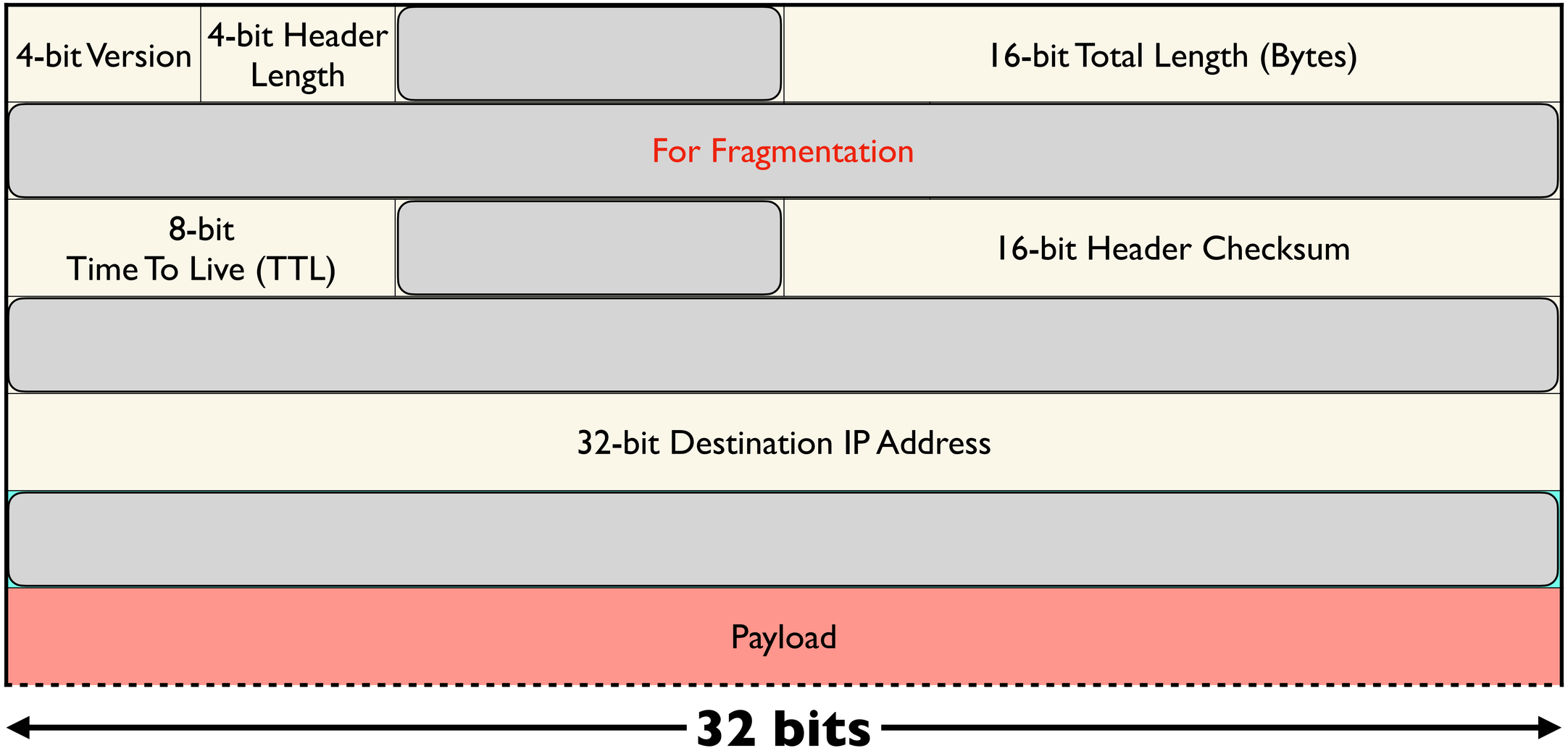
IP Packet Structure



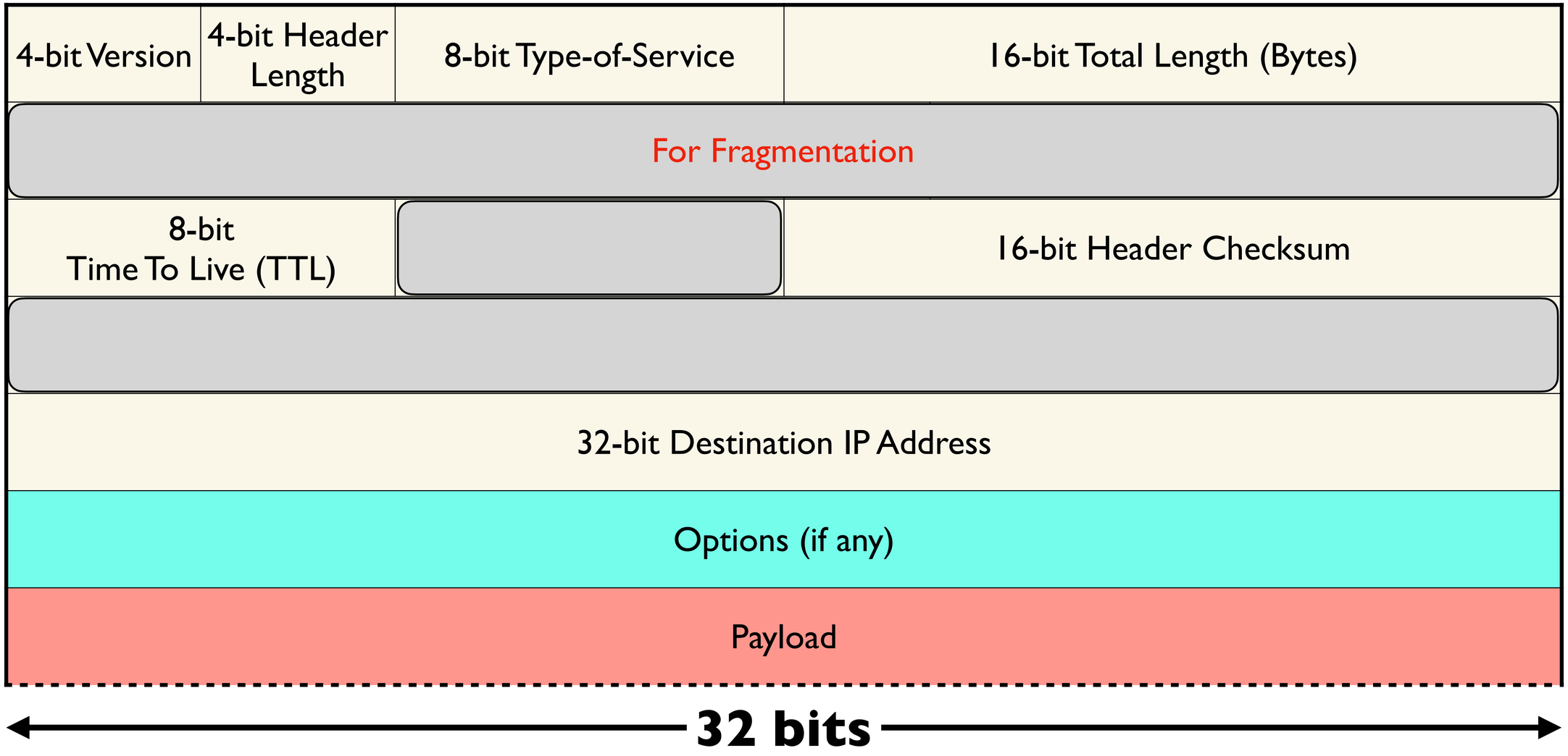
IP Packet Structure



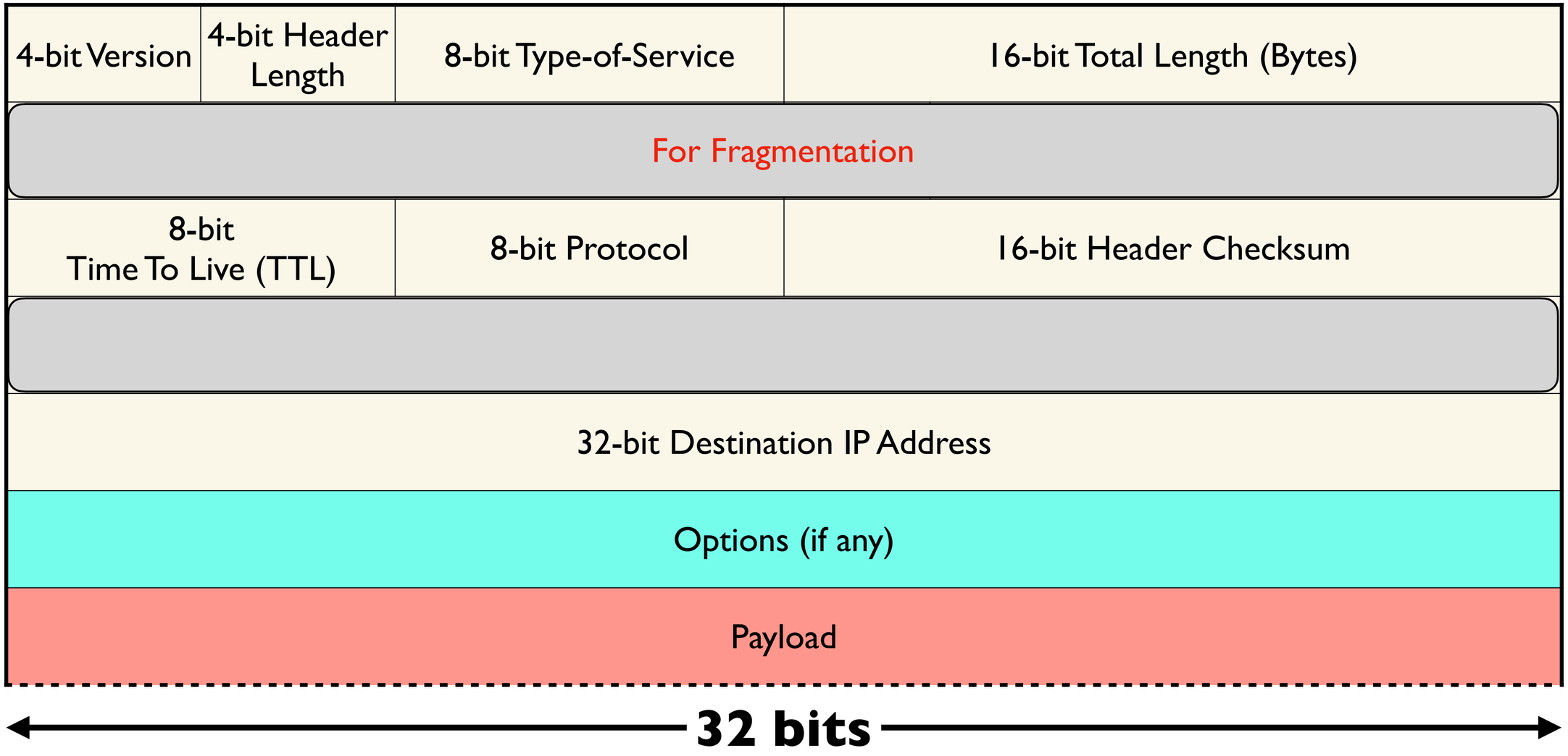
IP Packet Structure



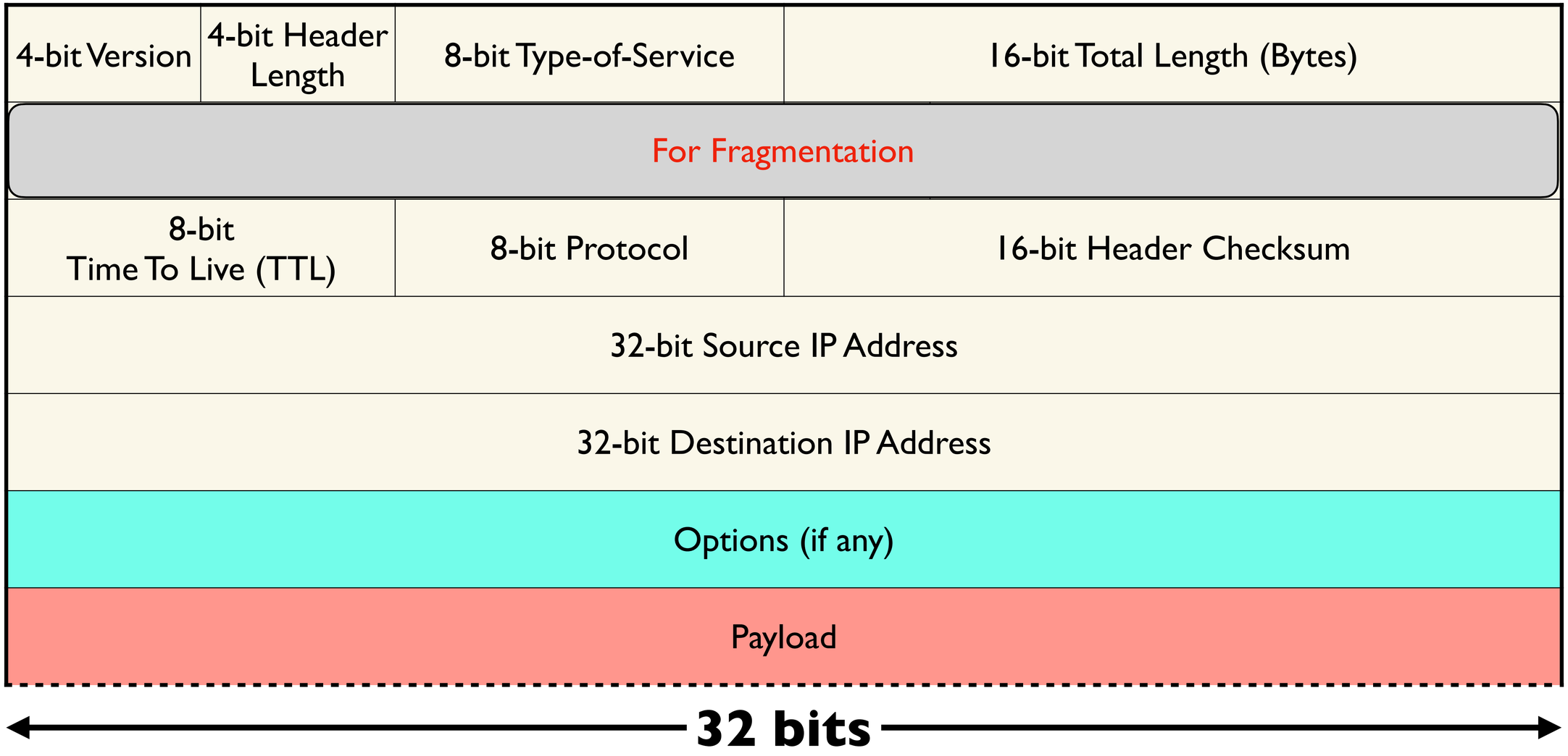
IP Packet Structure



IP Packet Structure



IP Packet Structure



A Closer Look at Fragmentation

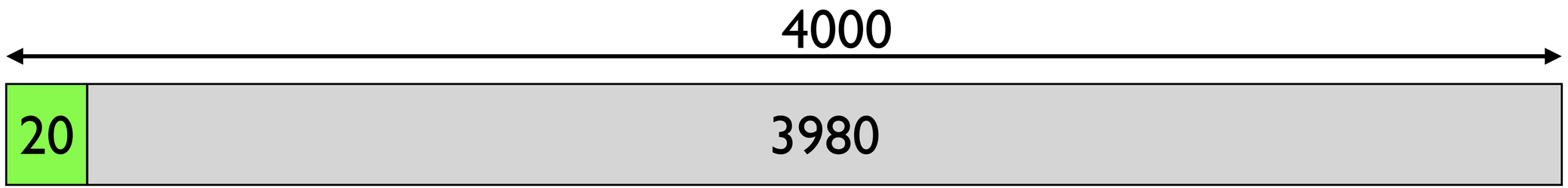
- **Every link has a "Maximum Transmission Unit" (MTU)**
  - Largest number of bits it can carry as one unit
- **A router can split a packet into multiple "fragments" if the packet size exceeds the link's MTU**
- **Must reassemble to recover original packet**

Example of Fragmentation

- **A 4000 byte packet crosses a link with MTU = 1500B**

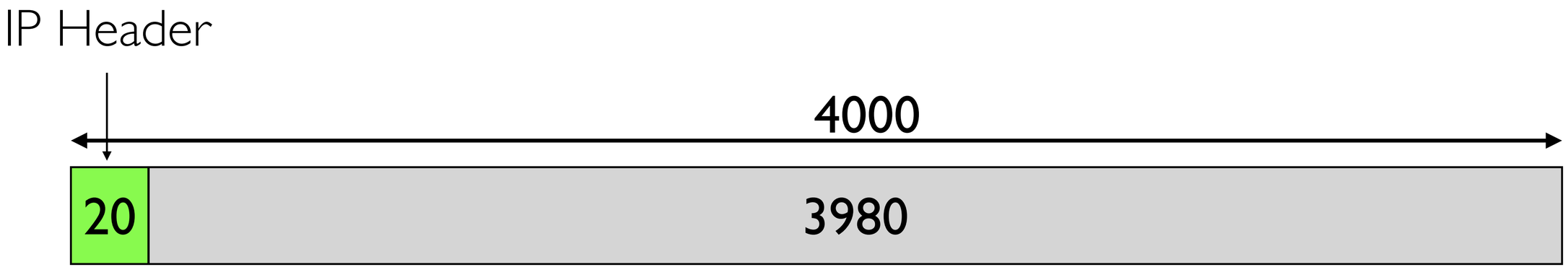
Example of Fragmentation

- **A 4000 byte packet crosses a link with MTU = 1500B**



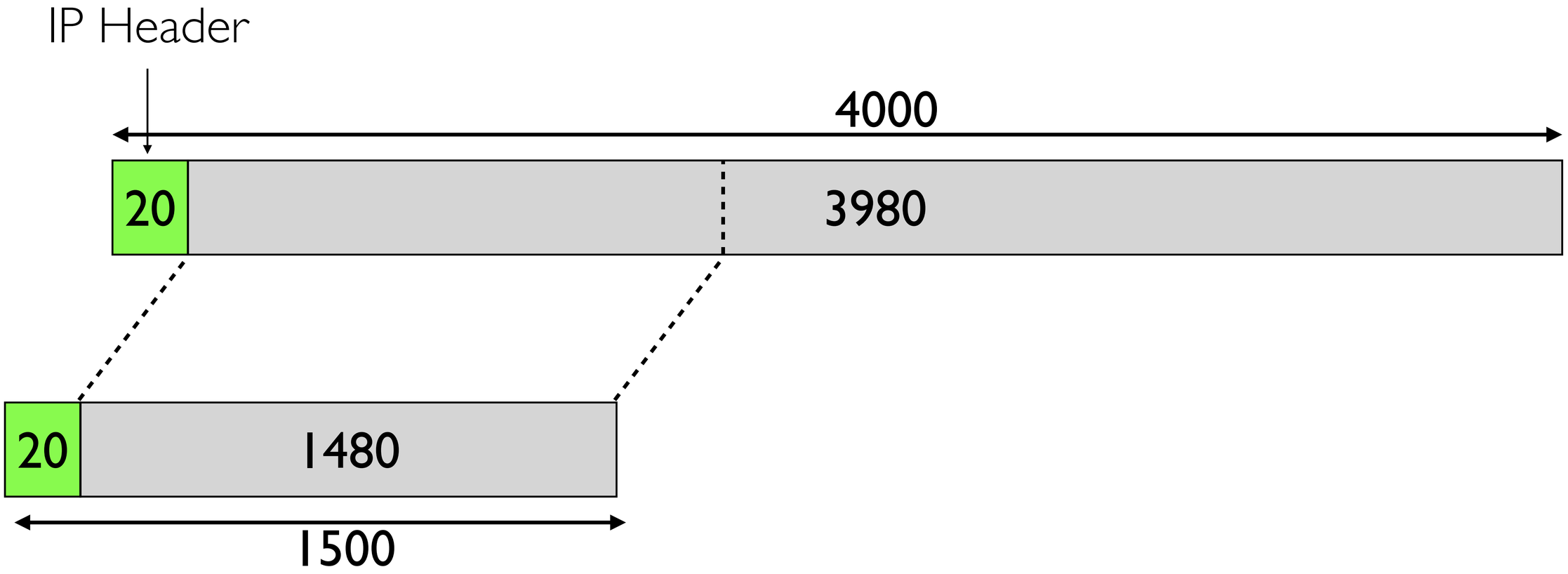
Example of Fragmentation

- **A 4000 byte packet crosses a link with MTU = 1500B**



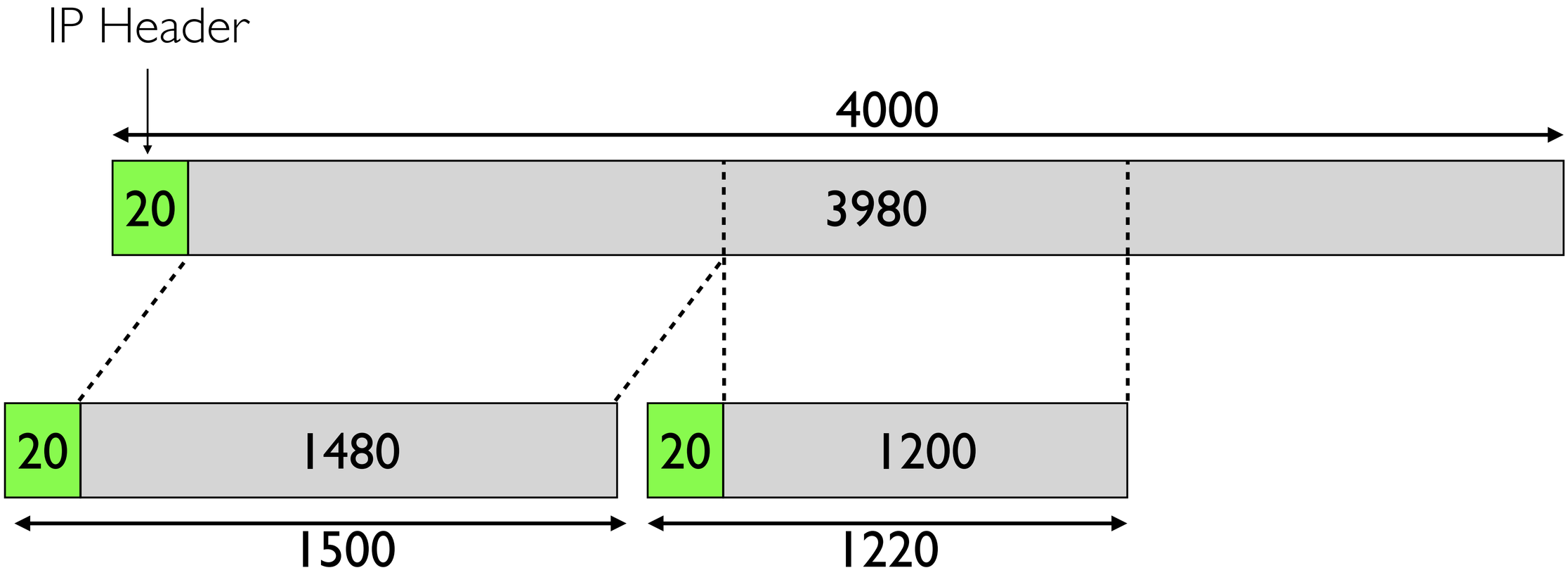
Example of Fragmentation

- **A 4000 byte packet crosses a link with MTU = 1500B**



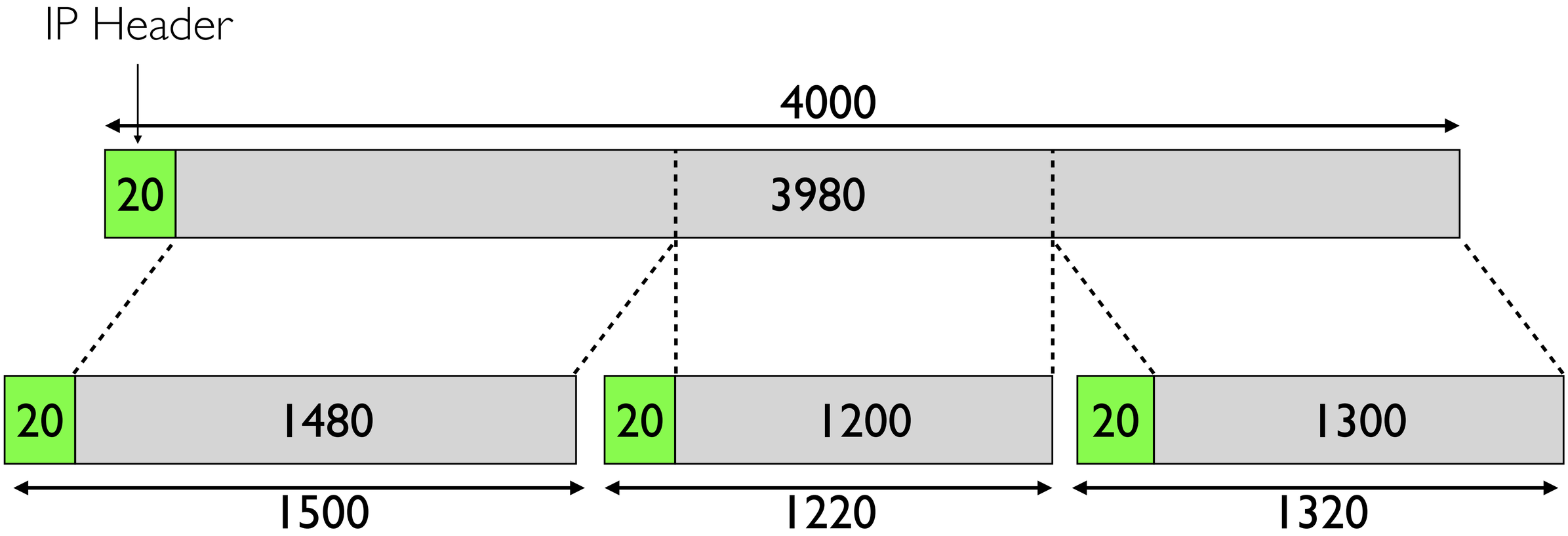
Example of Fragmentation

- **A 4000 byte packet crosses a link with MTU = 1500B**



Example of Fragmentation

- **A 4000 byte packet crosses a link with MTU = 1500B**



DIY Exercise in Header Engineering

DIY Exercise in Header Engineering

- **Take 3 mins to design fragmentation scheme**

DIY Exercise in Header Engineering

- **Take 3 mins to design fragmentation scheme**
- **At the end of 3 minutes, be ready to tell me:**
  - How many fields you need?
  - What are they and why?

DIY Exercise in Header Engineering

- **Take 3 mins to design fragmentation scheme**
- **At the end of 3 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!**

DIY Exercise in Header Engineering

- **Take 3 mins to design fragmentation scheme**
- **At the end of 3 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!**

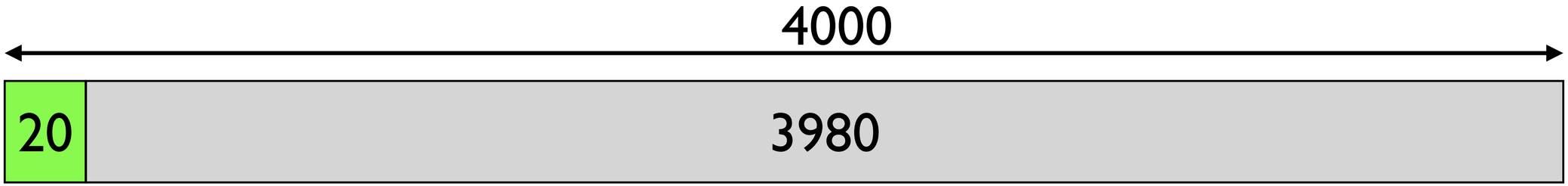
Time Left:

DIY Exercise in Header Engineering

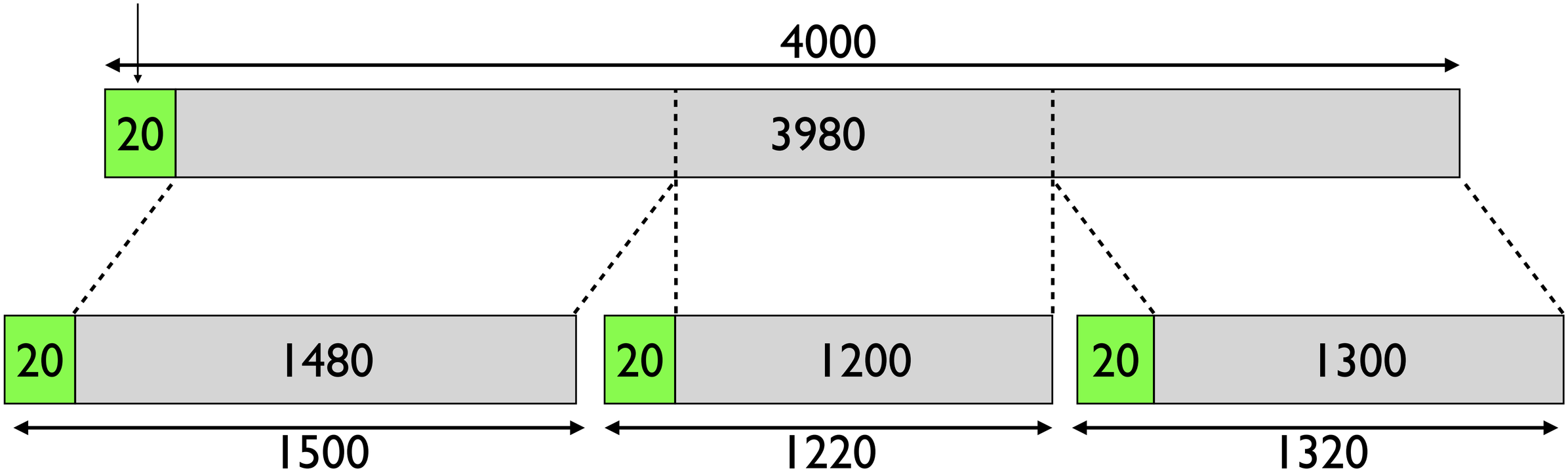
- **Take 3 mins to design fragmentation scheme**
- **At the end of 3 minutes, be ready to tell me:**
  - How many fields you need?
  - What are they and why?
- **Don't look for the answer elsewhere, the goal is to think about the problem!**

Time Left:

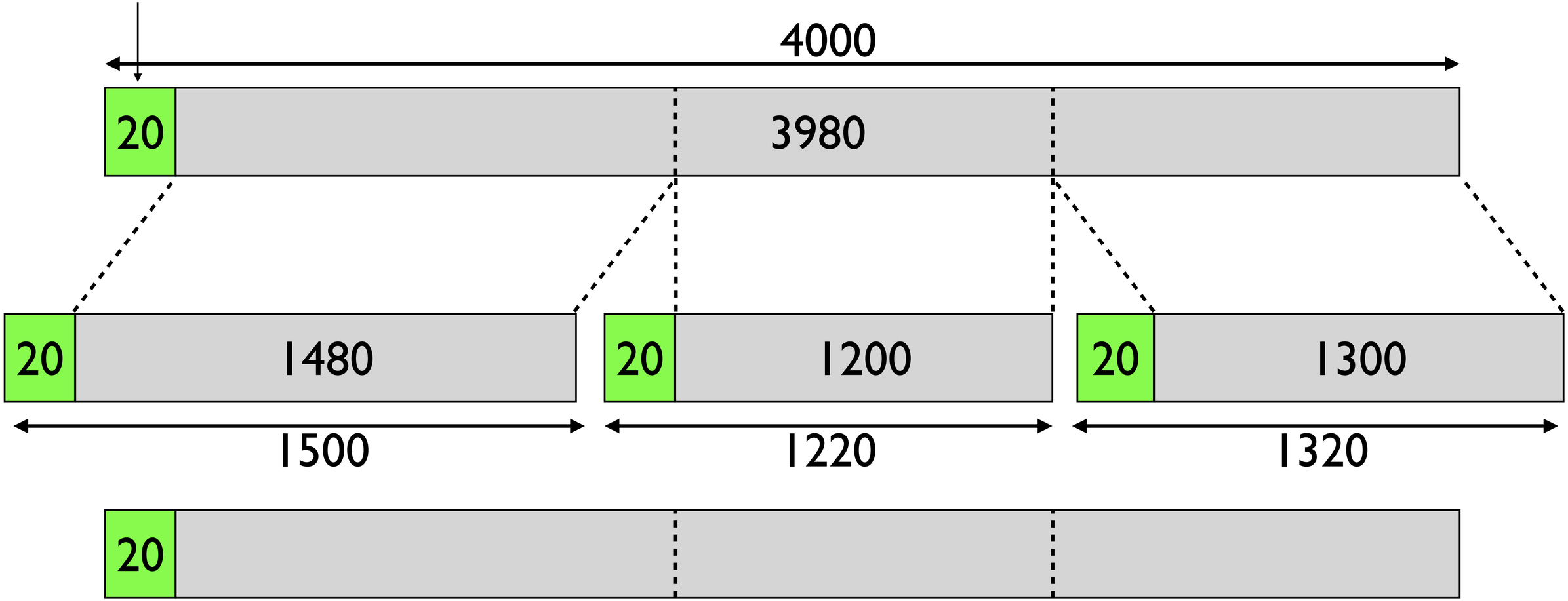
Why Reassemble?



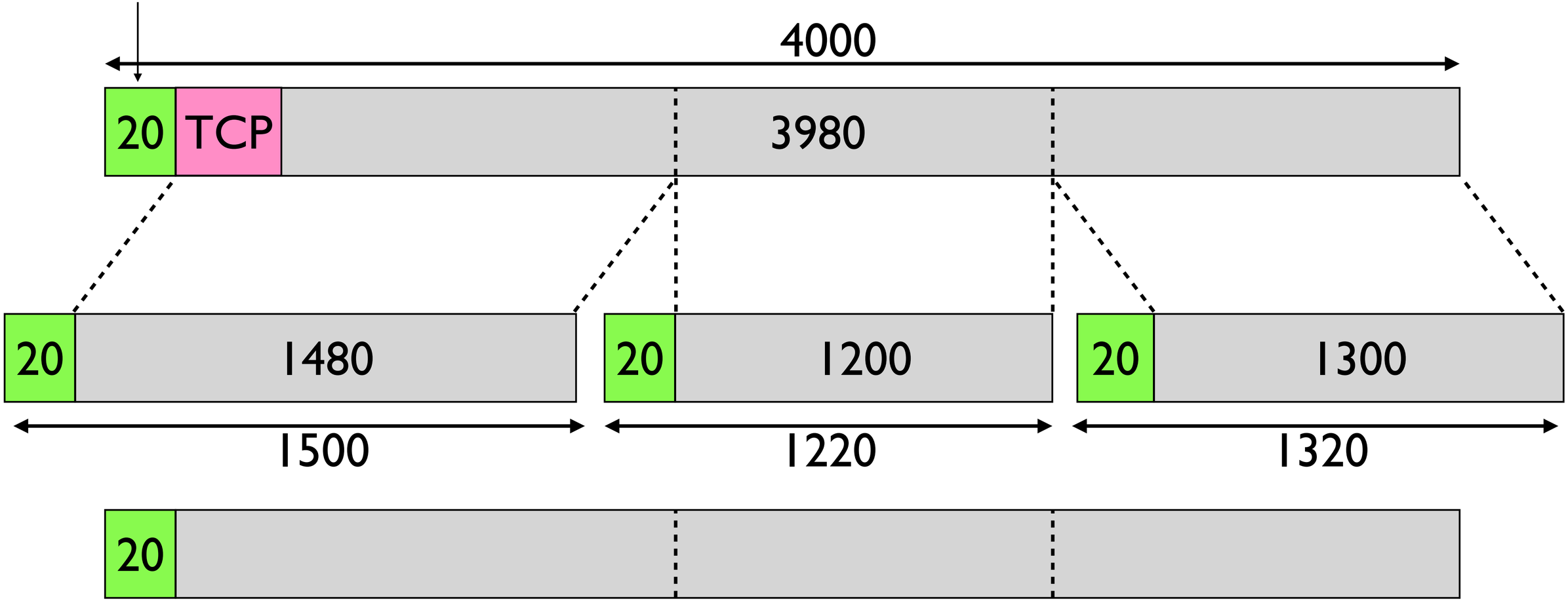
Why Reassemble?



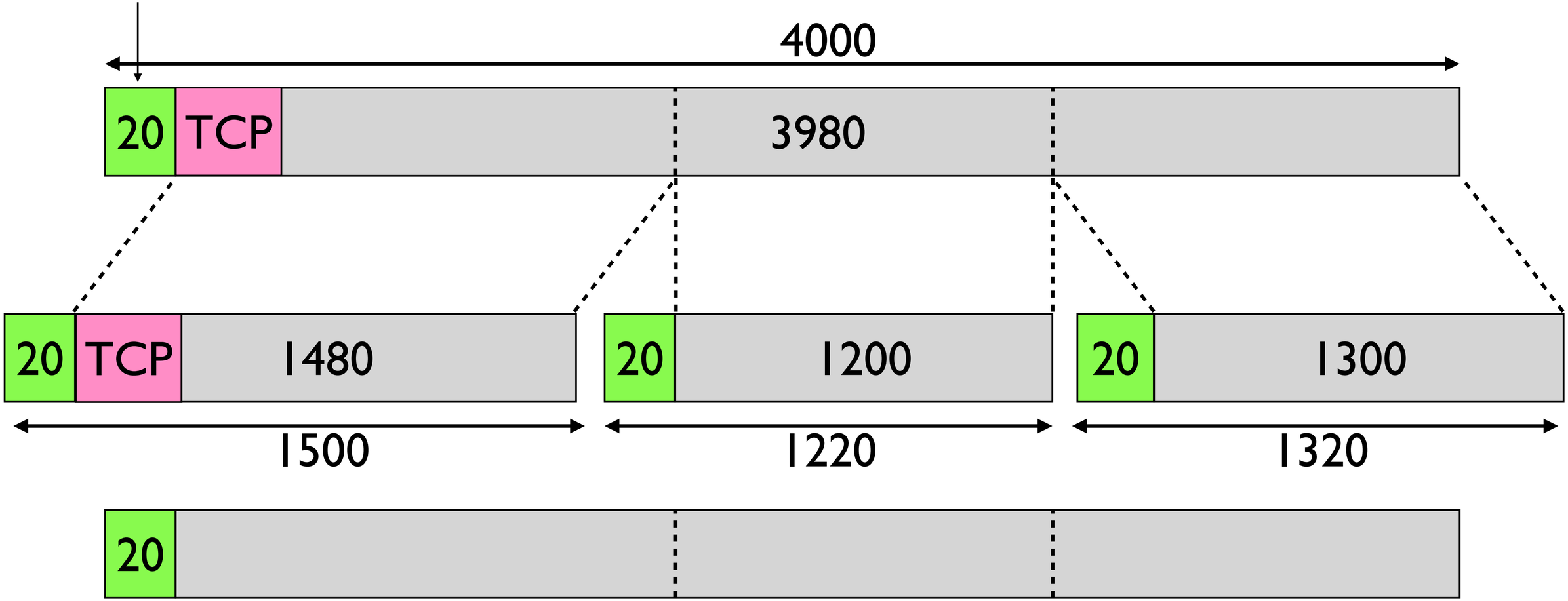
Why Reassemble?



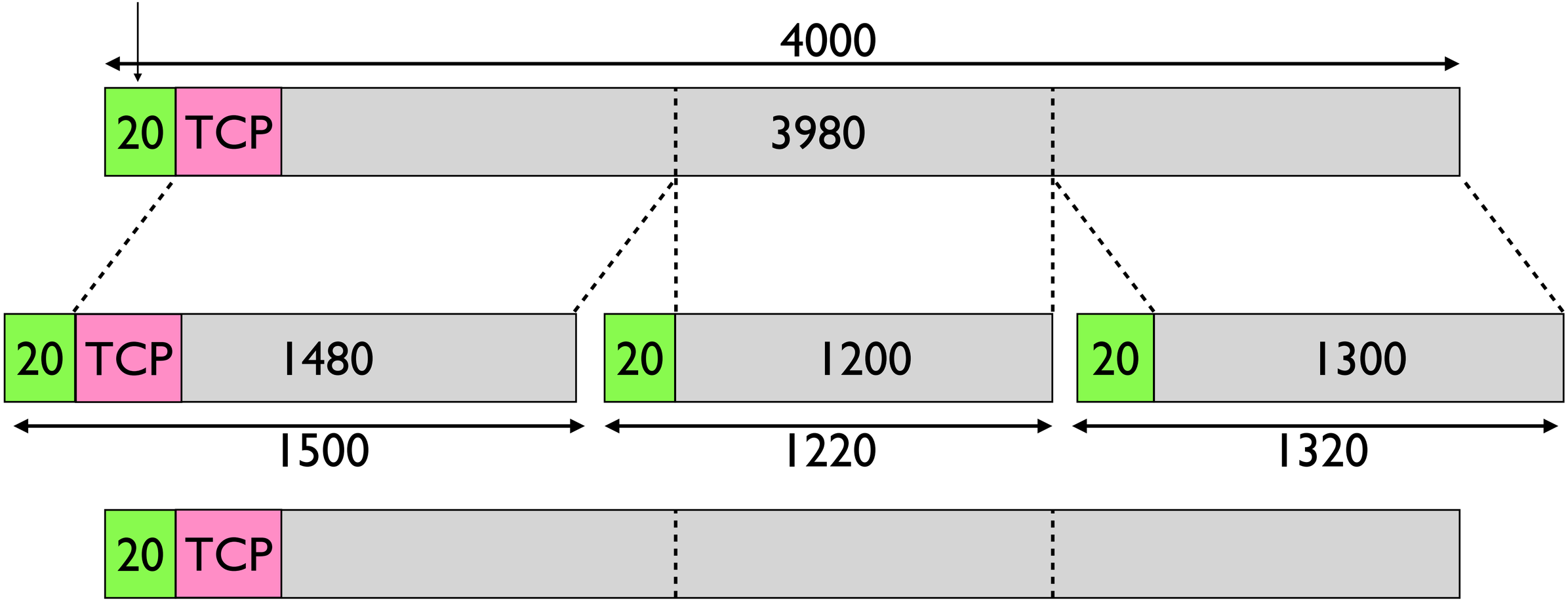
Why Reassemble?



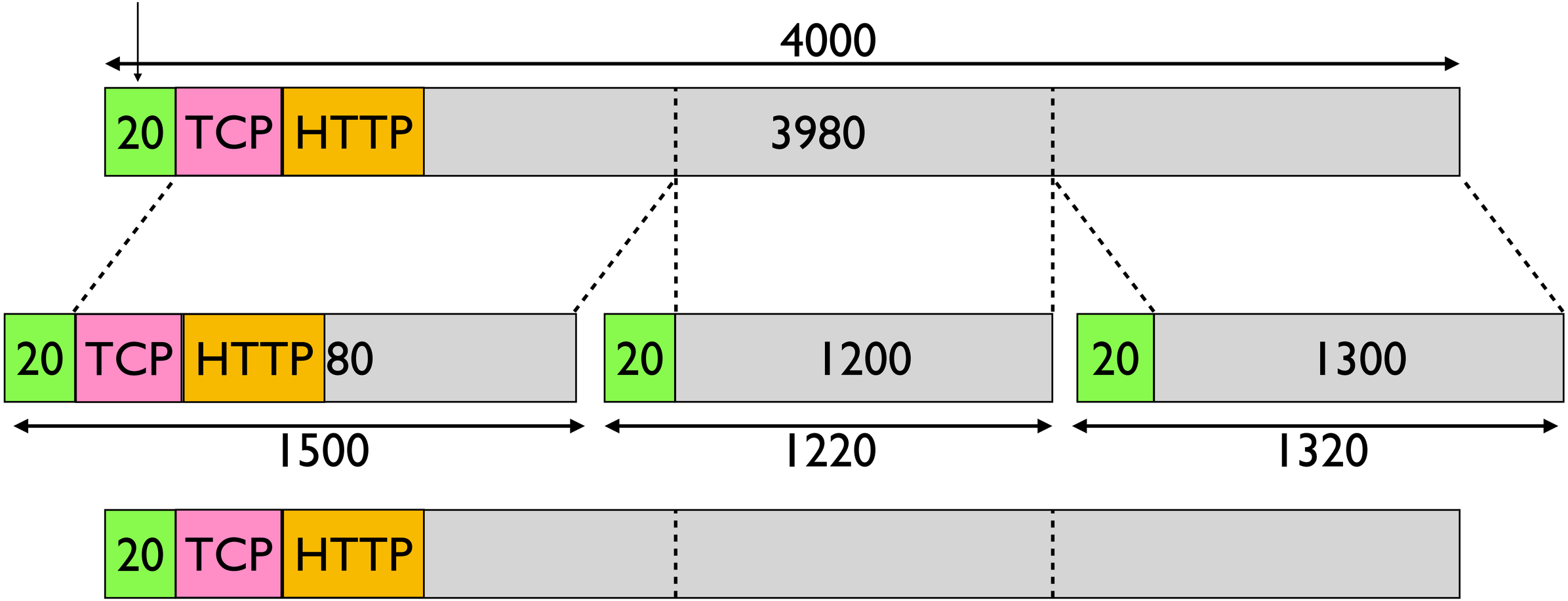
Why Reassemble?



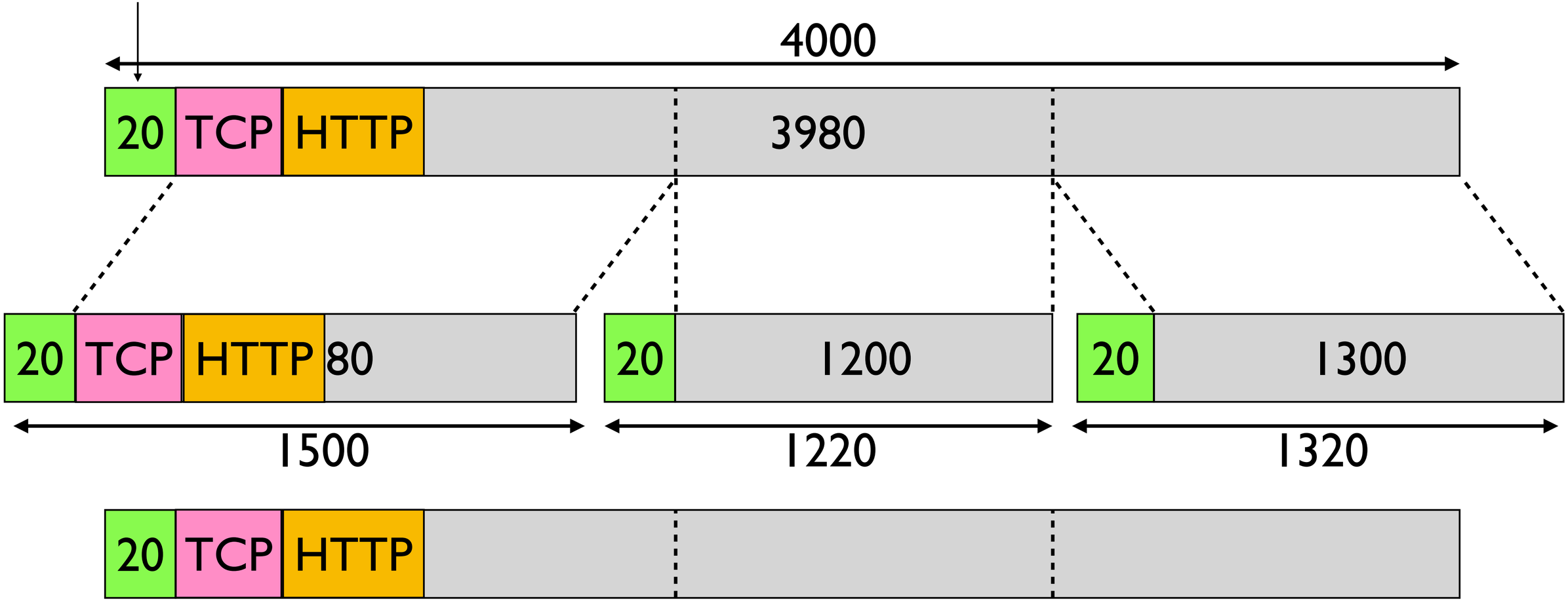
Why Reassemble?



Why Reassemble?



Why Reassemble?



Must reassemble before sending the packet to higher layers!

Where should reassembly occur?

Where should reassembly occur?

- **At next-hop router imposes burden on network**
  - *Complicated reassembly algorithm*
  - *Must hold on to fragments/state*

Where should reassembly occur?

- **At next-hop router imposes burden on network**
  - *Complicated reassembly algorithm*
  - *Must hold on to fragments/state*
- **Any other router may not work**
  - *Fragments may take different paths*

Where should reassembly occur?

- **At next-hop router imposes burden on network**
  - *Complicated reassembly algorithm*
  - *Must hold on to fragments/state*
- **Any other router may not work**
  - *Fragments may take different paths*
- **Little benefit, large cost for network reassembly**
  - ***Classic case of E2E principle***

Where should reassembly occur?

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

A Few Considerations

A Few Considerations

- **Fragments can get lost**

A Few Considerations

- **Fragments can get lost**
- **Fragments can follow different paths**

A Few Considerations

- **Fragments can get lost**
- **Fragments can follow different paths**
- **Fragments can get fragmented again**

Reassembly: what fields?

Reassembly: what fields?

- **How to identify which fragments belong together?**
  - Need an ***identifier***

Reassembly: what fields?

- **How to identify which fragments belong together?**
  - Need an ***identifier***
- **How to handle out-of-order or lost fragments?**
  - Need a ***sequence number*** or ***offset***

Reassembly: what fields?

- **How to identify which fragments belong together?**
  - Need an ***identifier***
- **How to handle out-of-order or lost fragments?**
  - Need a ***sequence number*** or ***offset***
- **How do we know we have all fragments?**
  - Need a ***max sequence number*** or ***flag***

IPv4's fragmentation fields

IPv4's fragmentation fields

- **Identifier:** which fragments belong together

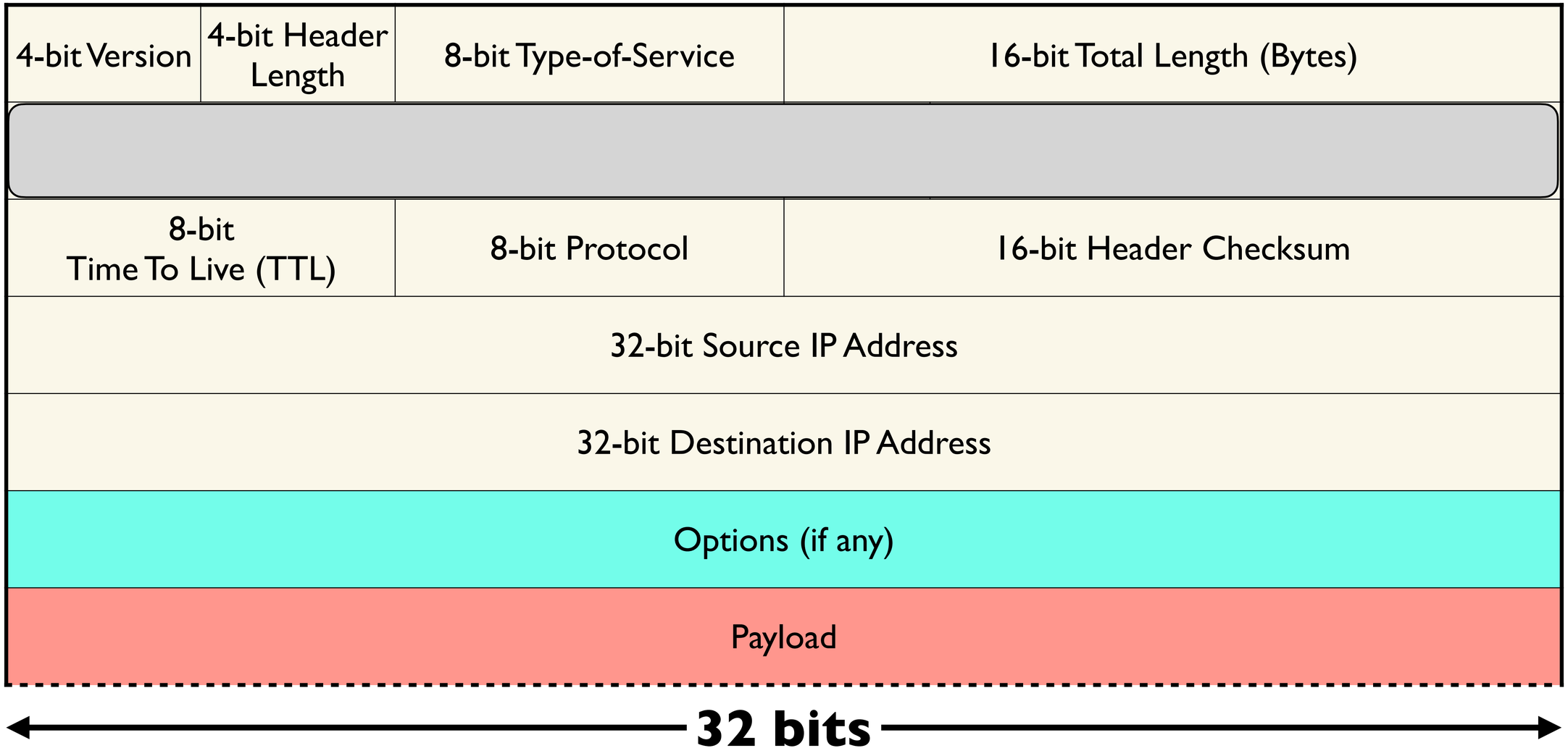
IPv4's fragmentation fields

- **Identifier:** which fragments belong together
- **Offset:** portion of original payload this fragment contains
  - In 8-byte units

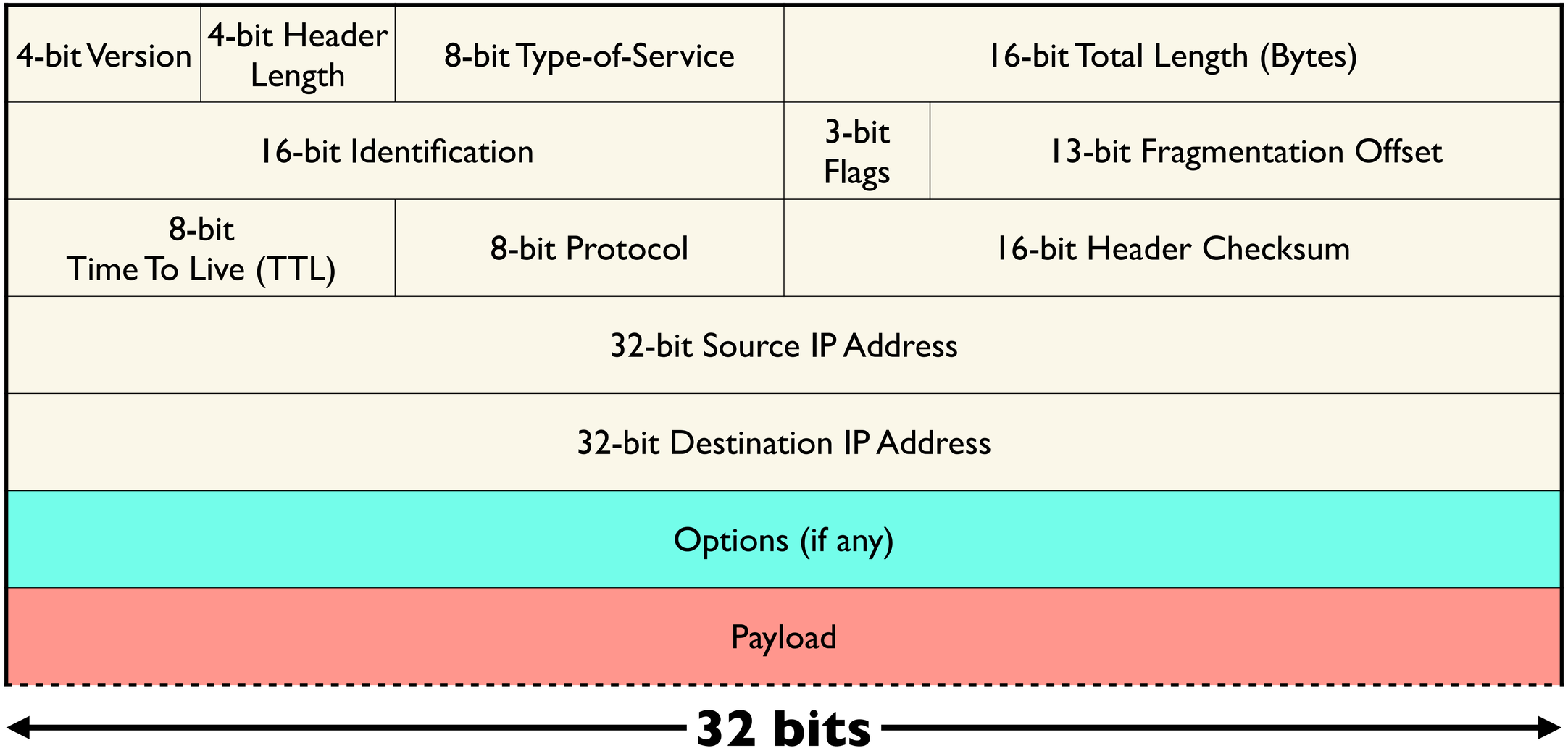
IPv4's fragmentation fields

- **Identifier:** which fragments belong together
- **Offset:** portion of original payload this fragment contains
  - In 8-byte units
- **Flags**
  - **R**eserved: ignore
  - **D**F: don't fragment
    - May trigger error message back to sender
  - **M**F: more fragments coming

IP Packet Structure



IP Packet Structure



Why This Works

Why This Works

- **Fragment without MF set (last fragment)**
  - Tells end-system which are the last bits in original payload

Why This Works

- **Fragment without MF set (last fragment)**
  - Tells end-system which are the last bits in original payload
- **All other fragments fill in holes**

Why This Works

- **Fragment without MF set (last fragment)**
  - Tells end-system 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

Why This Works

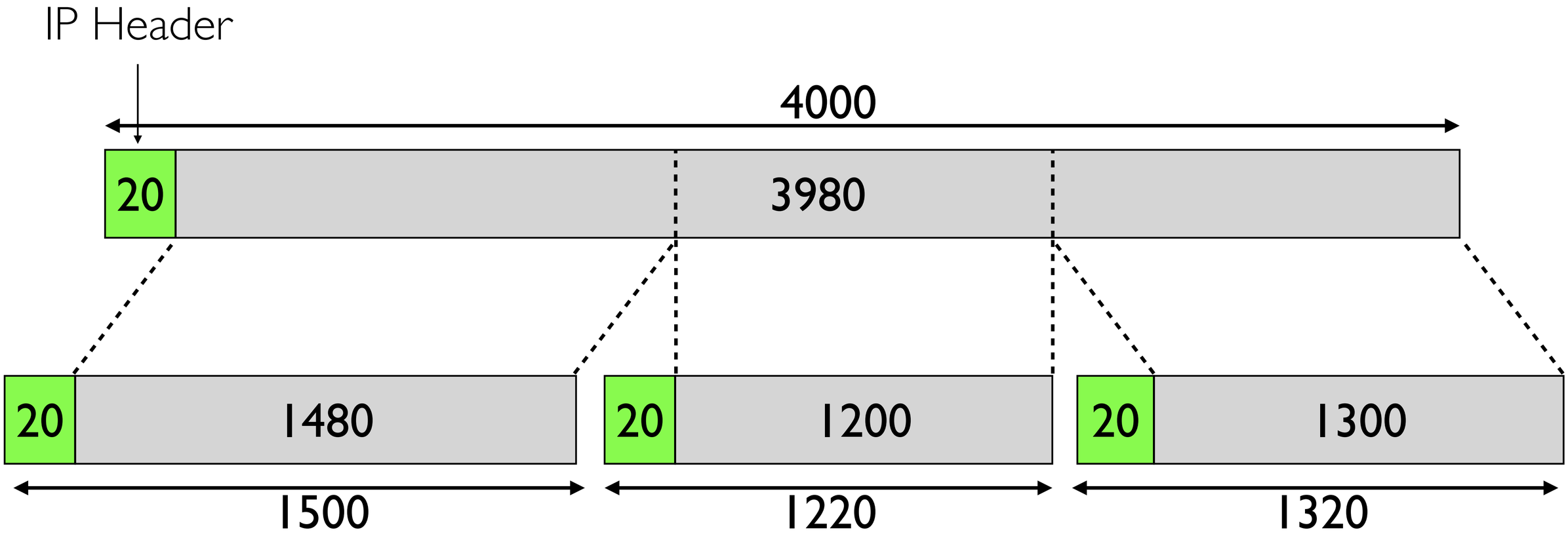
- **Fragment without MF set (last fragment)**
  - Tells end-system which are the last bits in original payload
- **All other fragments fill in holes**
- **Can tell when holes are filled, regardless of order**
  - Use offset field
- **Q:** Why use a byte-offset for fragments rather than numbering each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

Why This Works

- **Fragment without MF set (last fragment)**
  - Tells end-system 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?
- **A:** Allows further fragmentation of fragments

Example of Fragmentation (Contd.)

- **A 4000 byte packet crosses a link with MTU = 1500B**



Example of Fragmentation (Contd.)

- **4000B packet from host 1.2.3.4 to 5.6.7.8 traverses link with MTU 1500B**

Version 4	Header Length 5	Type-of-Service 0	Total Length (Bytes): 4000	
Identification: 56273			R/D/M 0/0/0	Fragmentation Offset: 0
Time To Live (TTL) 127		Protocol 6	Checksum: 44019	
Source IP Address: 1.2.3.4				
Destination IP Address: 5.6.7.8				

Example of Fragmentation (Contd.)

- **Possible first piece:**

<table>
<tr><td>Version
4</td><td>Header Length
5</td><td>Type-of-Service
0</td><td colspan="2">Total Length (Bytes): 1500</td></tr>
<tr><td colspan="3">Identification: 56273</td><td>R/D/M
0/0/1</td><td>Fragmentation Offset: 0</td></tr>
<tr><td colspan="2">Time To Live (TTL)
127</td><td>Protocol
6</td><td colspan="2">Checksum: xxx</td></tr>
<tr><td colspan="5">Source IP Address: 1.2.3.4</td></tr>
<tr><td colspan="5">Destination IP Address: 5.6.7.8</td></tr>
</table>

Example of Fragmentation (Contd.)

- **Possible second piece (Fragment#1 covered 1480 bytes)**

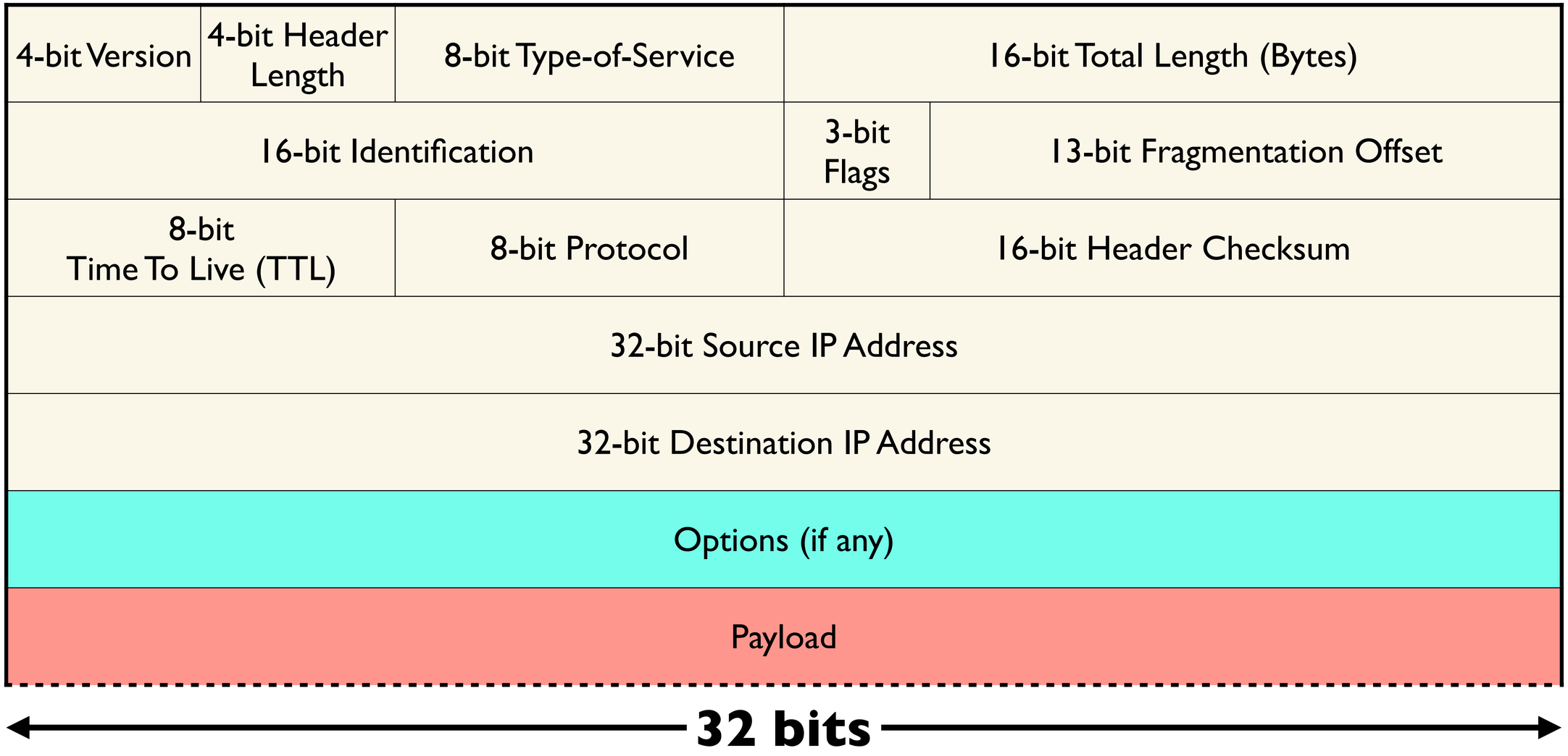
Version 4	Header Length 5	Type-of-Service 0	Total Length (Bytes): 1220	
Identification: 56273			R/D/M 0/0/1	Fragmentation Offset: 185 (185 * 8 = 1480)
Time To Live (TTL) 127		Protocol 6	Checksum: yyy	
Source IP Address: 1.2.3.4				
Destination IP Address: 5.6.7.8				

Example of Fragmentation (Contd.)

- **Possible third piece (Fragment#1&2 covered 1480+1200=2680 bytes)**

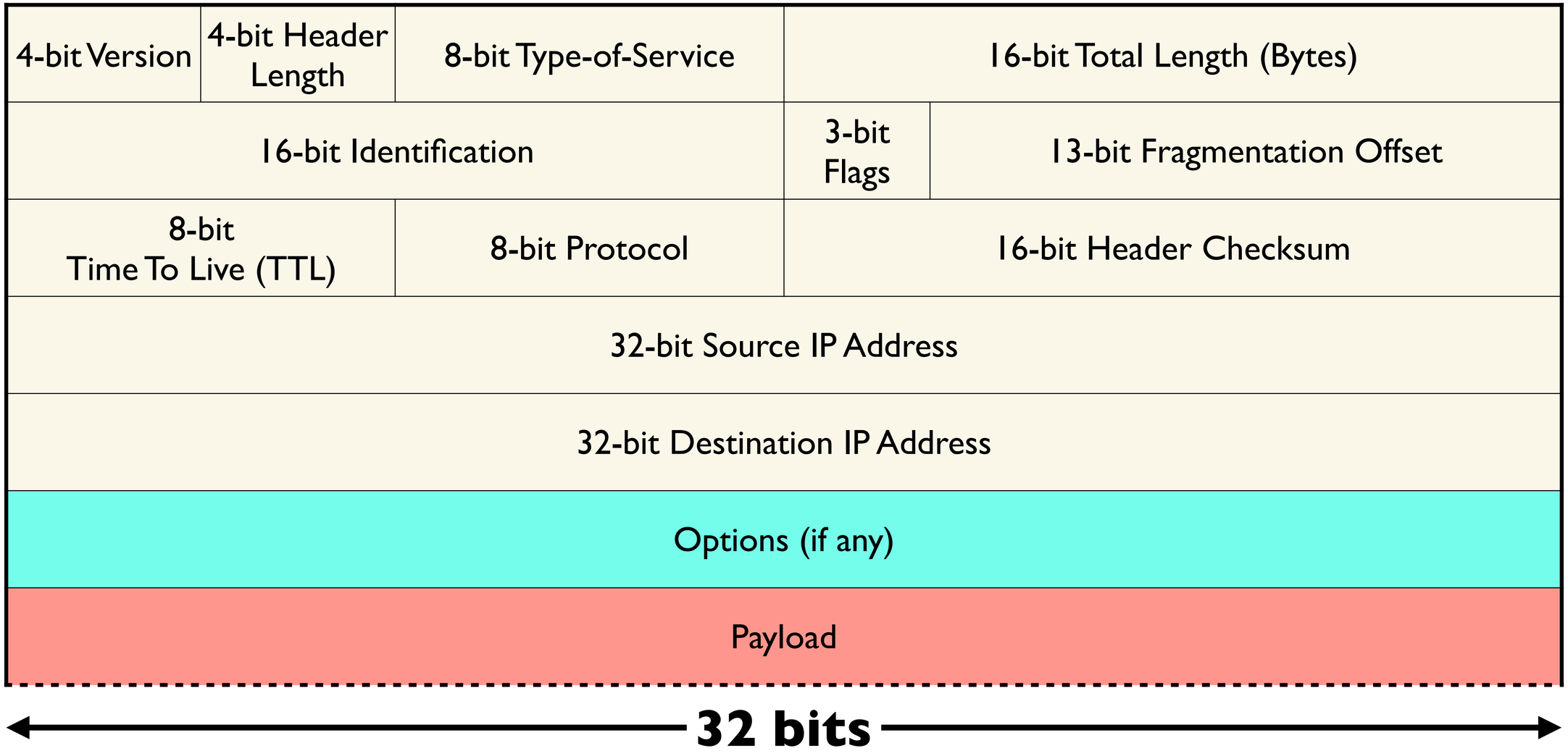
<table>
<tr><td>Version
4</td><td>Header Length
5</td><td>Type-of-Service
0</td><td colspan="2">Total Length (Bytes): 1320</td></tr>
<tr><td colspan="3">Identification: 56273</td><td>R/D/M
0/0/0</td><td>Fragmentation Offset: 335
(335 * 8 = 2680)</td></tr>
<tr><td colspan="2">Time To Live (TTL)
127</td><td>Protocol
6</td><td colspan="2">Checksum: zzz</td></tr>
<tr><td colspan="5">Source IP Address: 1.2.3.4</td></tr>
<tr><td colspan="5">Destination IP Address: 5.6.7.8</td></tr>
</table>

IP Packet Structure



Quick Security Analysis of IP Header

IP Packet Structure



IP Address Integrity

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

Implications of IP Address Integrity

- **Why would someone use a bogus source address?**

Implications of IP Address Integrity

- **Why would someone use a bogus source address?**
- **Launch a Denial-of-Service (DoS) 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

Implications of IP Address Integrity

- **Why would someone use a bogus source address?**
- **Launch a Denial-of-Service (DoS) 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

Implications of IP Address Integrity

- **Why would someone use a bogus source address?**
- **Launch a Denial-of-Service (DoS) 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

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?

Security Implications of ToS?

- **Attacker sets ToS priority for their traffic?**

Security Implications of ToS?

- **Attacker sets ToS priority for their traffic?**
  - If regular traffic does not set ToS, *then network preferentially treats the attack traffic*, greatly increasing damage

Security Implications of ToS?

- **Attacker sets ToS priority for their traffic?**
  - If regular traffic does not set ToS, *then network preferentially treats the attack traffic*, greatly increasing damage
- **What if network *charges* for ToS traffic…**

Security Implications of ToS?

- **Attacker sets ToS priority for their traffic?**
  - If regular traffic does not set ToS, *then network preferentially treats the attack traffic*, greatly increasing damage
- **What if network *charges* for ToS traffic…**
  - … and attacker spoofs the victim's source address?

Security Implications of ToS?

- **Attacker sets ToS priority for their traffic?**
  - If regular traffic does not set ToS, *then network preferentially treats 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**

Security Implications of ToS?

- **Attacker sets ToS priority for their traffic?**
  - If regular traffic does not set ToS, *then network preferentially treats 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**

Security Implications of ToS?

- **Attacker sets ToS priority for their traffic?**
  - If regular traffic does not set ToS, *then network preferentially treats 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?

Security Implications of Fragmentation?

- **Allows evasion of network monitoring/enforcement**

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"

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"

Offset=0

Nasty-at

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"

Offset=0	Offset=8
Nasty-at	**tack-bytes**

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"

Offset=0	Offset=8
Nasty-at	**tack-bytes**

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

Security Implications of TTL?

Security Implications of TTL?

- **Allows discovery of topology (a la *traceroute*)**

Security Implications of TTL?

- **Allows discovery of topology (a la *traceroute*)**
- **Can provide a hint that a packet is spoofed**

Security Implications of TTL?

- **Allows discovery of topology (a la *traceroute*)**
- **Can provide a hint that a packet is spoofed**
  - It arrives at a router with a TTL different from packets from that address (usually)

Security Implications of TTL?

- **Allows discovery of topology (a la *traceroute*)**
- **Can provide a hint that a packet is spoofed**
  - It arrives at a router with a TTL different from packets from that address (usually)
    - Because path from from attacker to router has different # hops

Security Implications of TTL?

- **Allows discovery of topology (a la *traceroute*)**
- **Can provide a hint that a packet is spoofed**
  - It arrives at a router with a TTL different from packets from that address (usually)
    - Because path from from attacker to router has different # hops
  - Through this is *brittle* in the presence of routing changes

Security Implications of TTL?

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

Security Implications of TTL?

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

Other Security Implications?

Other Security Implications?

- **No apparent problems with protocol field (8 bits)**

Other Security Implications?

- **No apparent problems with protocol field (8 bits)**
  - Its just a demuxing handle

Other Security Implications?

- **No apparent problems with protocol field (8 bits)**
  - Its just a demuxing handle
  - If set incorrectly, next layer will find packet ill-formed

Other Security Implications?

- **No apparent problems with protocol field (8 bits)**
  - Its just a demuxing 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**

Other Security Implications?

- **No apparent problems with protocol field (8 bits)**
  - Its just a demuxing 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

IPv6

- **Motivated by address exhaustion**

IPv6

- **Motivated by address exhaustion**
  - Addresses ***four*** times as big (128 bits)

IPv6

- **Motivated by address exhaustion**
  - Addresses ***four*** times as big (128 bits)
  - Number of possible addresses:

IPv6

- **Motivated by address exhaustion**
  - Addresses ***four*** times as big (128 bits)
  - Number of possible addresses:
    - 340 undecillion, or 340 billion billion billion billion

IPv6

- **Motivated by address exhaustion**
  - Addresses ***four*** times as big (128 bits)
  - Number of possible addresses:
    - 340 undecillion, or 340 billion billion billion billion
- **Steve Deering focused on simplifying IP**

IPv6

- **Motivated by address exhaustion**
  - Addresses ***four*** times as big (128 bits)
  - Number of possible addresses:
    - 340 undecillion, or 340 billion billion billion billion
- **Steve Deering focused on simplifying IP**
  - Got rid of all fields that were not absolutely necessary

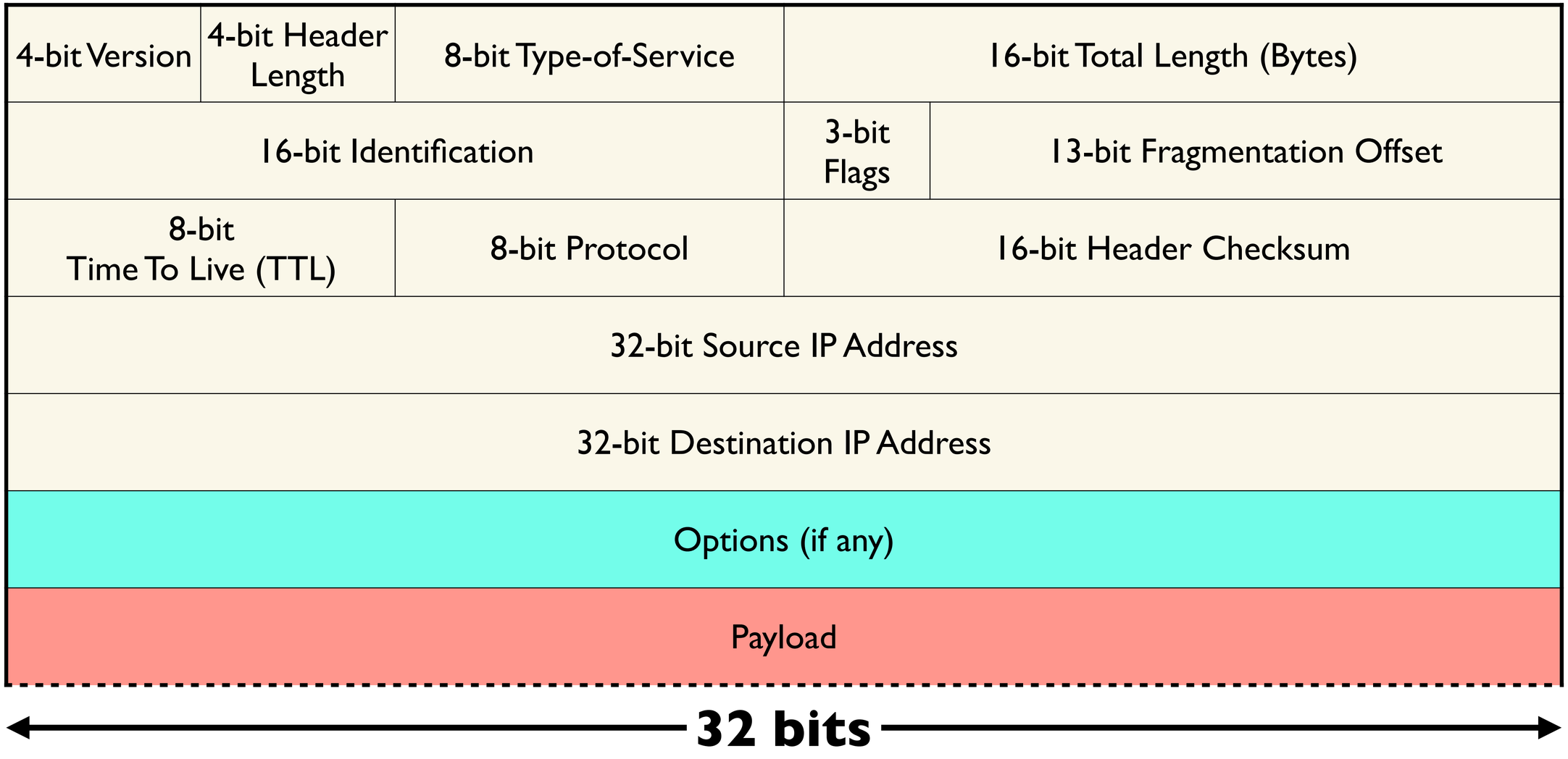
IPv6

- **Motivated by address exhaustion**
  - Addresses ***four*** times as big (128 bits)
  - Number of possible addresses:
    - 340 undecillion, or 340 billion billion billion billion
- **Steve Deering focused on simplifying IP**
  - Got rid of all fields that were not absolutely necessary
  - "Spring cleaning" for IP

IPv6

- **Motivated by address exhaustion**
  - Addresses ***four*** times as big (128 bits)
  - Number of possible addresses:
    - 340 undecillion, or 340 billion billion billion billion
- **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?



IPv6: Summary of Changes

IPv6: Summary of Changes

- **Eliminated fragmentation (*Why?*)**

IPv6: Summary of Changes

- **Eliminated fragmentation (*Why?*)**
- **Eliminated checksum (*Why?*)**

IPv6: Summary of Changes

- **Eliminated fragmentation (*Why?*)**
- **Eliminated checksum (*Why?*)**
- **New options mechanism (next header) (*Why?*)**

IPv6: Summary of Changes

- **Eliminated fragmentation (*Why?*)**
- **Eliminated checksum (*Why?*)**
- **New options mechanism (next header) (*Why?*)**
- **Eliminated header length (*Why?*)**

IPv6: Summary of Changes

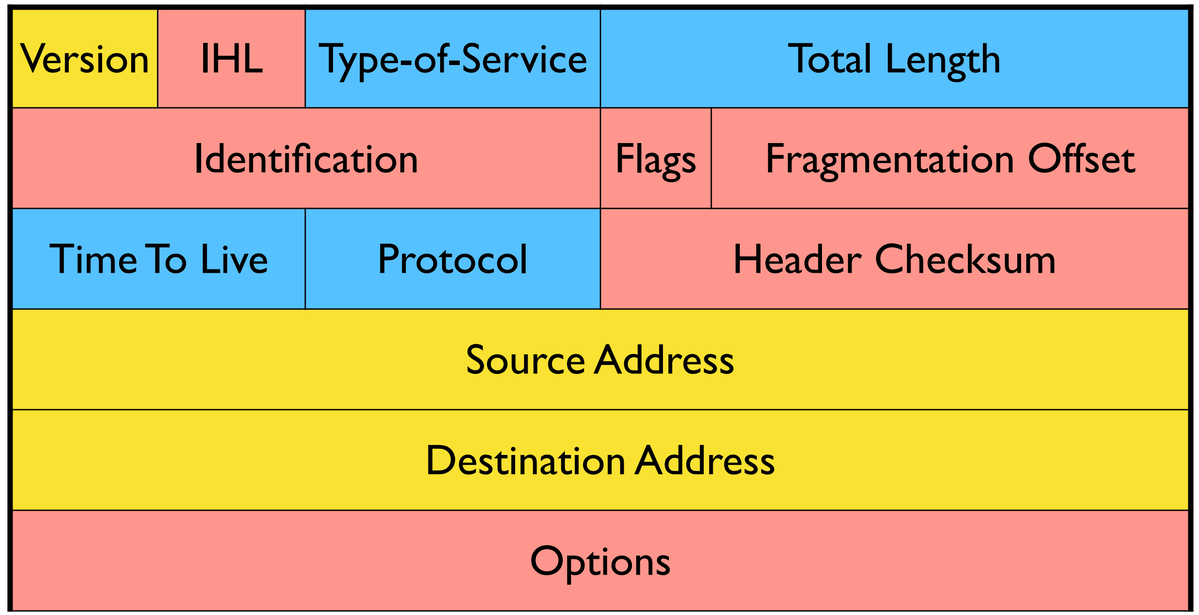
- **Eliminated fragmentation (*Why?*)**
- **Eliminated checksum (*Why?*)**
- **New options mechanism (next header) (*Why?*)**
- **Eliminated header length (*Why?*)**
- **Expanded addresses**

IPv6: Summary of Changes

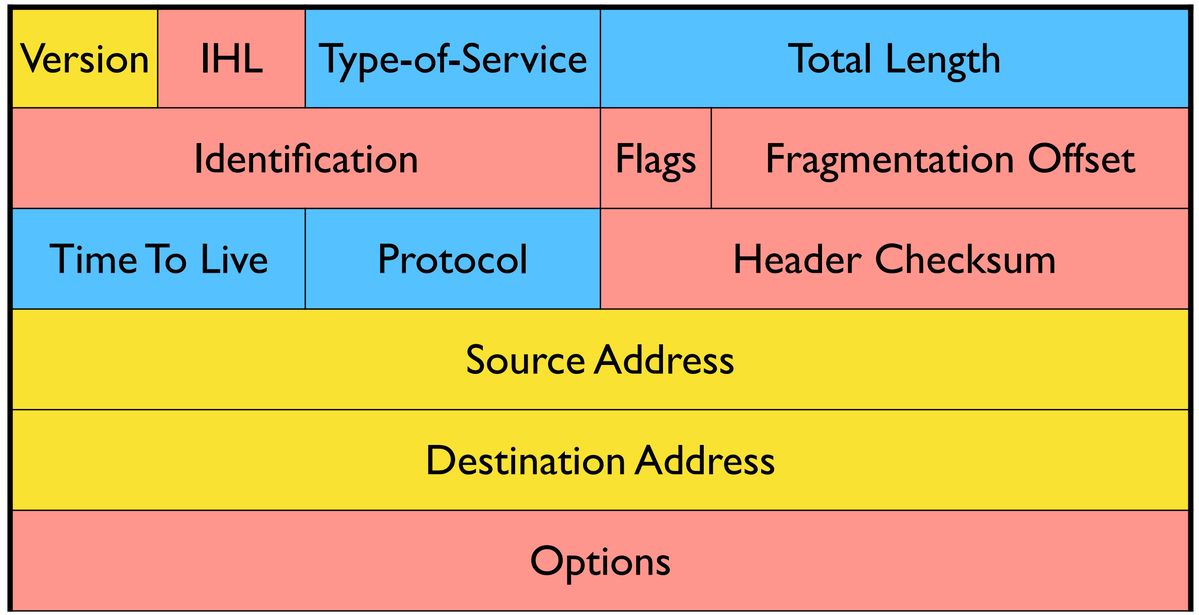
- **Eliminated fragmentation (*Why?*)**
- **Eliminated checksum (*Why?*)**
- **New options mechanism (next header) (*Why?*)**
- **Eliminated header length (*Why?*)**
- **Expanded addresses**
- **Added Flow Label**

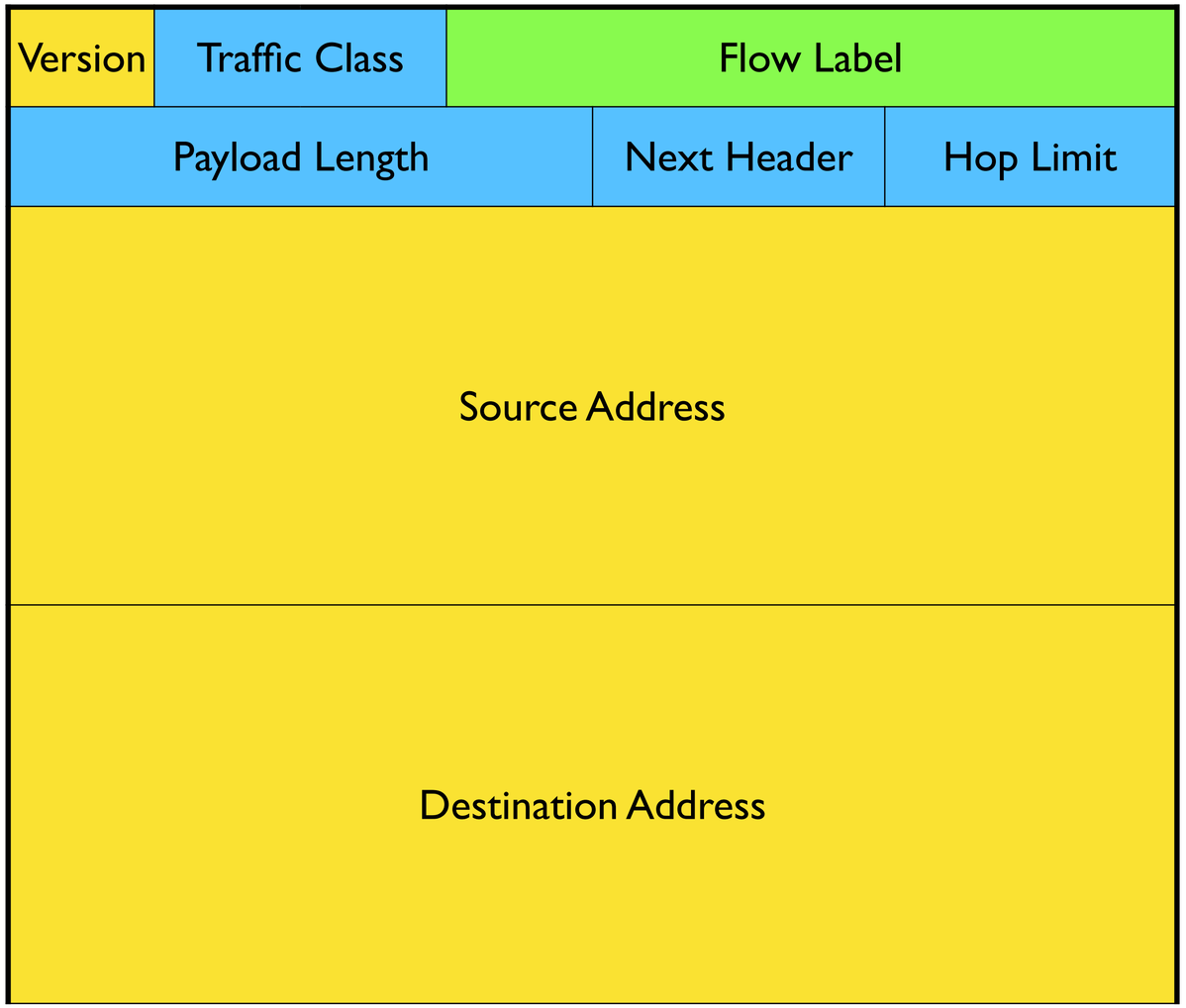
IPv4 and IPv6 Header Comparison

IPv4 and IPv6 Header Comparison

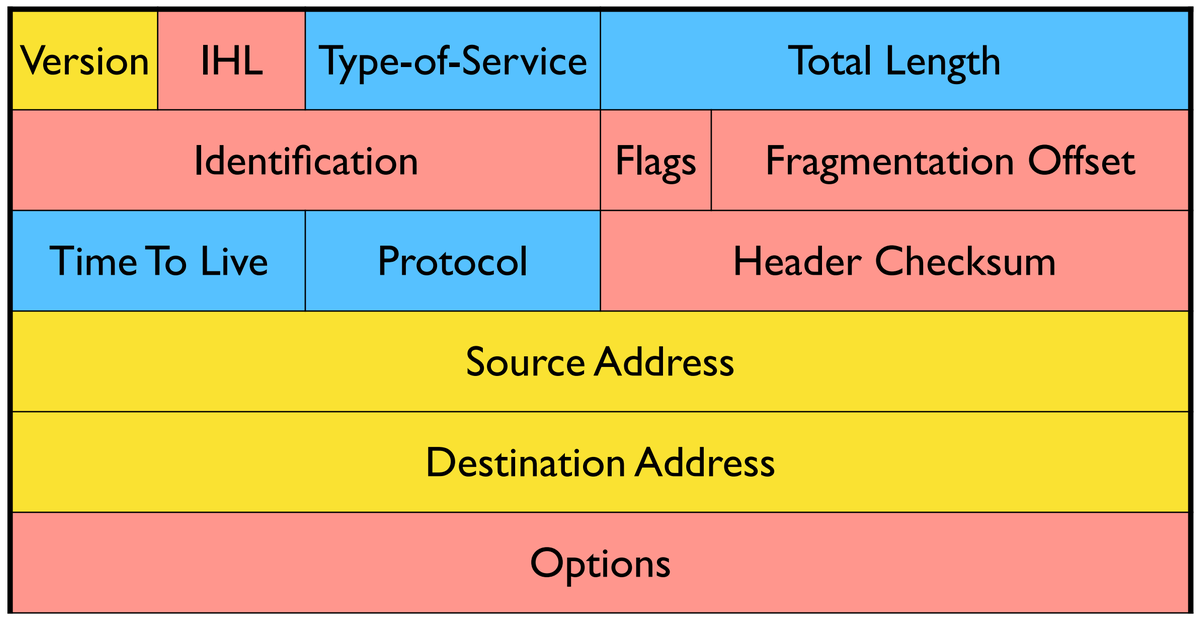


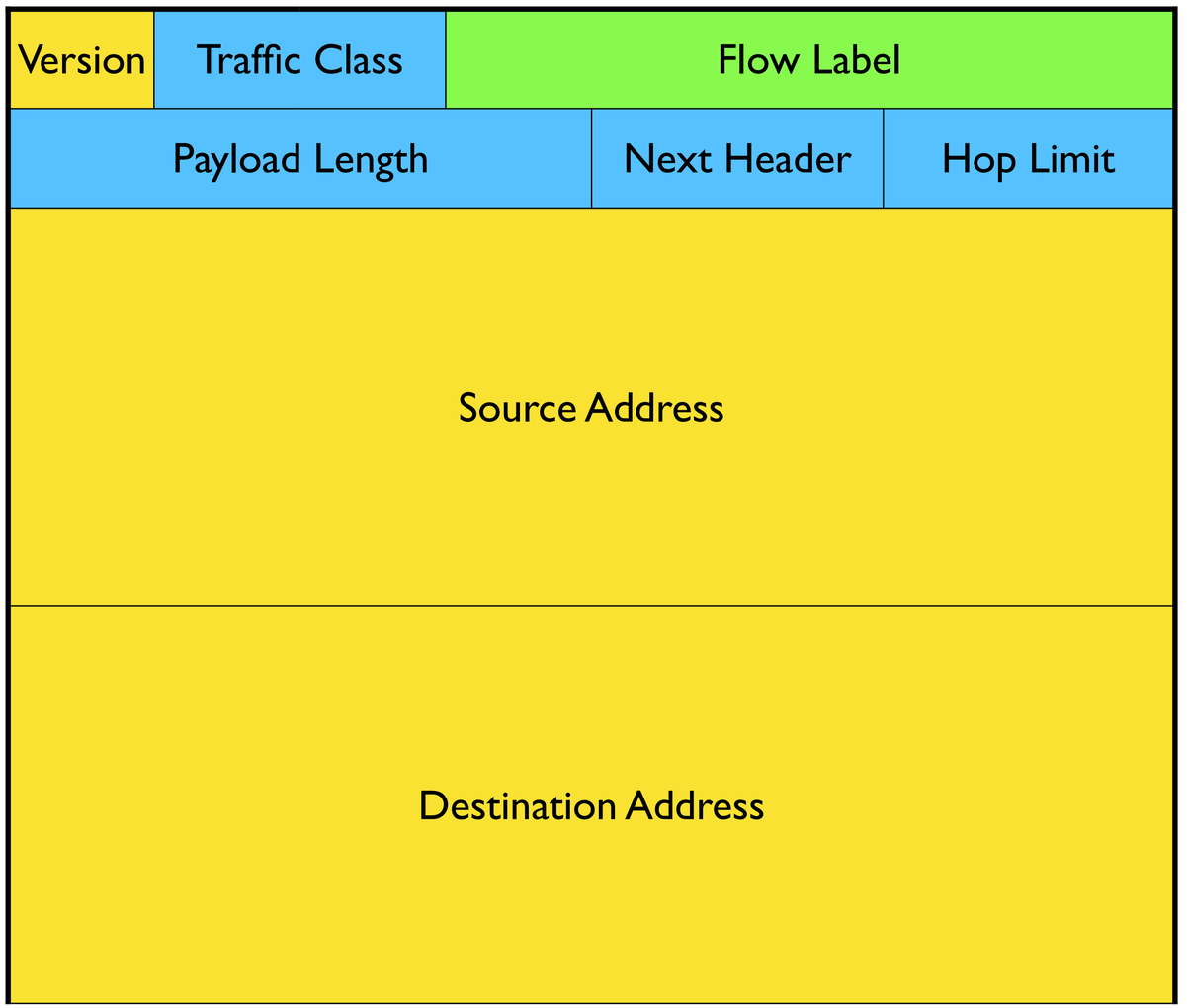
IPv4 and IPv6 Header Comparison

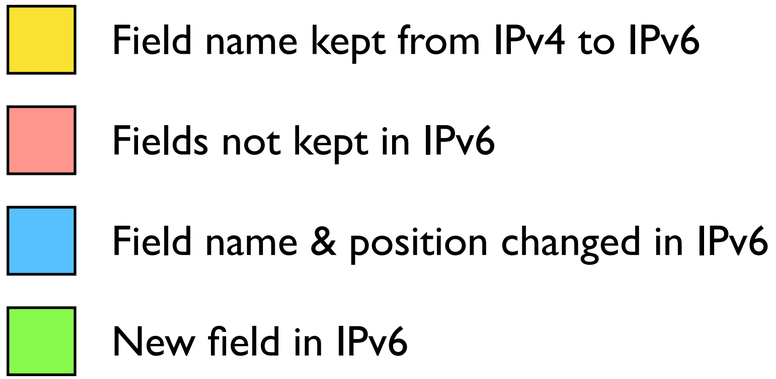




IPv4 and IPv6 Header Comparison







IPv6: Philosophy of changes?

IPv6: Philosophy of changes?

- **Don’t deal with problems: leave to ends**
  - Eliminated fragmentation
  - Eliminated checksum
  - *Why retain TTL?*

IPv6: 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?*

IPv6: 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

Summary: IP header design

Summary: IP header design

- **More nuanced than it first seems!**

Summary: IP header design

- **More nuanced than it first seems!**
- **Must juggle multiple goals**
  - Robustness
  - Efficiency
  - Security
  - Completeness

Summary: 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?

Summary: 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**