Lecture 17: Networking Background

https://cs161.org

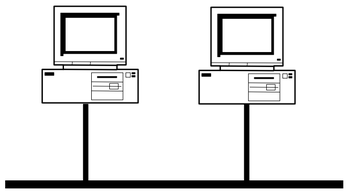
Announcements

- HW2B due tonight
- Project 2 design doc due Friday
- Networking tutorial, Saturday 3/7, 5-7pm, in HP Auditorium (306 Soda)

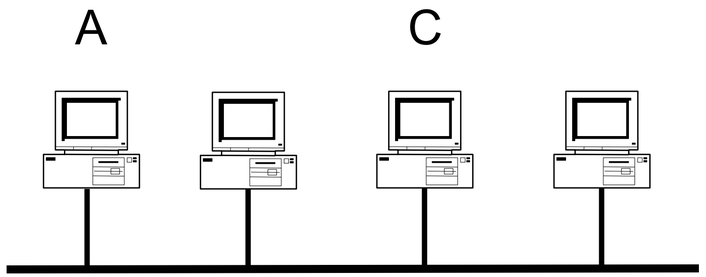
Network Security

- Today: background in networking, so we can explore network security for next 3 weeks
  - Speed running a month of networking in one lecture, so I'll focus on aspects that are security-relevant
  - Please ask questions when things are unclear!

Local-Area Networks



point-to-point

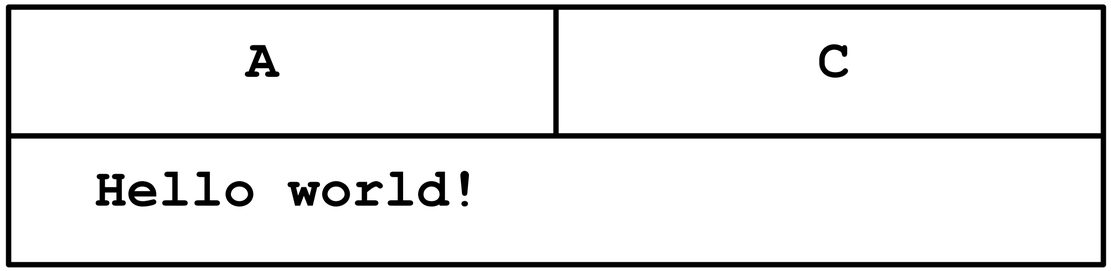


shared

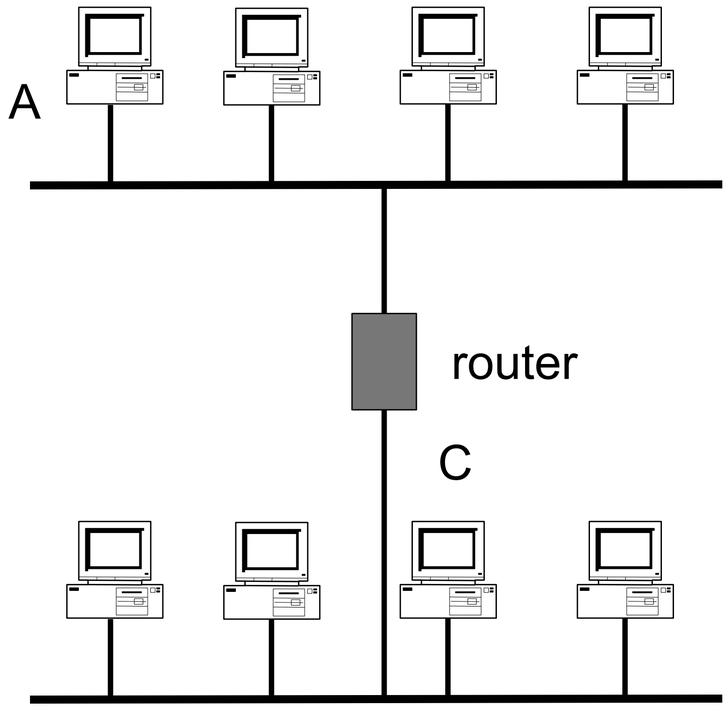
How does computer A send a message to computer C?

Local-Area Networks (LAN): Packets


```
Source: A
Destination: C
Message: Hello world!
```

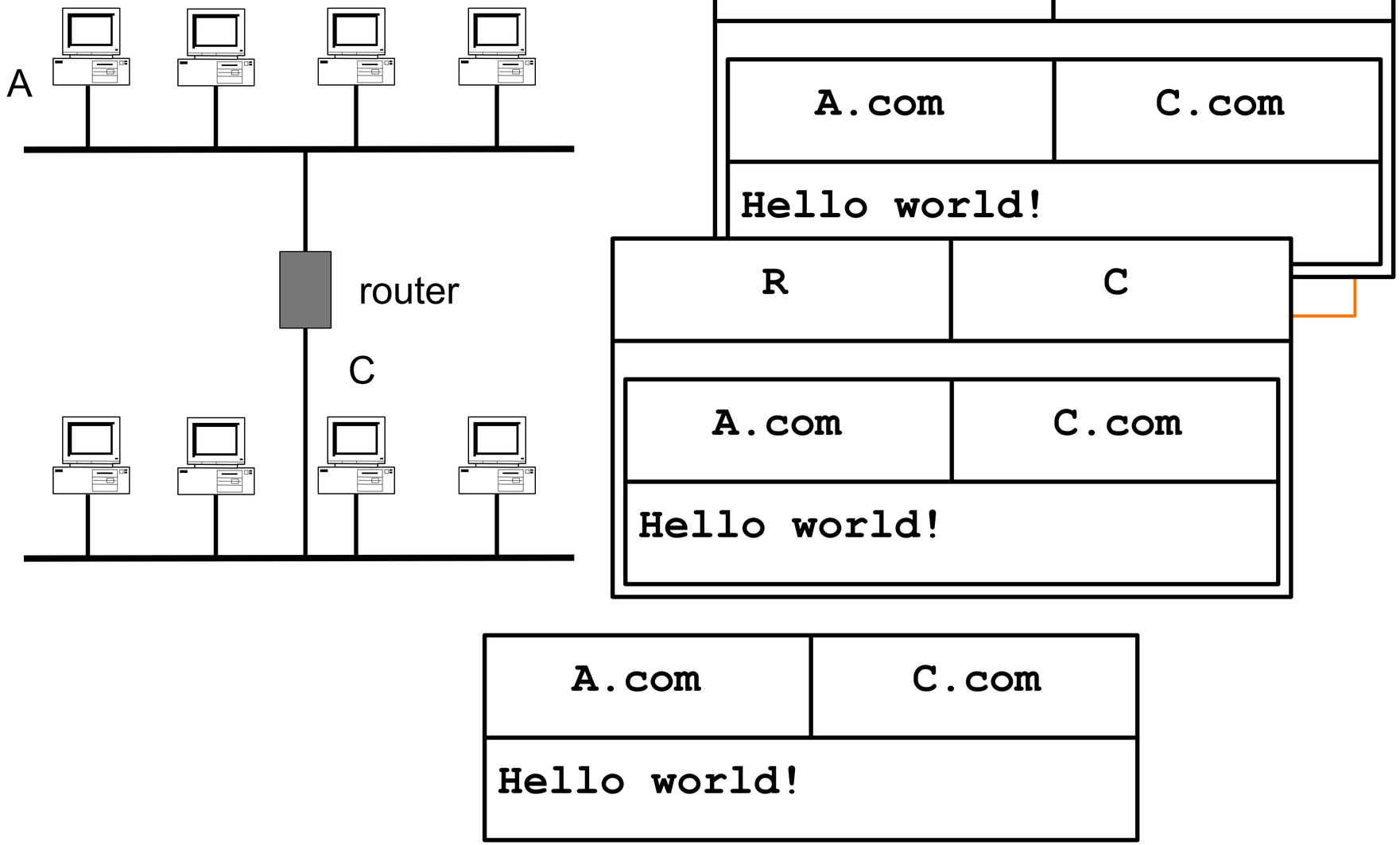


Wide-Area Networks



How do we connect two LANs?

Wide-Area Networks



Protocols

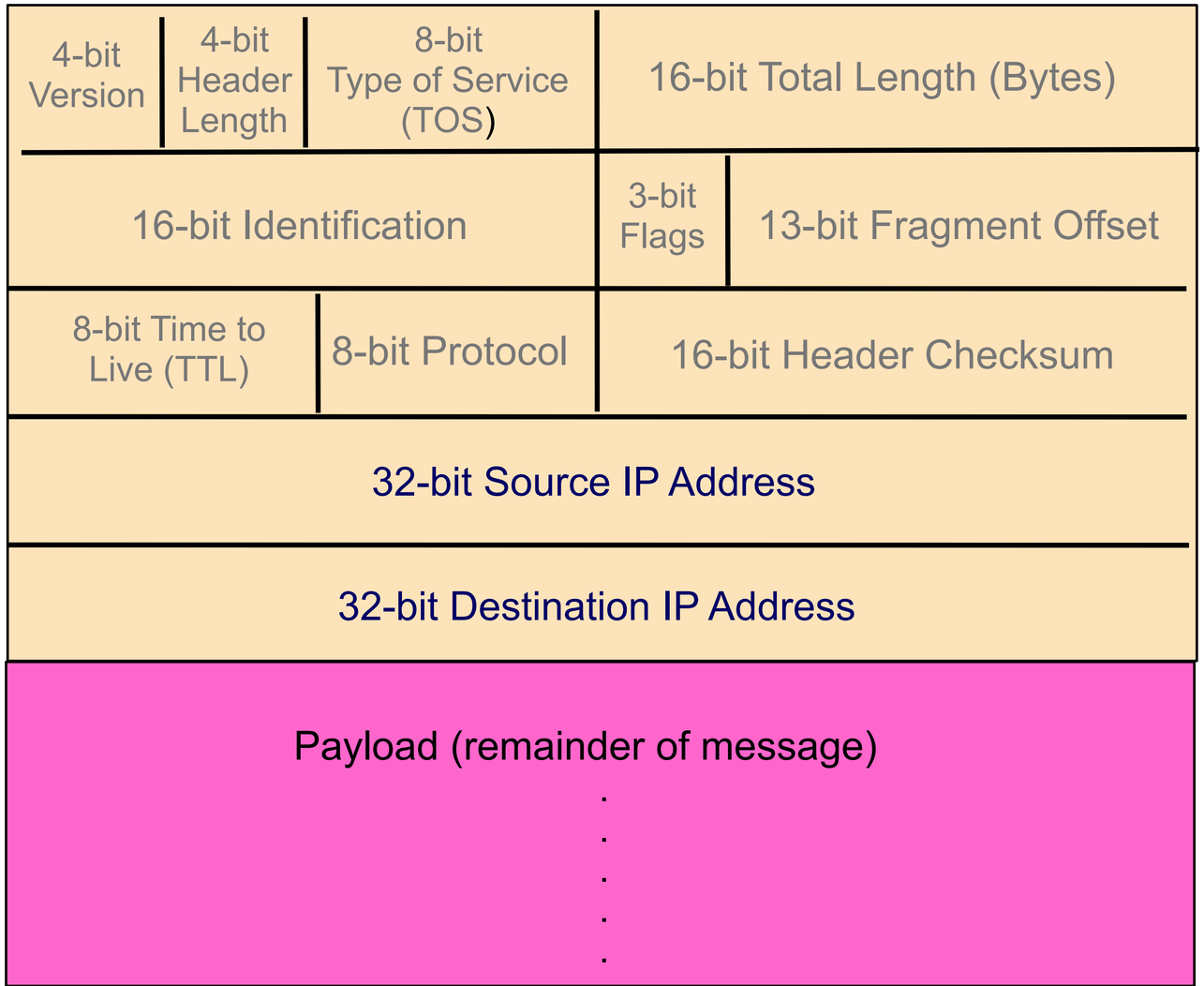
- A protocol is an agreement on how to communicate
- Includes syntax and semantics
  - How a communication is specified & structured
    - Format, order messages are sent and received
  - What a communication means
    - Actions taken when transmitting, receiving, or timer expires
- E.g.: making a comment in lecture?
  1. Raise your hand.
  2. Wait to be called on.
  3. Or: wait for speaker to **pause** and vocalize
  4. If unrecognized (after timeout): vocalize w/ “excuse me”

Network is Dumb

- Original Internet design: interior nodes ("routers") have no knowledge* of ongoing connections going through them
- Not how you picture the telephone system works
  - Which internally tracks all of the active voice calls
- Instead: the postal system!
  - Each Internet message ("packet") self-contained
  - Interior routers look at destination address to forward
  - If you want smarts, build it "*end-to-end*", not "hop-by-hop"
  - Buys simplicity & robustness at the cost of shifting complexity into end systems

* Today's Internet is full of hacks that violate this

Self-Contained IP Packet Format

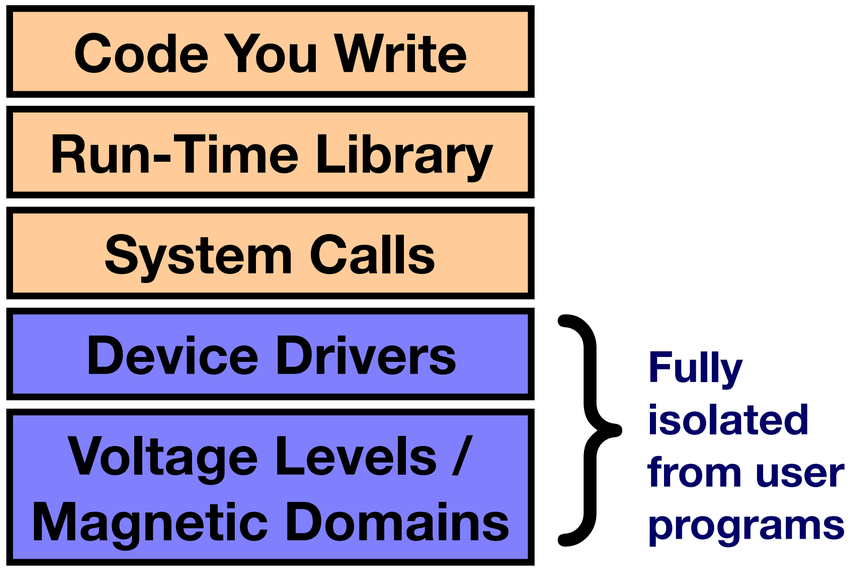


IP = Internet *Protocol*

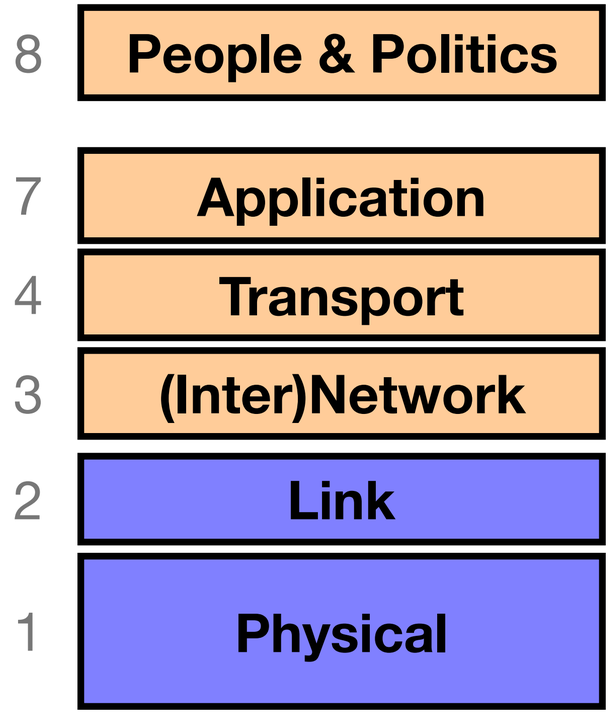
***Header* is like a letter envelope: contains all info needed for delivery**

Layering

- Internet design is partitioned into layers
  - Each layer relies on services provided by next layer below …
  - … and provides services to layer above it
- Analogy:
  - Consider structure of an application you've written and the "services" each layer relies on / provides



Internet Layering ("Protocol Stack"/"**OSI Model**")



Note on a point of potential confusion: these diagrams are always drawn with lower layers **below** higher layers …

But diagrams showing the layouts of packets are often the *opposite*, with the lower layers at the **top** since their headers precede those for higher layers

(And nobody remembers what layers 5 and 6 are for ("Session" and "Presentation) for the trivia buffs because they aren't really used)

(also, layer 8 is a "joke", but really is important)

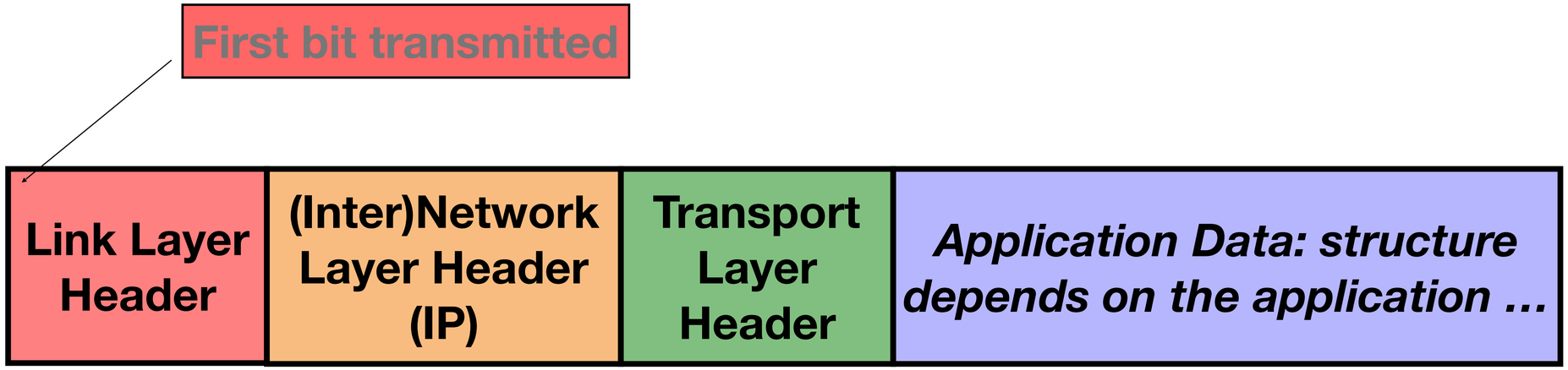
Packets and The Network

- Modern networks break communications up into packets
  - For our purposes, packets contain a variable amount of data up to a maximum specified by the particular network
- The sending computer breaks up the message and the receiving computer puts it back together
  - So the software doesn't actually see the packets per-se
  - Network itself is ***packet switched***: sending each packet on towards its next destination

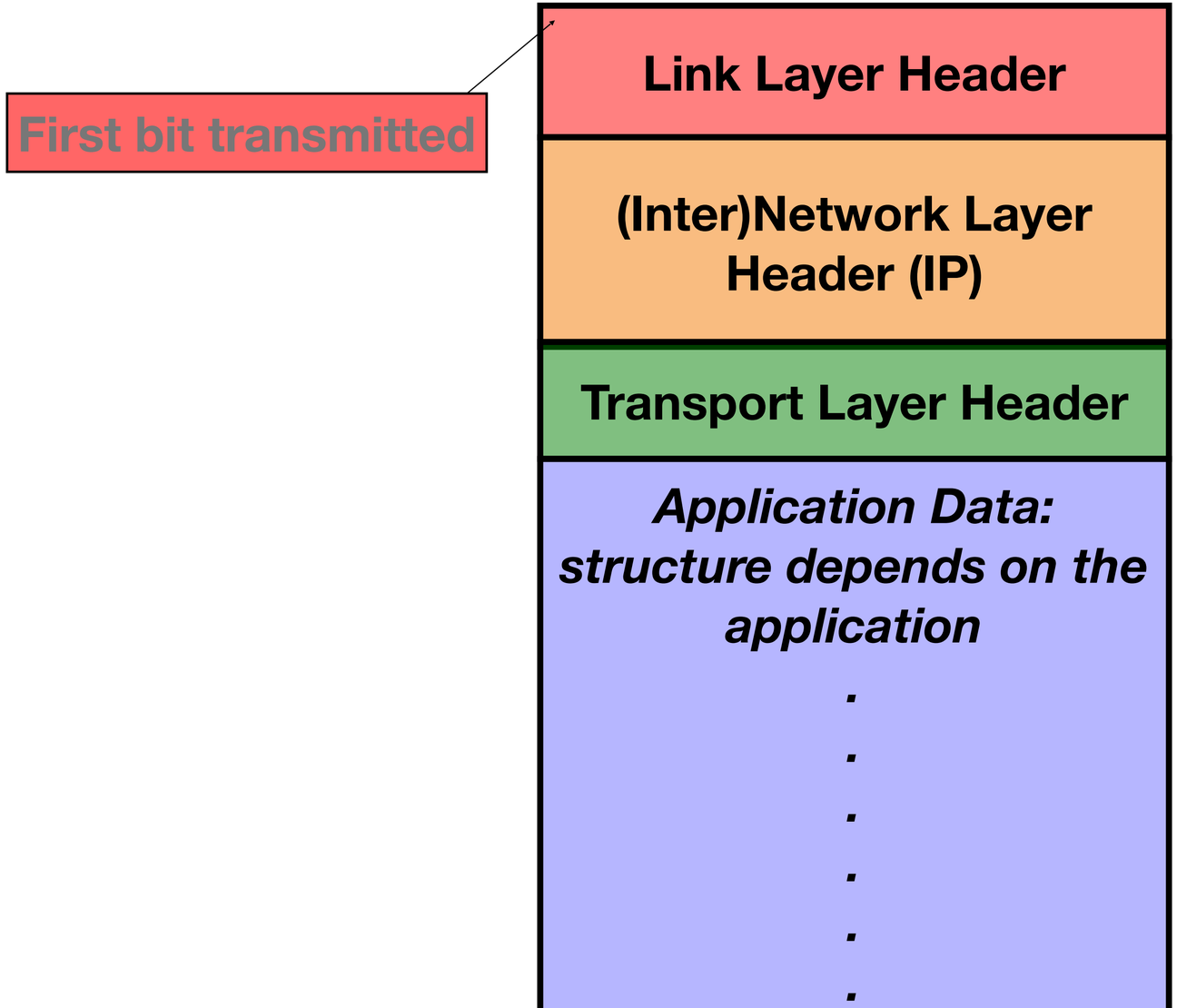
Reliability

- Packets are received ***correctly*** or not at all, if ***random*** errors occur
  - Packets have a checksum
  - No guarantees if adversary modifies packets (no cryptographic MACs)
- Packets may be ***unreliable*** and “dropped”
  - It’s up to higher-level protocols to make the connection reliable

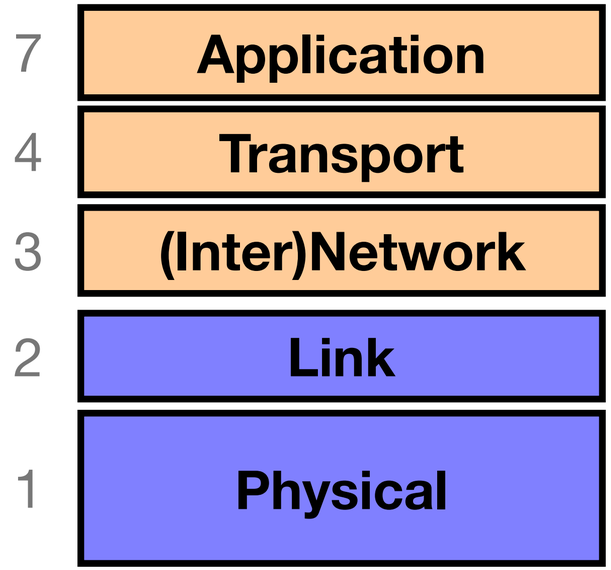
Horizontal View of a Single Packet



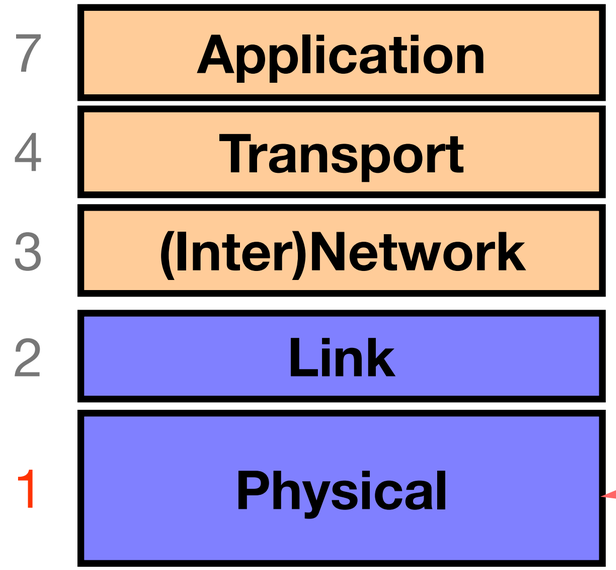
Vertical View of a Single Packet



Internet Layering (“Protocol Stack”)



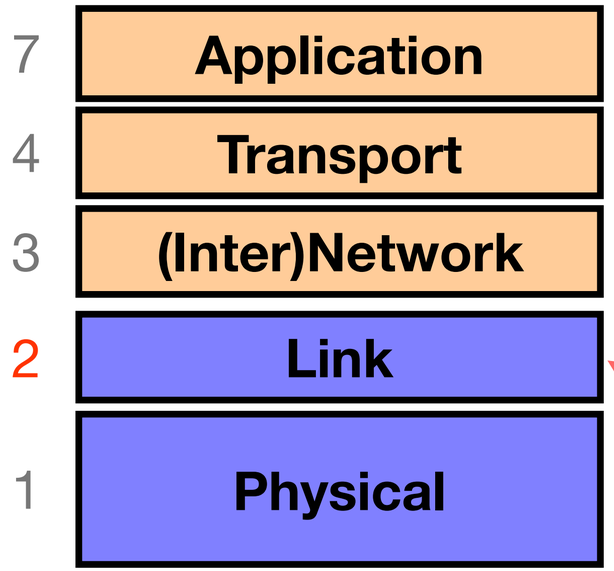
Layer 1: Physical Layer



Encoding bits to send them over a single **physical link**

e.g. patterns of
voltage levels /
photon intensities /
RF modulation

Layer 2: Link Layer

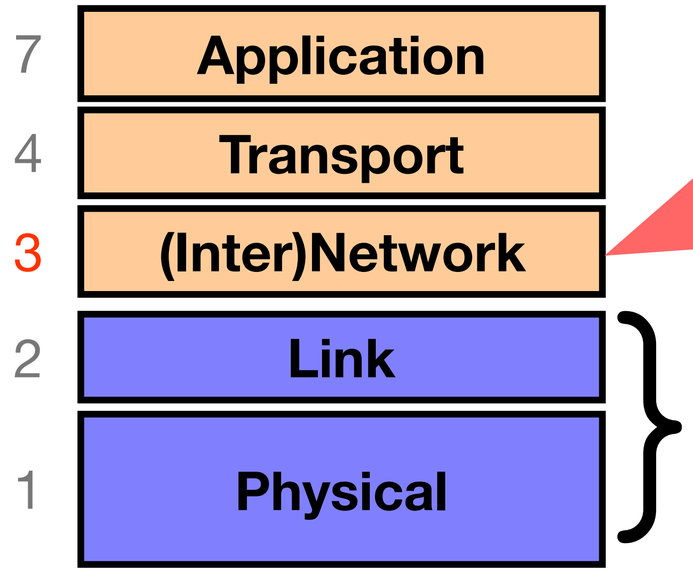


Framing and transmission of a collection of bits into individual messages sent across a single "subnetwork" (one physical technology)

Might involve multiple *physical links* (e.g., modern Ethernet)

Often technology supports broadcast transmission (**every** "node" connected to subnet receives)

Layer 3: (Inter)Network Layer *(IP)*



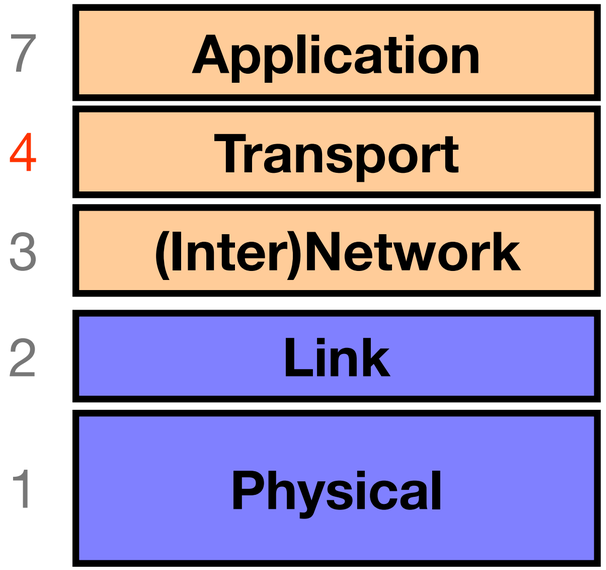
Bridges multiple "subnets" to provide *end-to-end* internet connectivity between nodes

- Provides global addressing

Works across different link technologies

***Different* for each Internet "hop"**

Layer 4: Transport Layer

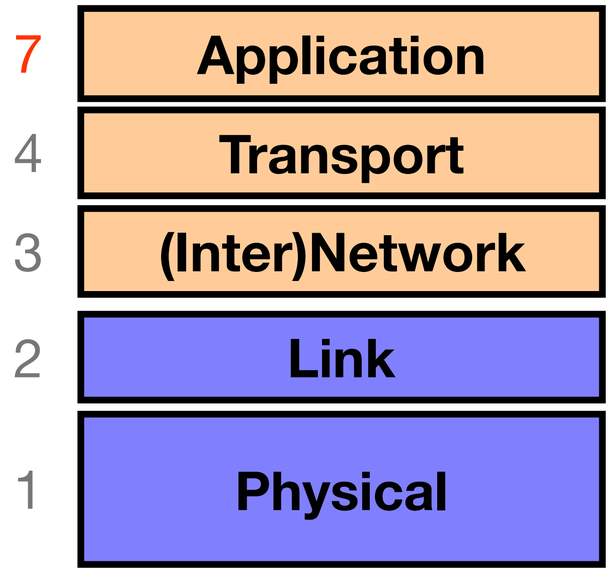


End-to-end communication between processes

Different services provided:
- TCP = reliable *byte stream*
- UDP = unreliable *datagrams*

(Datagram = single packet message)

Layer 7: Application Layer



Communication of whatever you wish

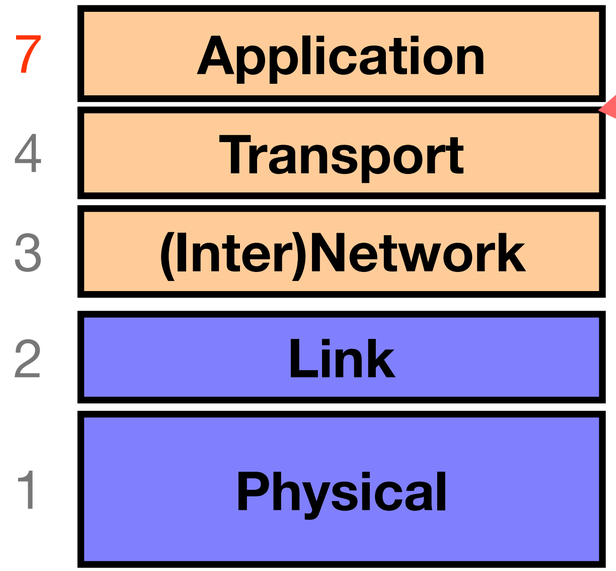
Can use whatever transport(s) is convenient

Freely structured

E.g.:

Skype, SMTP (email), HTTP (Web), Halo, BitTorrent

4.5: Some Crypto…

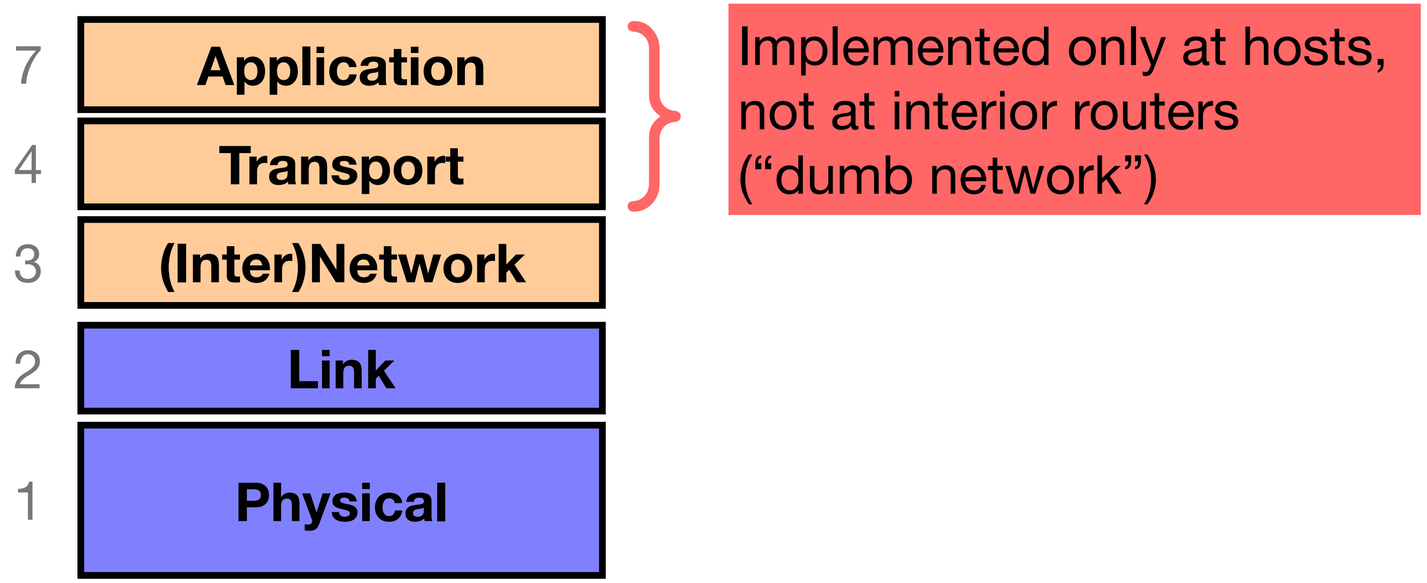


TLS cryptography
(aka the ’s’ in HTTPS)

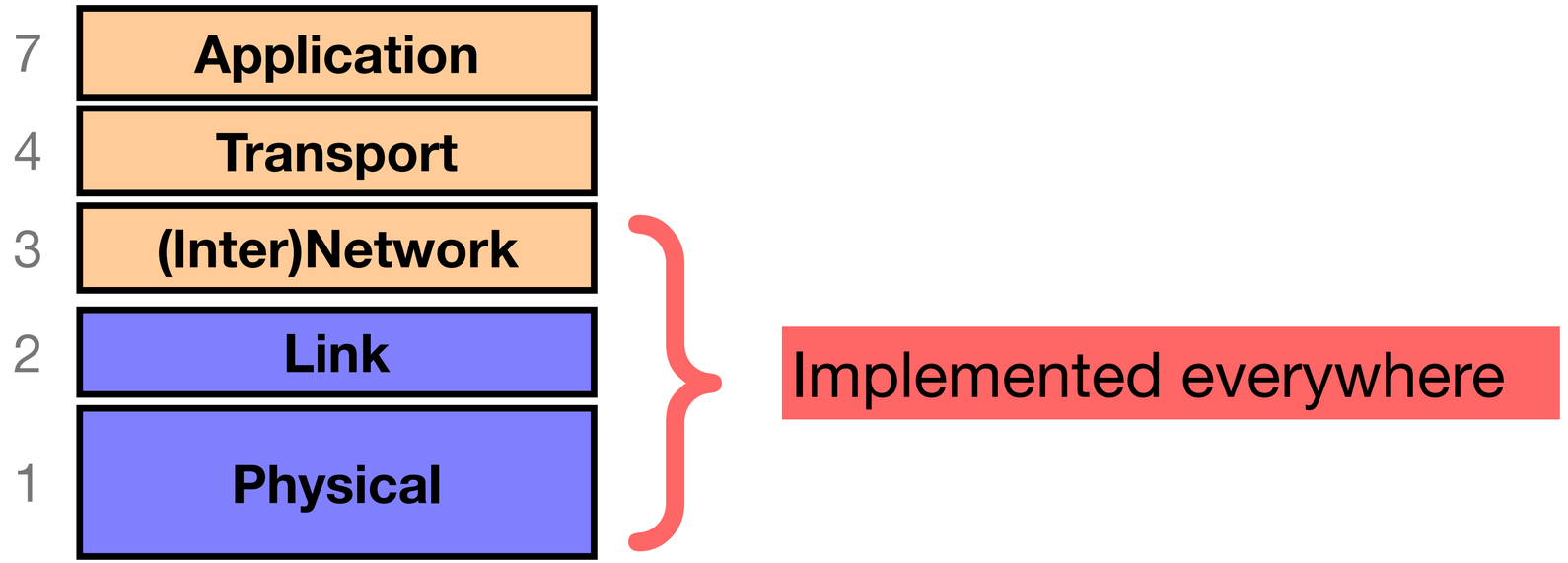
Often basically used as a “layer 4.5” transport layer to encrypt otherwise unencrypted network connections

Other times crypto may be at the application layer (e.g. ssh

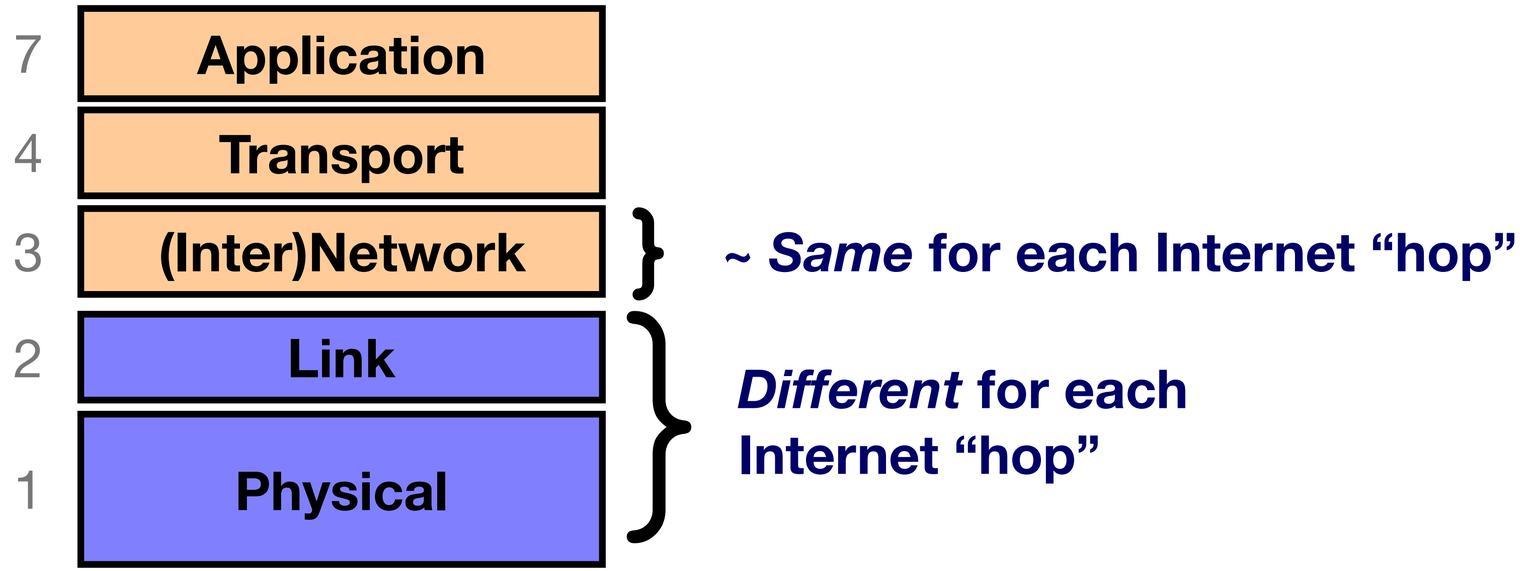
Internet Layering (“Protocol Stack”)



Internet Layering (“Protocol Stack”)

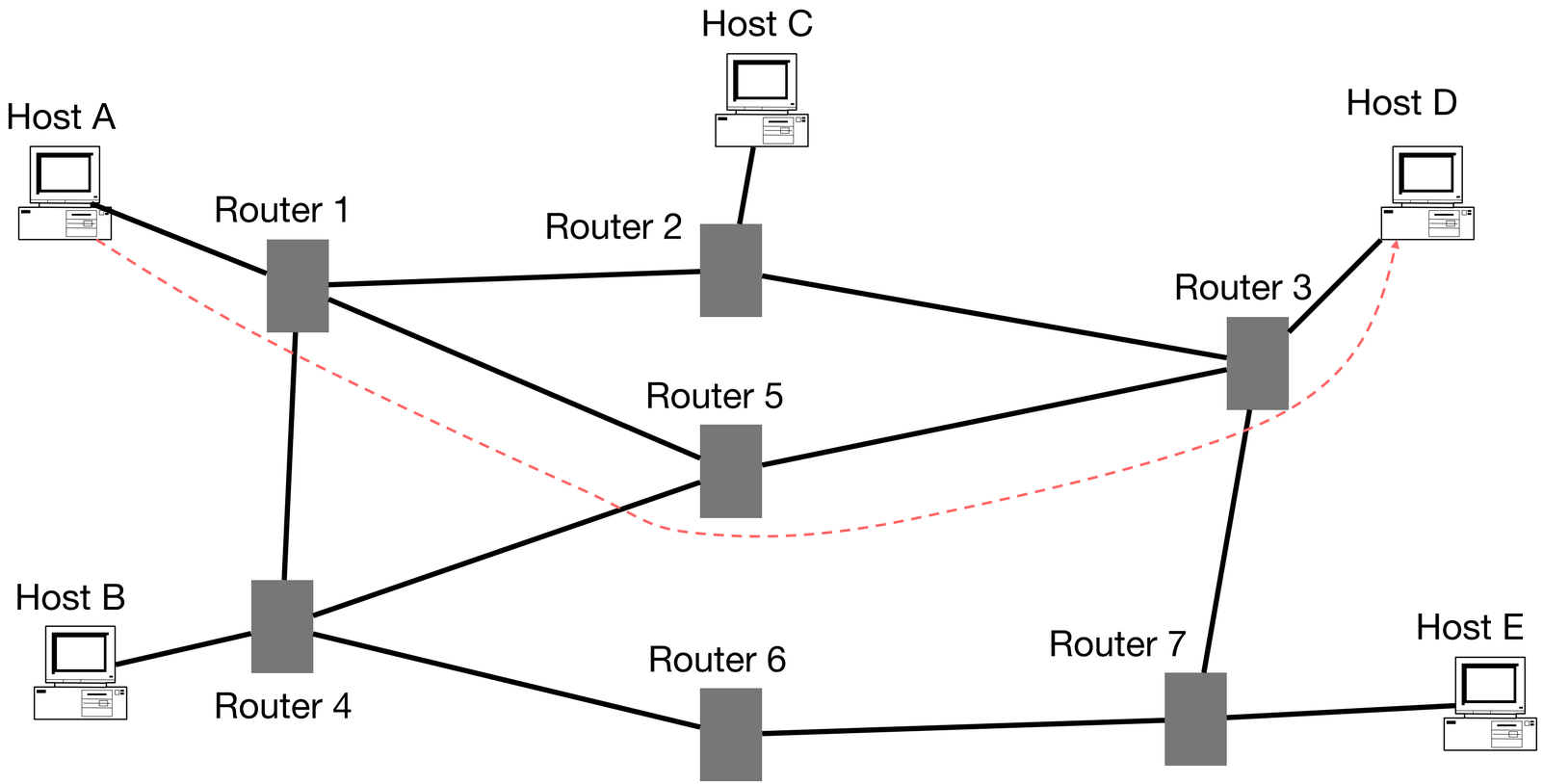


Internet Layering ("Protocol Stack")

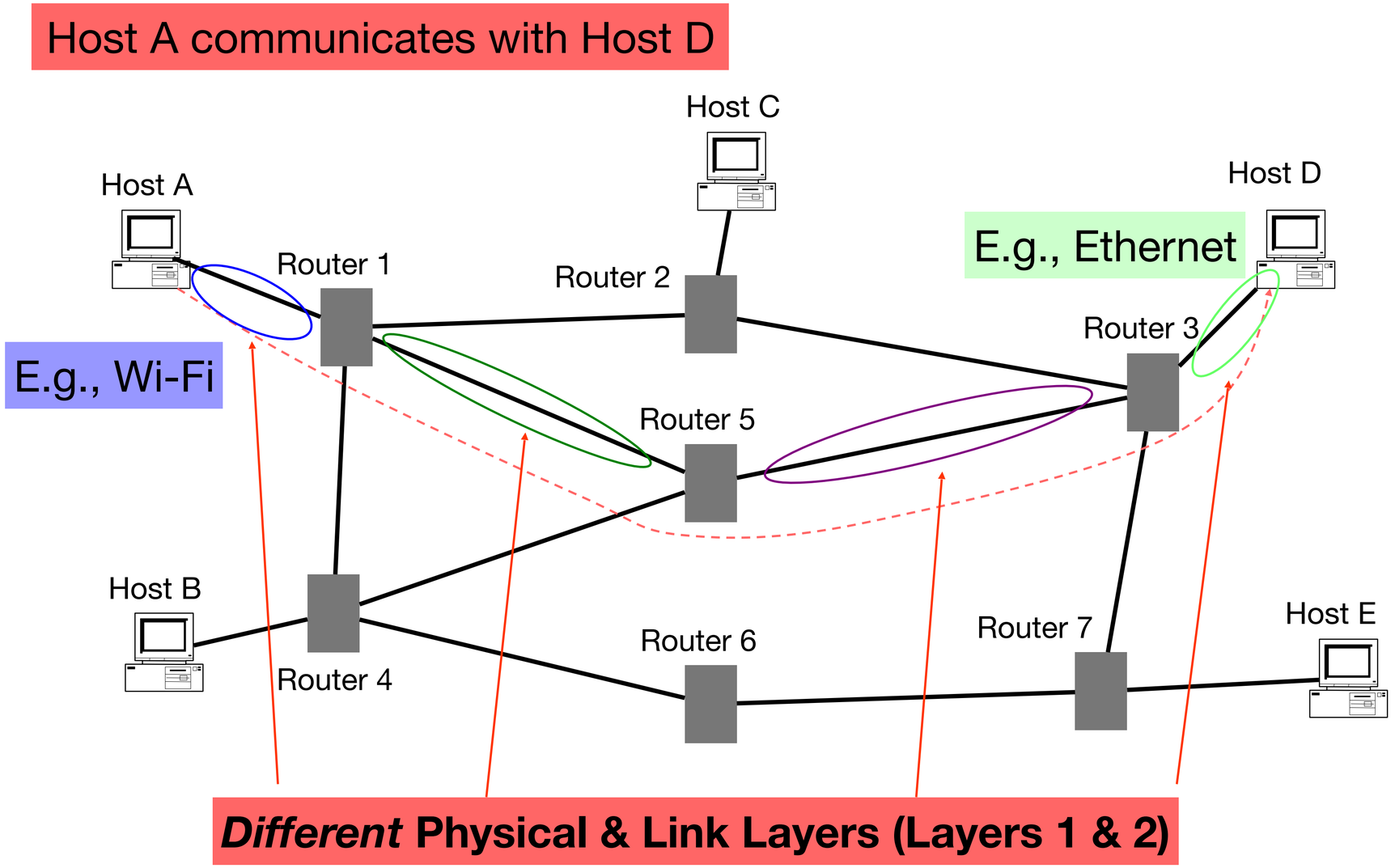


Hop-By-Hop vs. End-to-End Layers

Host A communicates with Host D

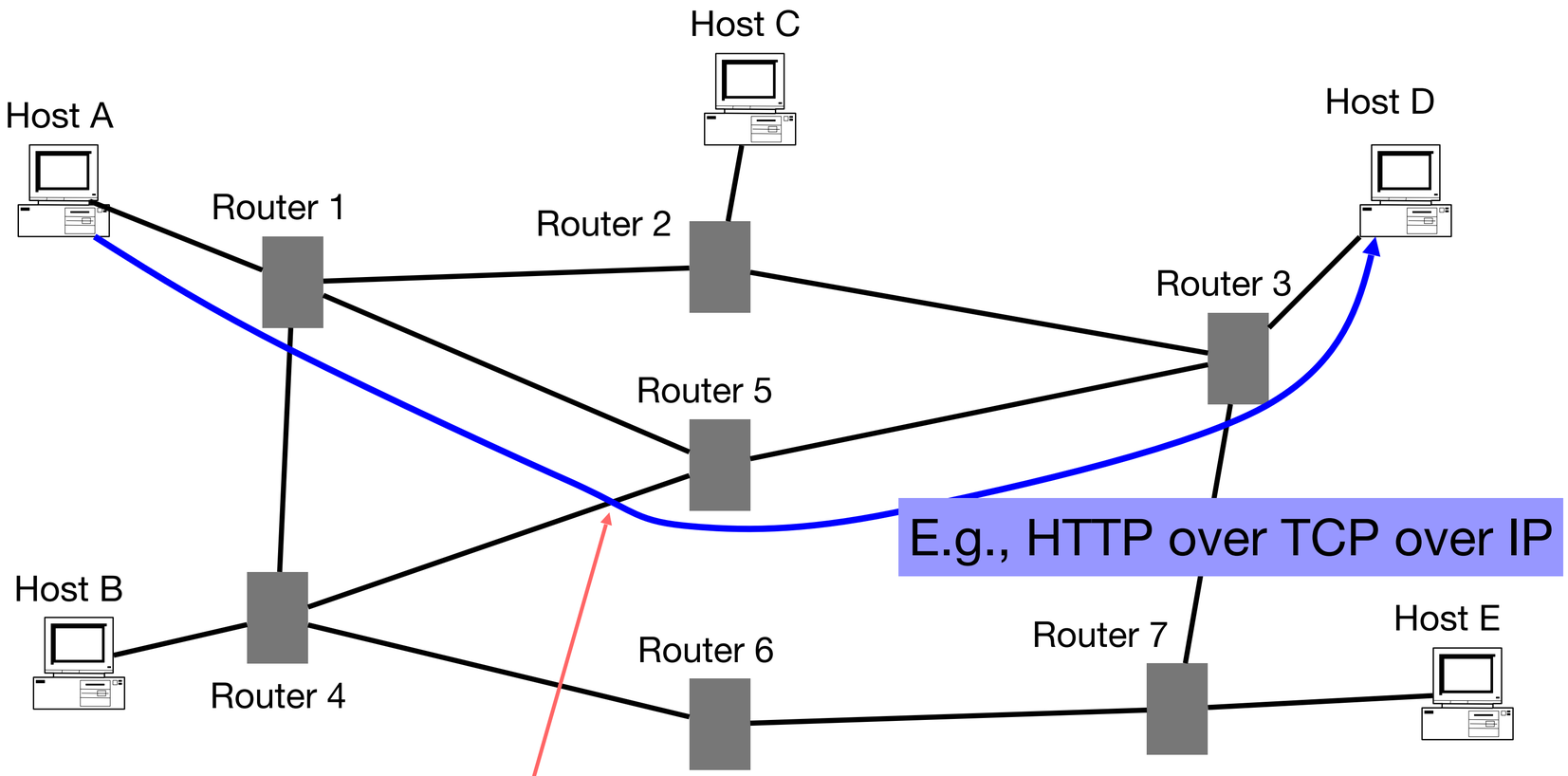


Hop-By-Hop vs. End-to-End Layers



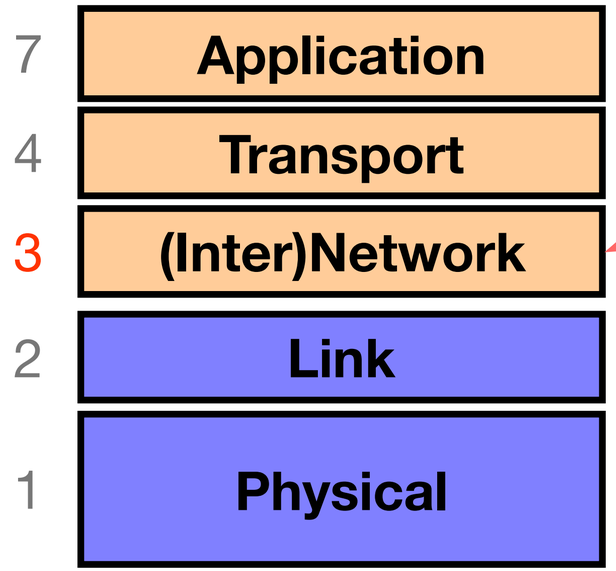
Hop-By-Hop vs. End-to-End Layers

Host A communicates with Host D



Same Network / Transport / Application Layers (3/4/7)
(Routers **ignore** Transport & Application layers)

Layer 3: (Inter)Network Layer *(IP)*

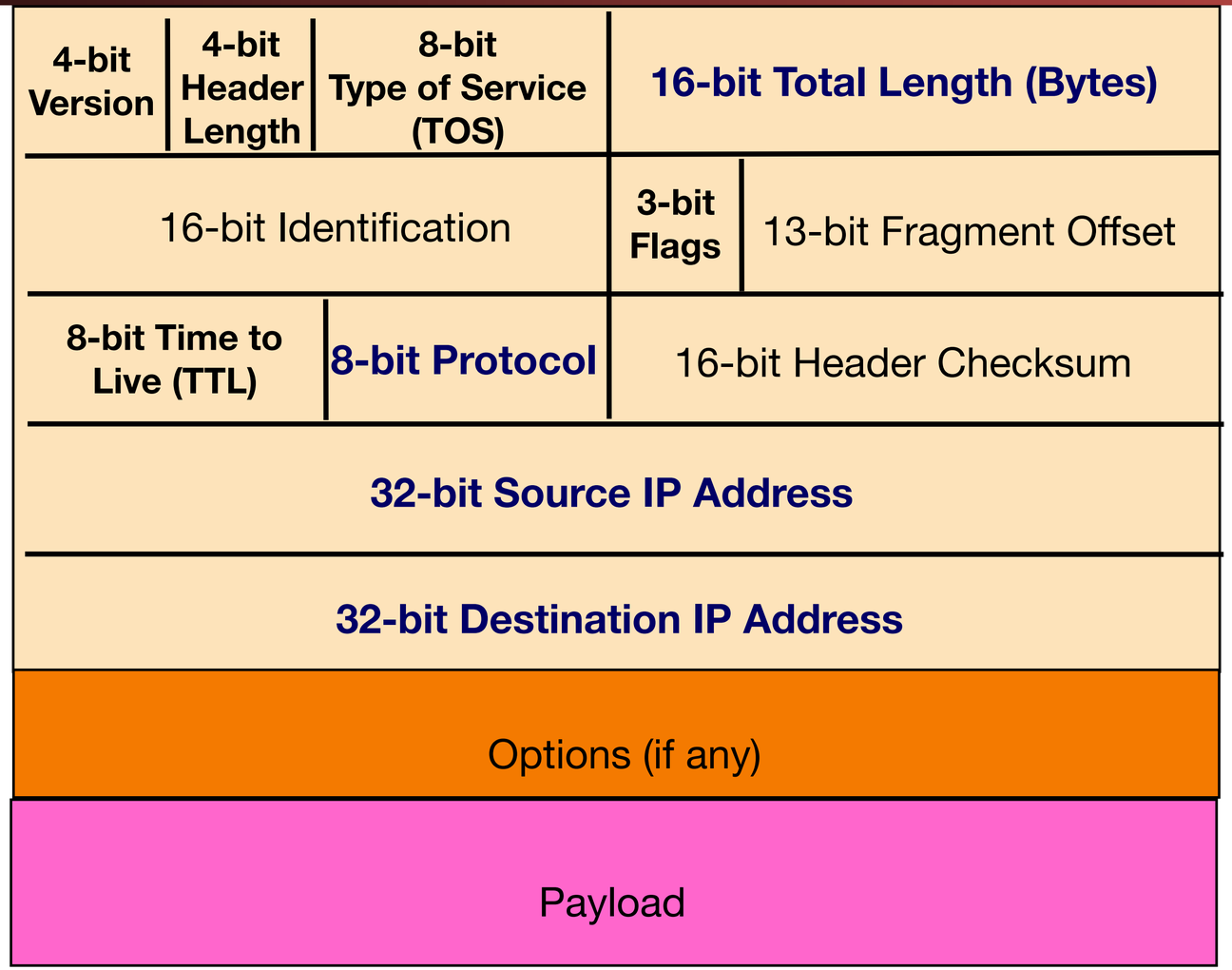


Bridges multiple “subnets” to provide *end-to-end* internet connectivity between nodes

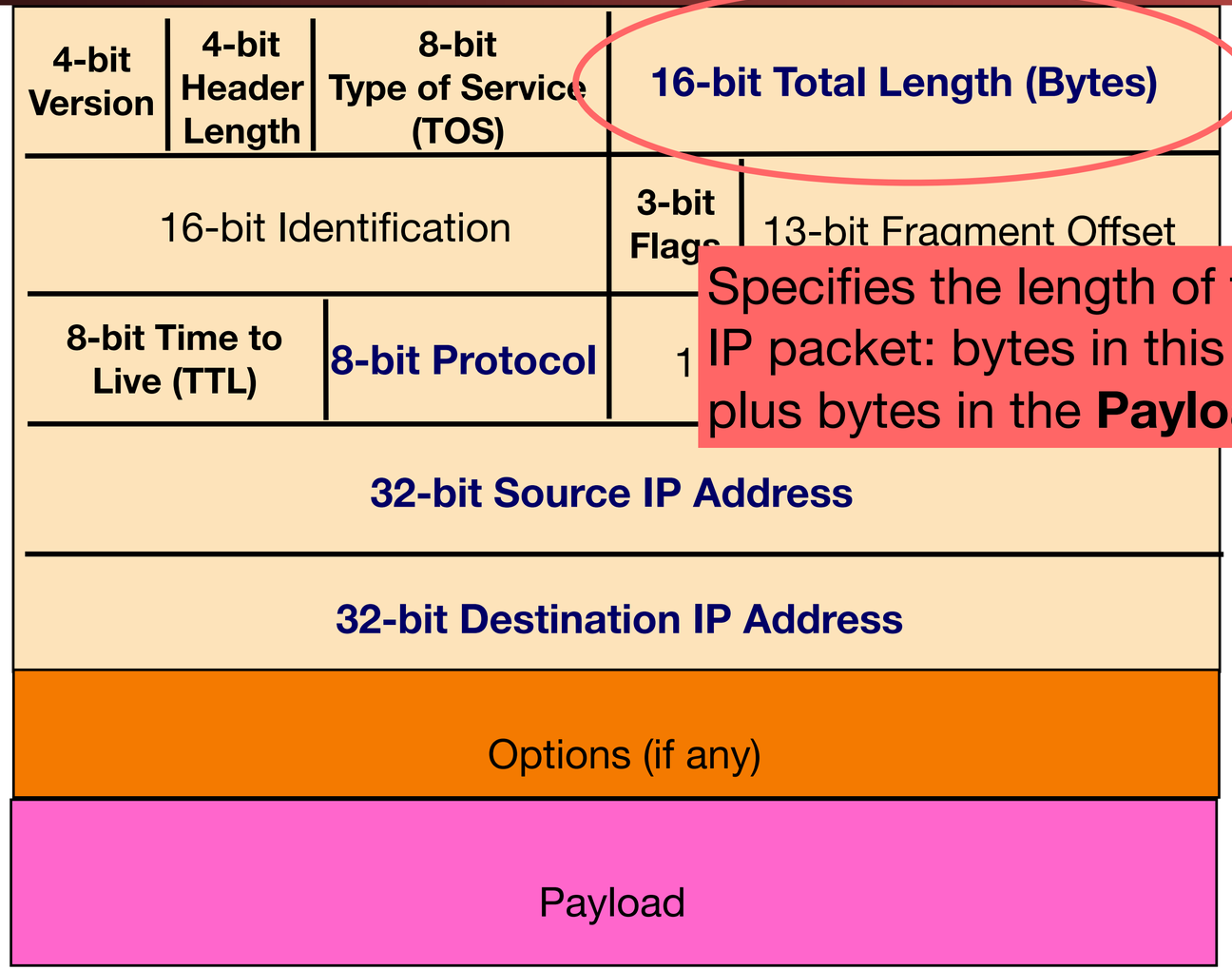
- Provides global addressing

Works across different link technologies

IPv4 Packet Structure (IP version 6 is different)

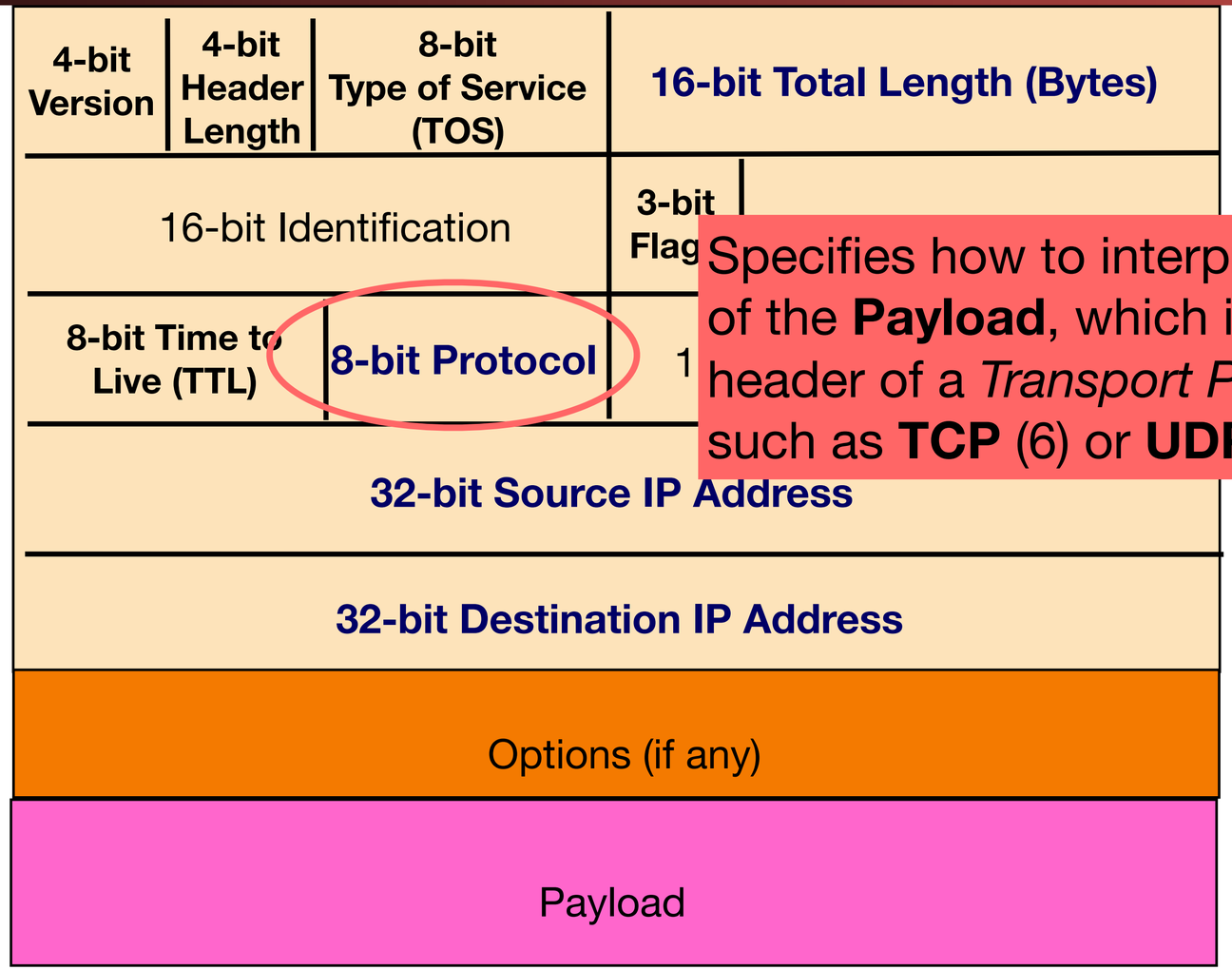


IP Packet Structure



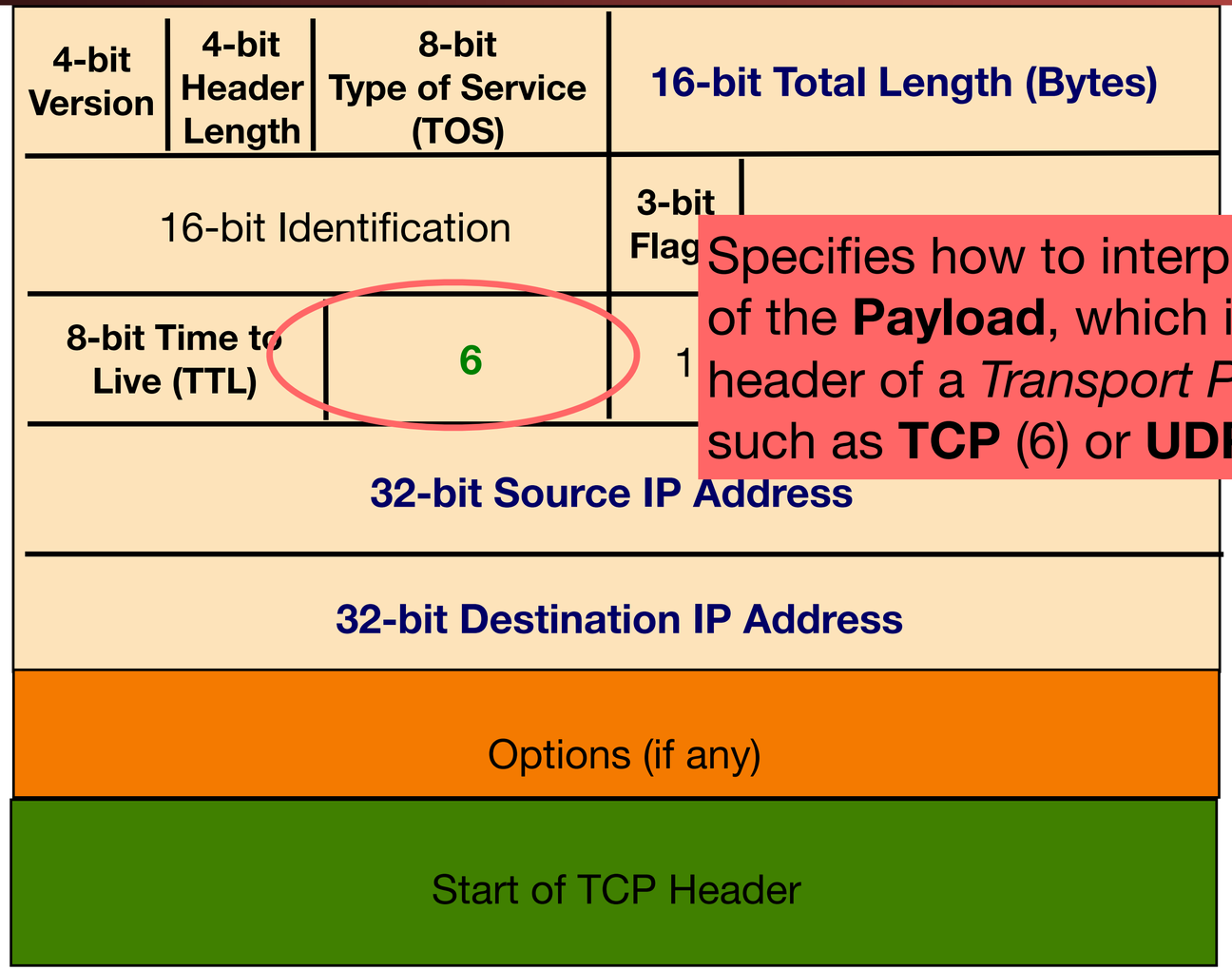
Specifies the length of the entire IP packet: bytes in this header plus bytes in the **Payload**

IP Packet Structure



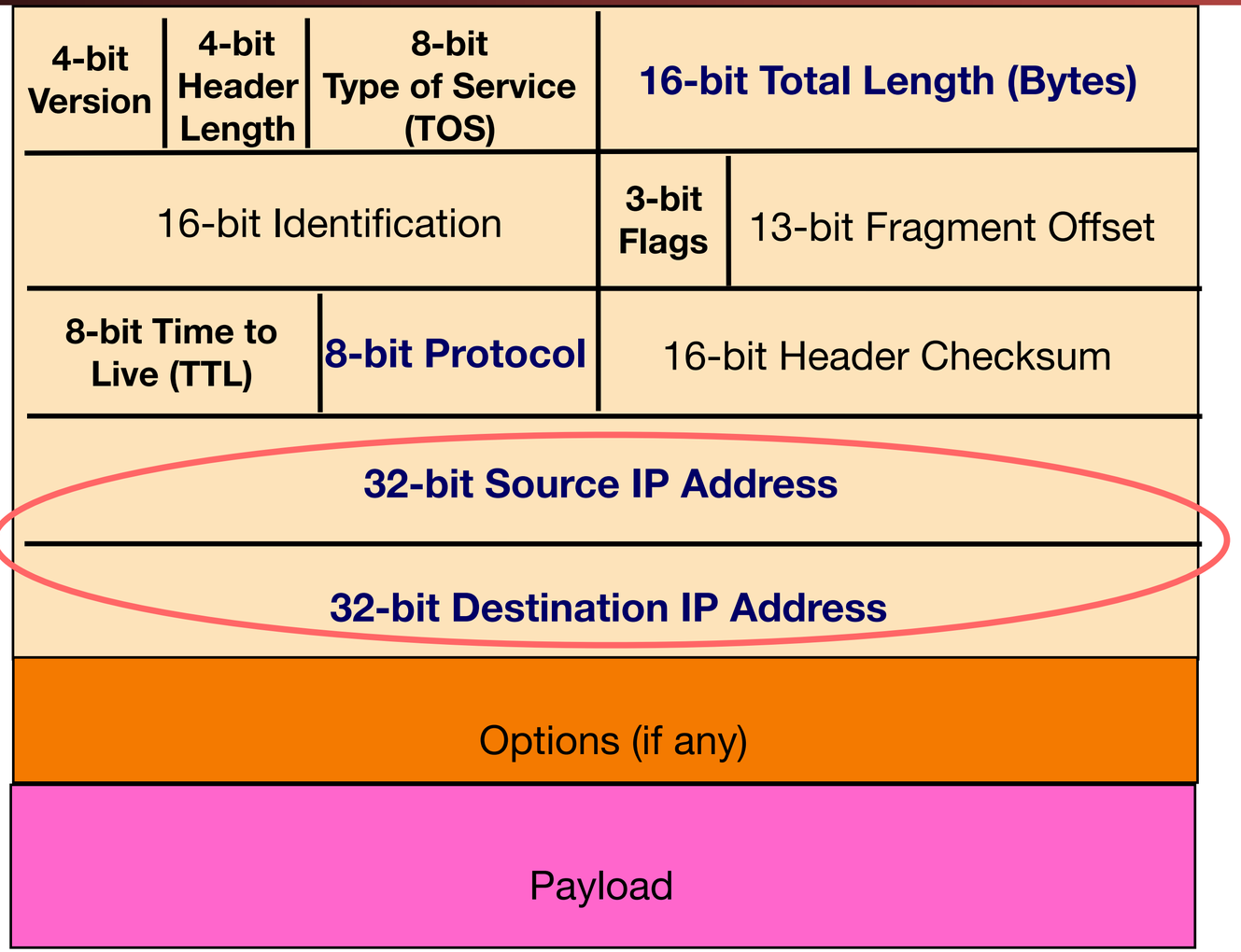
Specifies how to interpret the start of the **Payload**, which is the header of a *Transport Protocol* such as **TCP** (6) or **UDP** (17)

IP Packet Structure



Specifies how to interpret the start of the **Payload**, which is the header of a *Transport Protocol* such as **TCP** (6) or **UDP** (17)

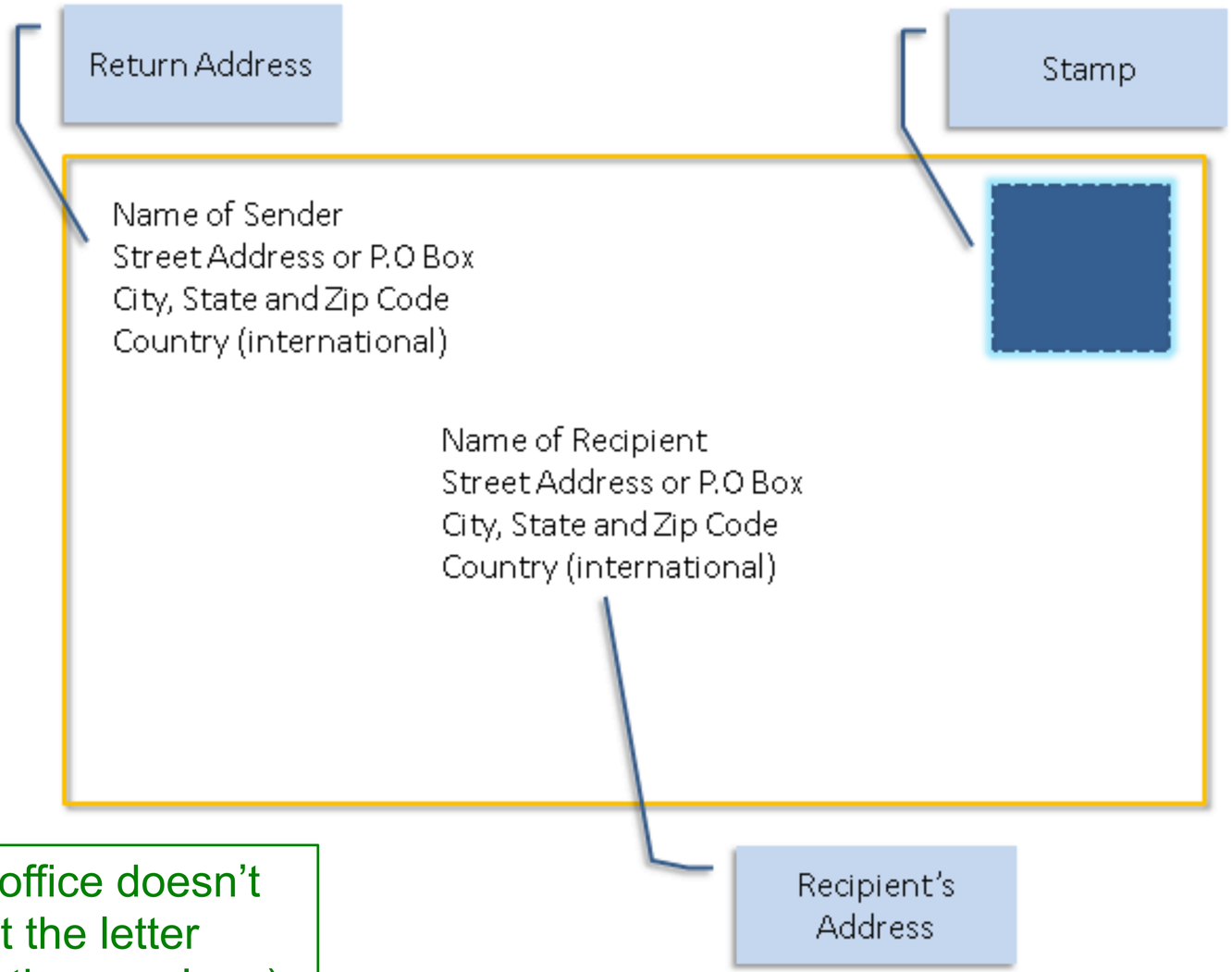
IP Packet Structure



IP Packet Header - IP addresses

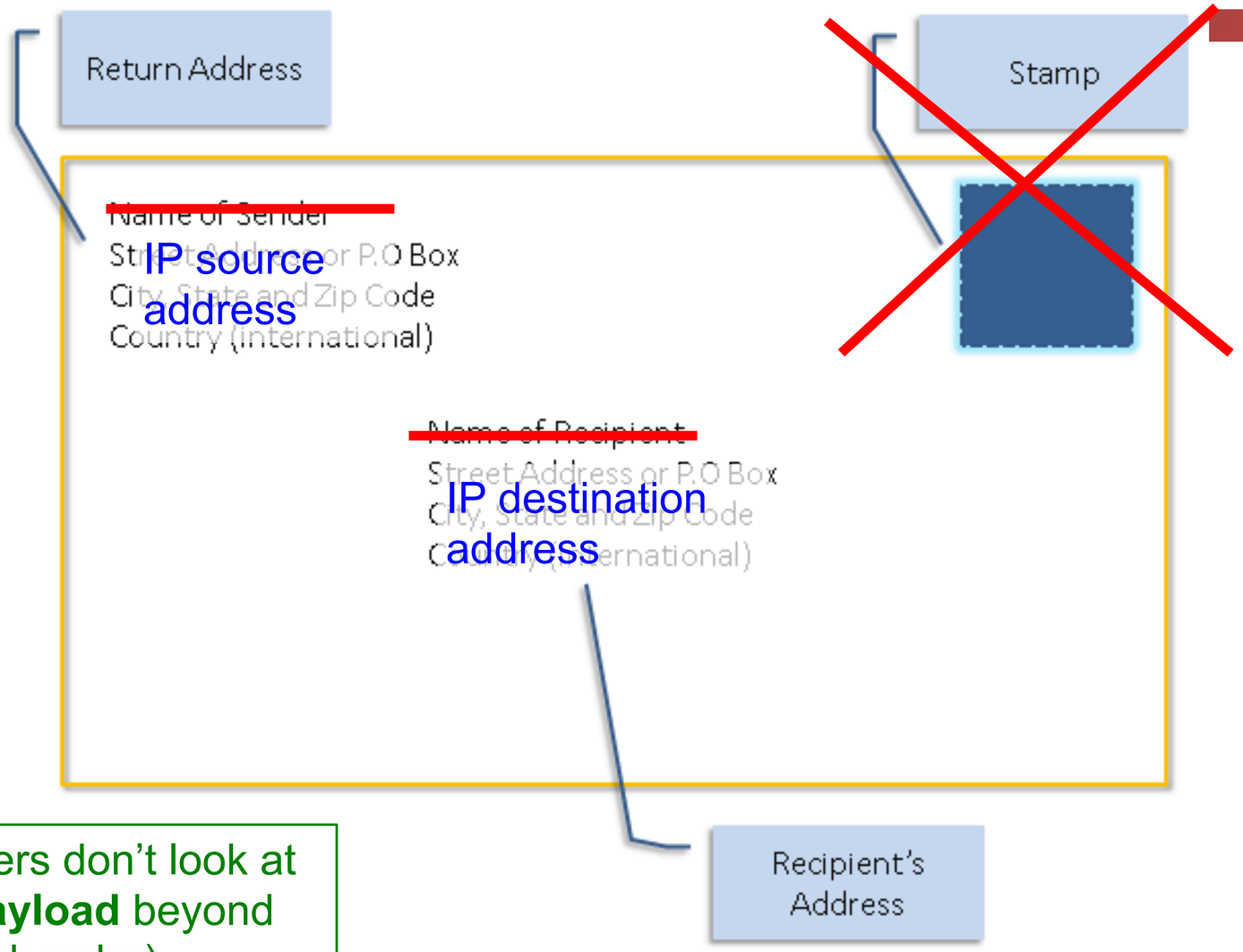
- Source address (32 bits)
  - Unique identifier/locator for the sending host
  - Recipient can decide whether to accept packet
  - Enables recipient to send reply back to source
- Destination address (32 bits)
  - Unique identifier/locator for the receiving host
  - Allows each node to make forwarding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

Postal Envelopes:



(Post office doesn't look at the letter inside the envelope)

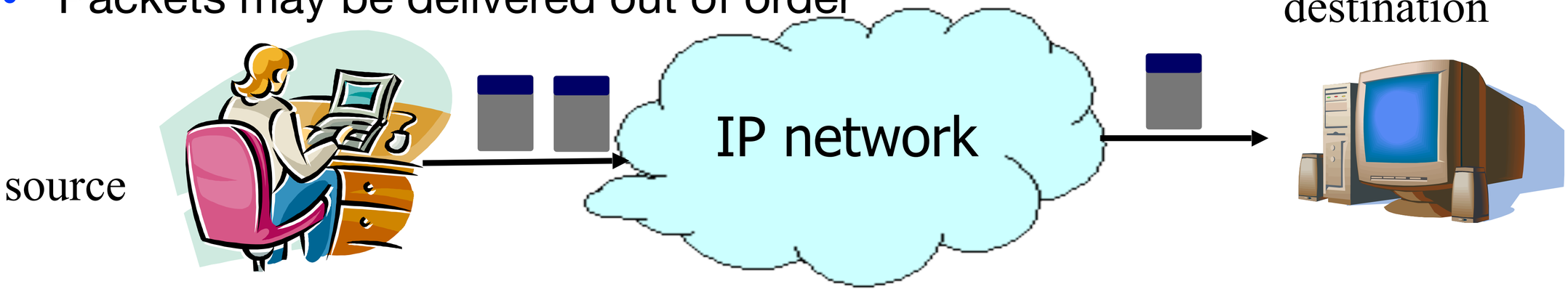
Analogy of IP to Postal Envelopes:



(Routers don't look at the **payload** beyond the IP header)

IP: "*Best Effort*" Packet Delivery

- Routers inspect destination address, locate "next hop" in forwarding table
  - Address = ~unique identifier/locator for the receiving host
- Only provides a "*I'll give it a try*" delivery service:
  - Packets may be lost
  - Packets may be corrupted
  - Packets may be delivered out of order



“Best Effort” is Lame! What to do?

- It’s the job of our Transport (layer 4) protocols to build services our apps need out of IP’s modest layer-3 service

“Best Effort” is Lame! What to do?

- #1 workhorse: TCP (Transmission Control Protocol)
- Service provided by TCP:
  - Connection oriented (explicit set-up / tear-down)
    - End hosts (processes) can have multiple concurrent long-lived communication
  - **Reliable**, in-order, *byte-stream* delivery
    - Robust detection & retransmission of lost data