Error Detection

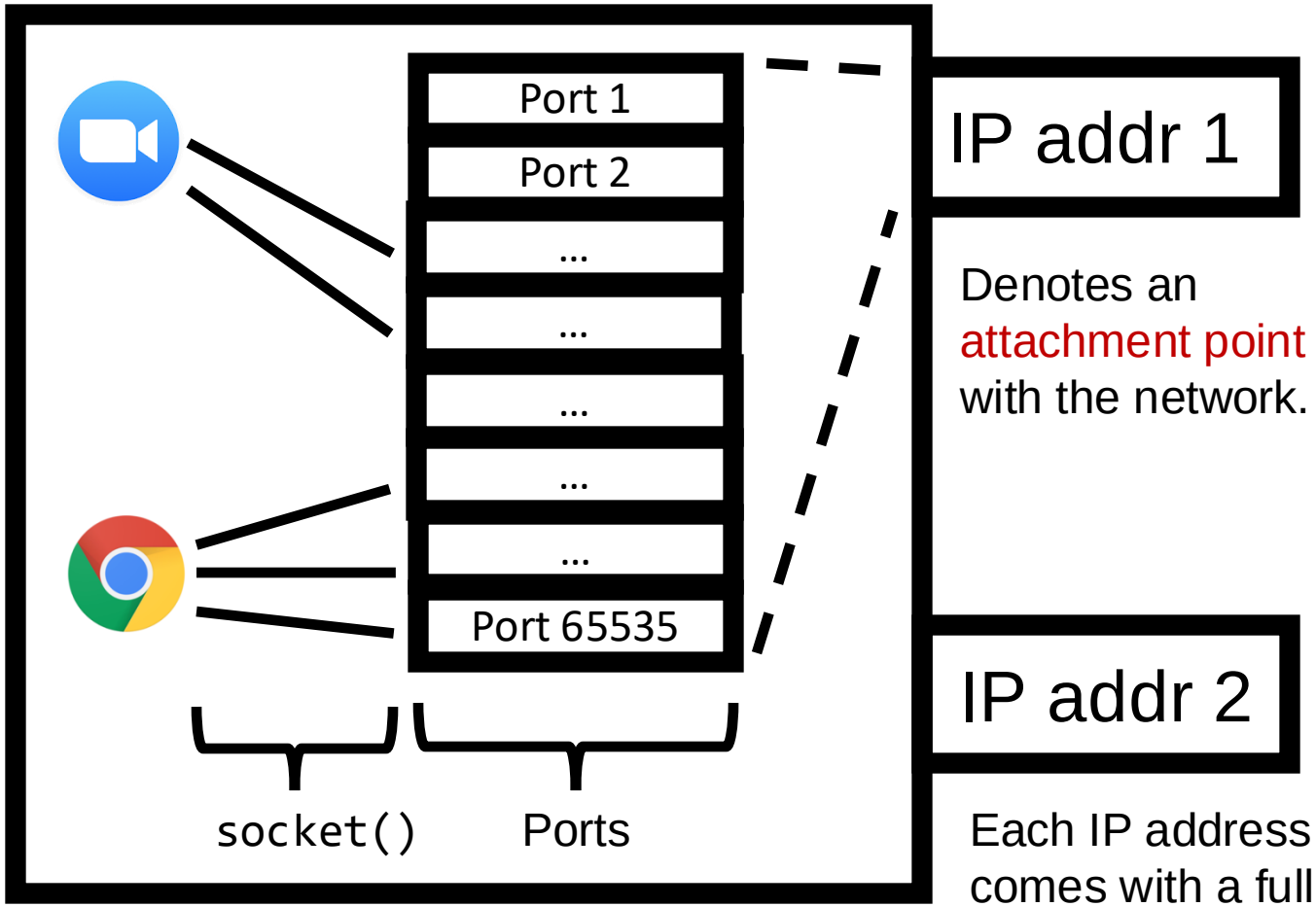
Lecture 12

http://www.cs.rutgers.edu/~sn624/352-F24

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

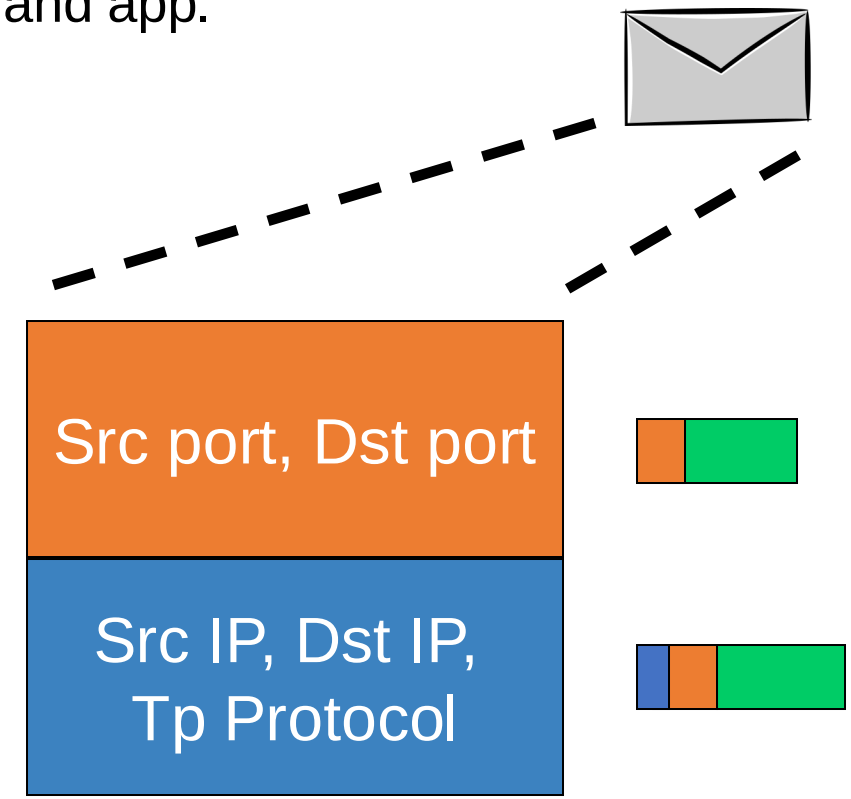
Review: Demultiplexing



Denotes an attachment point with the network.

Each IP address comes with a full copy of its own ports.

Connection lookup: The operating system does a lookup using these data to determine the right socket and app.



Listing sockets and connections

- `ss`
- `iperf –s` and `iperf –s -u`

User Datagram Protocol

UDP: User Datagram Protocol [RFC 768]

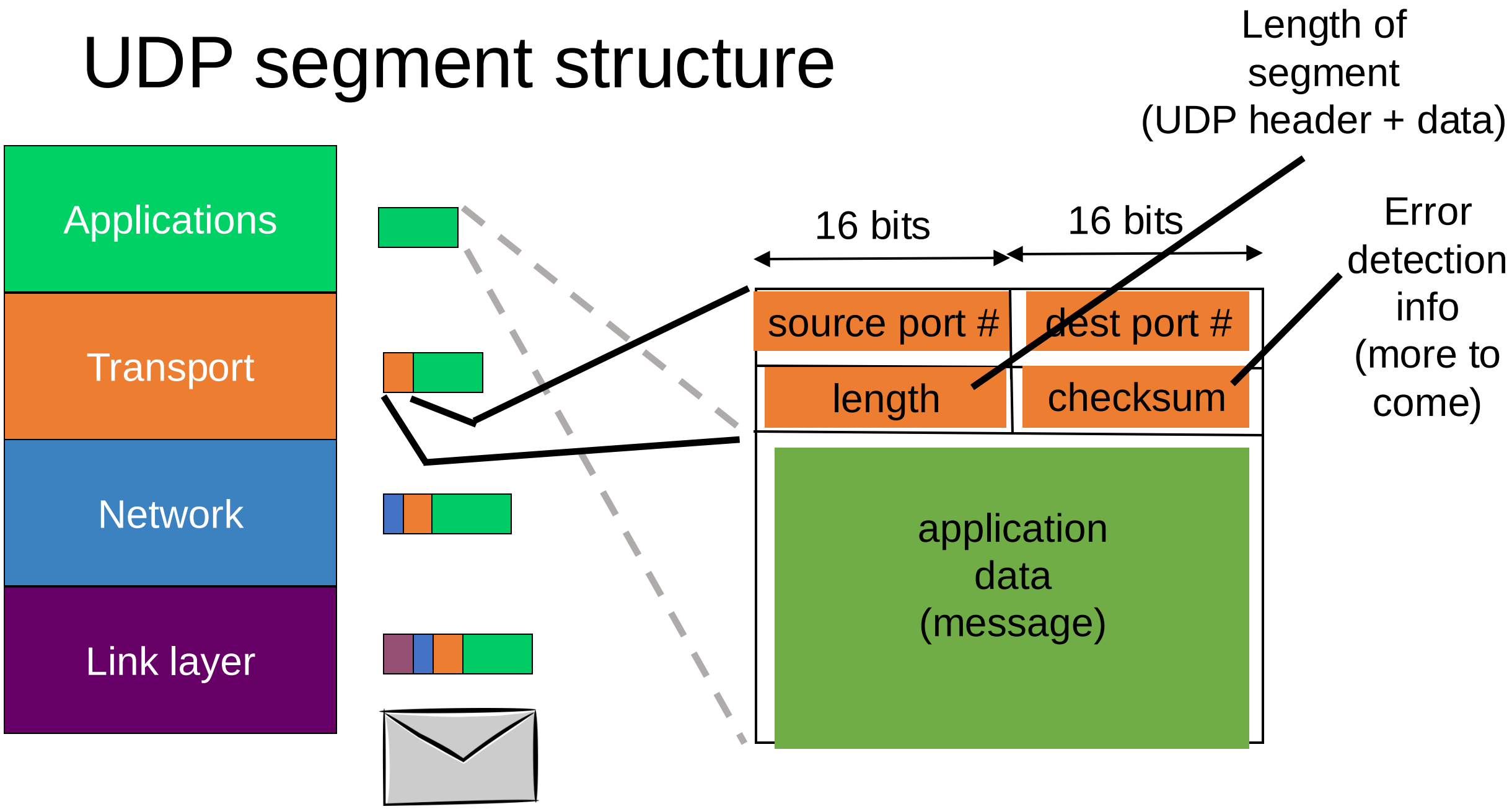
- Best effort service
  - UDP segments may be lost, corrupted, reordered
- UDP is connectionless
  - Each UDP segment handled independently of others (i.e. no "memory" across packets)
- Suitable for one-off req/resp
  - E.g., DNS uses UDP
- Early multimedia apps used UDP
  - Delay-sensitive but loss tolerant

Why are UDP's guarantees even okay?

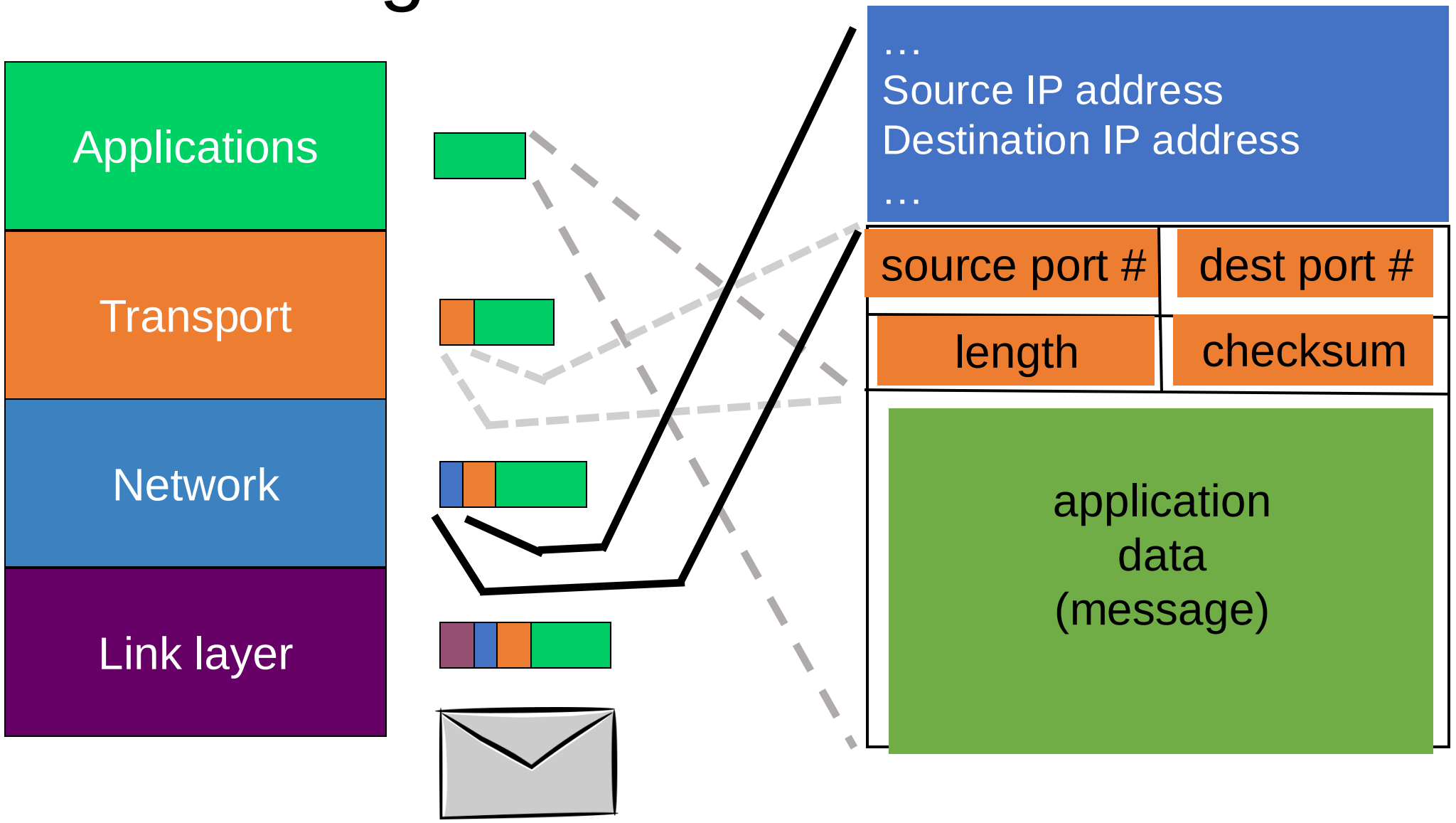
Simple & low overhead compared to TCP:

- No delays due to "connection establishment" (which TCP does)
  - UDP can send a packet immediately
- Small segment header (TCP's is larger)
- UDP can blast data without control
  - TCP is more balanced and measured
- Less memory for connection "state" at sender & receiver relative to TCP

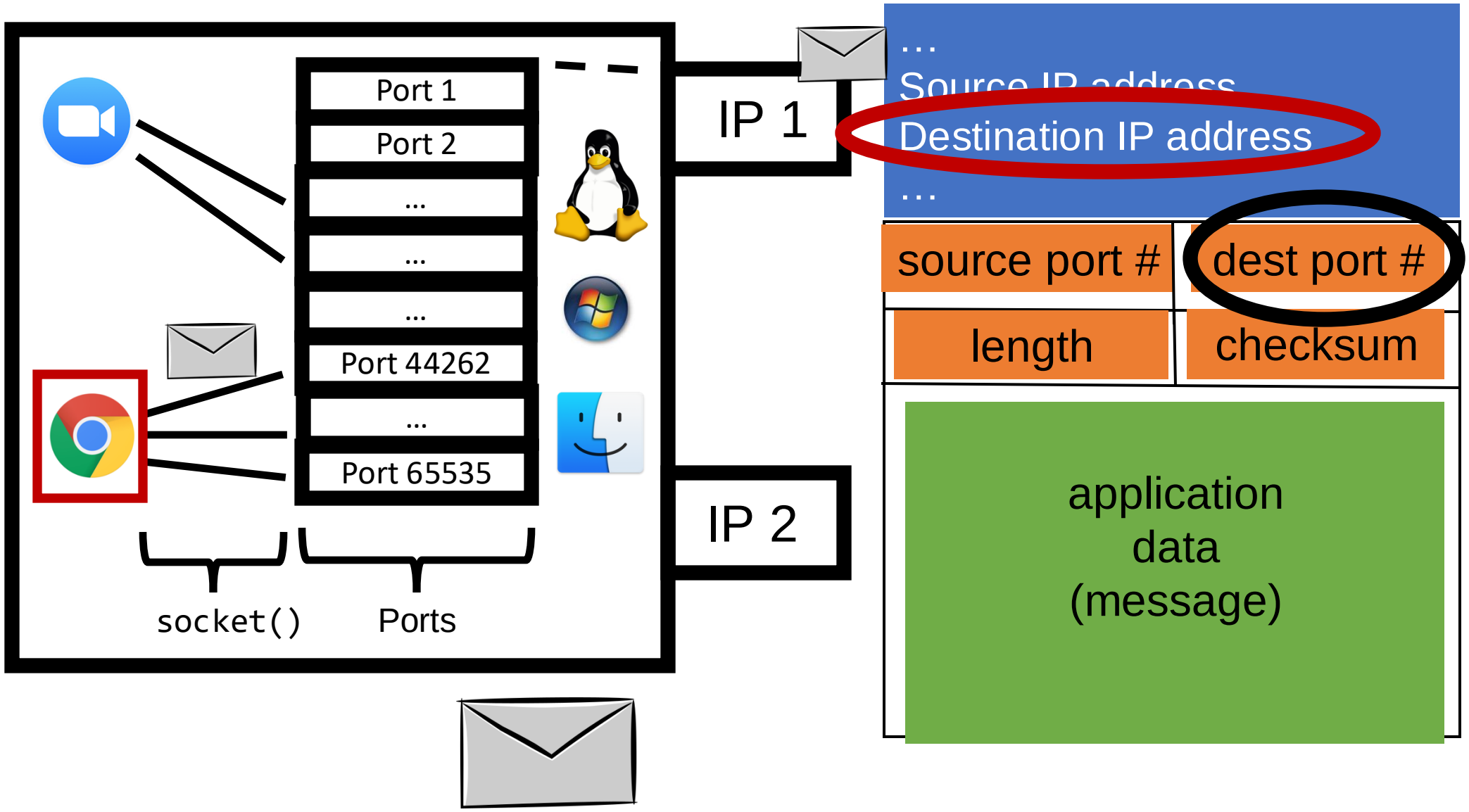
UDP segment structure



UDP segment structure



Review: UDP demultiplexing



Seeing UDP packets in action

- How to craft and send (UDP) packets?
  - It’s simpler than you think!

- `sudo tcpdump -i lo -XAvvv udp # observe packets`
- `sudo scapy # tool used to send crafted packets`
- Example:
  - `send(IP(dst="127.0.0.1")/UDP(sport=1024, dport=2048)/"hello world”, iface="lo")`
- See other fields of UDP using `UDP().fields_desc`
- Scapy can send and receive crafted packets!
  - However, it requires sudo (superuser privileges)

Error Detection in the Transport Layer

Why error detection?

- Network provides best effort service
- UDP is a simple and low overhead transport
  - Data may be corrupted along the way (e.g., 1 -> 0)
- However, simple error detection is possible!
  - Was the data I received the same data the remote machine sent?
- Error detection is a useful feature for all transport protocols including TCP

- Q: Suppose you're sending a package to a friend. How would you detect tampering with that package?

Error Detection in UDP and TCP

- Key idea: have sender compute a function over the data
  - Store the result in the packet
  - Receiver can check the function's value in received packet

- An analogy: you're sending a package of goodies and want your recipient to know if goodies were leaked along the way

- Your idea: weigh the package; stamp the weight on the package
  - Have the recipient weigh the package and cross-check the weight with the stamped value

Requirements on error detection function

- Function must be easy to compute
- Function value must change if the packet changes
  - If the packet was modified through “likely” changes, the function value must change
- Function must be easy to verify

- UDP and TCP use a class of function called a checksum
  - Very common idea: used in multiple parts of networks and computer systems

UDP & TCP's Checksum function

Sender:

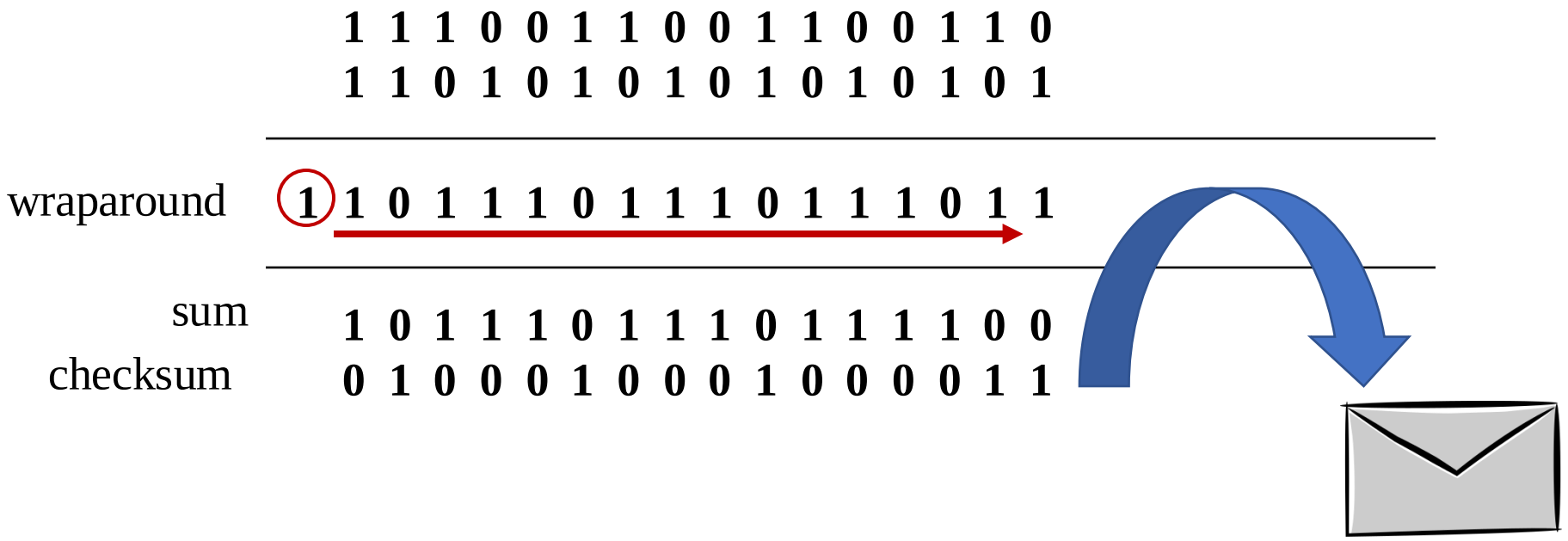
- treat segment contents as sequence of 16-bit integers
- checksum: addition (1's complement sum) of segment contents
- sender puts checksum value into UDP/TCP checksum field

Receiver:

- compute a checksum of the received segment, including the checksum in packet itself
- check if the resulting (computed) checksum is 0
- NO – an error is detected
- YES – *assume* no error

Computing 1's complement sum

- Very similar to regular (unsigned) binary addition.
- However, when adding numbers, a carryout from the most significant bit needs to be added to the result
- Example: add two 16-bit integers



From the UDP specification (RFC 768)

- Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

- The pseudo header conceptually prefixed to the UDP header contains the source address, the destination address, the protocol, and the UDP length.

Warning: Technical language ahead