Missing Pieces, and Designing IP

EE122 Fall 2012

Scott Shenker

http://inst.eecs.berkeley.edu/~ee122/

Materials with thanks to Jennifer Rexford, Ion Stoica, Vern Paxson and other colleagues at Princeton and UC Berkeley

Questions about Project 1

Announcements

- HW formatting: don’t screw it up.
  - **You have been warned!**

- HW2 out later tonight

- Midterm review???

Today’s Lecture: Two Topics

- Covering some “missing pieces”
  - Maybe networking isn’t as simple as I said….
- Designing IP
  - What should it be doing?
  - What needs to be included in the packet header?

Missing Pieces

Where are we?

- We have covered the "fundamentals"
  - How to deliver packets (routing)
  - How to build reliable delivery on an unreliable network
- With this, we could build a decent network
- But couldn't actually *do* anything with the network
  - Too many missing pieces
- We now want to identify those pieces
  - Will guide what we cover rest of semester

Scenario: Joan Wants Her Music

- Joan is sitting in her dorm room, with a laptop

- Has overwhelming urge to listen to John Cage
  - In particular, his piece *4'33"*
  - *Let's listen to the opening movement…* **(quiet!!)**

- What needs to happen to make this possible?
  - Not in terms of today's protocols…
  - ……but in terms of basic tasks

Did I miss anything?

- Accessing the network from laptop
  - Wireless or ethernet
  - N… it work)
- M…
- M…
- D…
- A…

Before I answer, jot down a few steps. This portion of the lecture won't mean much if you don't try to figure it out.

Talk to your neighbors about it,
talk to yourself about it,
don't just sit there and read your mail….

What Are The Steps Involved?

- Accessing the network from laptop
  - **Wireless or ethernet**
  - Network management (someone needs to make it work)
- Mapping “real world name” to “network name”
- Mapping network name to location
- Download content from location
- Addressing general security concerns
  - Verifying that this is the right content
  - And that no one can tell what she’s downloading

Access Networks

- If access network is "switched", we understand it
  - Just like any other packet-switched network
- If the access network is ***shared*** medium, then we need to figure out how to share the medium
  - Wireless
  - Classical ethernet

Media Access Control (MAC)

- Carrier sense: (CSMA)
  - Don't send if someone else is sending
- Collision detection: (CD)
  - Stop if you detect someone else was also sending
- Collision avoidance: (CA)
  - How to arrange transmissions so that they don't collide

And you know how old people like me like to relive their youth…..

What Are The Steps Involved?

- Accessing the network from laptop
  - Wireless or ethernet
  - **Network management (need to make it work)**
- Mapping “real world name” to “network name”
- Mapping network name to location
- Download content from location
- Addressing general security concerns
  - Verifying that this is the right content
  - And that no one can tell what she’s downloading

Network Management

- Control how network interconnects to Internet
  - Interdomain routing
- Keep unwanted traffic off network
  - Firewalls and access control
- Share limited number of public addresses
  - NAT
- Keep links from overloading
  - Traffic engineering

Most undeveloped part of Internet architecture

Current Network Management

- No abstractions, no layers
- Just complicated distributed algorithms
  - Such as routing algorithms
- Or manual configuration
  - Such as Access Control Lists and Firewalls

Future Network Management

- Clean abstractions
- No complicated distributed algorithms
- Treat networks like systems…

Two lectures later in semester!

Find out why stick shifts are the root of all evil in networking!

What Are The Steps Involved?

- Accessing the network from laptop
  - Wireless or ethernet
  - Network management (someone needs to make it work)
- **Mapping “real world name” to “network name”**
- Mapping network name to location
- Download content from location
- Addressing general security concerns
  - Verifying that this is the right content
  - And that no one can tell what she’s downloading

“Real World Name” to “Network Name”

- Joan knows what music she wants
- Doesn’t know how to tell network what she wants
- Needs to map “real world .

How can we do this?

- …..to a name that the infrastructure understands
  - We will call this the “network name” but this isn’t a name at the IP level, but another portion of the infrastructure
- **Search engine!**
  - Maps keywords to URL

What is a “Network Name”?

- HTTP://www.youtube.com/watch?v=hUJagb7hL0E
- HTTP is host-to-host protocol
- www.youtube.com is a “host name”
  - Widely replicated, but still represents a host
- watch?v=hUJagb7hL0E is meaningful to host

What Are The Steps Involved?

- Accessing the network from laptop
  - Wireless or ethernet
  - Network management (someone needs to make it work)
- Mapping “real world name” to “network name”
- **Mapping network name to location**
- Download content from location
- Addressing general security concerns
  - Verifying that this is the right content
  - And that no one can tell what she’s downloading

Map Network Name to Location

- "Name resolution" converts name to location
  - Location is IP address of host

- We would like location to be nearby copy
  - Speeds up download
  - Reduce load on backbone and access networks

How is this done today?

- Name resolution: Domain Name System (DNS)
  - Hand in a hostname, get back an IP address
- Nearby copy of the data?
  - CDNs: content distribution networks (like Akamai)
- P2P systems can also point you to nearby content

What Are The Steps Involved?

- Accessing the network from laptop
  - Wireless or ethernet
  - Network management (someone needs to make it work)
- Mapping “real world name” to “network name”
- Mapping network name to location
- **Download content from location**
- Addressing general security concerns
  - Verifying that this is the right content
  - And that no one can tell what she’s downloading

Download Data from Location

- Need a reliable transfer protocol: TCP
  - Must share network with others: congestion control

- But must be able to use URL to retreive content
  - Need higher-level protocol like HTTP to coordinate

What Are The Steps Involved?

- Accessing the network from laptop
  - Wireless or ethernet
  - Network management (someone needs to make it work)
- Mapping “real world name” to “network name”
- Mapping network name to location
- Download content from location
- **Addressing general security concerns**
  - Verifying that this is the right content
  - And that no one can tell what she’s downloading

Ensuring Security

- **Privacy**: prevent sniffers from knowing what she downloaded ("it was for EE122, I promise!")
- **Integrity**: ensure data wasn't tampered with during its trip through network
- **Provenance**: ensure that music actually came from the music company (and not some imposter)

How do we do this today?

- Cryptographic measures enable us to do all three
- Public Key cryptography is crucial
  - No need to share secrets beforehand

Scenario Requires

- Media Access Control
- Network management
- Naming and name resolution
- Content distribution networks
- And perhaps P2P
- Congestion control
- HTTP
- Cryptographic measures to secure content

Rest of Course

- Details of IP and TCP
  - Bringing reality to general concepts
- Filling in pieces of name resolution and HTTP
- Congestion control
- Advanced routing
- Security
- Ethernet and Wireless
- Network Management
- What if we were to redesign Internet from scratch

Break

The Design of IP

We are about to make a transition!

From heady principles…
...to packet headers

From essentials…
…to esoterica

From fundamentals…
…to no-fun-at-all

What I'll try to get through….

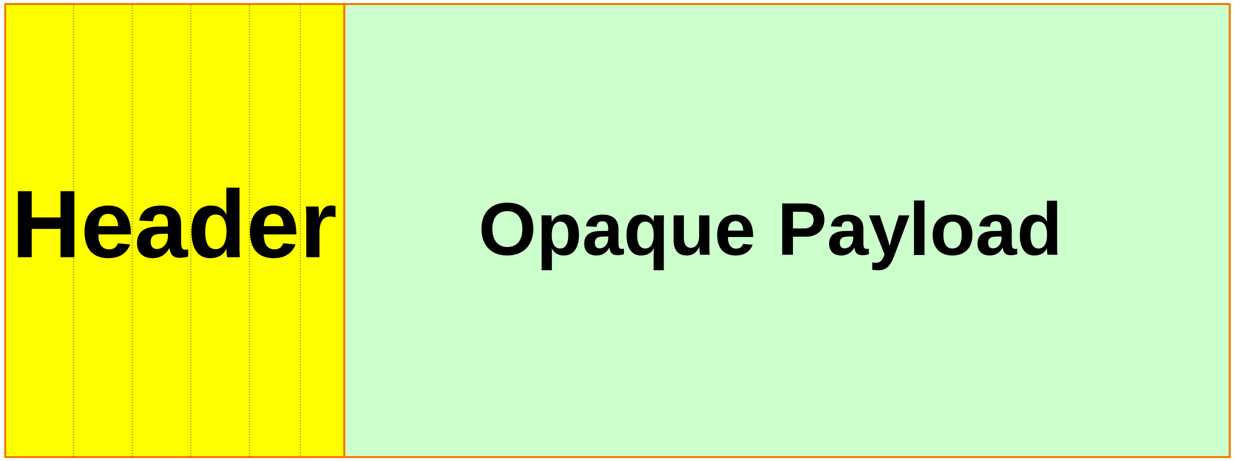
- Design-it-yourself packet header
- IP header (maybe)
- Comparison with IPv6 (not a chance)

What is “designing” a protocol?

- Specifying the ***syntax*** of its messages
  - Format
- Specifying their ***semantics***
  - Meaning
  - Responses

What is Designing IP?

- Syntax: format of packet
  - Nontrivial part: packet “header”
  - Rest is opaque payload *(why opaque?)*



- Semantics: meaning of header fields
  - Required processing

Packet Header as Interface

- Think of packet header as interface
  - Only way of passing information from packet to switch
- Designing interfaces:
  - What task are you trying to perform?
  - What information do you need to accomplish it?
- Header reflects information needed for basic tasks

In-Class Exercise

- Five minutes to design the IPv7 packet header
  - **Do not** look at book, or otherwise copy IPv4 or IPv6
  - ***Do work in groups***
- Goal not to get right answer, but to think about:
  - What tasks are involved?
  - How can a packet header accomplish it?
- Note: IPv4 is not a great model
  - Try to do better!

I'll Take Two or Three Answers

- You tell me your:
  - Task list
  - Corresponding information in header
  - *And any deep insights about architecture?* **(Optional!**)

- *Example:*
  - Task 1: *get packet to destination*
  - Header information: *destination address*

Answer #1:

- Destination address
- TTL

What Tasks Do We Need to Do?

- Read packet correctly
- Get packet to the destination
- Get responses to the packet back to source
  - Not really, but humor me….
- Carry data
- Tell host what to do with packet once arrived
- Specify any special network handling of the packet
- Deal with problems that arise along the path

Reading Packet Correctly

- Where does header end?
- Where does packet end?
- What version of IP?
  - *Why is this so important?*

Getting to the Destination

- Provide destination address (duh!)

- Should this be location or identifier?
  - And what's the difference?

- If a host moves, should its address change?
  - If not, how can you build scalable Internet?
  - If so, then what good is an address for identification?

Getting Response Back to Source

- Source address (duh!)
- You’ve already heard my rant on this….

Carry Data

- Payload (duh!)

Telling Dest'n How to Process Packet

- Indicate which protocols should handle packet
- What layer should this protocol be in?
- What are some options for this today?
- How does the source know what to enter here?

Special Handling

- Type-of-service: Priority, etc.
- Options: discuss later

Dealing with Problems

- Is packet caught in loop?
  - TTL
- Header Corrupted:
  - Detect with Checksum
  - What about payload checksum?
- Packet too large?
  - Deal with fragmentation
  - Split packet apart
  - Keep track of how to put together

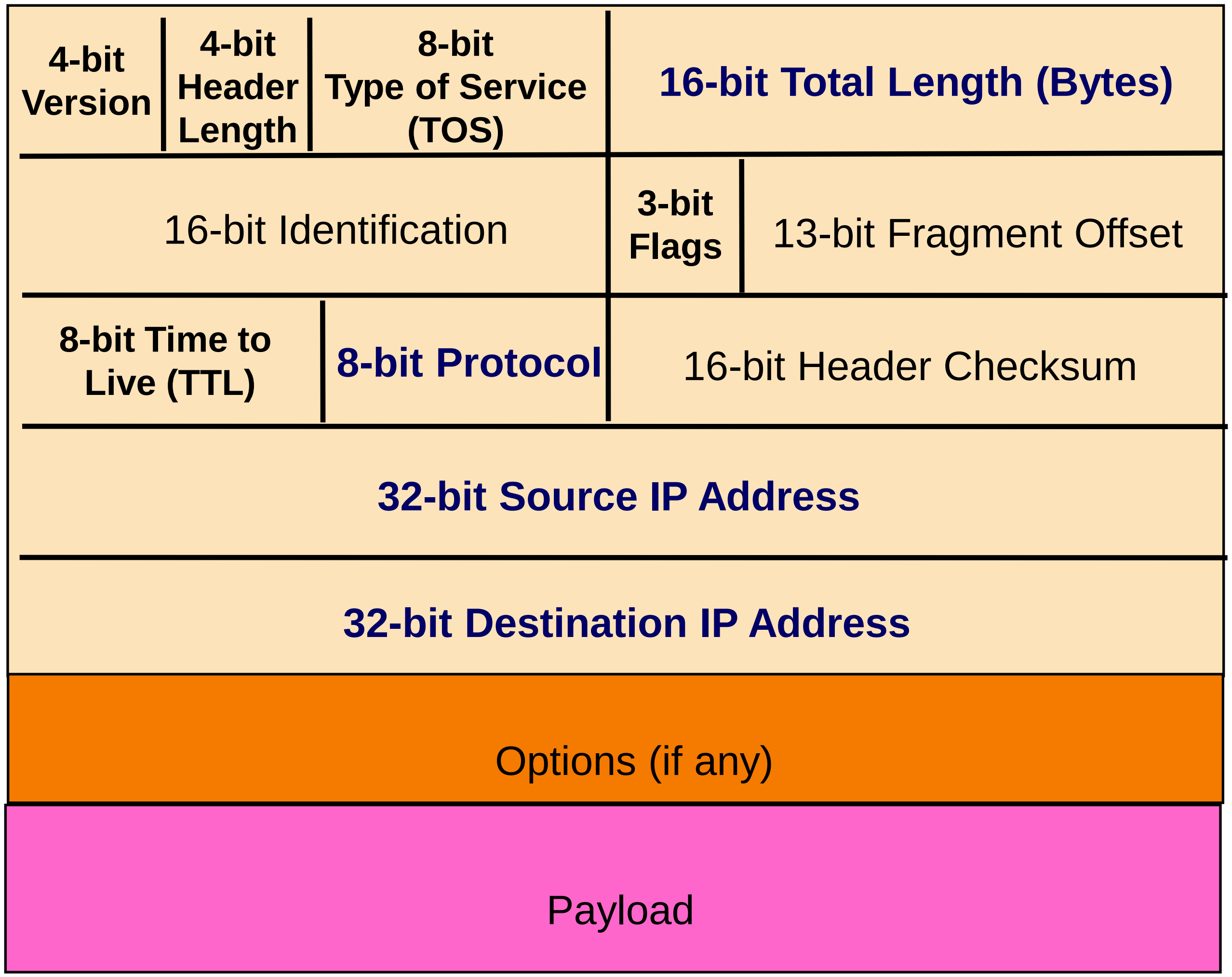
Are We Missing Anything?

- Read packet correctly
- Get packet to the destination
- Get responses to the packet back to source
- Carry data
- Tell host what to do with packet once arrived
- Specify any special network handling of the packet
- Deal with problems that arise along the path

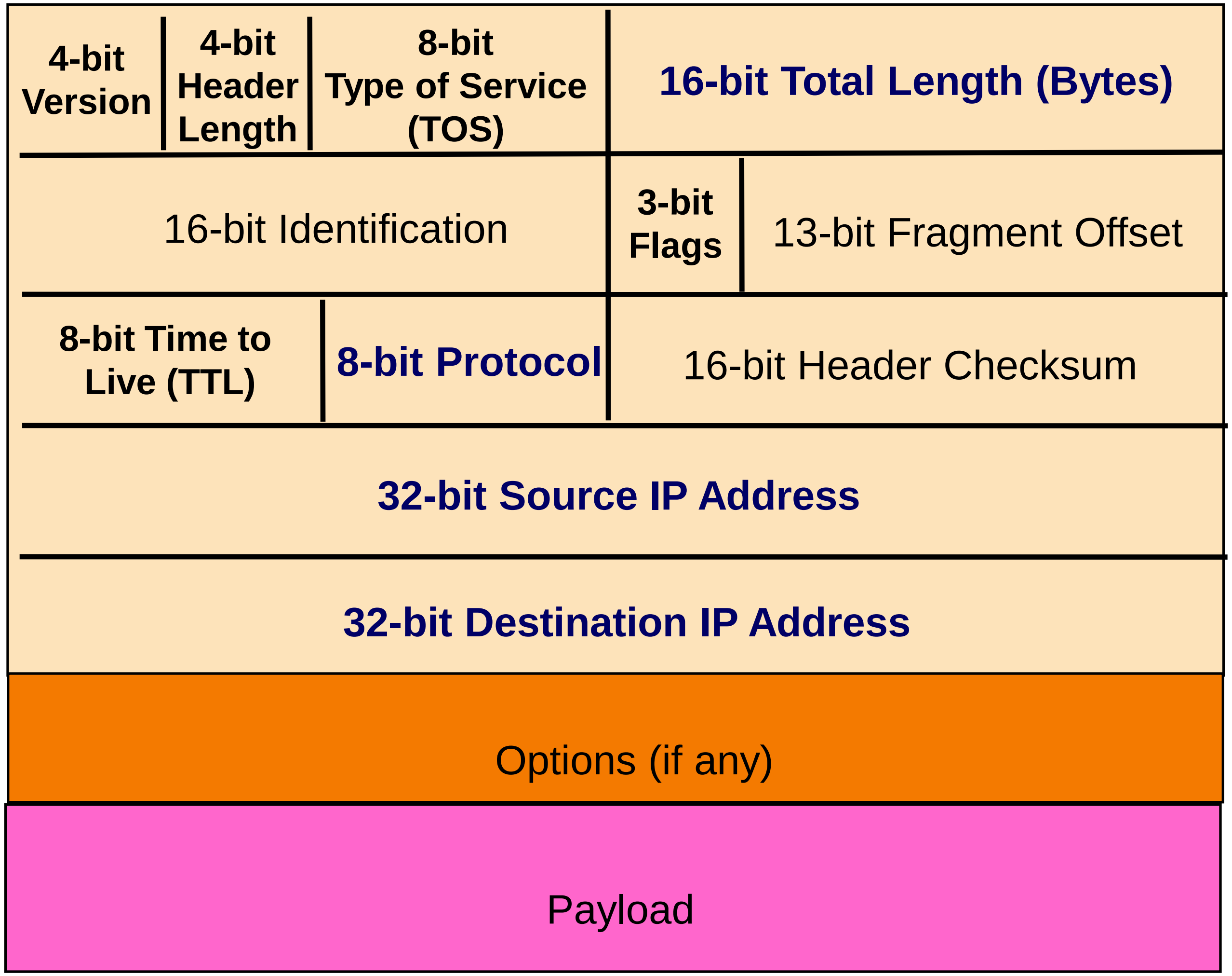
From Semantics to Syntax

- The past few slides discussed the kinds of information the header must provide
- Will now show the syntax (layout) of IPv4 header, and discuss the semantics in more detail

IP Packet Structure



20 Bytes of Standard Header, then Options



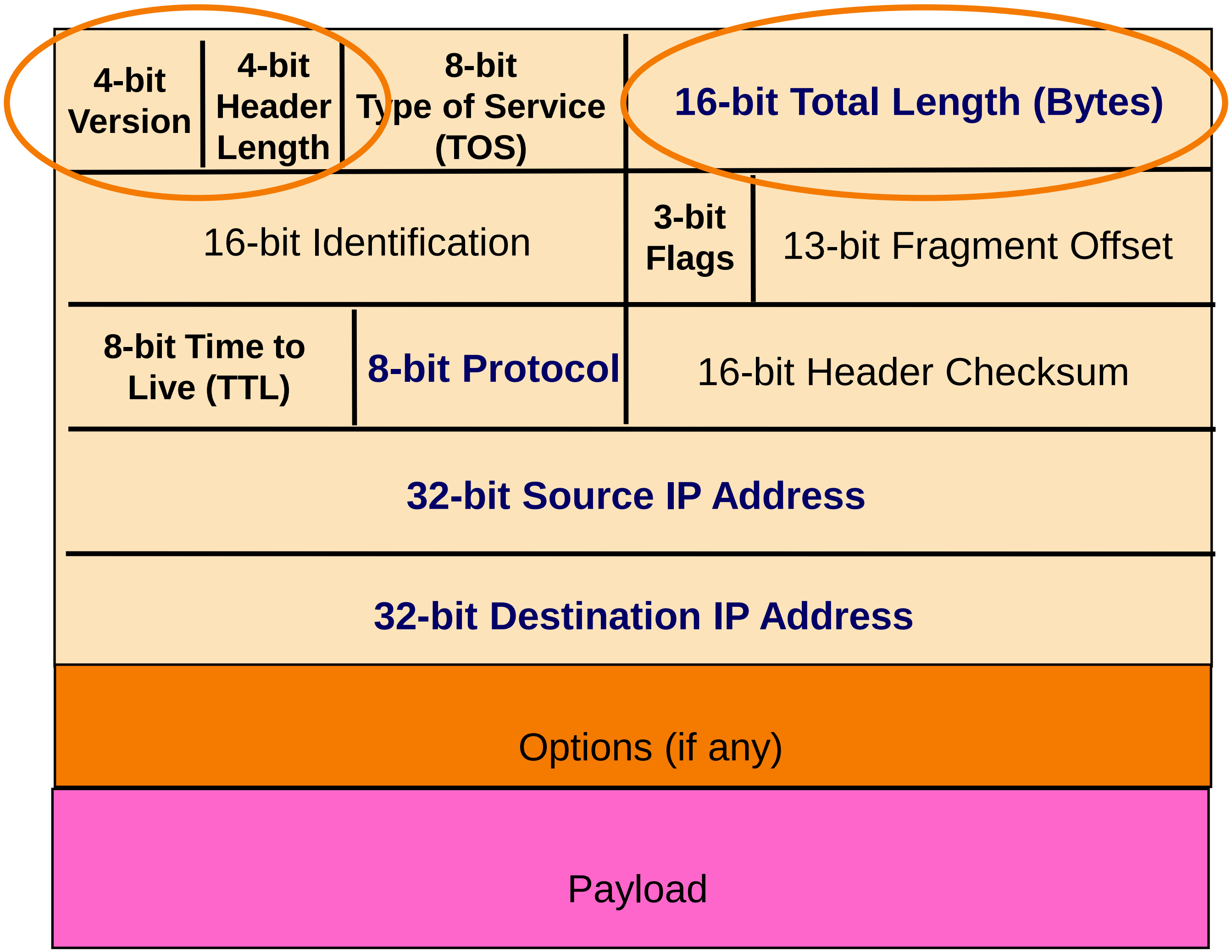
Go Through Tasks One-by-One

- Read packet correctly
- Get packet to the destination
- Get responses to the packet back to source
- Carry data
- Tell host what to do with packet once arrived
- Specify any special network handling of the packet
- Deal with problems that arise along the path

Reading Packet Correctly

- Version number (4 bits)
  - Indicates the version of the IP protocol
  - Necessary to know what other fields to expect
  - Typically “4” (for IPv4), and sometimes “6” (for IPv6)
- 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
- Total length (16 bits)
  - Number of bytes in the packet
  - Maximum size is 65,535 bytes (2^{16} -1)
  - … though underlying links may impose smaller limits

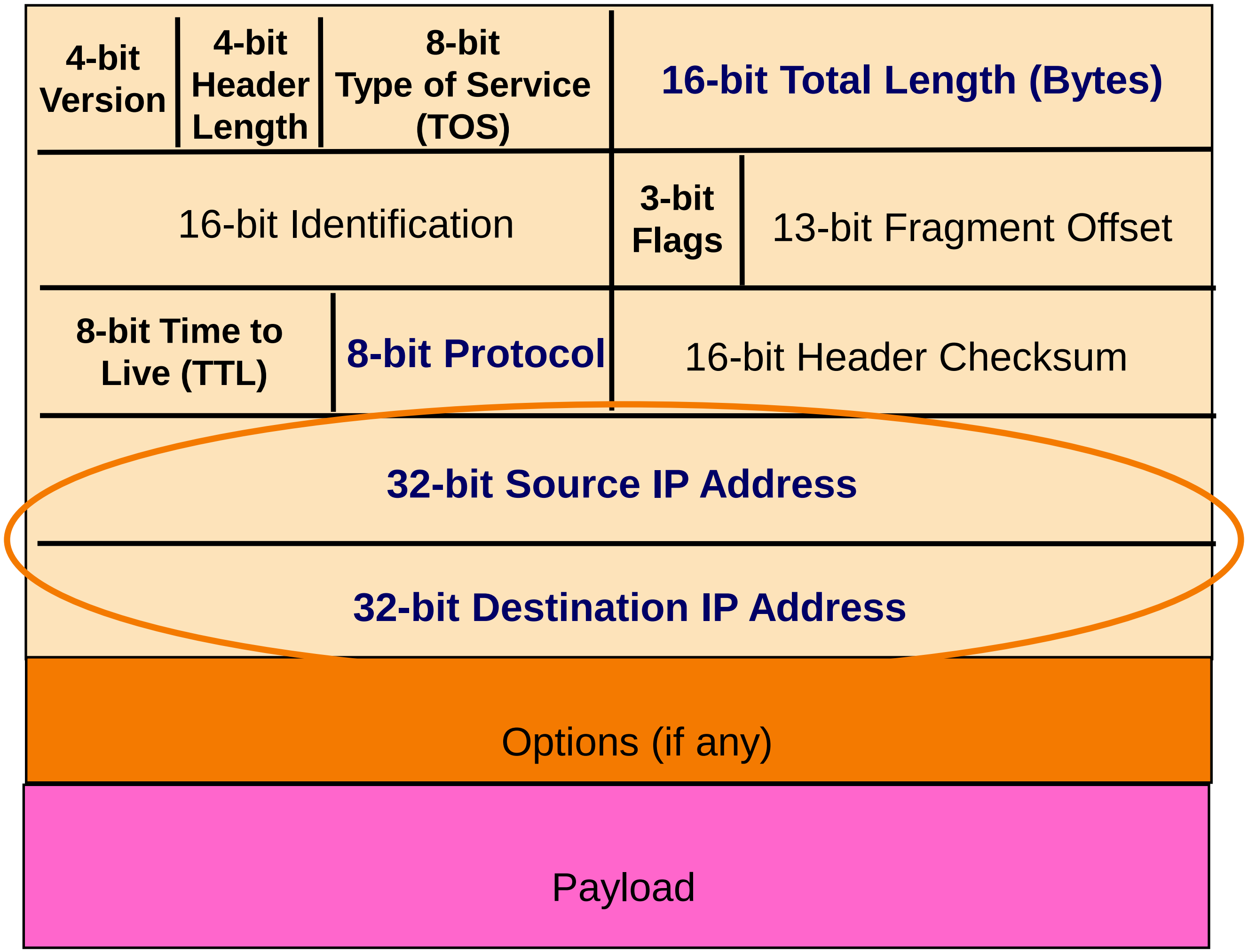
Fields for Reading Packet Correctly



Getting Packet to Destination and Back

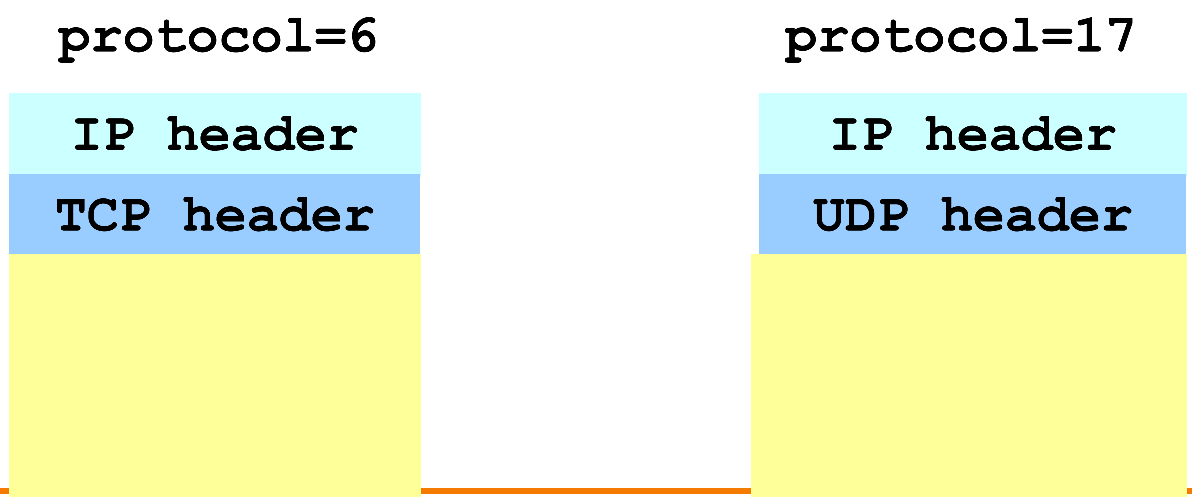
- Two IP addresses
  - Source IP address (32 bits)
  - Destination IP address (32 bits)
- Destination address
  - Unique identifier/locator for the receiving host
  - Allows each node to make forwarding decisions
- Source address
  - Unique identifier/locator for the sending host
  - Recipient can decide whether to accept packet
  - Enables recipient to send a reply back to source

Fields for Packet Reaching Destination

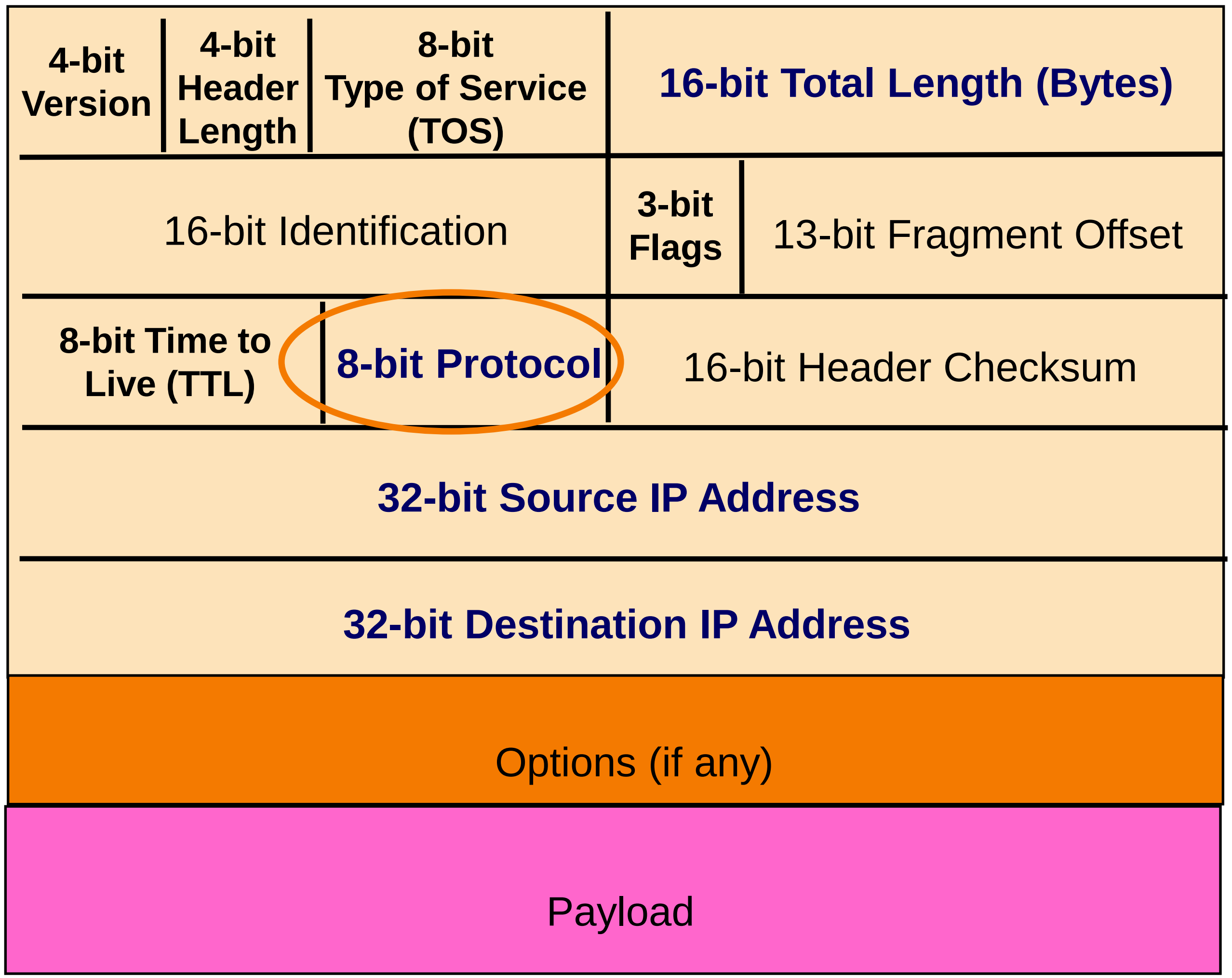


Telling Host How to Handle Packet

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



Field for Next Protocol

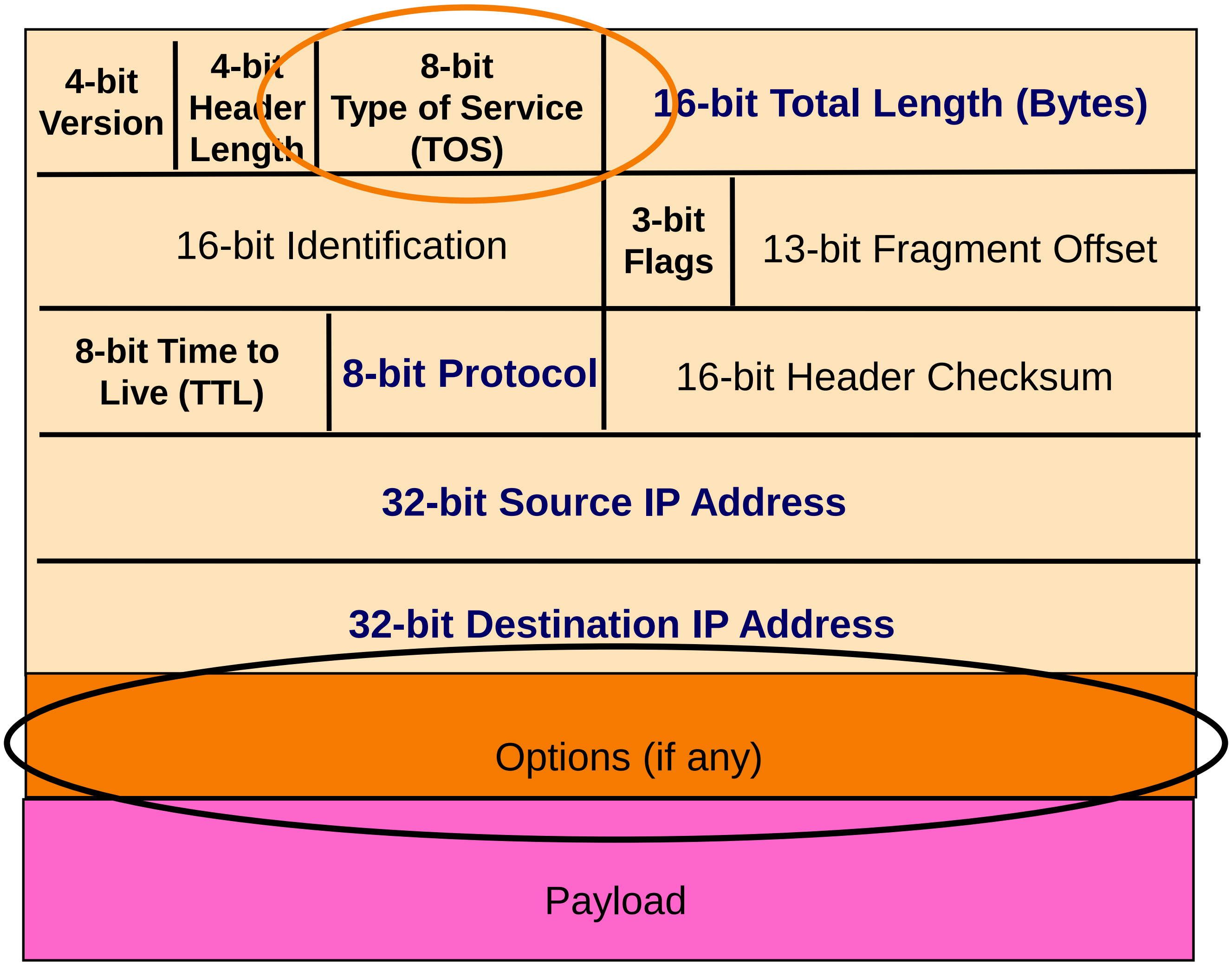


Special Handling

- Type-of-Service (8 bits)
  - Allow packets to be treated differently based on needs
  - E.g., low delay for audio, high bandwidth for bulk transfer
  - Has been redefined several times, will cover late in QoS

- Options

Fields for Special Handling



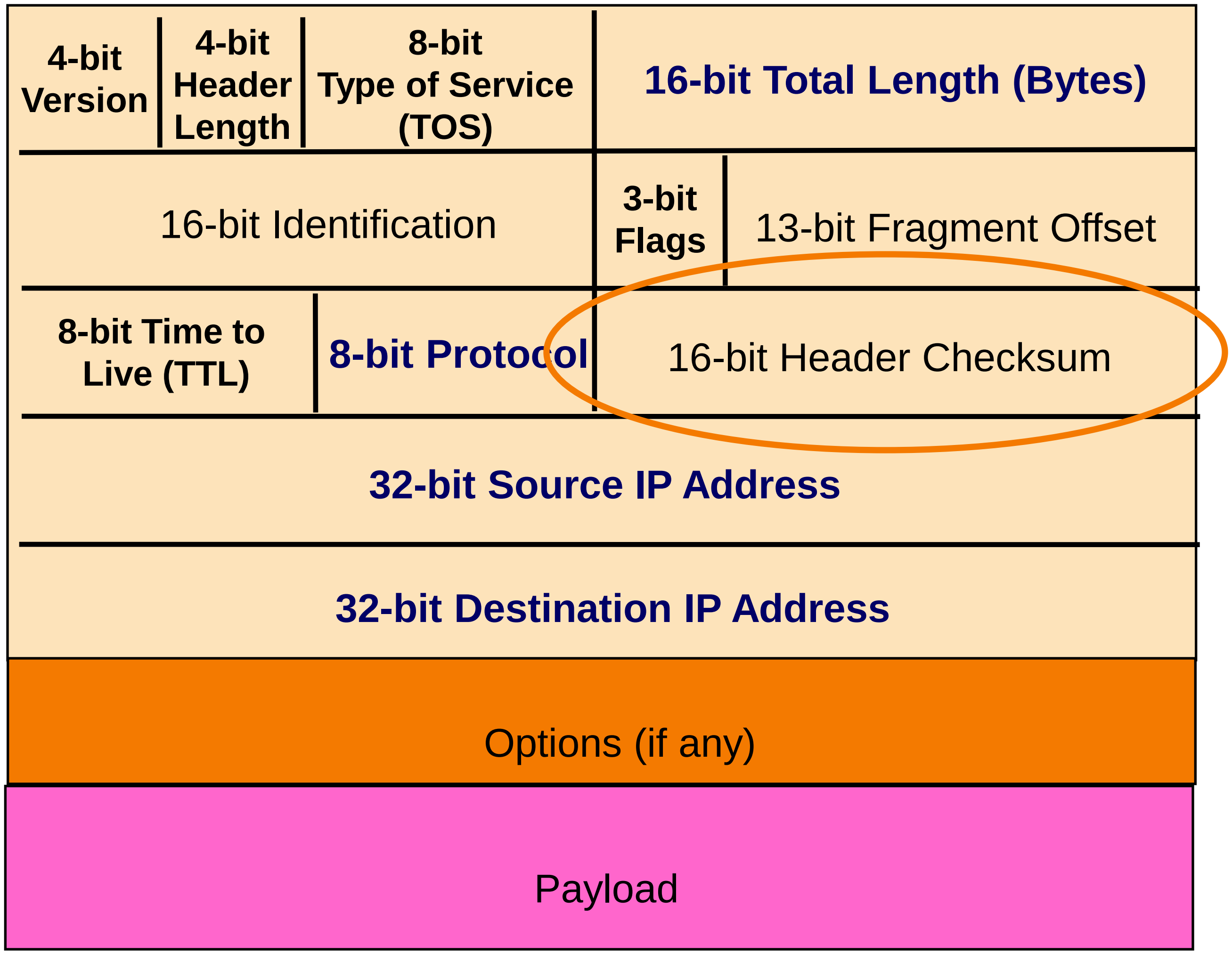
Potential Problems

- Header Corrupted: **Checksum**
- Loop: **TTL**
- Packet too large: **Fragmentation**

Header Corruption

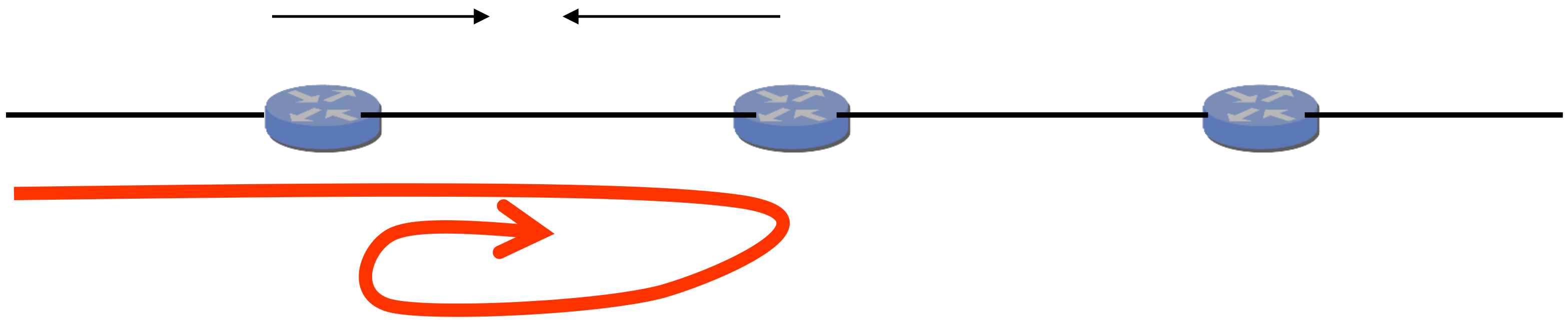
- Checksum (16 bits)
  - Particular form of checksum over packet header
- If not correct, router discards packets
  - So it doesn't act on bogus information
- Checksum recalculated at every router
  - **Why?**
  - **Why include TTL?**
  - **Why only header?**

Checksum Field



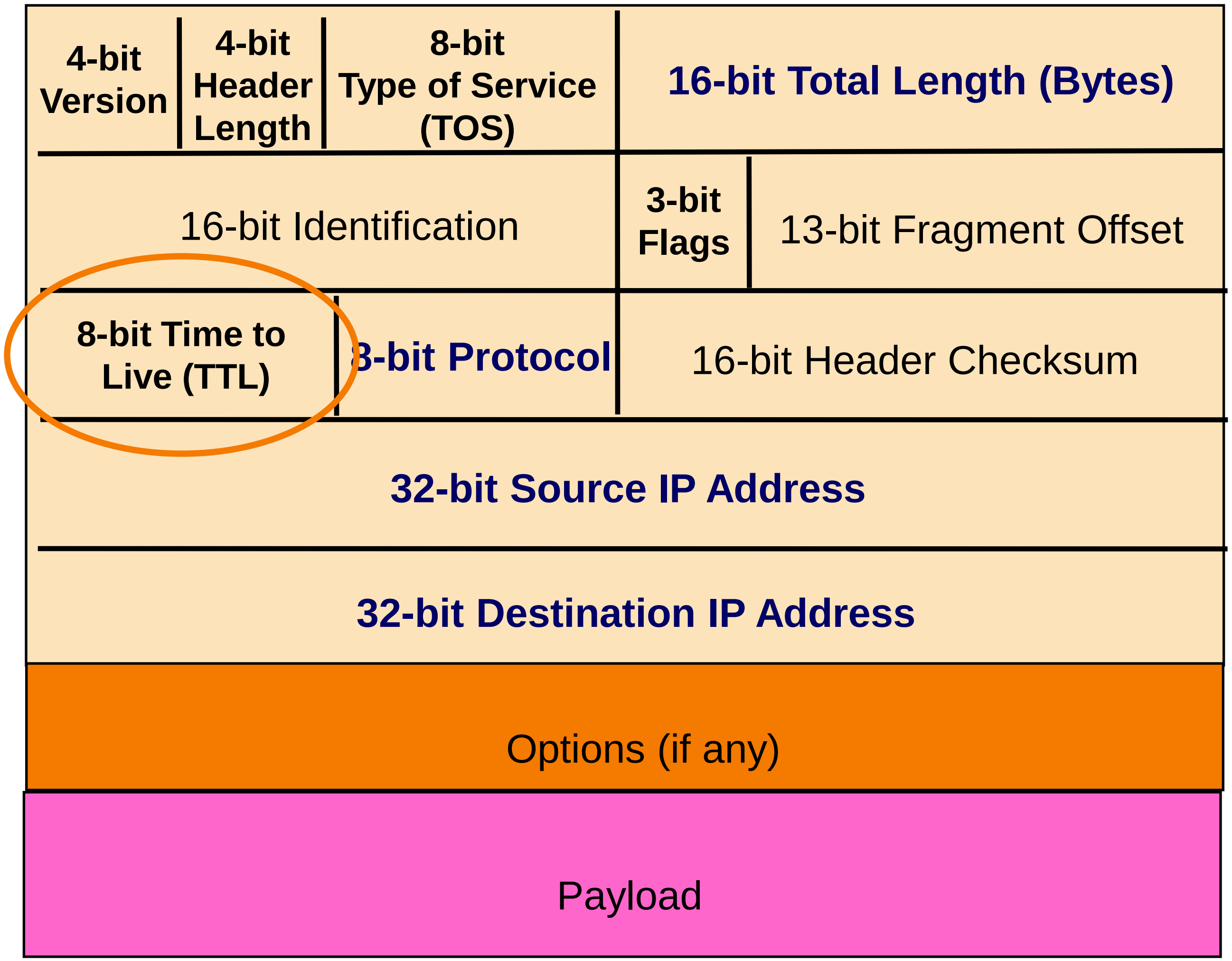
Preventing Loops

- Forwarding loops cause packets to cycle forever
  - As these accumulate, eventually 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
    - o Using "ICMP" control message; basis for **traceroute**

TTL Field



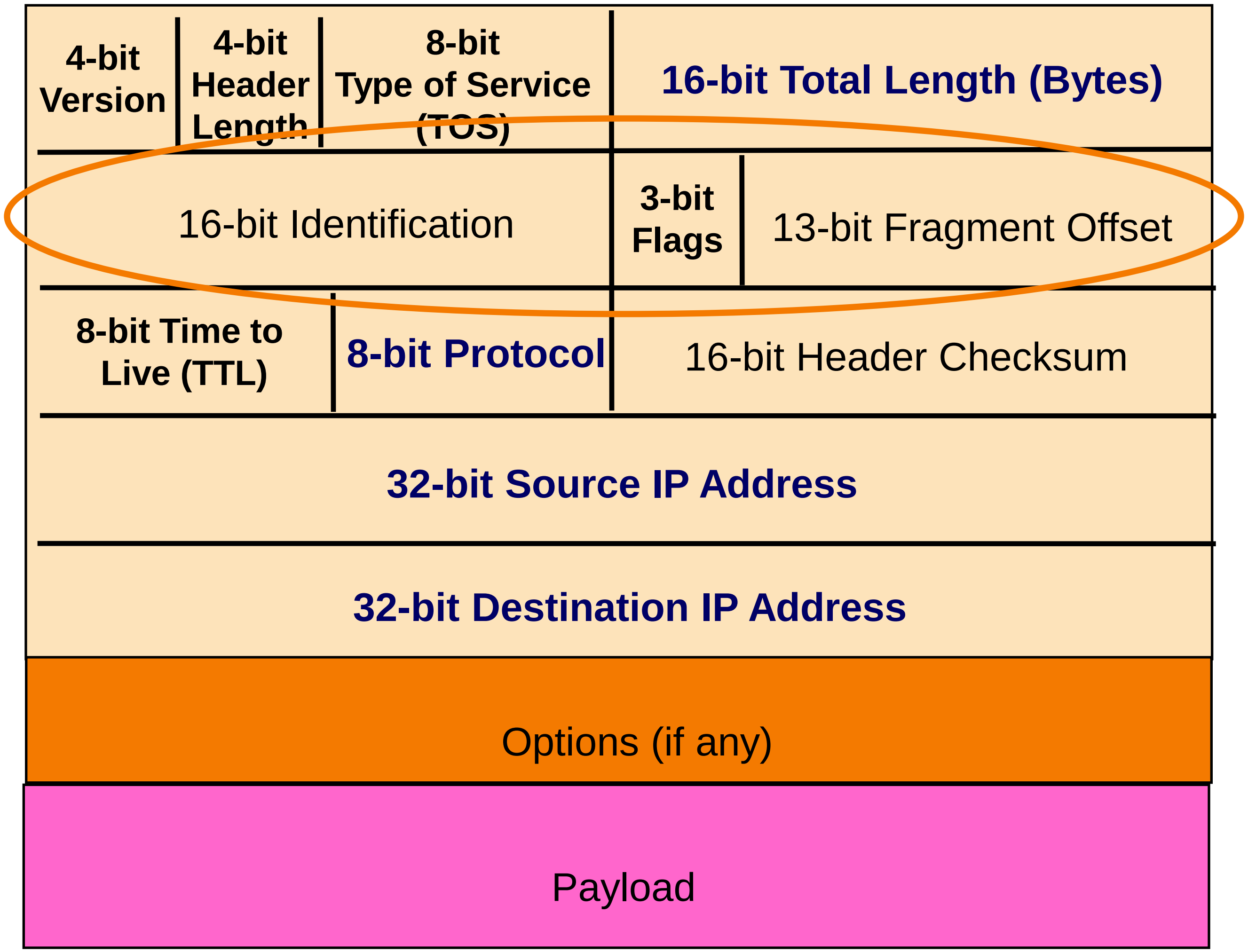
4-bit Version
4-bit Header Length
8-bit Type of Service (TOS)
16-bit Total Length (Bytes)
16-bit Identification
3-bit Flags
13-bit Fragment Offset
8-bit Time to Live (TTL)
8-bit Protocol
16-bit Header Checksum
32-bit Source IP Address
32-bit Destination IP Address
Options (if any)
Payload

Fragmentation

- Fragmentation: when forwarding a packet, an Internet router can split it into multiple pieces ("fragments") if too big for next hop link

- Must reassemble to recover original packet
  - Need fragmentation information (32 bits)
  - Packet identifier, flags, and fragment offset

IP Packet Structure



Option Field Layout

Field	Size (bits)	Description
Copied	1	Set if field copied to all fragments
Class	2	0=control, 2=debugging/measurement
Number	5	Specifies option
Length	8	Size of entire option
Data	Variable	Option-specific data

Examples of Options

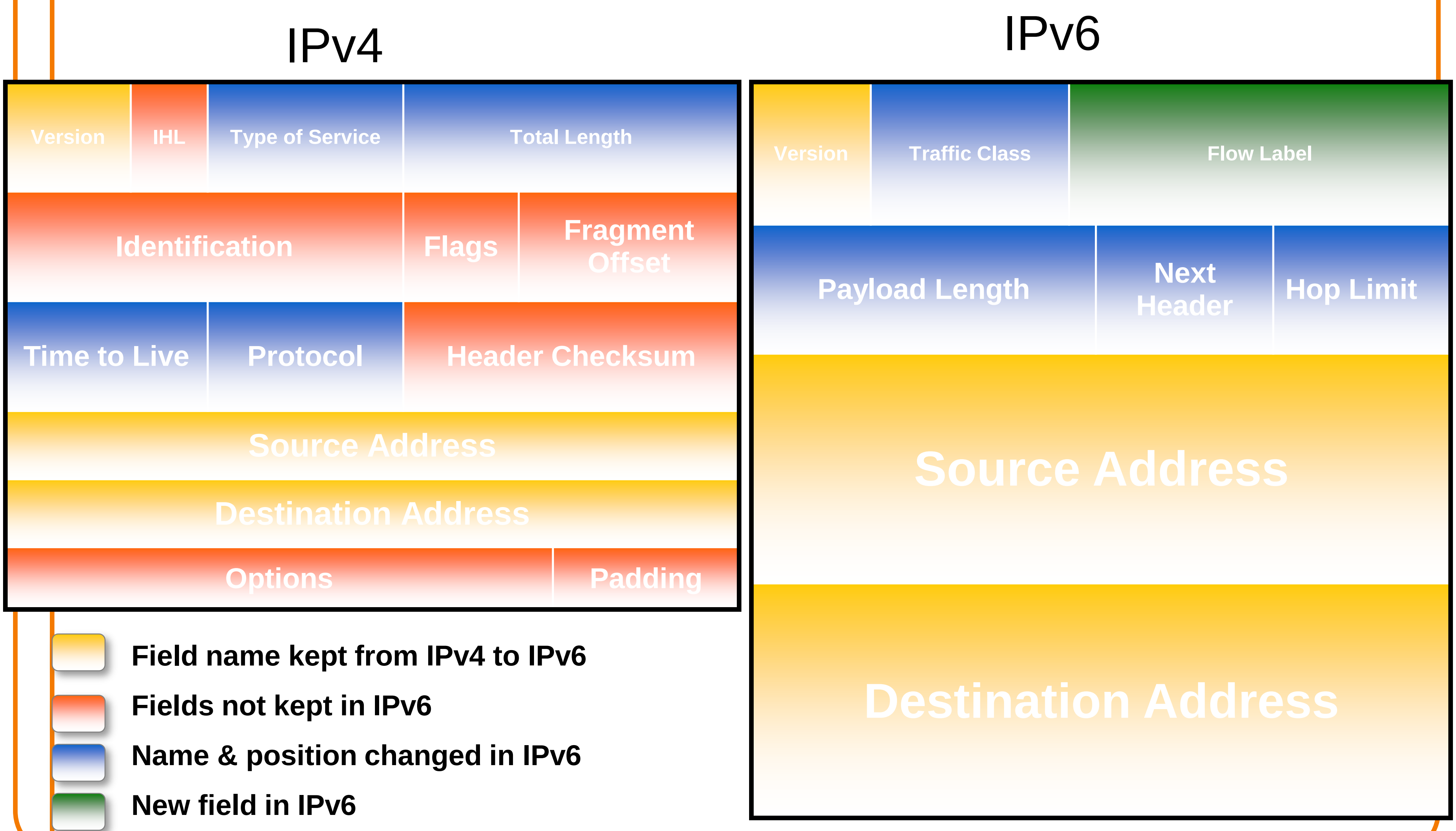
- End of Options List
- No Operation (padding between options)
- Record Route
- Strict Source Route
- Loose Source Route
- Timestamp
- Traceroute
- Router Alert
- .....

IPv6

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

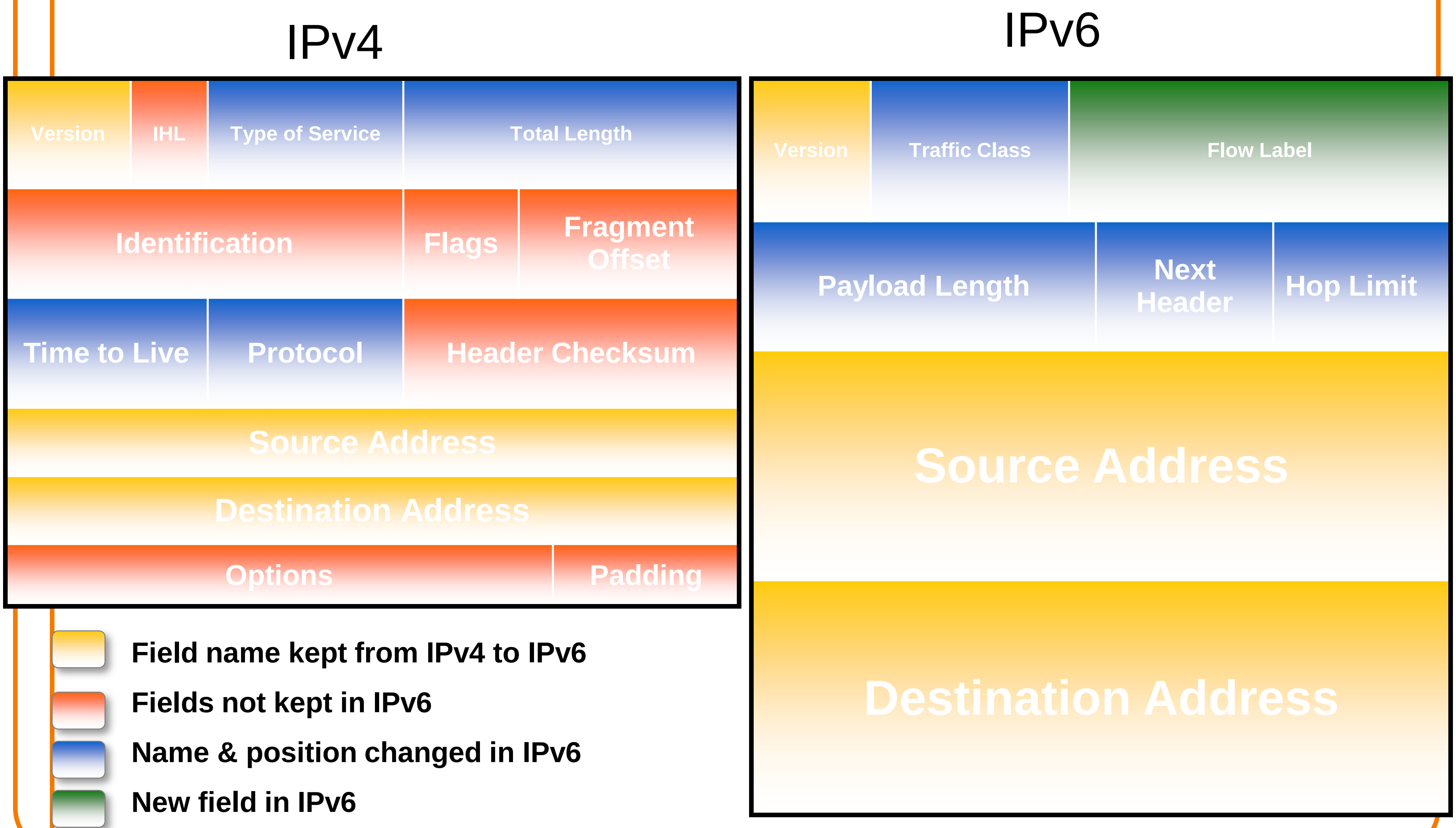
IPv4 and IPv6 Header Comparison



Summary of Changes

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

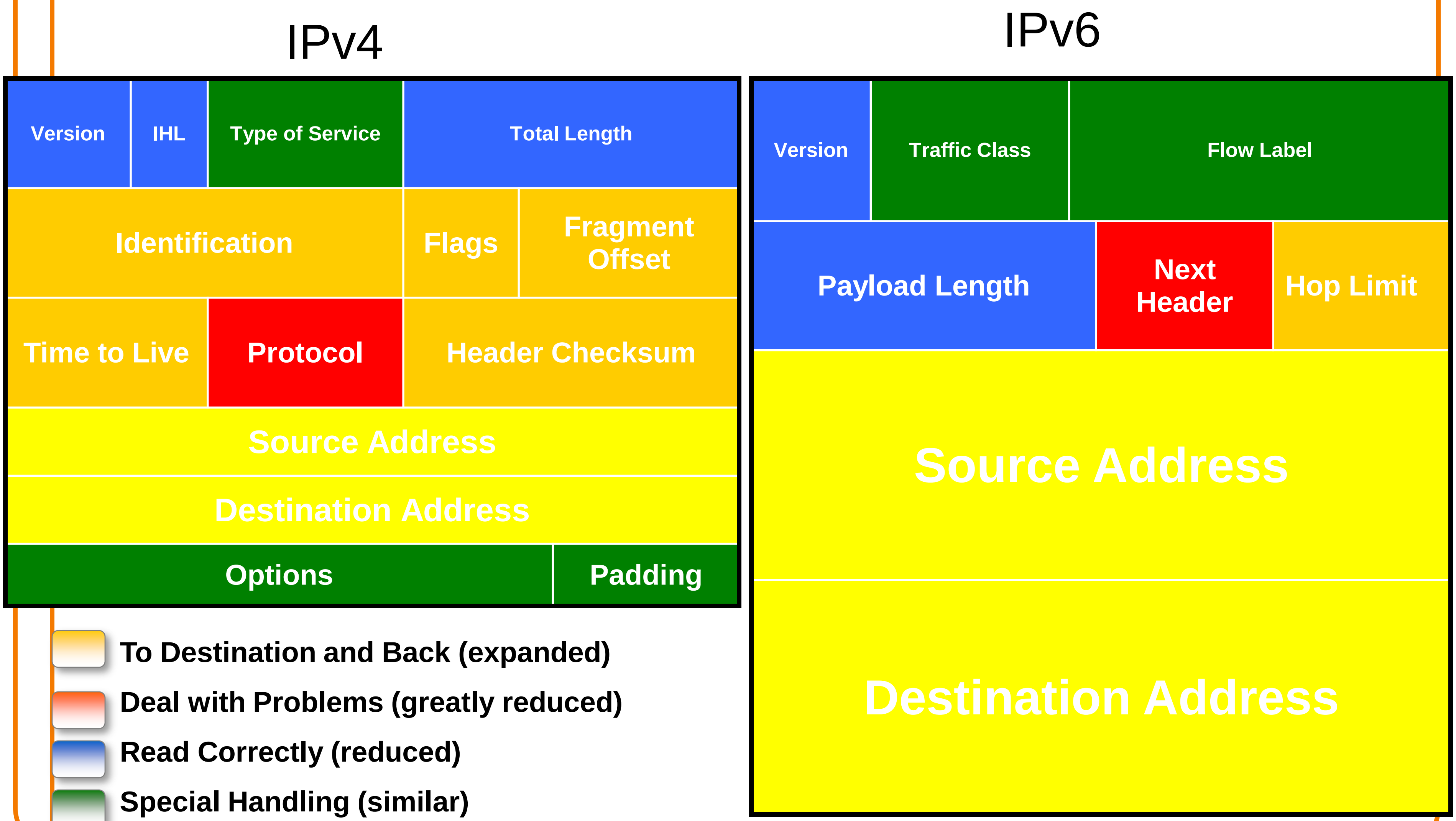
IPv4 and IPv6 Header Comparison



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

Comparison of Design Philosophy



Improving on IPv4 and IPv6?

- Why include unverifiable source address?
  - Would like accountability ***and*** anonymity (now neither)
  - Return address can be communicated at higher layer
- Why packet header used at edge same as core?
  - Edge: host tells network what service it wants
  - Core: packet tells switch how to handle it
    - One is local to host, one is global to network
- Some kind of payment/responsibility field?
  - Who is responsible for paying for packet delivery?
  - Source, destination, other?
- Other ideas?