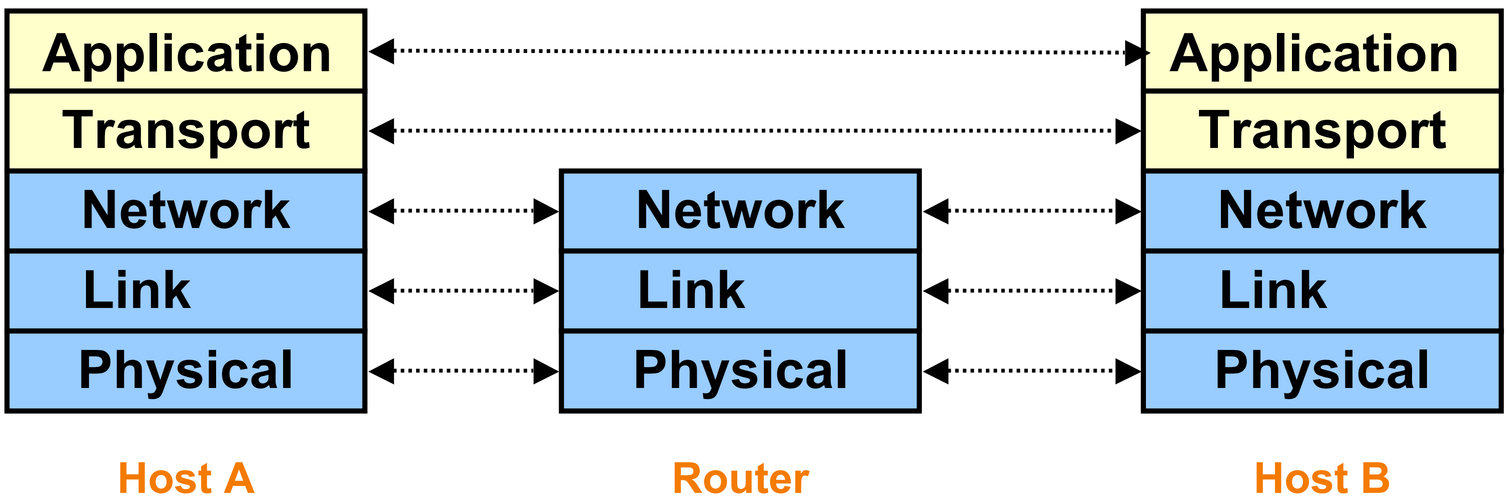
Supplementary Networking Slides

- These slides provide more detail than we covered in lecture
- We don't in general anticipate drawing upon these extra points
  – If/when we do, we'll strive to explicitly cover them in lecture
- But they may prove helpful in absorbing the networking background material

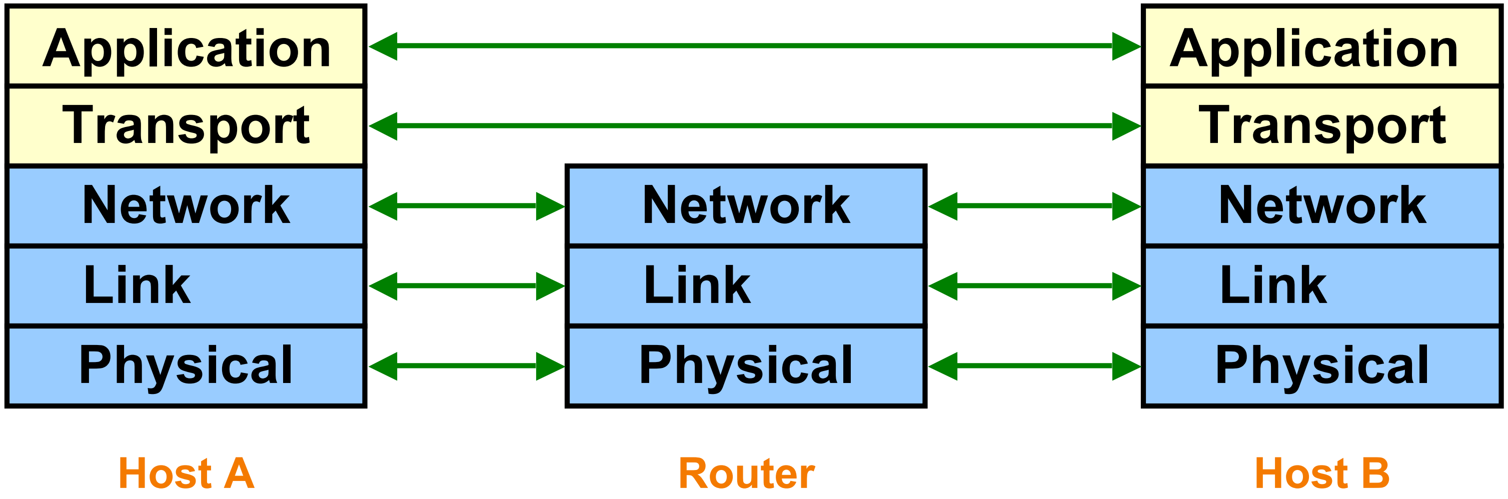
Who Does What?

- Five layers
  - Lower three layers implemented everywhere
  - Top two layers implemented only at hosts



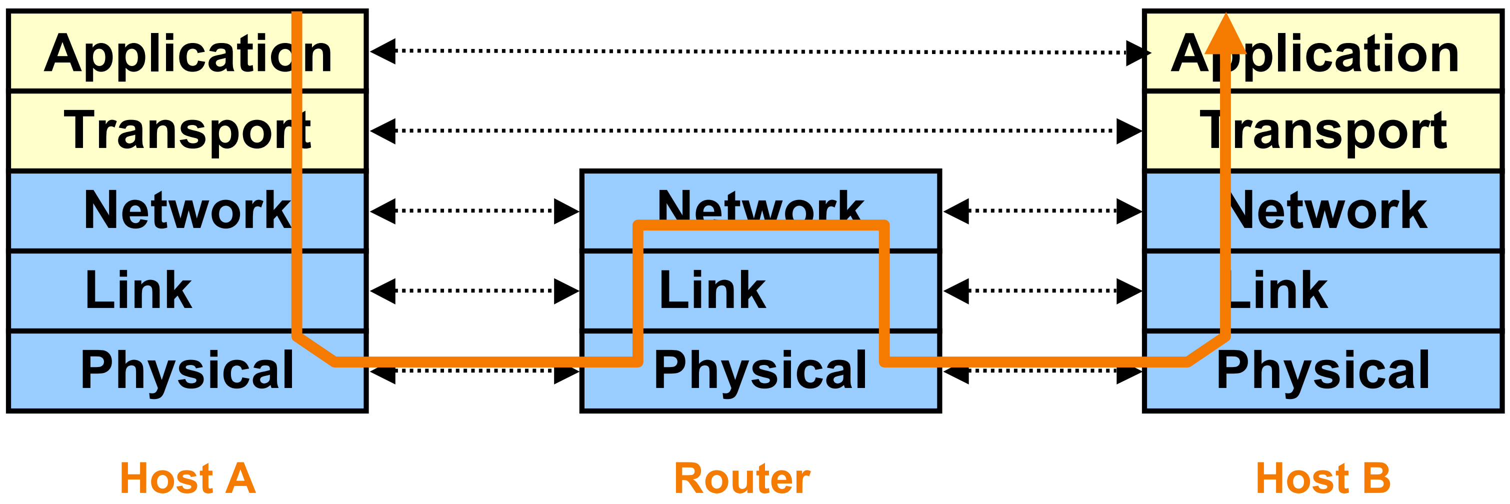
Logical Communication

- Layers interacts with peer's corresponding layer

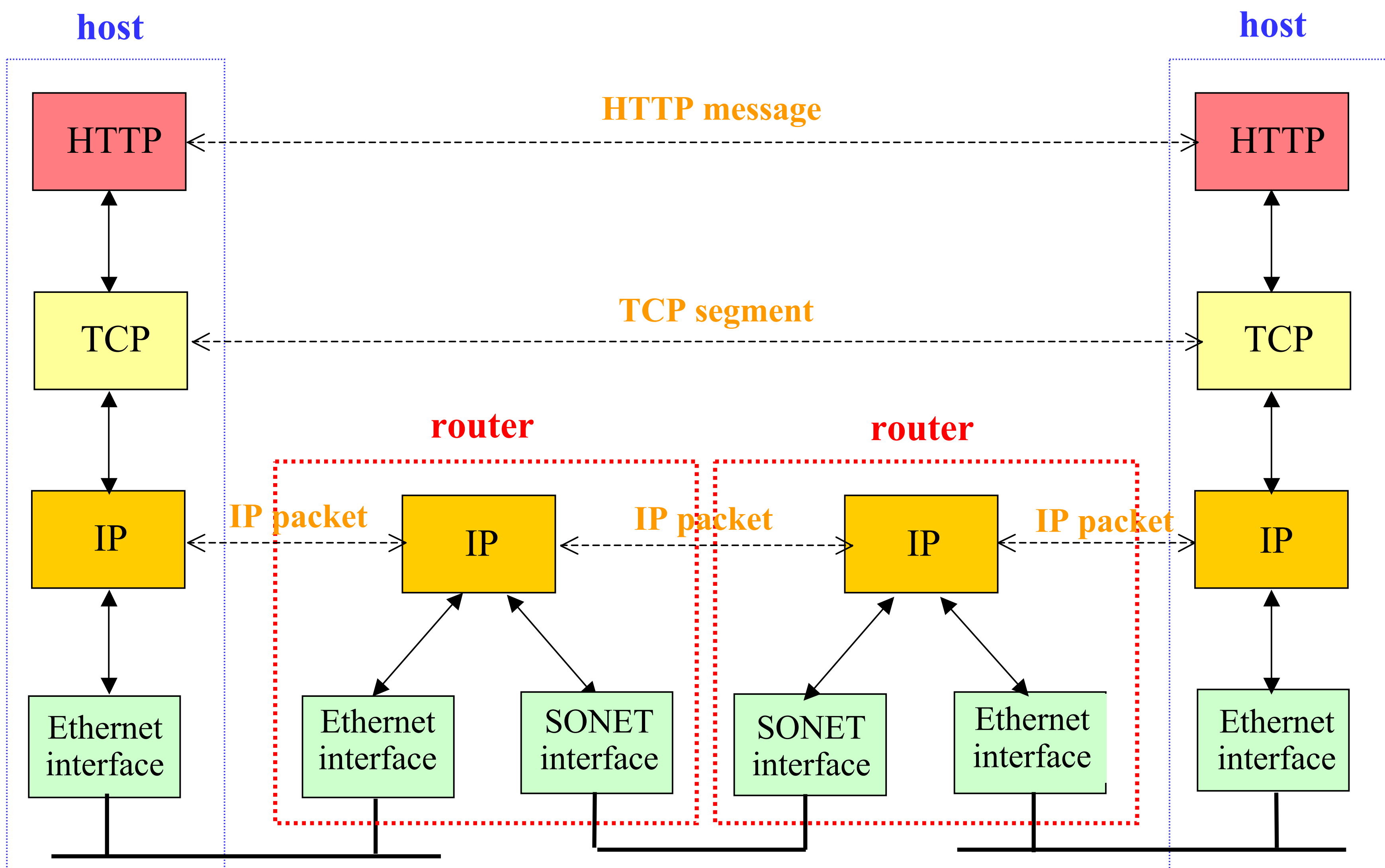


Physical Communication

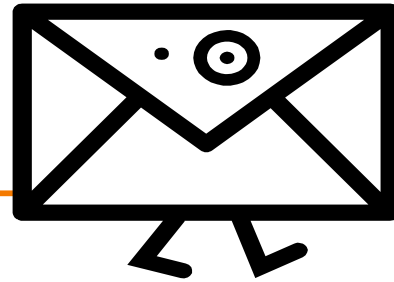
- Communication goes down to physical network
- Then from network peer to peer
- Then up to relevant layer

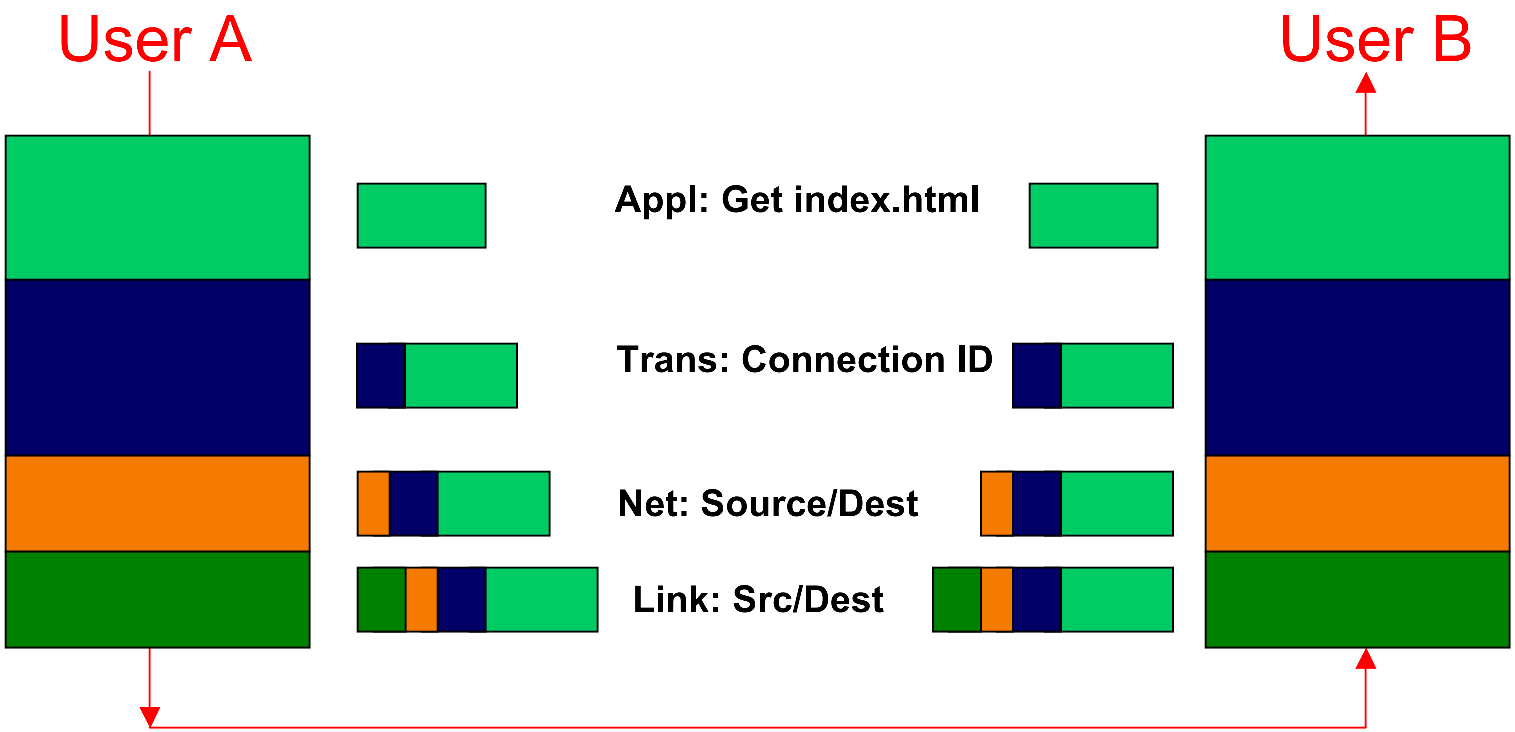


IP Suite: End Hosts vs. Routers



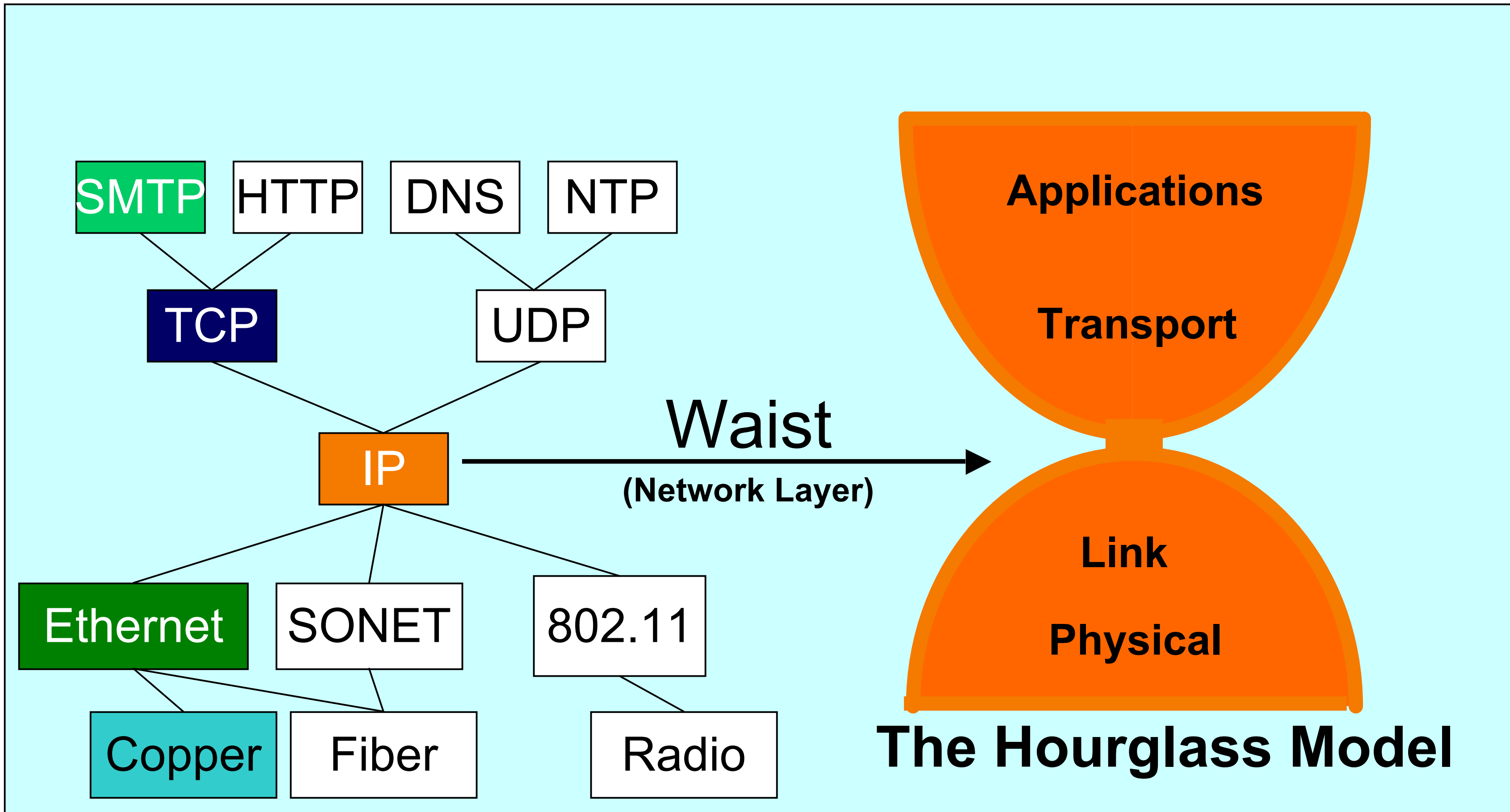
Layer Encapsulation





Common case: 20 bytes TCP header + 20 bytes IP header + 14 bytes Ethernet header = *54 bytes overhead*

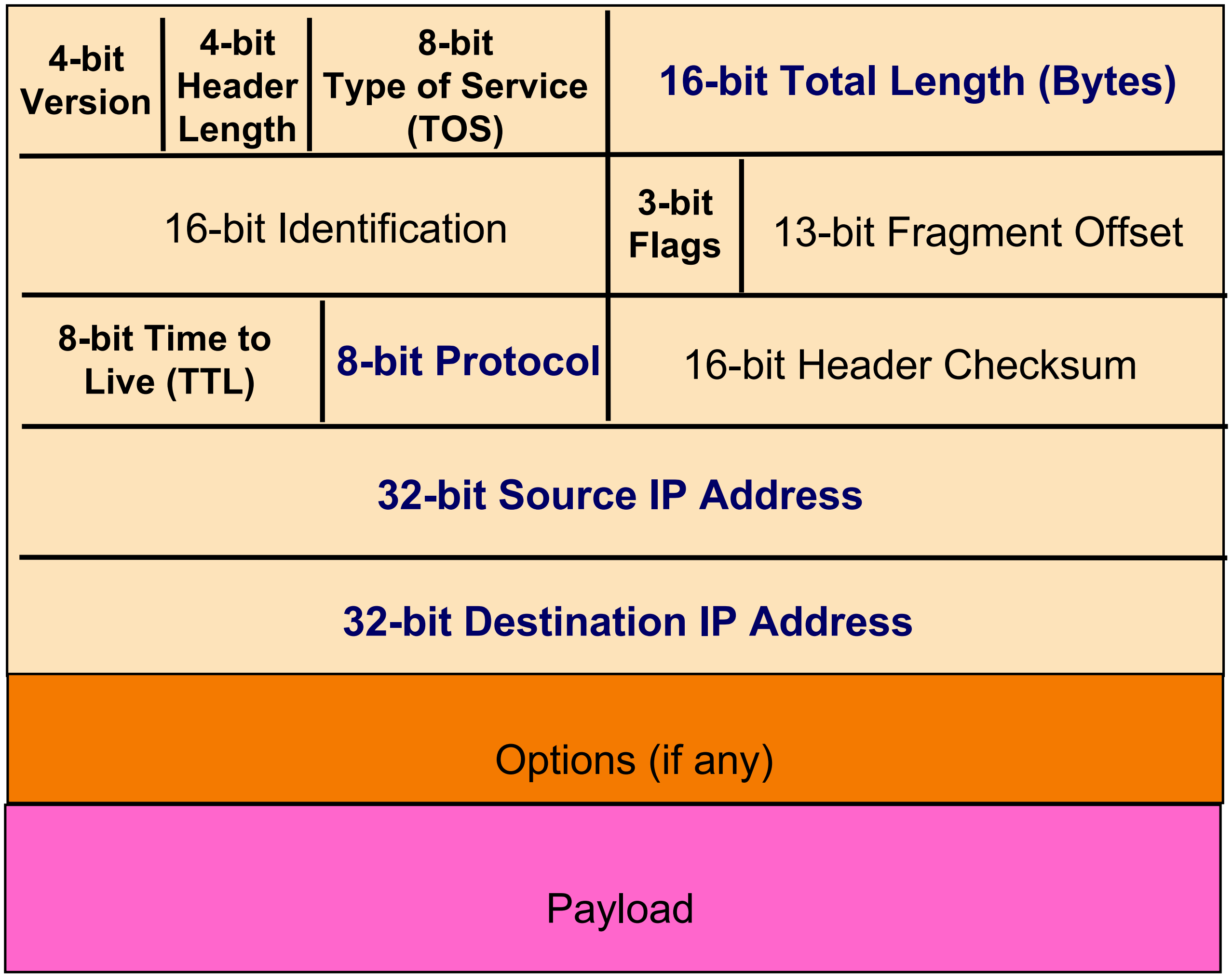
The Internet *Hourglass*



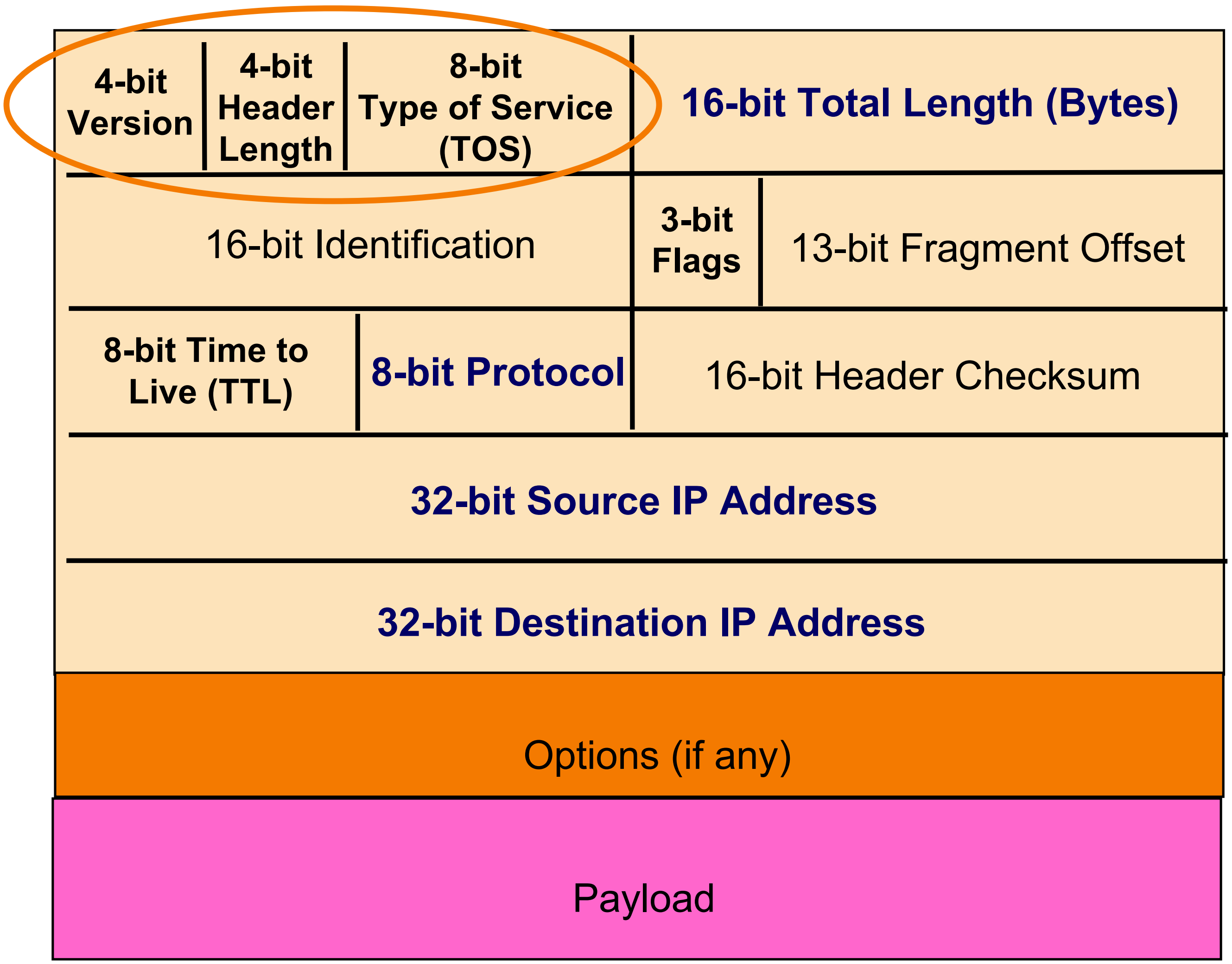
There is just one network-layer protocol, **IP**.

The "narrow waist" facilitates interoperability.

IP Packet Structure



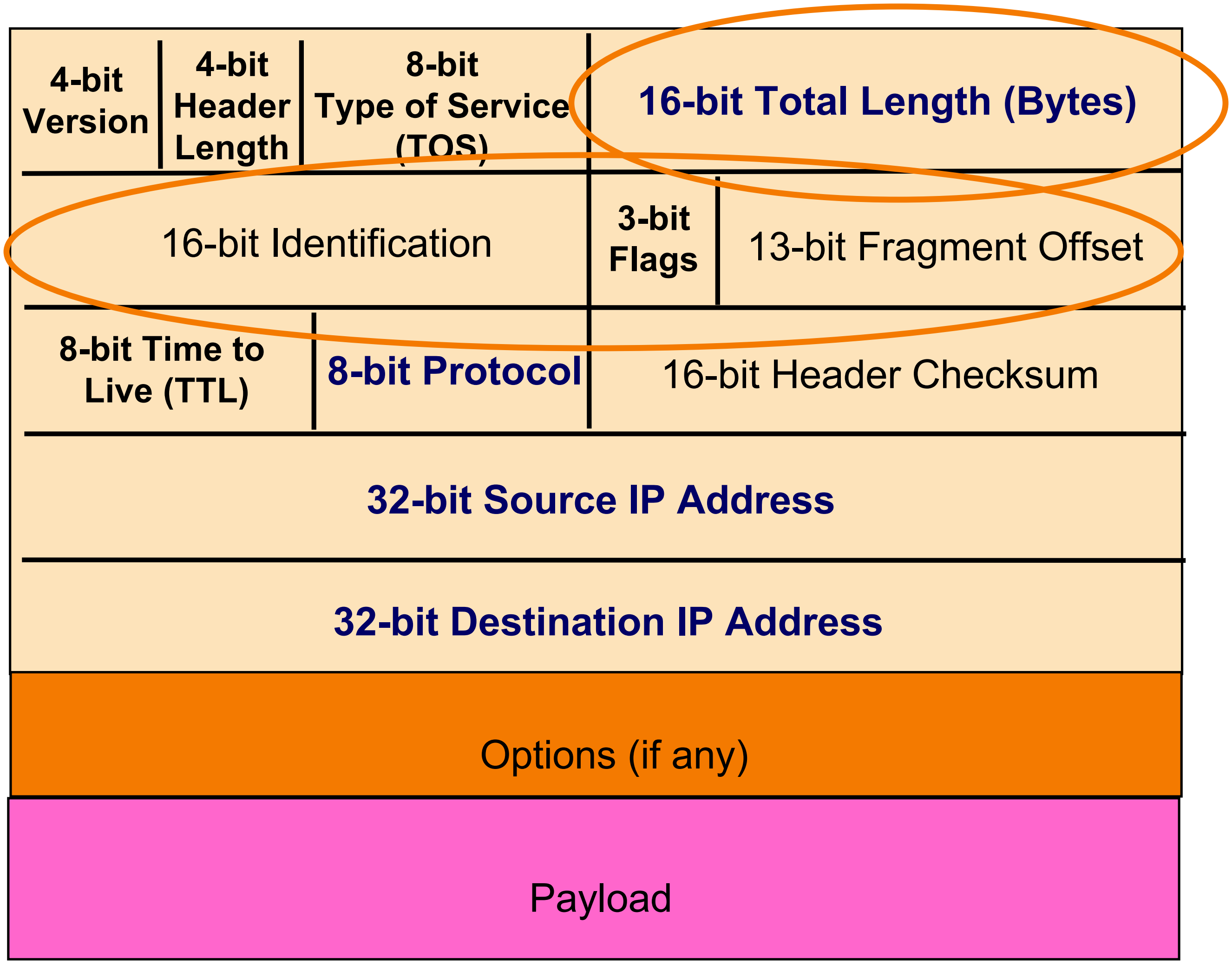
IP Packet Structure



IP Packet Header Fields

- 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
- 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

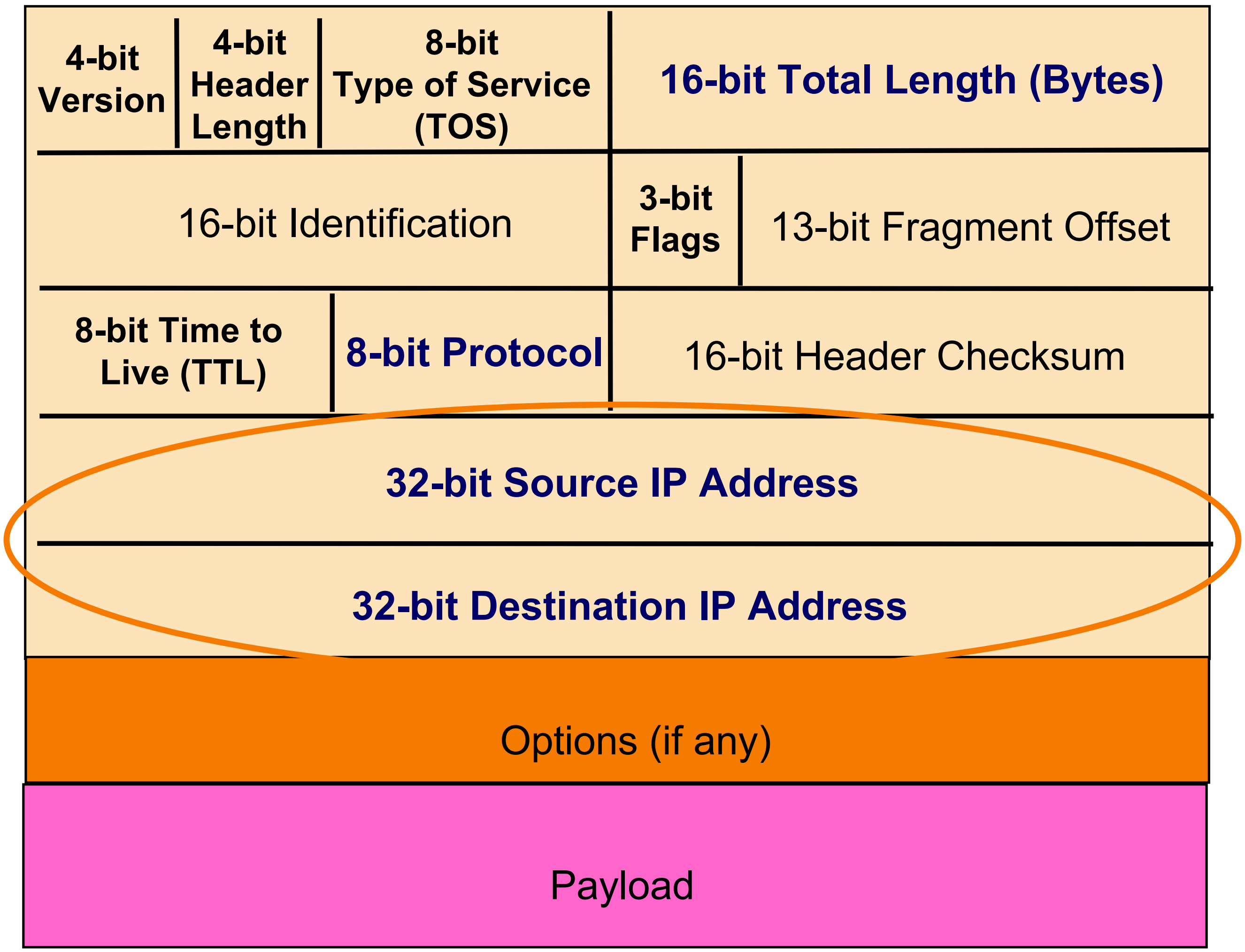
IP Packet Structure



IP Packet Header Fields (Continued)

- 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
- Fragmentation: when forwarding a packet, an Internet router can split it into multiple pieces ("fragments") if too big for next hop link
- End host reassembles to recover original packet
- Fragmentation information (32 bits)
  - Packet identifier, flags, and fragment offset
  - Supports dividing a large IP packet into fragments
  - … in case a link cannot handle a large IP packet

IP Packet Structure



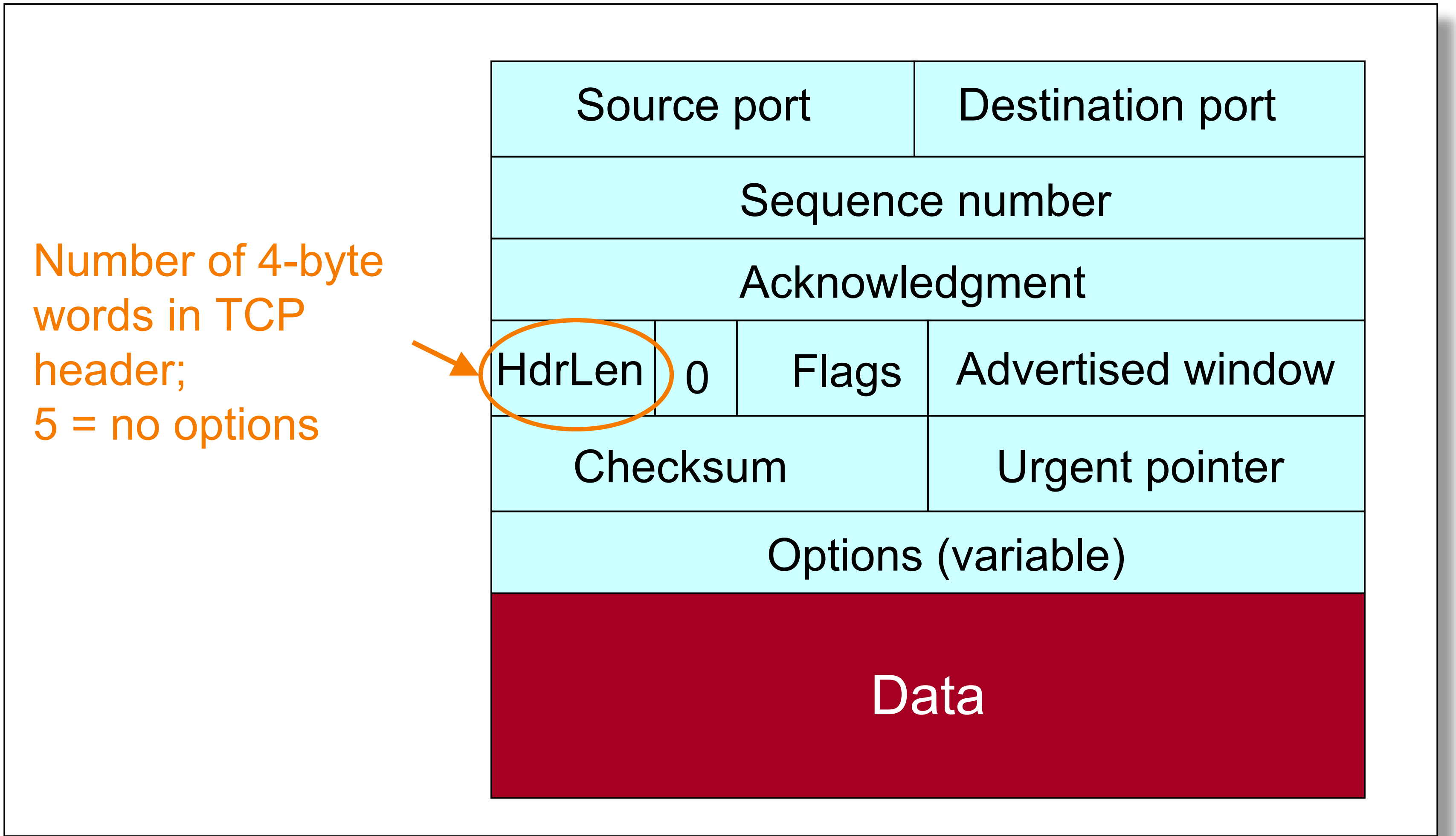
IP Packet Header (Continued)

- 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

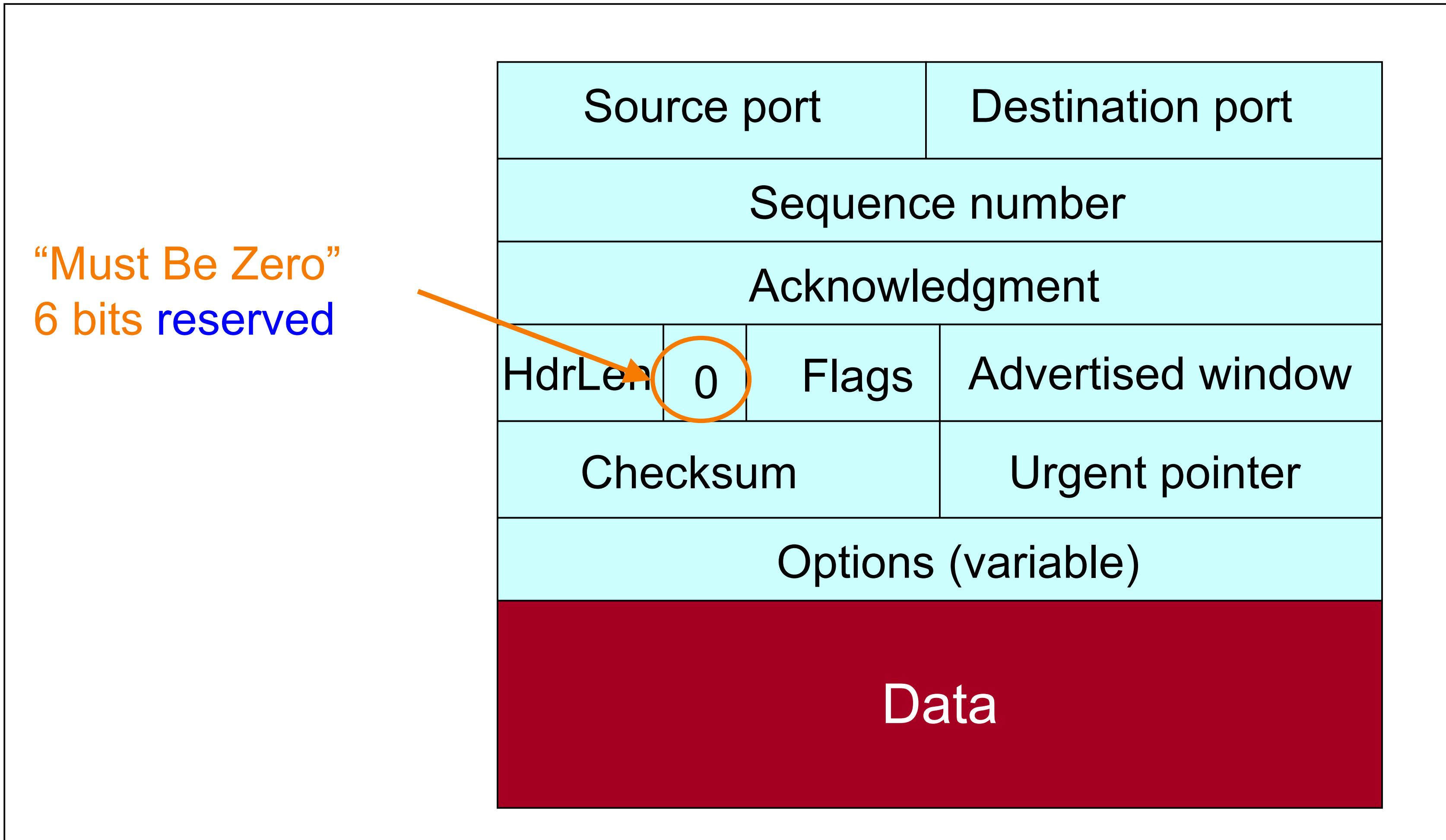
TCP Support for Reliable Delivery

- Checksum
  - Used to detect corrupted data at the receiver
  - …leading the receiver to drop the packet
- Sequence numbers
  - Used to detect missing data
  - ... and for putting the data back in order
- Retransmission
  - Sender retransmits lost or corrupted data
  - Timeout based on estimates of round-trip time
  - *Fast retransmit* algorithm for rapid retransmission

TCP Header


Number of 4-byte words in TCP header;
5 = no options
Source port
Destination port
Sequence number
Acknowledgment
HdrLen
0
Flags
Advertised window
Checksum
Urgent pointer
Options (variable)
Data

TCP Header

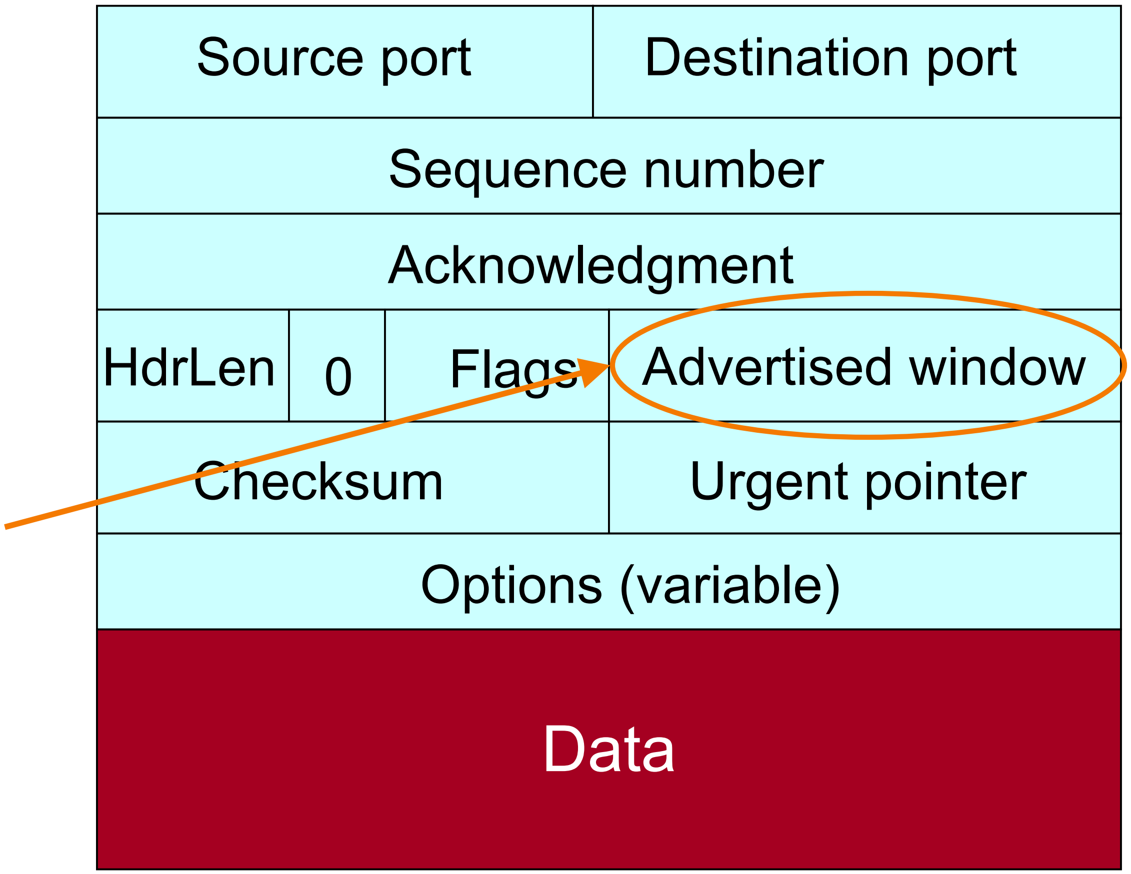


"Must Be Zero"
6 bits reserved
Source port
Destination port
Sequence number
Acknowledgment
HdrLen
0
Flags
Advertised window
Checksum
Urgent pointer
Options (variable)
Data

TCP Header

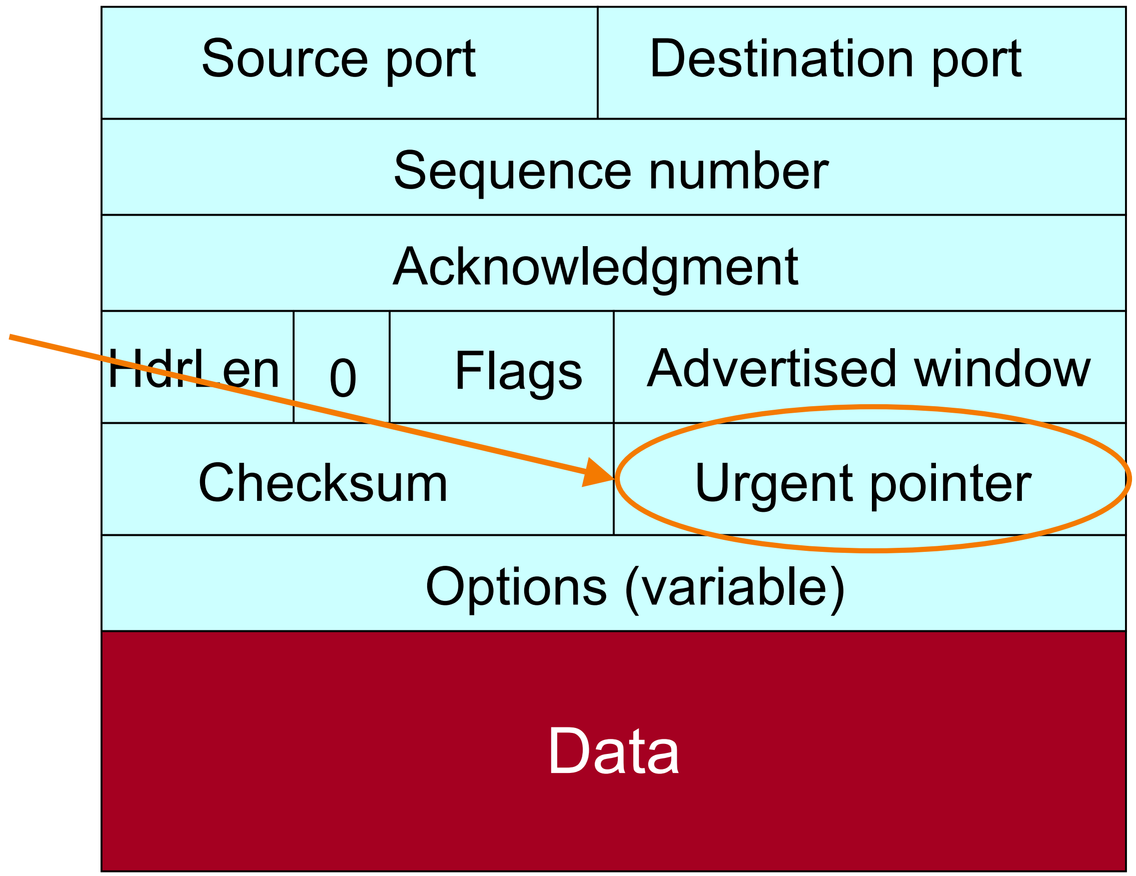
Buffer space available for receiving data. Used for TCP's sliding window.

Interpreted as offset beyond Acknowledgment field's value.

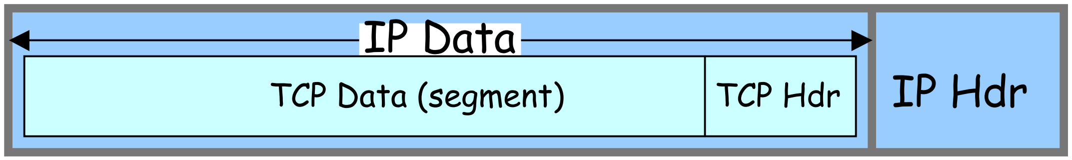


TCP Header

Used with **URG** flag to indicate urgent data (not discussed further)

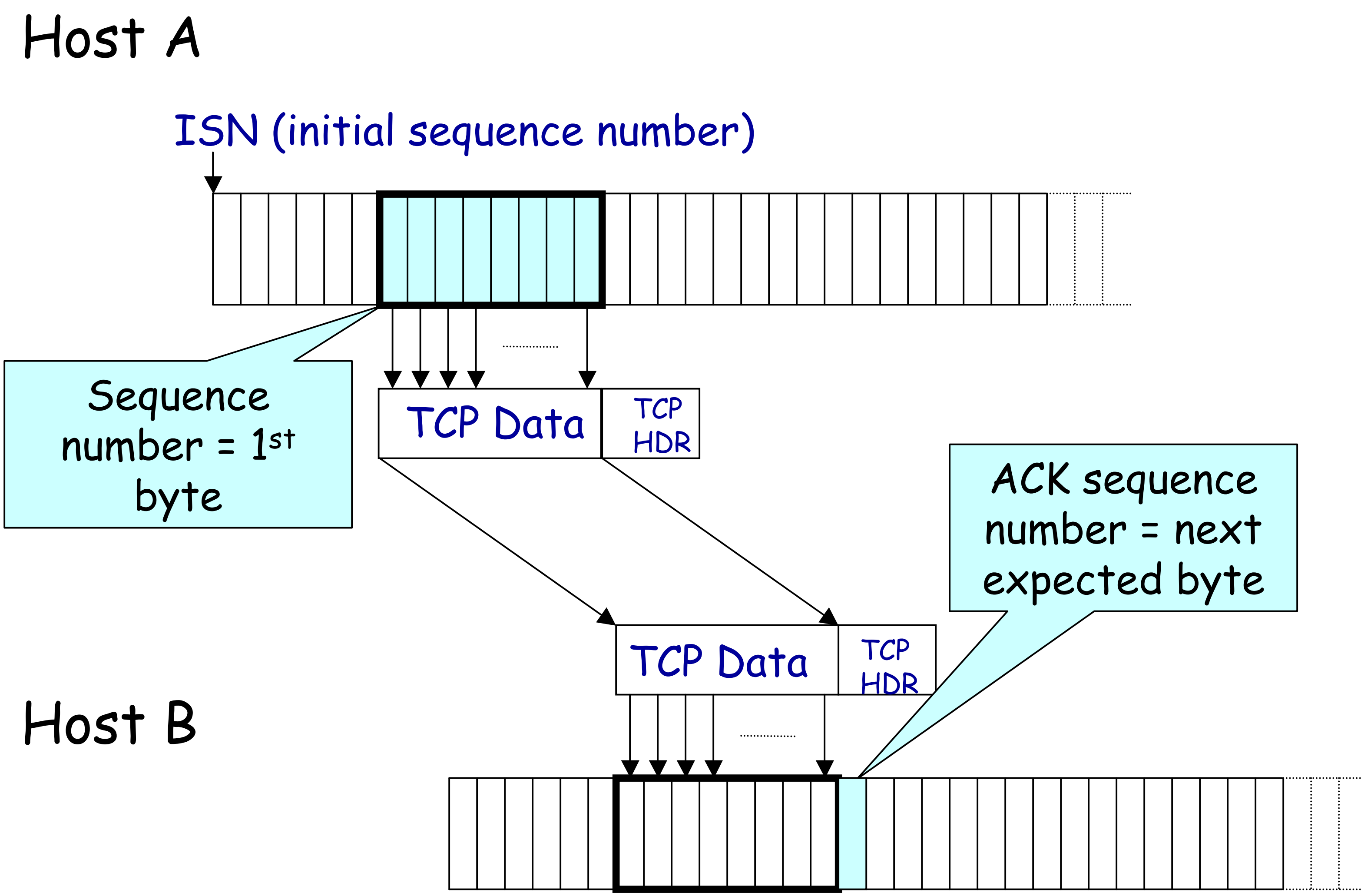


TCP Segment



- IP packet
  - No bigger than Maximum Transmission Unit (MTU)
  - E.g., up to 1,500 bytes on an Ethernet
- TCP packet
  - IP packet with a TCP header and data inside
  - TCP header ≥ 20 bytes long
- TCP **segment**
  - No more than Maximum Segment Size (MSS) bytes
  - E.g., up to 1460 consecutive bytes from the stream

Sequence Numbers

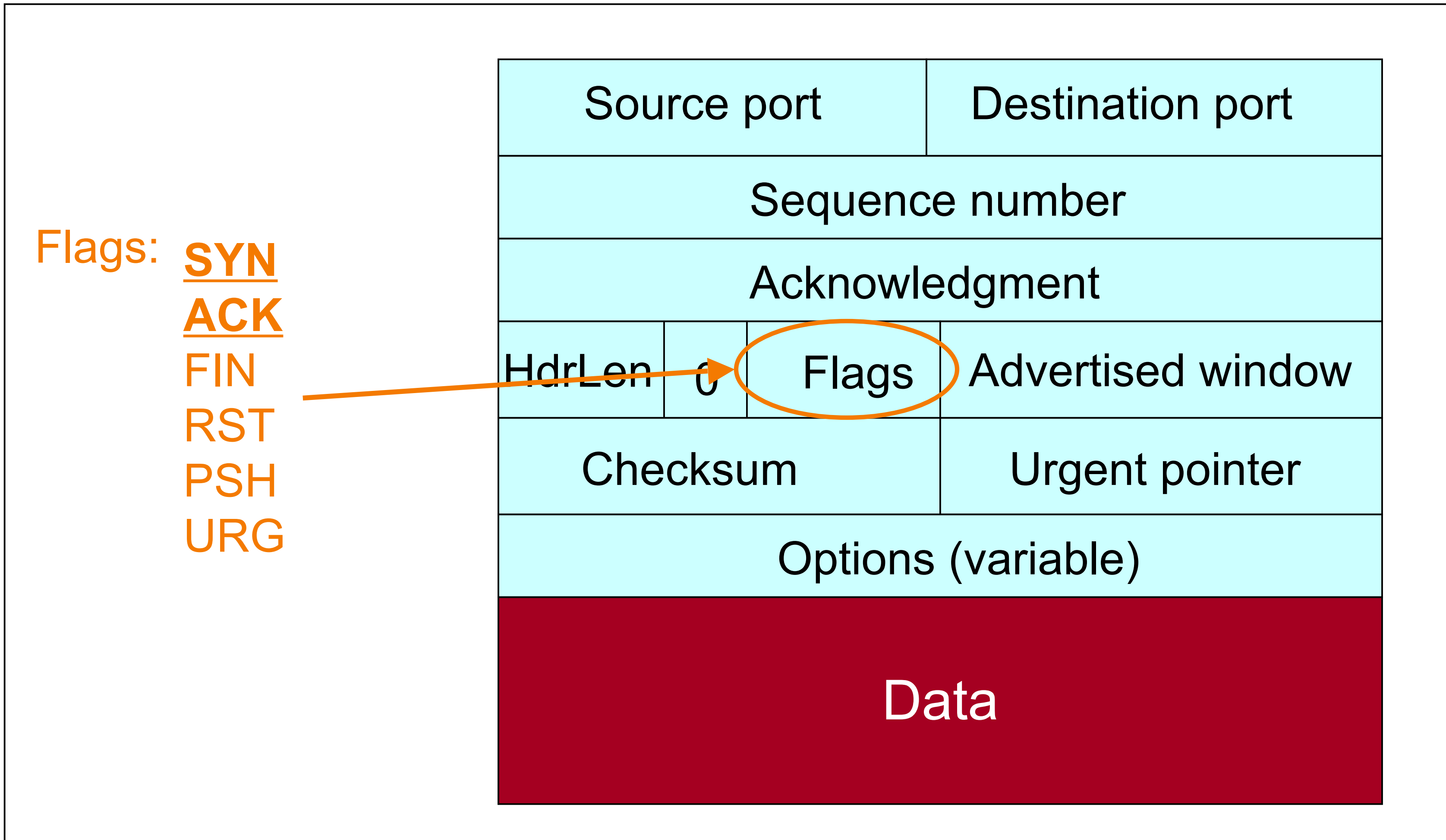


Initial Sequence Number (ISN)

- Sequence number for the very first byte
  - E.g., Why not just use ISN = 0?
- Practical issue
  - IP addresses and port #s uniquely identify a connection
  - Eventually, though, these port #s do get used again
  - … ∃ a chance an old packet is still in flight
  - … and might be associated with new connection
- ∴ TCP requires (RFC793) changing ISN over time
  - Set from 32-bit clock that ticks every 4 microseconds
  - … only wraps around once every 4.55 hours
- To establish a connection, hosts exchange ISNs

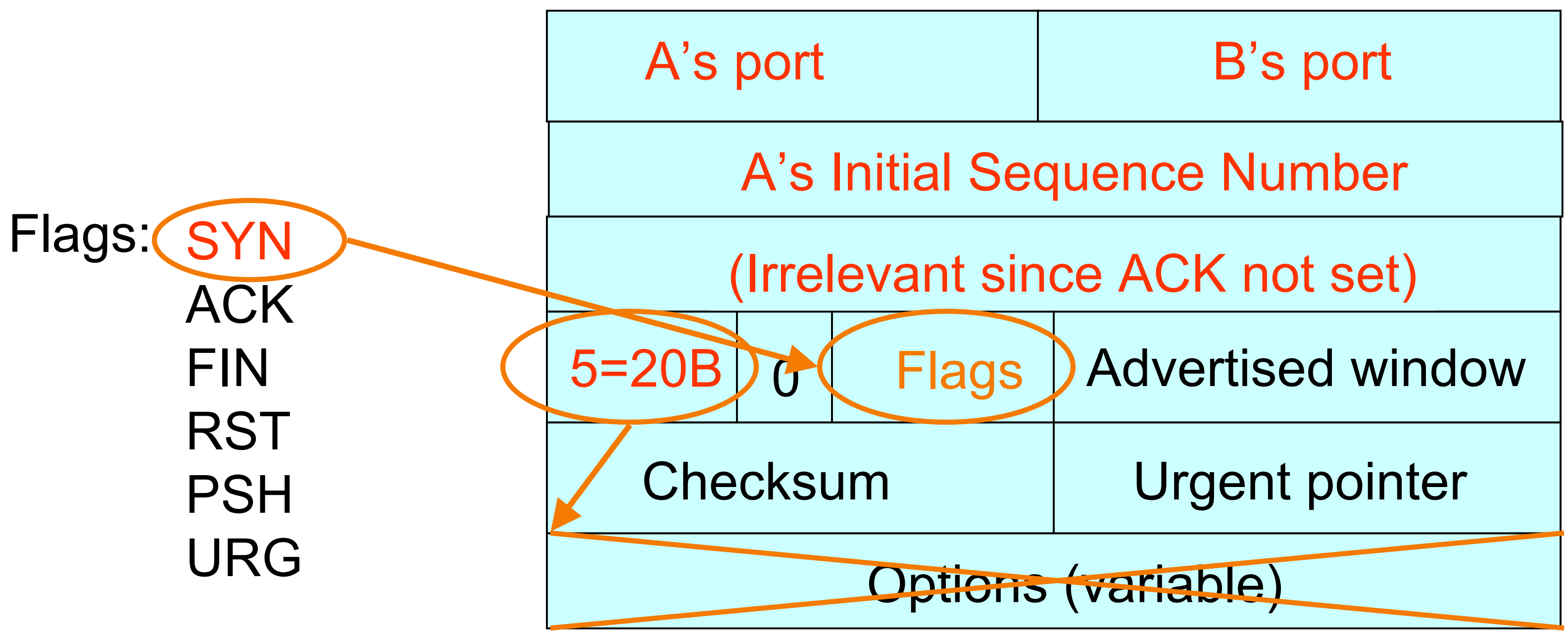
Connection Establishment: TCP's *Three-Way Handshake*

TCP Header



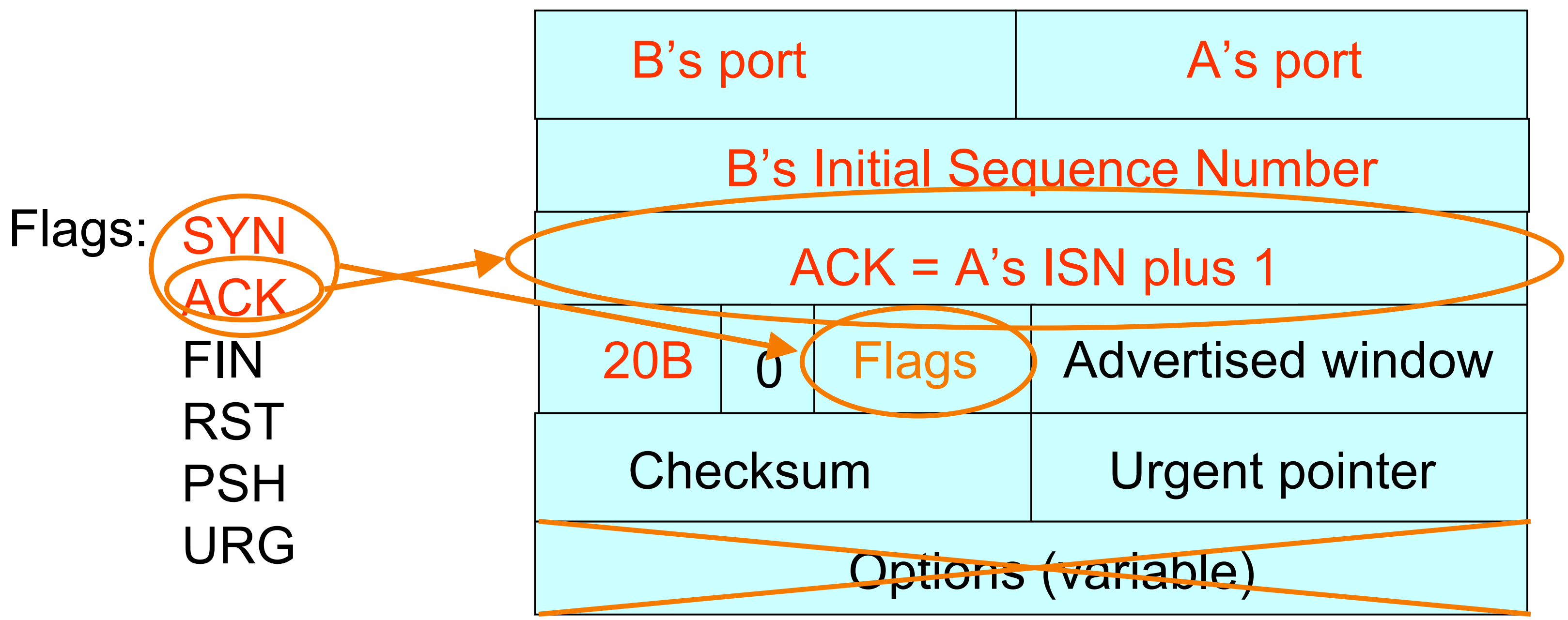
See /usr/include/netinet/tcp.h on Unix Systems

Step 1: A's Initial SYN Packet



A tells B it wants to open a connection…

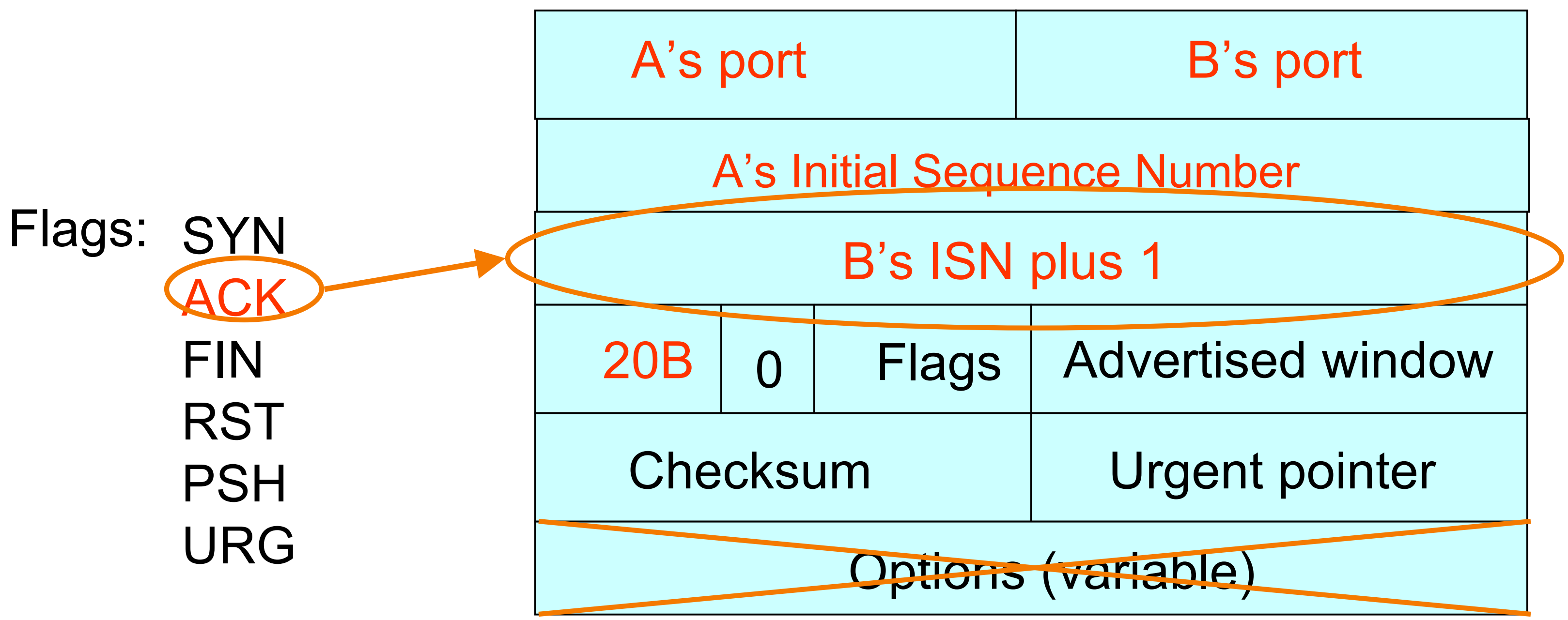
Step 2: B's SYN-ACK Packet



B tells A it accepts, and is ready to hear the next byte…

… upon receiving this packet, A can start sending data

Step 3: A's ACK of the SYN-ACK



A tells B it's likewise okay to start sending

… upon receiving this packet, B can start sending data

What if the SYN Packet Gets Lost?

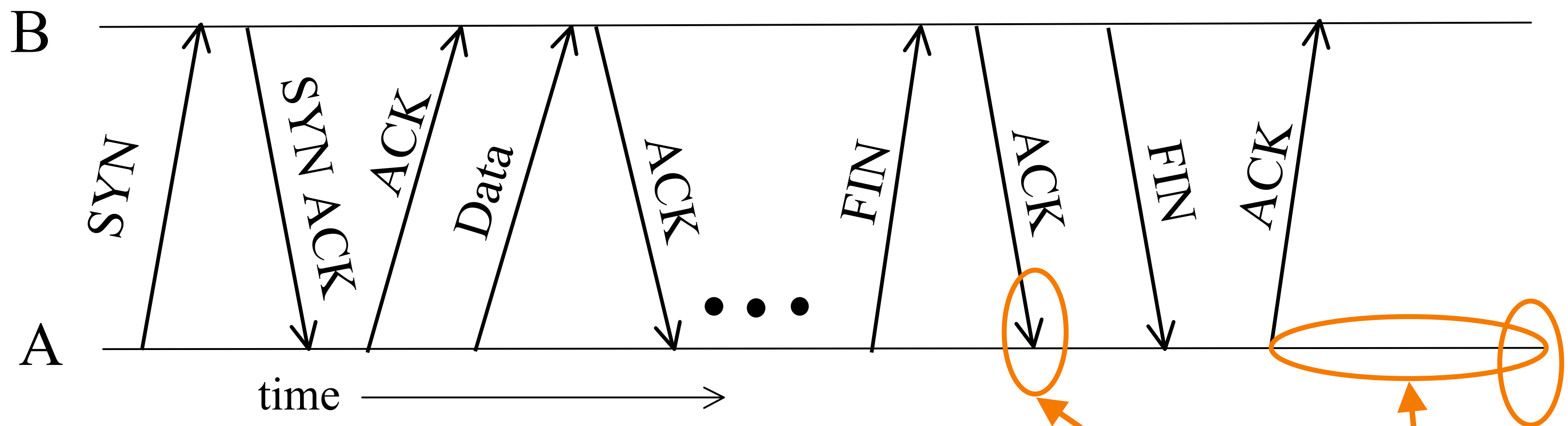
- Suppose the SYN packet gets lost
  - Packet is lost inside the network, or:
  - Server discards the packet (e.g., listen queue is full)
- Eventually, no SYN-ACK arrives
  - Sender sets a timer and waits for the SYN-ACK
  - … and retransmits the SYN if needed
- How should the TCP sender set the timer?
  - Sender has no idea how far away the receiver is
  - Hard to guess a reasonable length of time to wait
  - **SHOULD** (RFCs 1122 & 2988) use default of 3 seconds
    - Other implementations instead use 6 seconds

SYN Loss and Web Downloads

- User clicks on a hypertext link
  - Browser creates a socket and does a "connect"
  - The "connect" triggers the OS to transmit a SYN
- If the SYN is lost…
  - 3-6 seconds of delay: can be very long
  - User may become impatient
  - … and click the hyperlink again, or click "reload"
- User triggers an "abort" of the "connect"
  - Browser creates a new socket and another "connect"
  - Essentially, forces a faster send of a new SYN packet!
  - Sometimes very effective, and the page comes quickly

Tearing Down the Connection

Normal Termination, One Side At A Time



- Finish (**FIN**) to close and receive remaining bytes
  - **FIN** occupies one octet in the sequence space
- Other host ack's the octet to confirm
- Closes A's side of the connection, but not B's
  - Until B likewise sends a **FIN**
  - Which A then acks

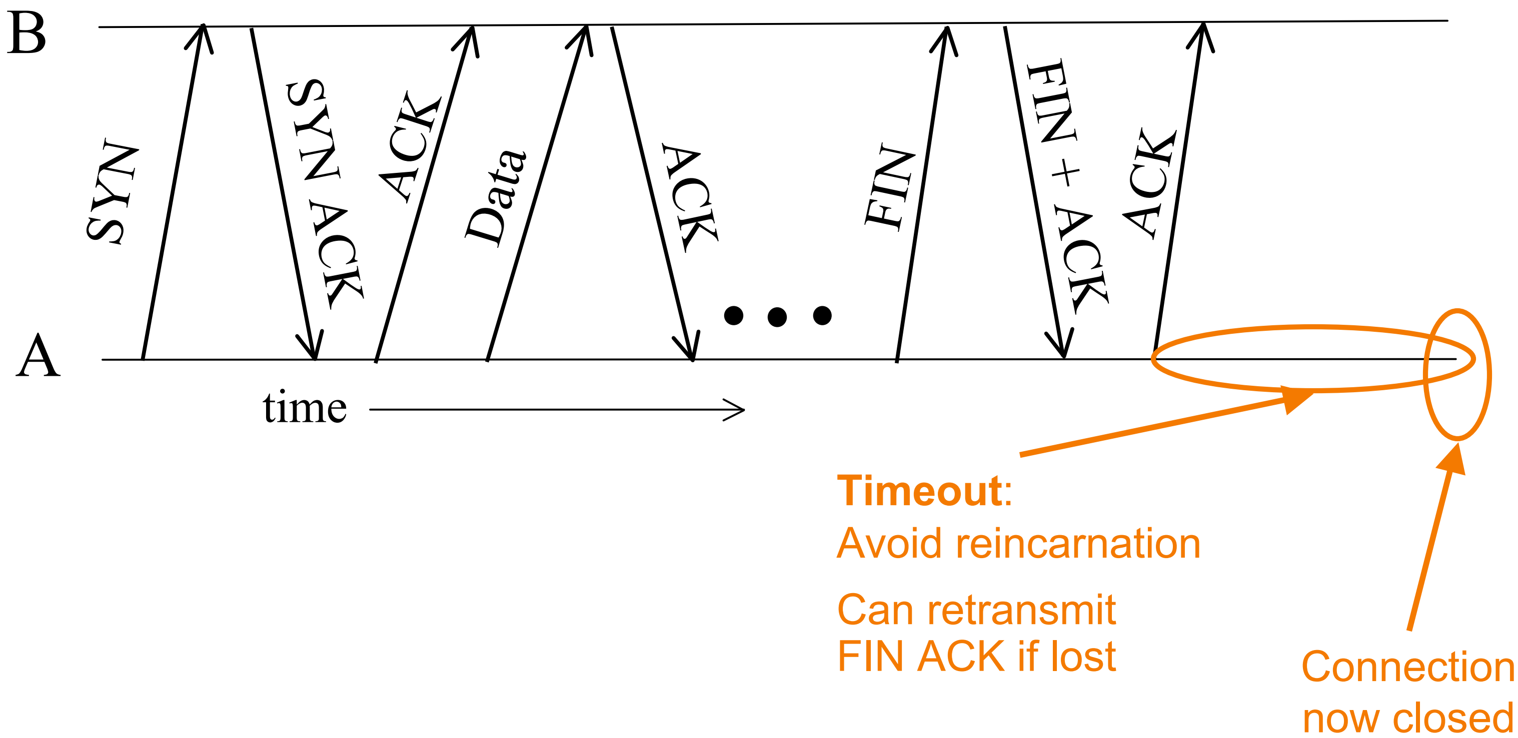
Connection now **closed**

Connection now **half-closed**

Timeout:
Avoid *reincarnation*

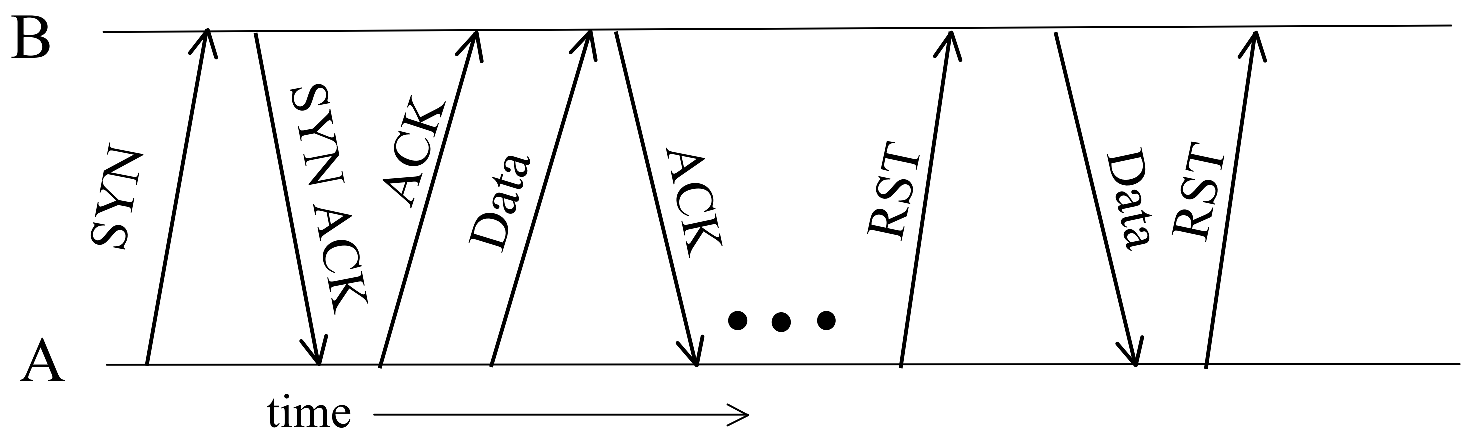
Can retransmit FIN ACK if lost

Normal Termination, Both Together



- Same as before, but B sets **FIN** with their ack of A's **FIN**

Abrupt Termination



- A sends a RESET (**RST**) to B
  - E.g., because app. process on A crashed
- That's it
  - B does not ack the **RST**
  - Thus, **RST** is not delivered reliably
  - And: any data in flight is lost
  - But: if B sends anything more, will elicit another **RST**

Layer 7 Example: E-Mail Message Using MIME

MIME version

method used to encode data

type and subtype

encoded data

```
From: jrex@cs.princeton.edu
To: feamster@cc.gatech.edu
Subject: picture of my cat
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

Base64 encoded data ….
JVBERi0xLjMNJeLjz9MNMSAwI
.........................
......base64 encoded data
```

Example With Received Header

Return-Path: <casado@cs.stanford.edu>
Received: from ribavirin.CS.Princeton.EDU (ribavirin.CS.Princeton.EDU [128.112.136.44])
by newark.CS.Princeton.EDU (8.12.11/8.12.11) with SMTP id k04M5R7Y023164
for <jrex@newark.CS.Princeton.EDU>; Wed, 4 Jan 2006 17:05:37 -0500 (EST)
Received: from bluebox.CS.Princeton.EDU ([128.112.136.38])
by ribavirin.CS.Princeton.EDU (SMSSMTP 4.1.0.19) with SMTP id M2006010417053607946
for <jrex@newark.CS.Princeton.EDU>; Wed, 04 Jan 2006 17:05:36 -0500
Received: from smtp-roam.Stanford.EDU (smtp-roam.Stanford.EDU [171.64.10.152])
by bluebox.CS.Princeton.EDU (8.12.11/8.12.11) with ESMTP id k04M5XNQ005204
for <jrex@cs.princeton.edu>; Wed, 4 Jan 2006 17:05:35 -0500 (EST)
Received: from [192.168.1.101] (adsl-69-107-78-147.dsl.pltn13.pacbell.net [69.107.78.147])
(authenticated bits=0)
by smtp-roam.Stanford.EDU (8.12.11/8.12.11) with ESMTP id k04M5W92018875
(version=TLSv1/SSLv3 cipher=DHE-RSA-AES256-SHA bits=256 verify=NOT);
Wed, 4 Jan 2006 14:05:32 -0800
Message-ID: <43BC46AF.3030306@cs.stanford.edu>
Date: Wed, 04 Jan 2006 14:05:35 -0800
From: Martin Casado <casado@cs.stanford.edu>
User-Agent: Mozilla Thunderbird 1.0 (Windows/20041206)
MIME-Version: 1.0
To: jrex@CS.Princeton.EDU
CC: Martin Casado <casado@cs.stanford.edu>
Subject: Using VNS in Class
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: 7bit

Layer 7 Example: SMTP interaction

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: From: alice@crepes.fr
C: To: hamburger-list@burger-king.com
C: Subject: Do you like ketchup?
C:
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

Message header

Message body

Lone period marks end of message

MAC Address vs. IP Address

- MAC addresses
  - Hard-coded in read-only memory when adaptor is built
  - Like a social security number
  - Flat name space of 48 bits (e.g., 00-0E-9B-6E-49-76)
  - Portable, and can stay the same as the host moves
  - Used to get packet between interfaces on same network
- IP addresses
  - Configured, or learned dynamically
  - Like a postal mailing address
  - Hierarchical name space of 32 bits (e.g., 12.178.66.9)
  - Not portable, and depends on where the host is attached
  - Used to get a packet to destination IP subnet