Transport Layer

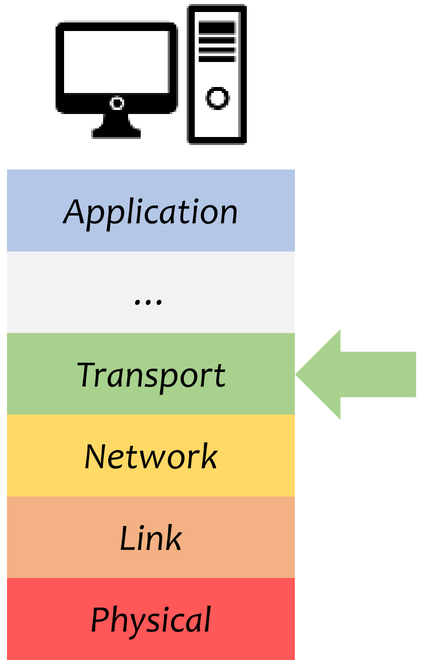
Prof. Anja Feldmann, Ph.D.

(Based on slide deck of “Network Protocol Architecture” course at TU Berlin)

Transport Layer

Facilitates logical communication between processes (or applications)

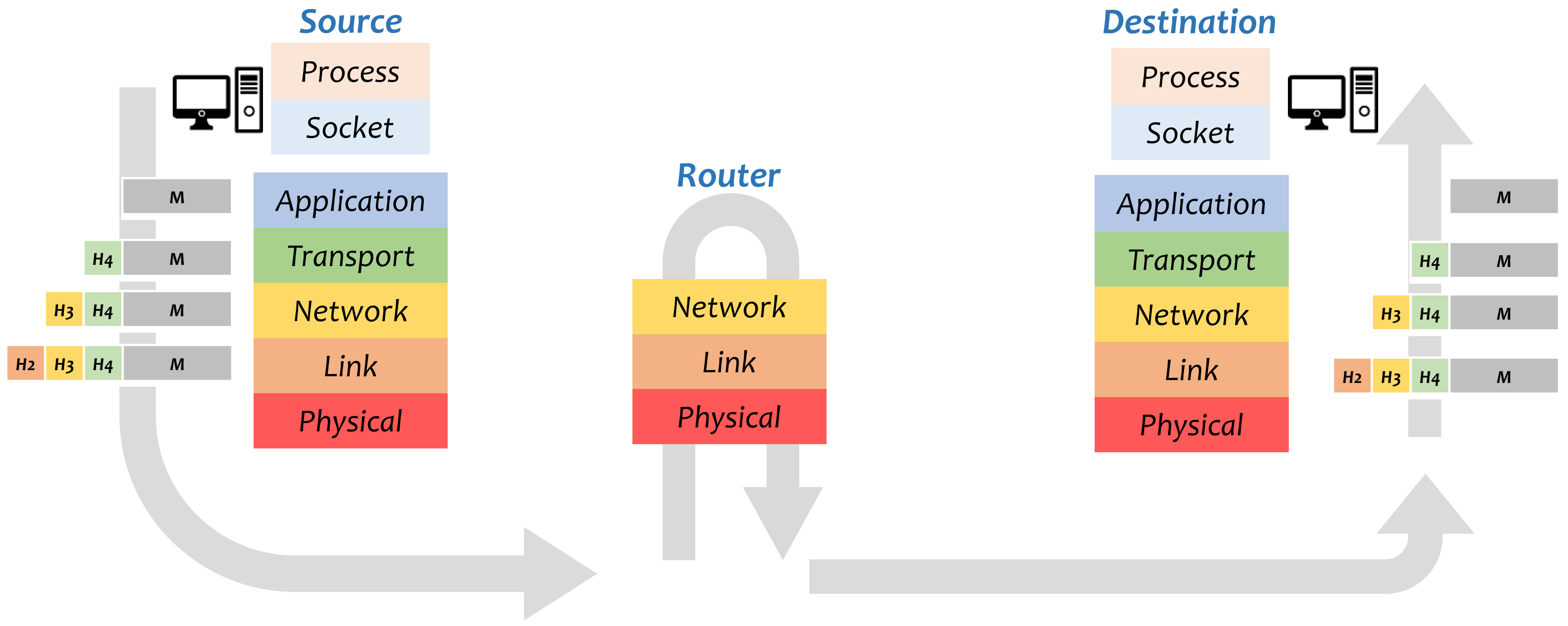
- Builds on network layer
- Uses ***ports*** for ***addressing***
- Options: ***TCP*** & ***UDP***



Transport Layer: Sockets

Socket API

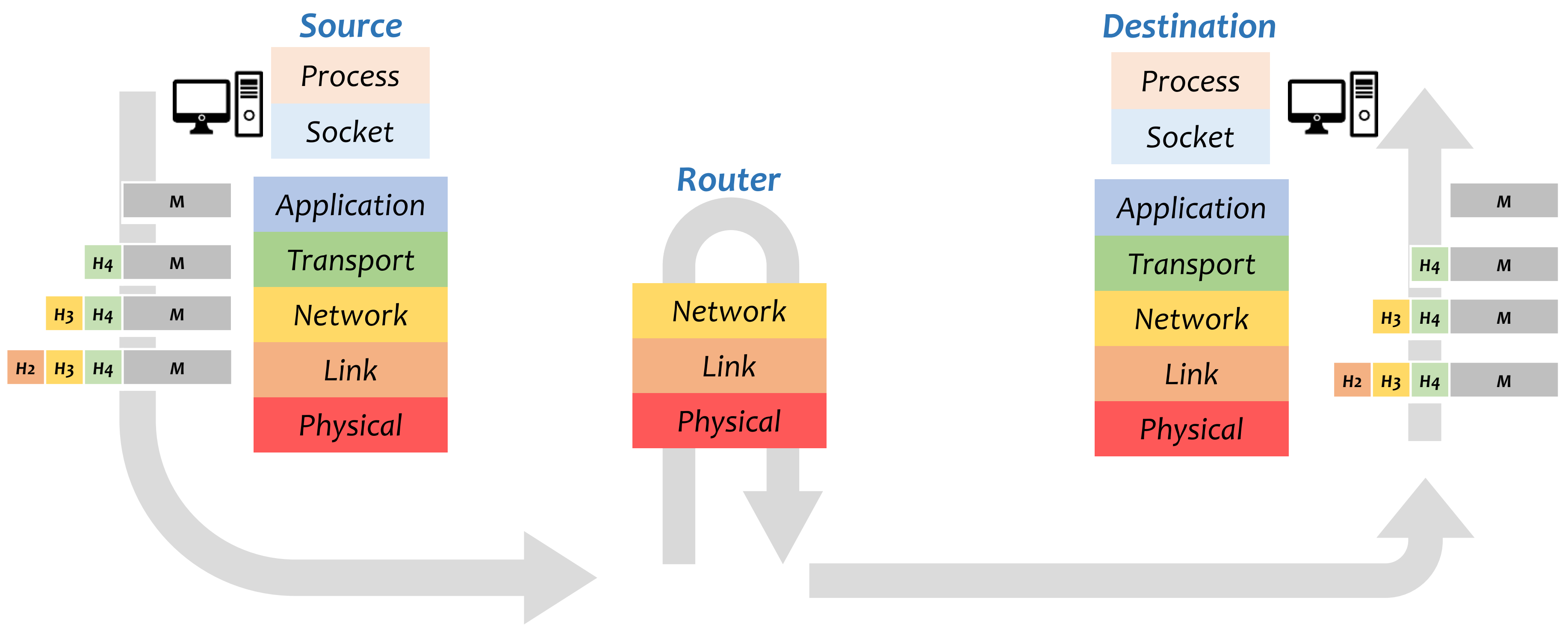
- Introduced in ***BSD4.1 UNIX***, ***1981***
- *explicitly* created, used, and released by apps.; client-server paradigm



Transport Layer: Sockets

Socket API

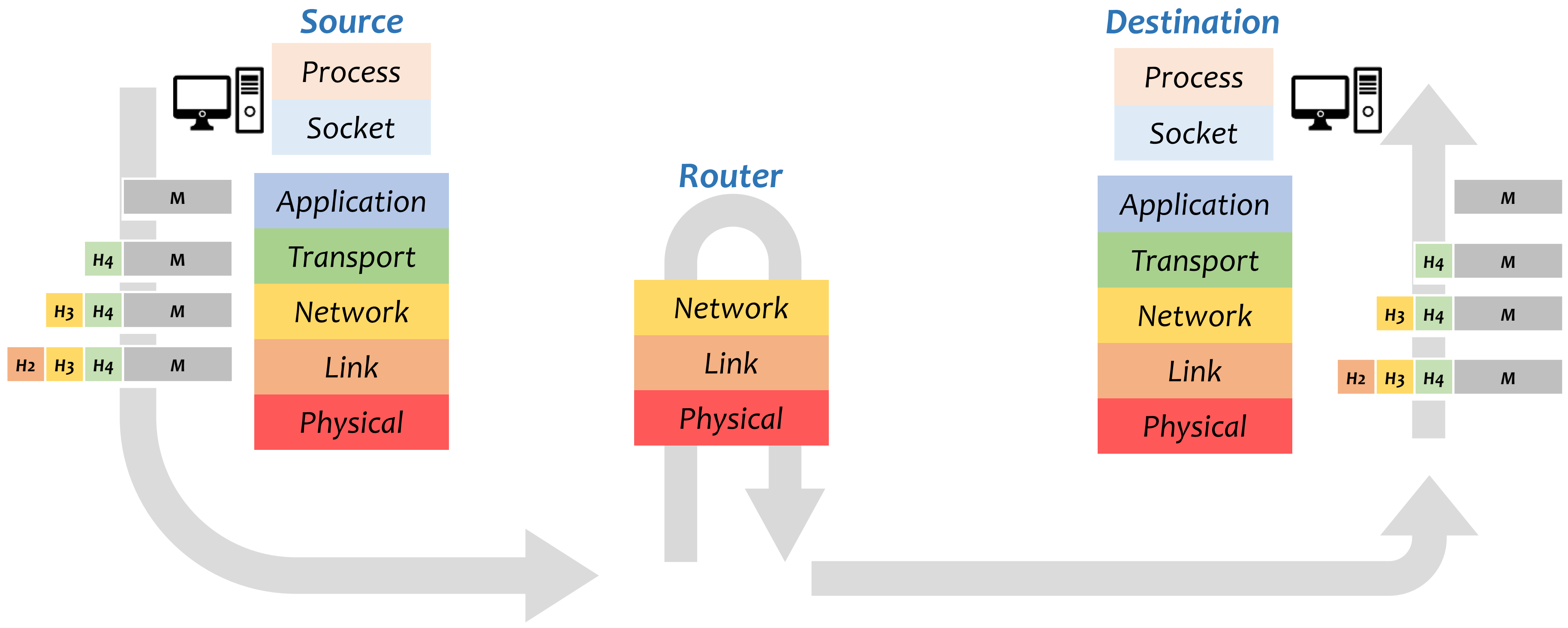
- Two types of transport service: ***Unreliable datagram*** and ***reliable, byte stream-oriented***



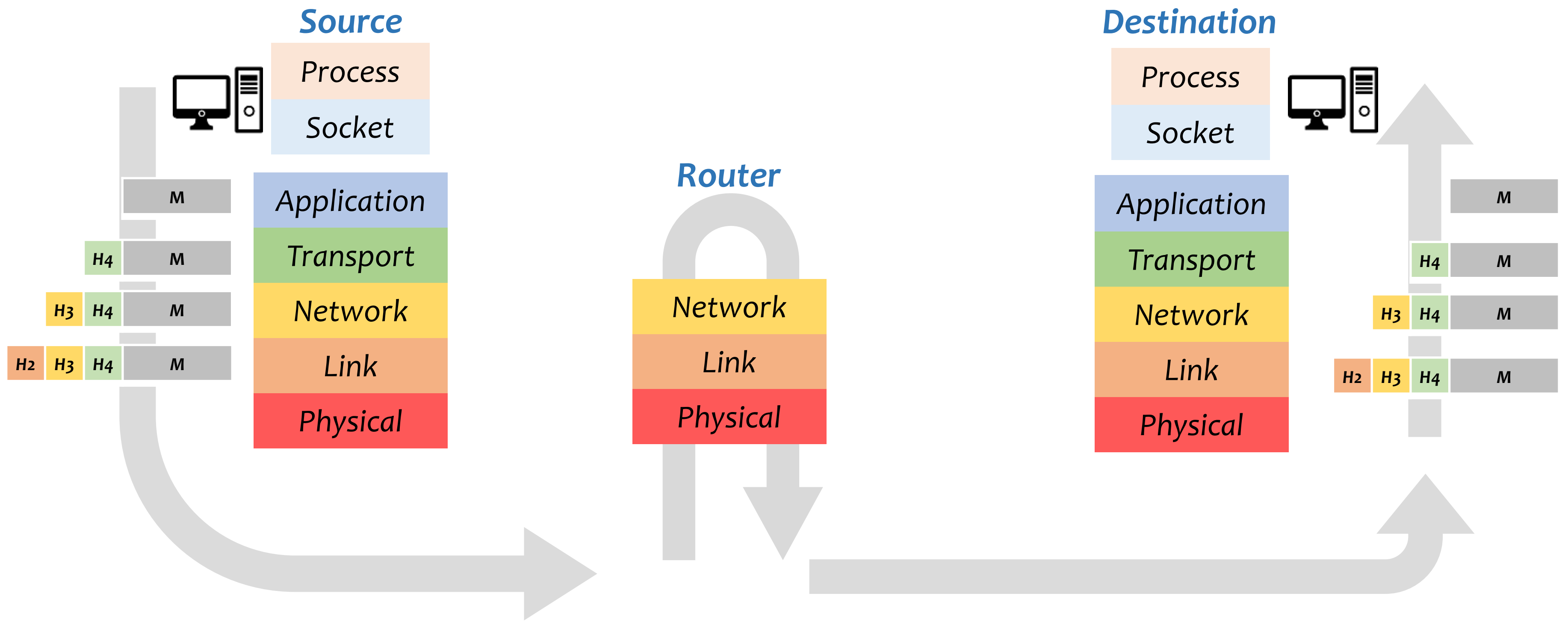
Transport Layer: Sockets

Sockets

- *A* ***door*** *between application process and end-end-transport layer protocol*



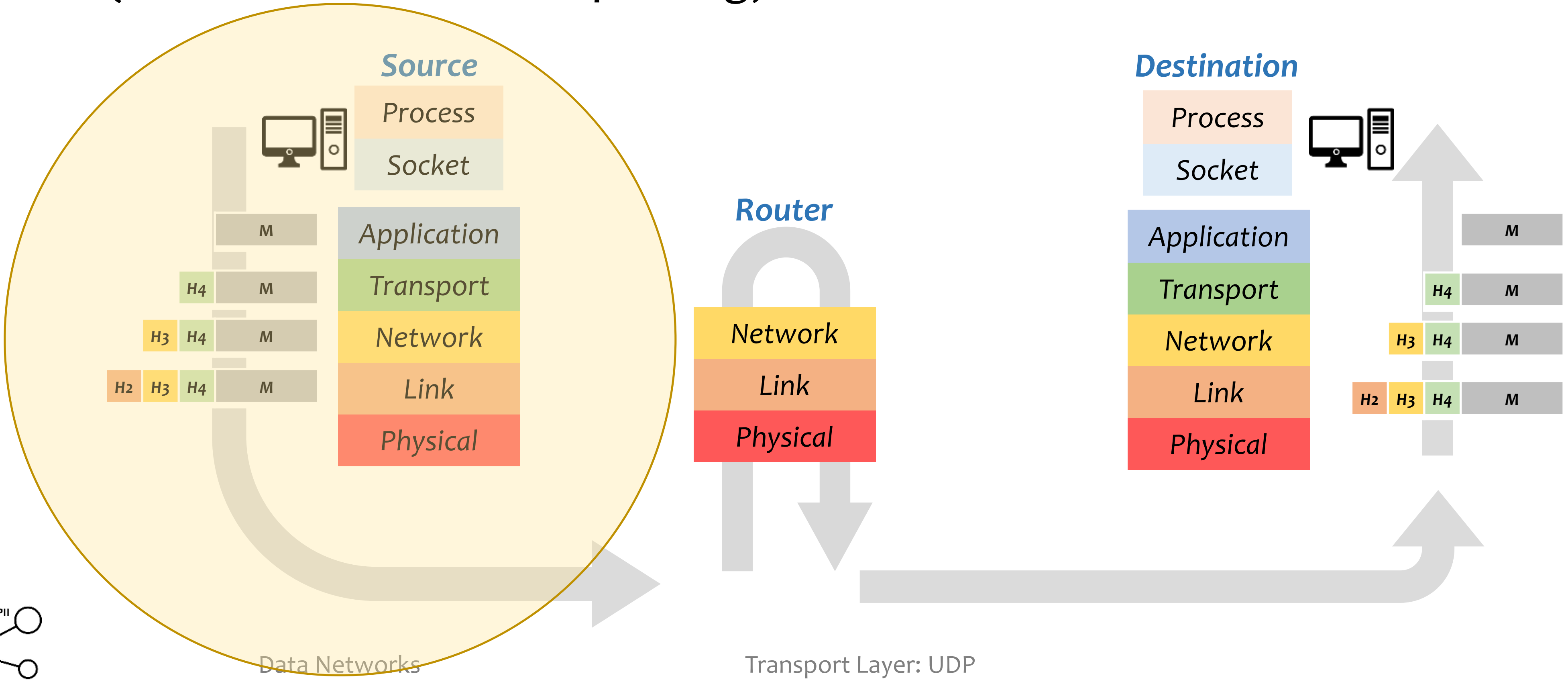
Transport Layer: Multiplexing/Demultiplexing



Transport Layer: Multiplexing/Demultiplexing

Multiplexing at source host (or sender)

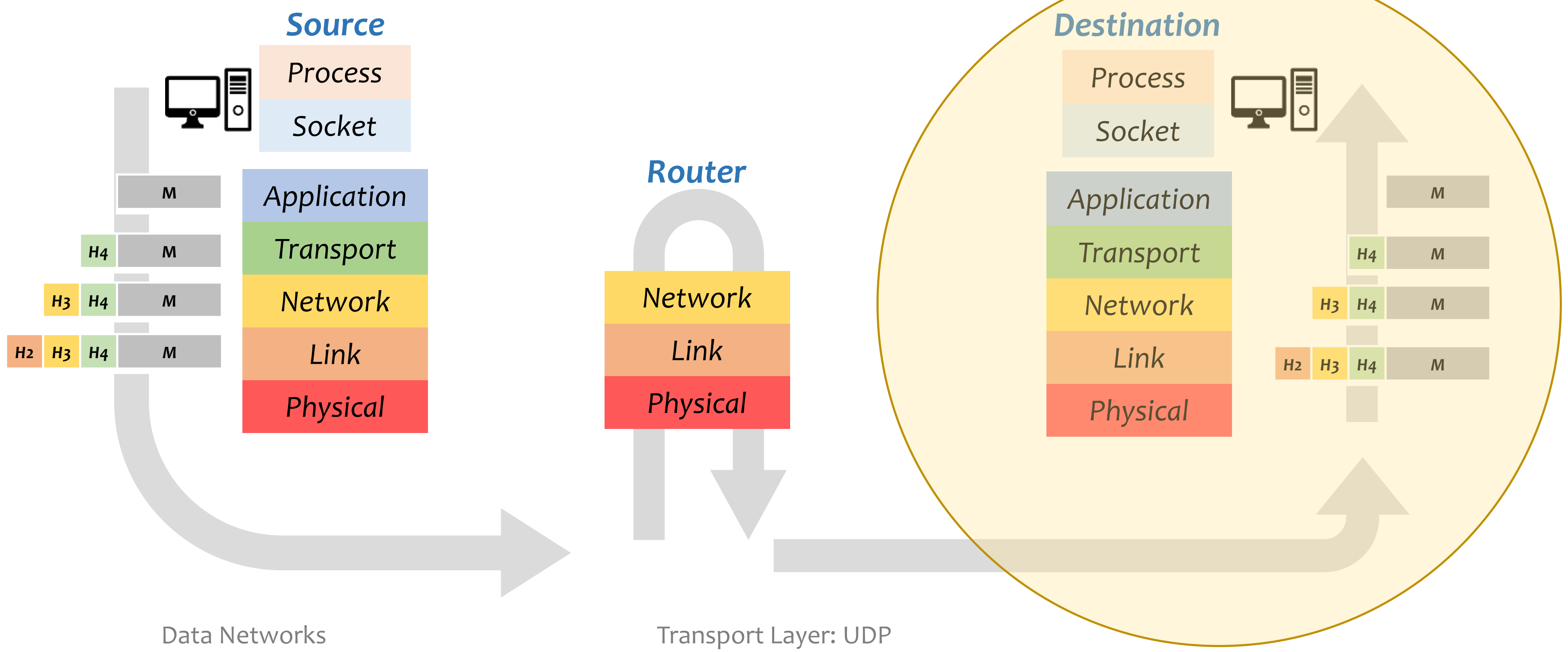
- Gathering data from multiple apps. (sockets), enveloping data with header (later used for demultiplexing)



Transport Layer: Multiplexing/Demultiplexing

Demultiplexing at destination host (or receiver)

- Delivering received *segments* to correct application (socket)



Transport Layer: Ports

Multiplexing/demultiplexing

- Based on sender, receiver ***port numbers***
- *Well-known* port numbers for specific applications
  - *80: HTTP, 443: HTTPS, 23: SMTP, 53: DNS, …*

Segments??

Segment

- *Protocol data unit (PDU)*

PDU of transport layer is a ***segment***;
PDU of network layer is a ***packet***

(Image courtesy: Bruce Mars, www.pexels.com)

Transport Layer: UDP

User Datagram Protocol (UDP)

- *“Bare bones”* Internet transport protocol
- *RFC 768*

“Best effort” service!

UDP *segments* may be

- ***Lost***
- Delivered ***out of order*** to application

Transport Layer: UDP

Connectionless

- No ***handshakes*** between UDP sender, receiver
- Each UDP segment handled ***independently*** of others

UDP Segments: ***Datagrams***

But why do we need UDP?

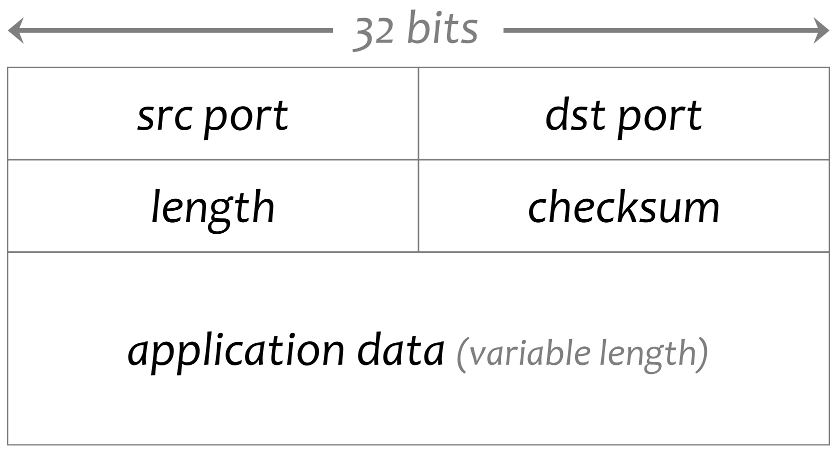
- ***No setup delay***, since there is connection establishment
- ***Simple***: No connection ***state*** at sender and receiver
- ***Small*** segment ***header***
- ***No congestion control***:
  Blast away as fast as desired

(Image courtesy: Alexander Dummer, www.pexels.com)

UDP: Segment Structure

Each user request transferred in a single datagram

- UDP has a receive buffer, but **no** sender buffer

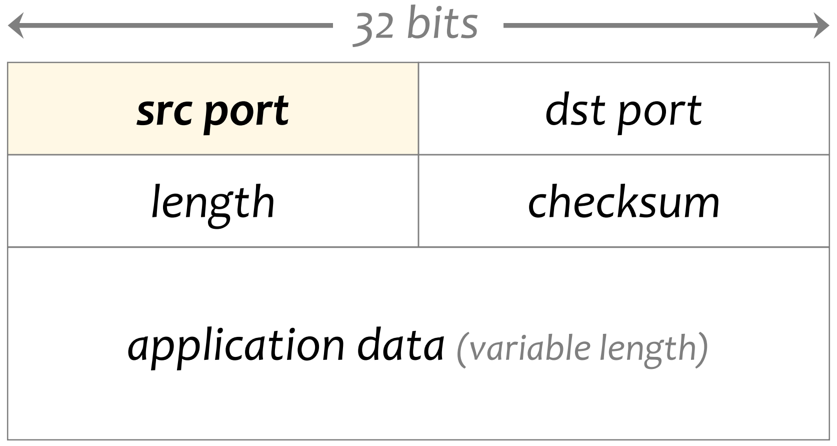


UDP: Segment Structure

Each user request transferred in a single datagram

- UDP has a receive buffer, but **no** sender buffer

Source port number (16 bits)

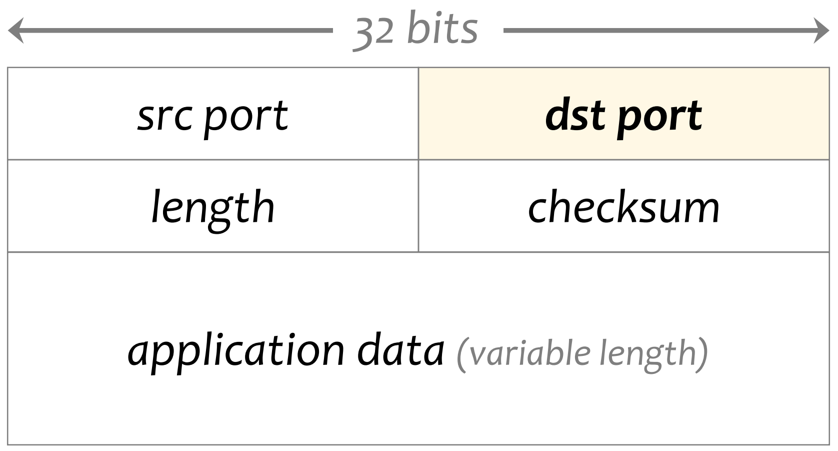


UDP: Segment Structure

Each user request transferred in a single datagram

- UDP has a receive buffer, but **no** sender buffer

Destination port number (16 bits)

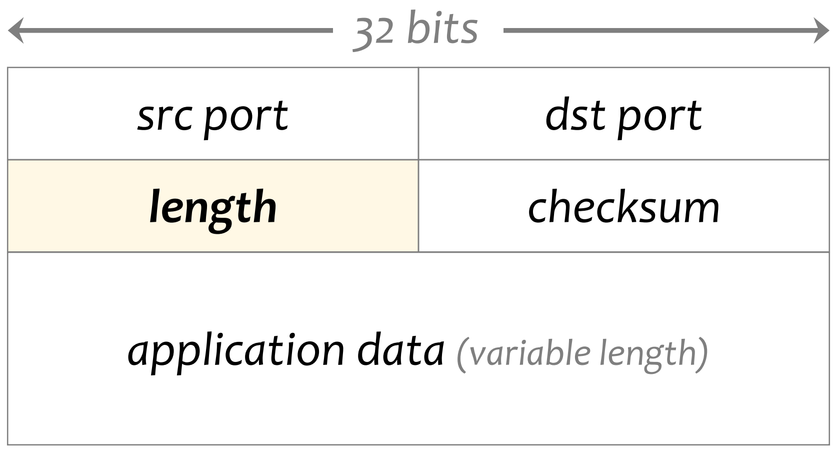


UDP: Segment Structure

Each user request transferred in a single datagram

- UDP has a receive buffer, but **no** sender buffer

Length of segment, including header (in bytes)

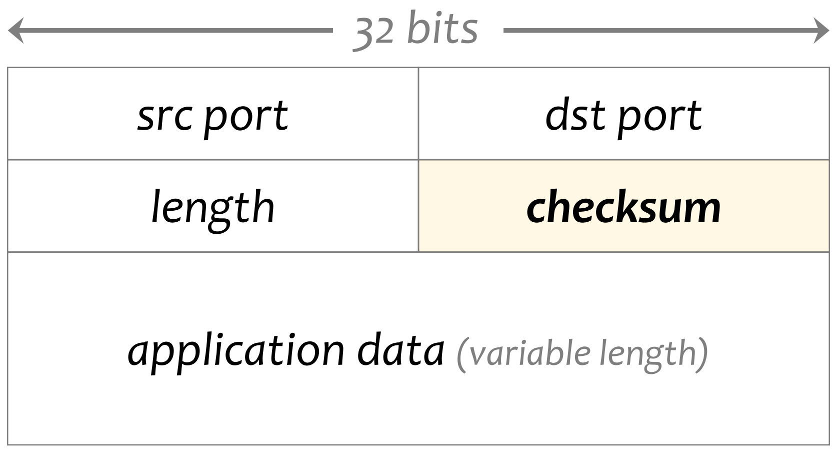


UDP: Checksum

Each user request transferred in a single datagram

- UDP has a receive buffer, but **no** sender buffer

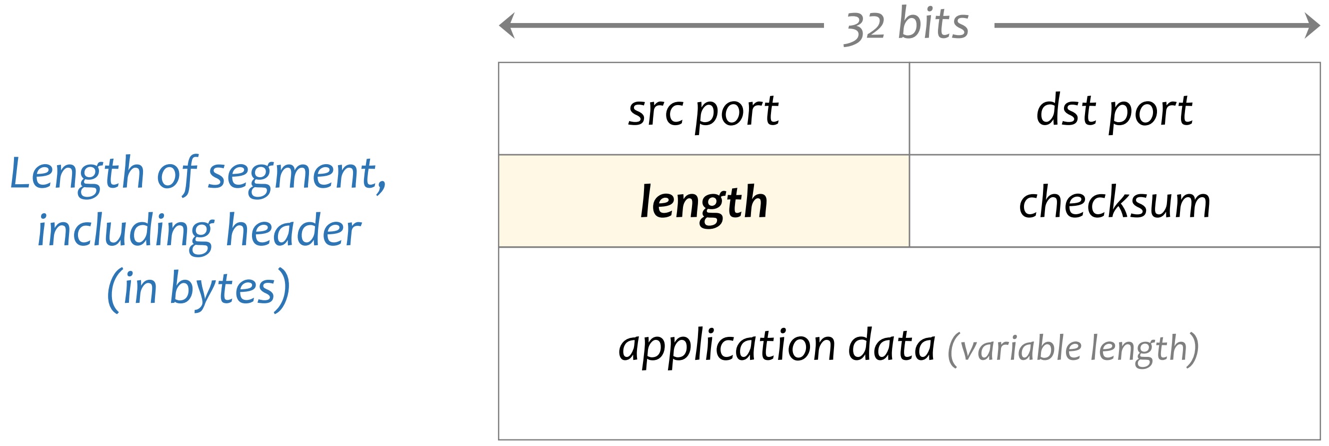
Checksum
(16 bits)



UDP: Checksum

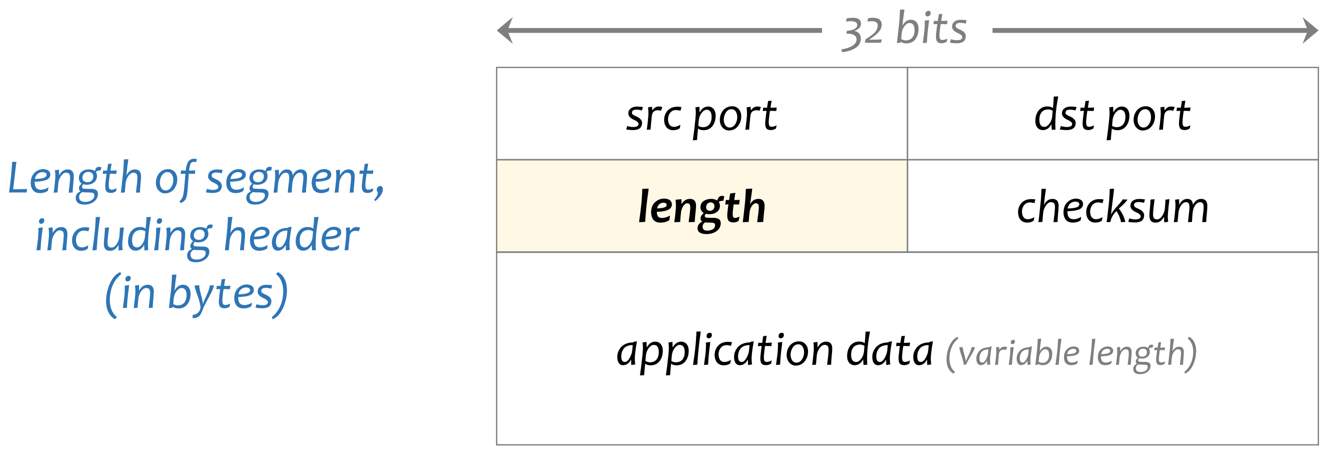
Ensures that packet has reached the correct host

- ***Ones-complement*** of *16-bit* words
- Covers *data* plus a *12-byte pseudo header*
  - *IP addresses, protocol identifier, length*


32 bits
src port
dst port
length
checksum
application data (variable length)
Length of segment, including header (in bytes)

UDP: Checksum

- Pad byte in case of an odd packet length
- *Optional:* Checksum=0 indicates no checksum
  - Should always be enabled

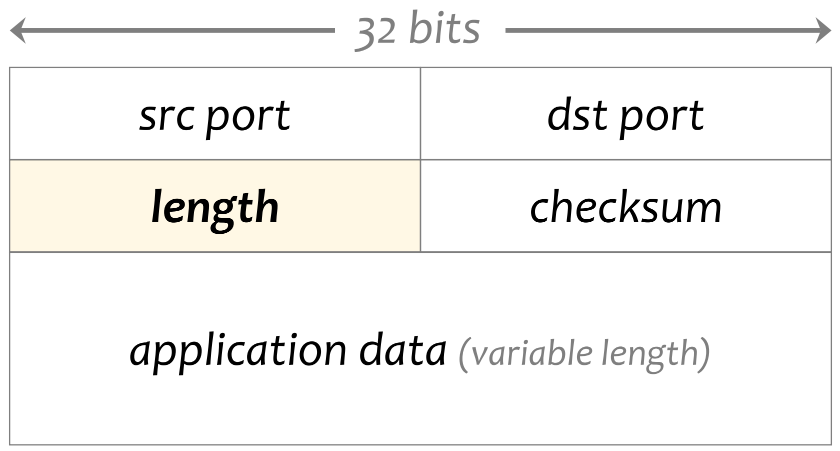

32 bits
src port
dst port
length
checksum
application data (variable length)
Length of segment, including header (in bytes)

UDP: Checksum

Ensures that packet has reached the correct host

- Receiver *has to verify* checksum

Length of segment, including header (in bytes)



That’s all folks!

- UDP
    - Connectionless and unreliable, but fast!

One common widely used UDP-based application?