CS 344: Operating Systems I

02.20: Part III: Networking I

M/W 12:00 – 1:50 PM (LINC #200)

Sanghyun Hong

sanghyun.hong@oregonstate.edu

SAIL

Secure AI Systems Lab

Notice

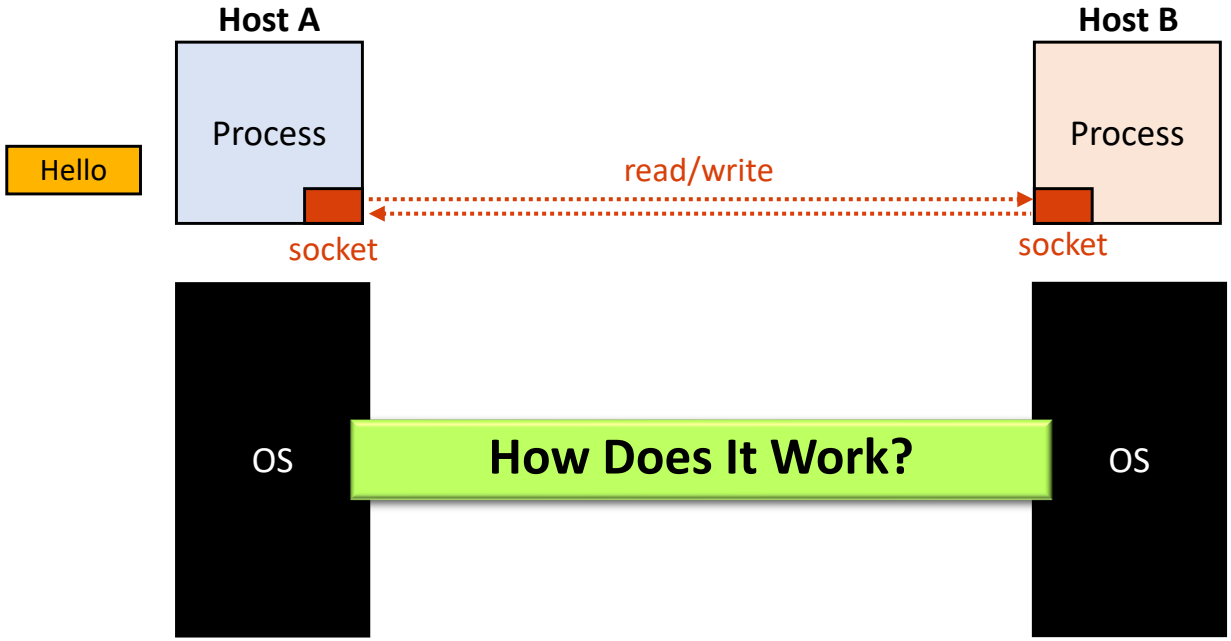
- Announcements
  - No lecture on the 27th
    - A slot for quizzes, assignments, and extra opportunities
    - SH will be on Discord
  - 2 more extra credit opportunities on Canvas
    - Build an ML classifier (+2%)
    - Multi-process data loader (+3%)

Topics for today

- Part III: Networking
  - Provide abstraction
    - OSI models
    - Packet encapsulation
  - Offer standard interface
    - RPC mechanisms (e.g., sockets)
  - Manage resources
    - Packet encapsulation in detail

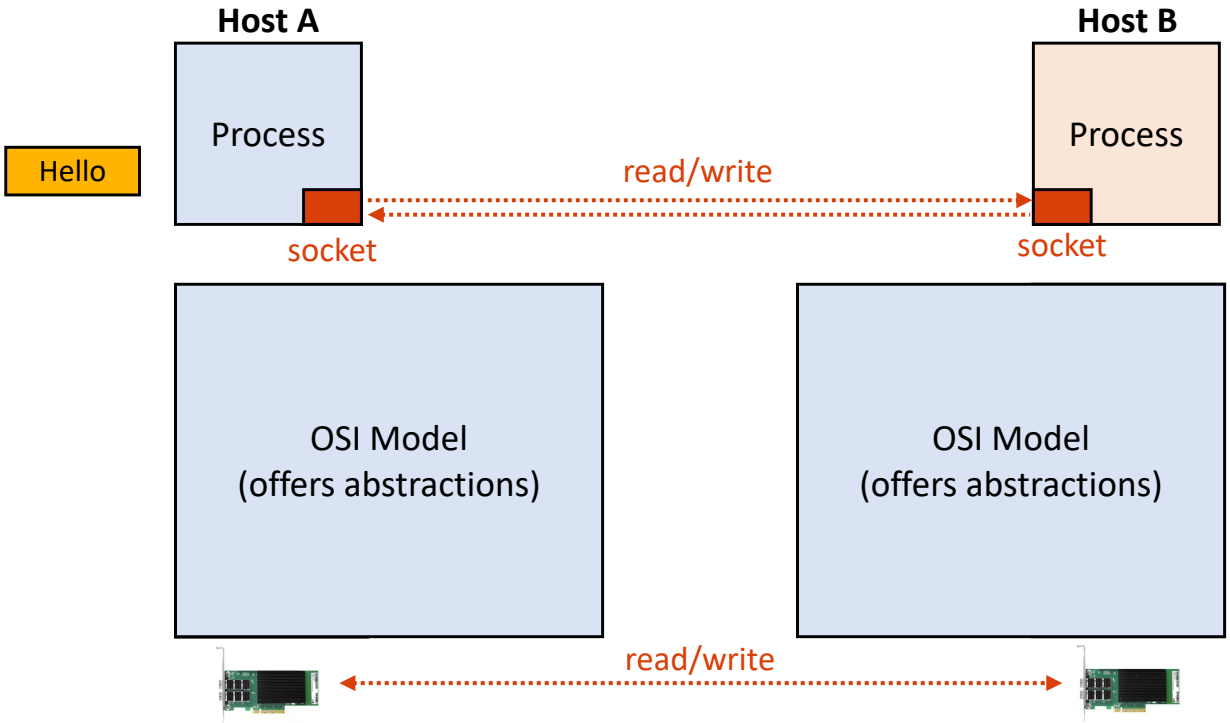
(Computer) Networking

- Networking
  - **Definition:** two or more applications on different computers (hosts) exchanging data



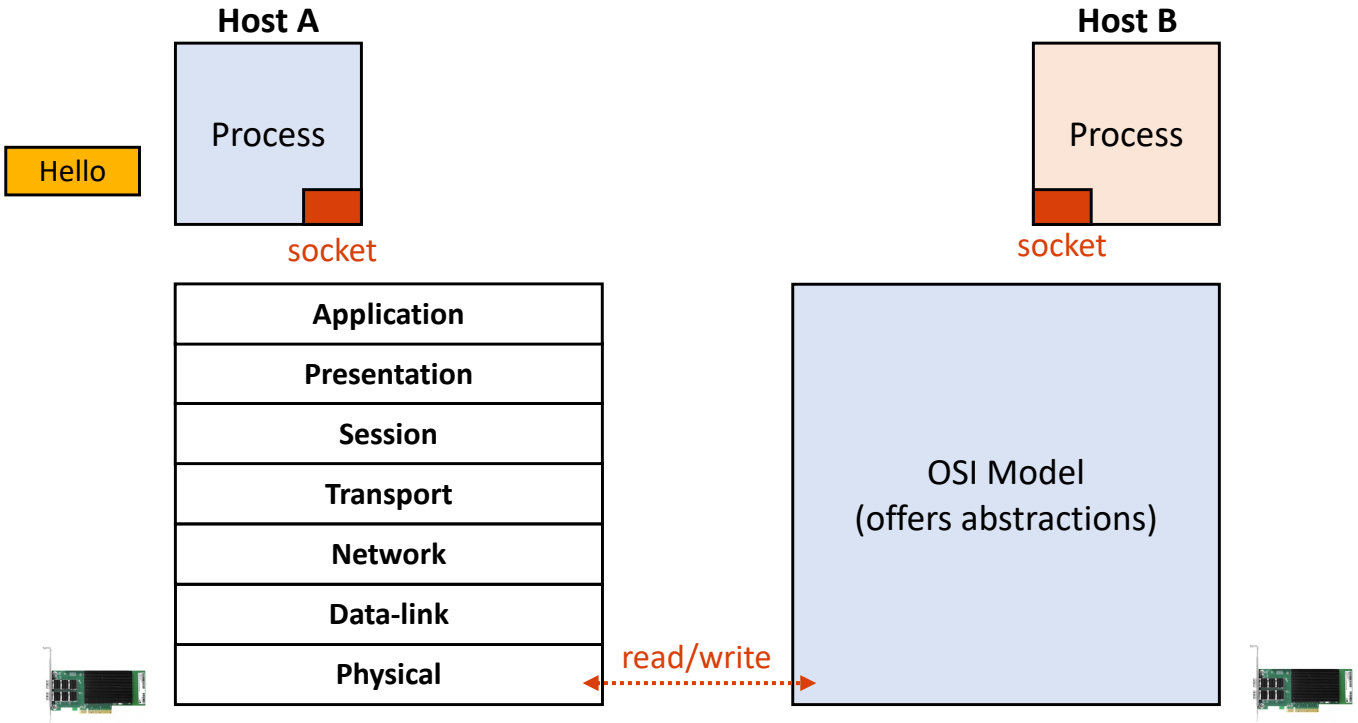
Provide abstraction

- **O**pen **I**nternet **I**nterface (**OSI**) model



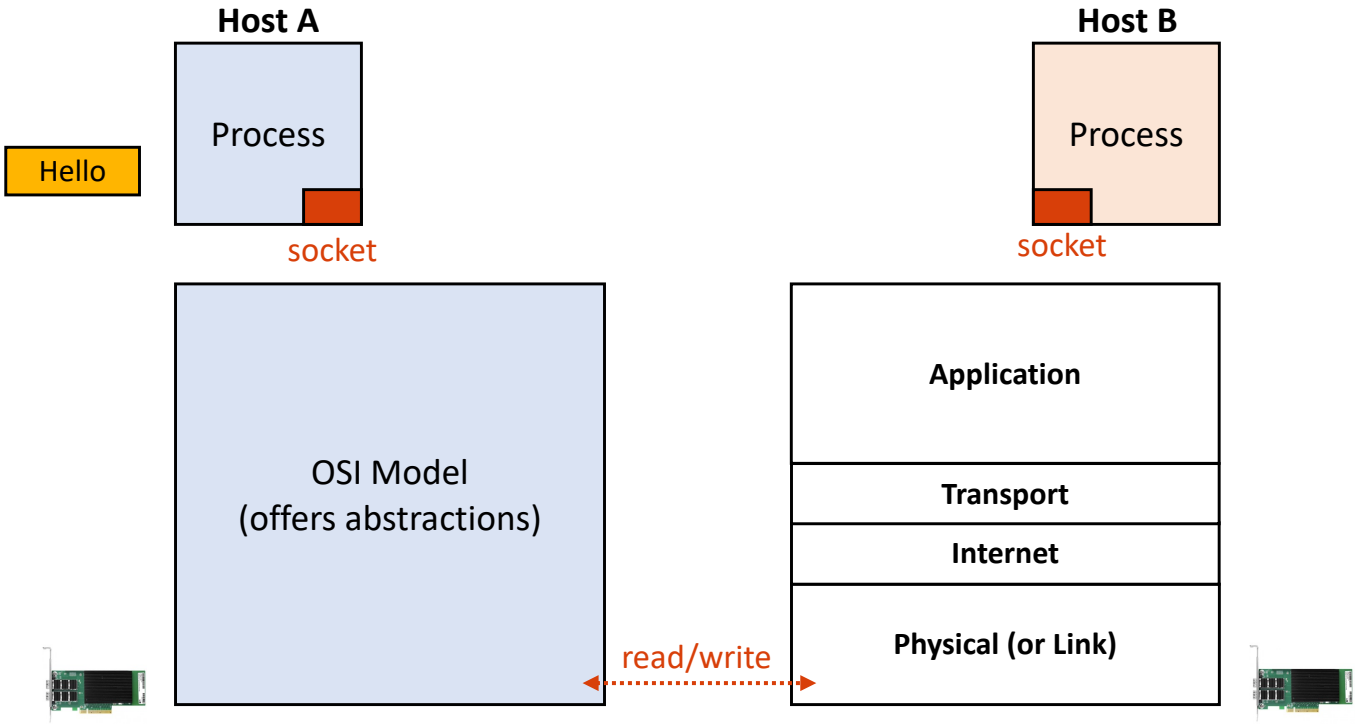
Provide abstraction: 7-layer model

- Open Internet Interface (OSI) model



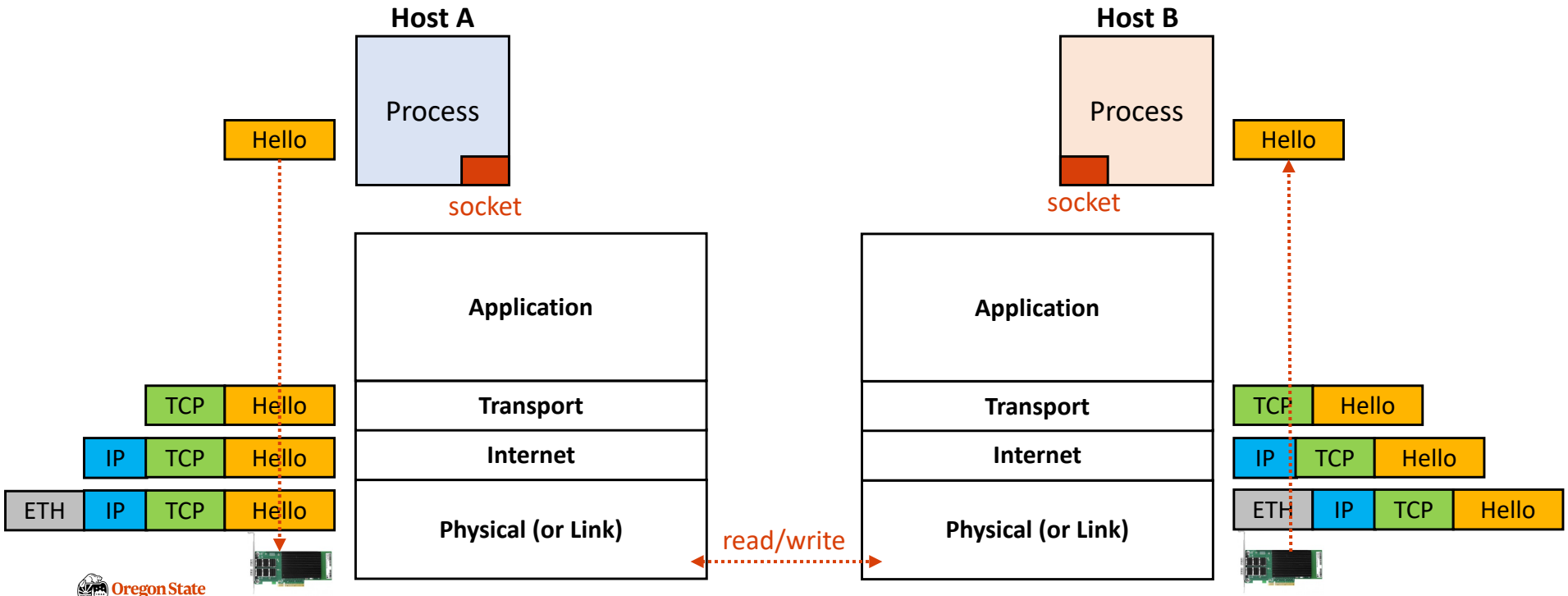
Provide abstraction: TCP/IP 4-layer model

- **O**pen **I**nternet **I**nterface (**OSI**) model



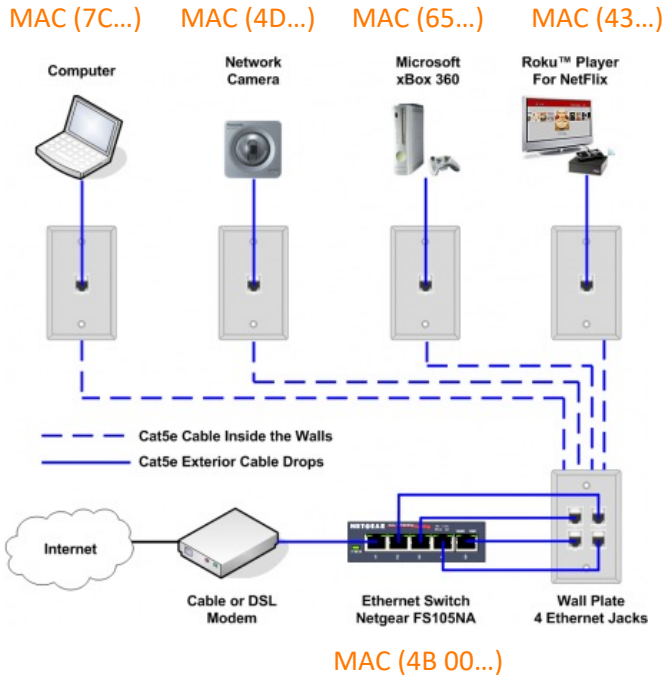
Provide abstraction: packet encapsulation

- In the TCP/IP 4-layer model



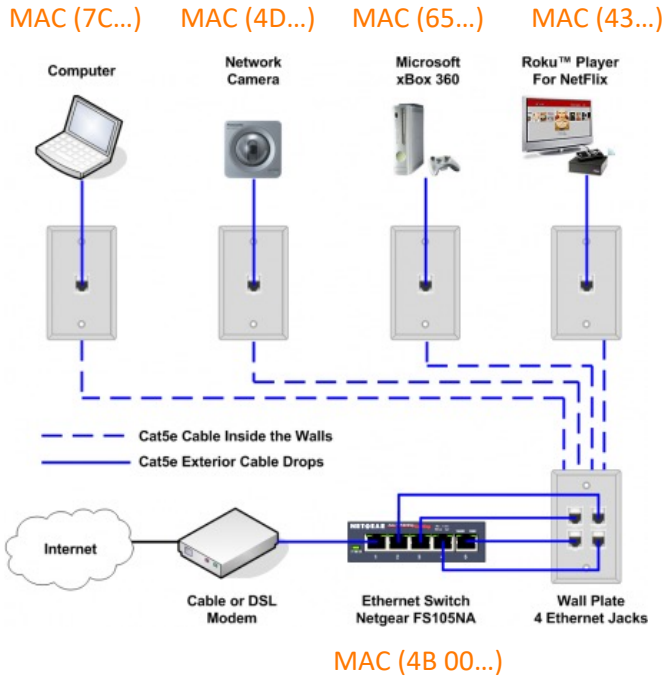
Provide abstraction: ethernet (physical layer)

- Ethernet Protocol (~80s)
  - Each network device (NIC) has 48-bit **MAC address**
  - Each NIC is connected via Ethernet **cable**



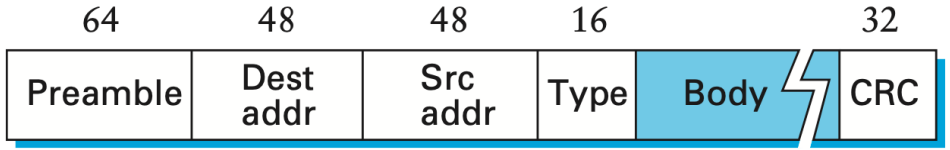
Provide abstraction: ethernet (physical layer)

- Ethernet Protocol (~80s)
  - Each network device (NIC) has 48-bit **MAC address**
  - Each NIC is connected via Ethernet **cable**
  - ETH **header** contains:
    - (64 bit) Preamble (0x111111111… or a unique data)
    - (48-bit) Destination MAC address
    - (48-bit) Source MAC address
    - (16-bit) Type
    - (up to 1500 bytes) Data
    - (32-bit) CRC for error correcting



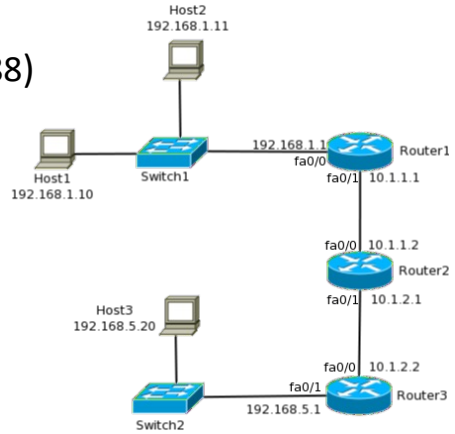
Provide abstraction: ethernet (physical layer)

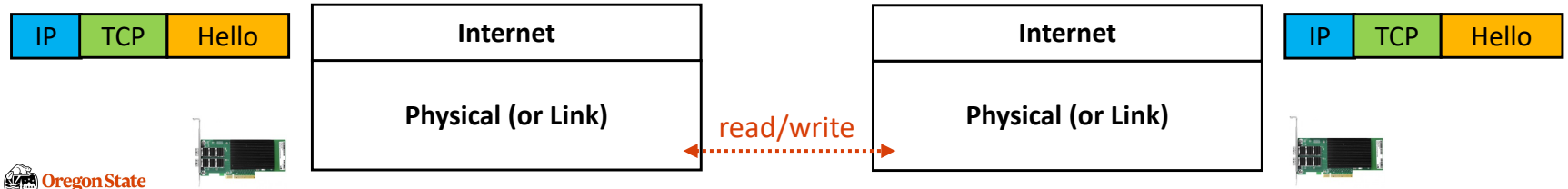
- Ethernet Protocol (~80s)
  - Each network device (NIC) has 48-bit **MAC address**
  - Each NIC is connected via Ethernet **cable**
  - ETH **header** contains:
    - (64 bit) Preamble (0x111111111… or a unique data)
    - (48-bit) Destination MAC address
    - (48-bit) Source MAC address
    - (16-bit) Type
    - (up to 1500 bytes) Data
    - (32-bit) CRC for error correcting



Provide abstraction: IP layer

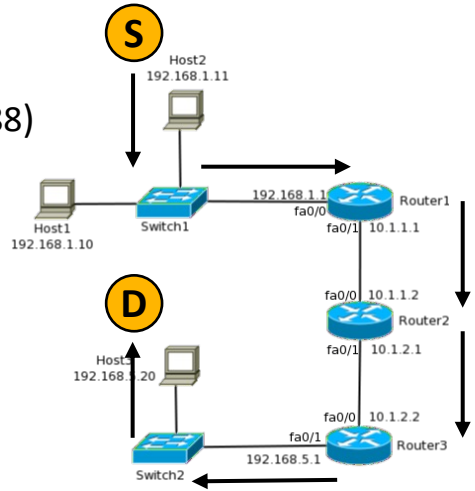
- Internet Protocol (IP)
  - IP allows us to connect multiple networks
  - Each host has a unique IP address
    - **IPv4**: 32-bit address (e.g., 147.56.28.101)
    - **IPv6**: 128-bit address (e.g., 2001:db8:3333:4444:5555:6666:7777:8888)

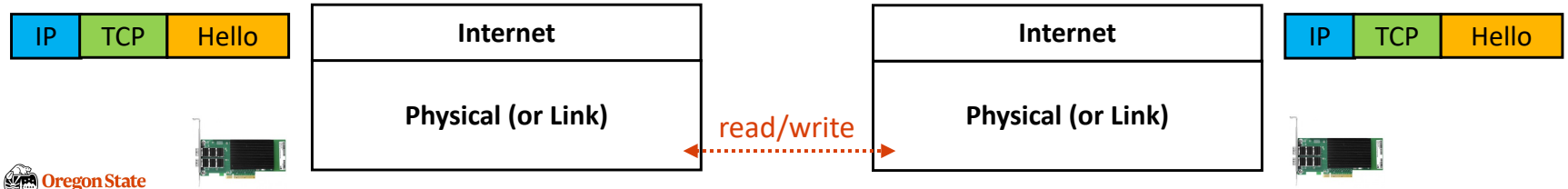




Provide abstraction: IP layer

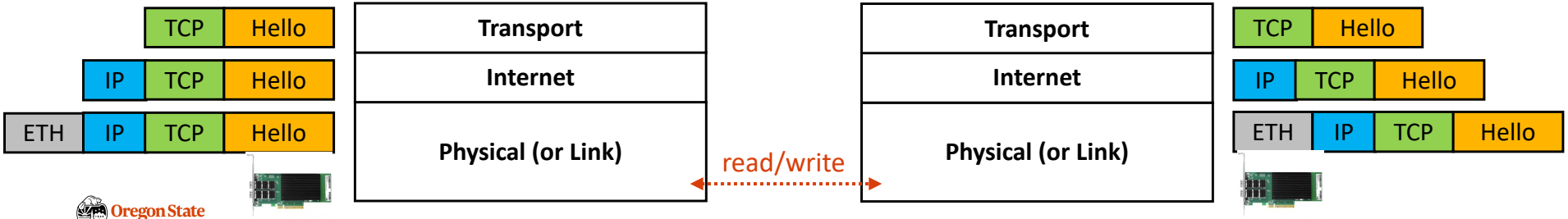
- Internet Protocol (IP)
  - IP allows us to connect multiple networks
  - Each host has a unique IP address
    - **IPv4**: 32-bit address (e.g., 147.56.28.101)
    - **IPv6**: 128-bit address (e.g., 2001:db8:3333:4444:5555:6666:7777:8888)
  - IP data (packets) is **routed** based on **destination IP**





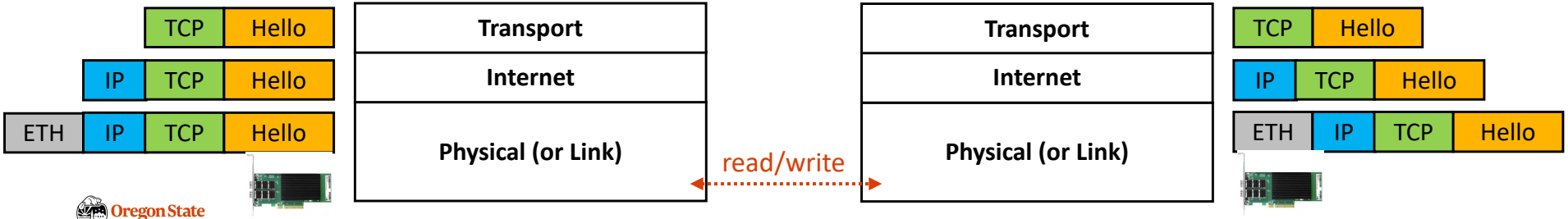
Provide abstraction: transport layer

- TCP vs UDP **Protocol**
  - **T**ransmission **C**ontrol **P**rotocol: **TCP** Packet
    - (16-bit, for each) Source and destination ports
    - (32-bit) Sequence number
    - (32-bit) Acknowledgement number
    - Others: flags, checksums, window-size, pointer, …
  - **U**ser **D**atagram **P**rotocol: **UDP** Packet
    - (16-bit, for each) Source and destination port
    - (16-bit, for each) Length and checksum



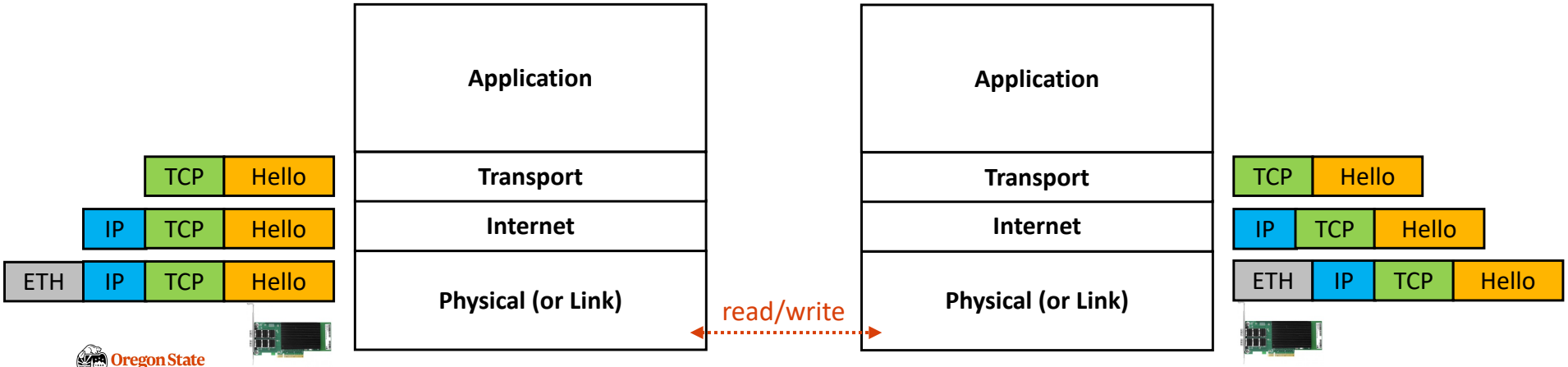
Provide abstraction: transport layer

- TCP vs UDP **Protocol**
  - TCP requires an established connection, but UDP is not (broadcast)
  - TCP can use sequences, but UDP is not
  - TCP is like a PIPE; data won't be lost, but UDP will (can lose data)
  - TCP guarantees delivery, but UDP does not
  - TCP is slower than UDP (suppose that we deliver all the packets)



Provide abstraction: application layer

- Application layer
  - Support various user-defined or OS-defined protocols (on top of TCP/UDP)
  - **TCP-based :** HTTPS, HTTP, SMTP, POP, FTP, …
  - **UDP-based:** Video streaming, conferencing, DNS, VoIP, …



Topics for today

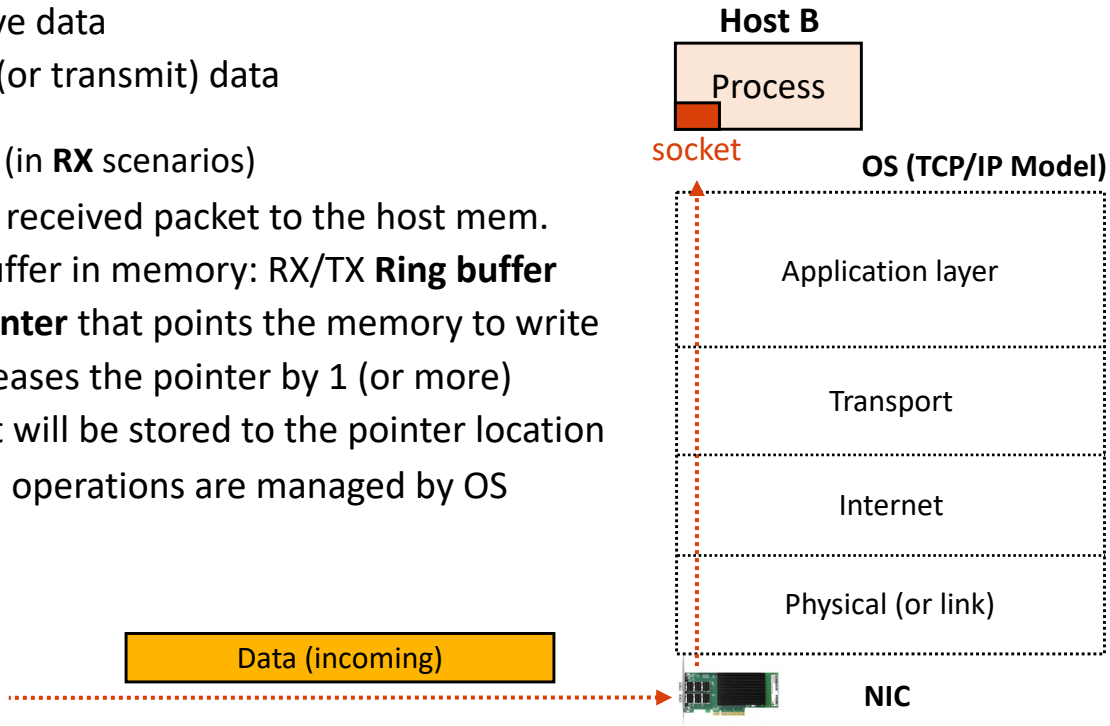
- Part III: Networking
  - Provide abstraction
    - OSI models
    - Packet encapsulation
  - Offer standard interface
    - RPC mechanisms (e.g., sockets)
  - Manage resources
    - Packet encapsulation in detail

Packet encapsulation in detail

- Dive into the encapsulation
  - Hardware: NIC (and the network driver)
  - Physical: MAC address-based communication
  - Internet: IP address-based communication
  - Transport: Define protocols, *e.g.*, TCP or UDP
  - Application: Define custom protocols, *e.g.*, HTTP, HTTPS, FTP, etc.

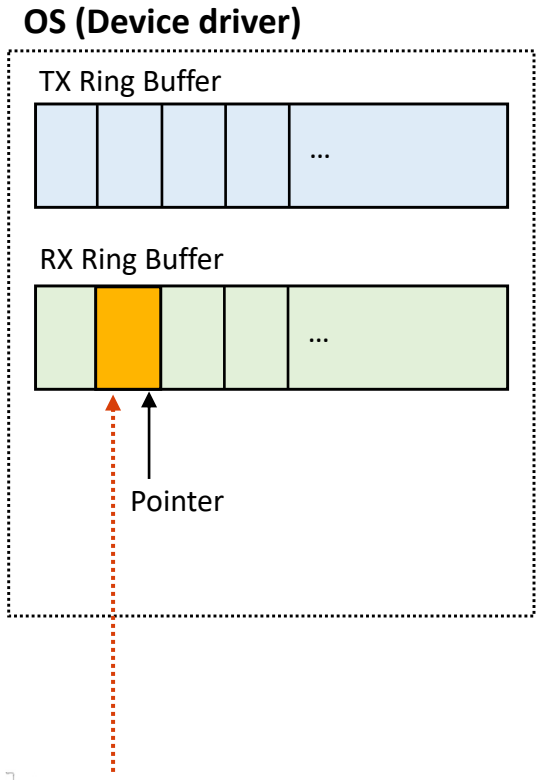
Packet encapsulation: NIC

- Network Interface Card
  - Networking terminology
    - Receive (RX) : receive data
    - Transmit (TX): send (or transmit) data
  - NIC and OS interaction (in **RX** scenarios)
    - First, NIC copies the received packet to the host mem.
      - The OS has a buffer in memory: RX/TX **Ring buffer**
      - It also has **a pointer** that points the memory to write
    - Second, the OS increases the pointer by 1 (or more)
      - The next packet will be stored to the pointer location
    - The first and second operations are managed by OS



Packet encapsulation: NIC

- **N**etwork **I**nterface **C**ard
  - Networking terminology
    - Receive (RX) : receive data
    - Transmit (TX): send (or transmit) data
  - NIC and OS interaction (in **RX** scenarios)
    - First, NIC copies the received packet to the host mem.
      - The OS has a buffer in memory: RX/TX **Ring buffer**
      - It also has **a pointer** that points the memory to write
    - Second, the OS increases the pointer by 1 (or more)
      - The next packet will be stored to the pointer location
    - The first and second operations are managed by OS

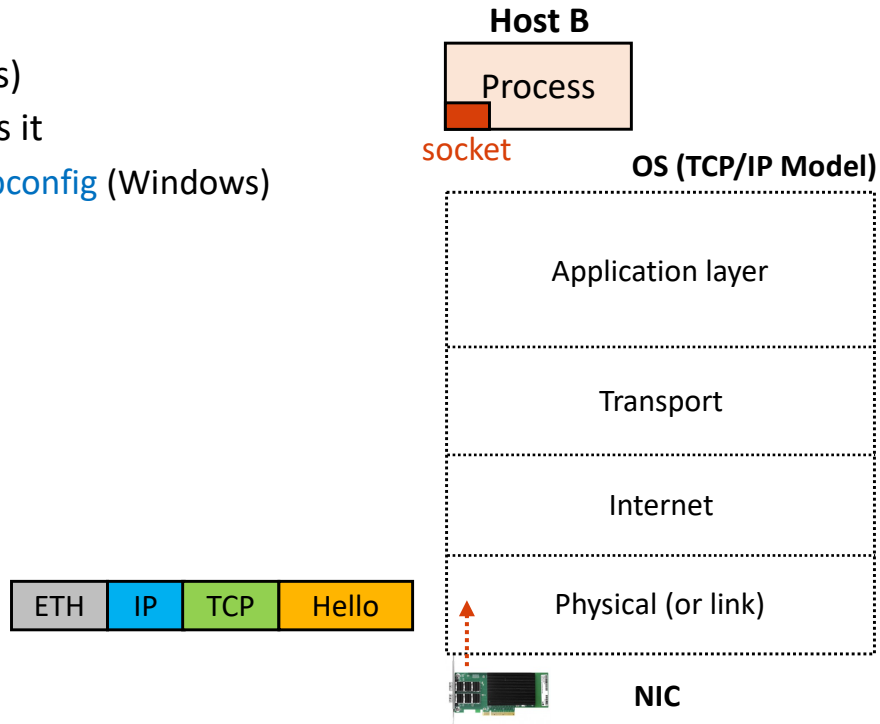


Packet encapsulation in detail

- Dive into the encapsulation
  - Hardware: NIC (and the network driver)
  - Physical: MAC address-based communication
  - Internet: IP address-based communication
  - Transport: Define protocols, *e.g.*, TCP or UDP
  - Application: Define custom protocols, *e.g.*, HTTP, HTTPS, FTP, etc.

Packet encapsulation: physical (or link) layer

- Physical (or link) layer
  - Ethernet Protocol (ETH)
    - Developed in 80s
    - Used for local area networks (LANs)
    - Use 48-bit MAC addresses; NIC has it
      - DIY check: ifconfig (Linux) and ipconfig (Windows)



Packet encapsulation: physical (or link) layer

- Physical (or link) layer
  - Ethernet Protocol (ETH)
    - Developed in 80s
    - Used for local area networks (LANs)
    - Use 48-bit MAC addresses; NIC has it
      - DIY check: ifconfig (Linux) and ipconfig (Windows)

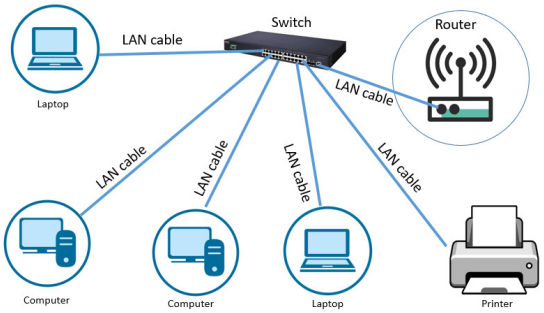
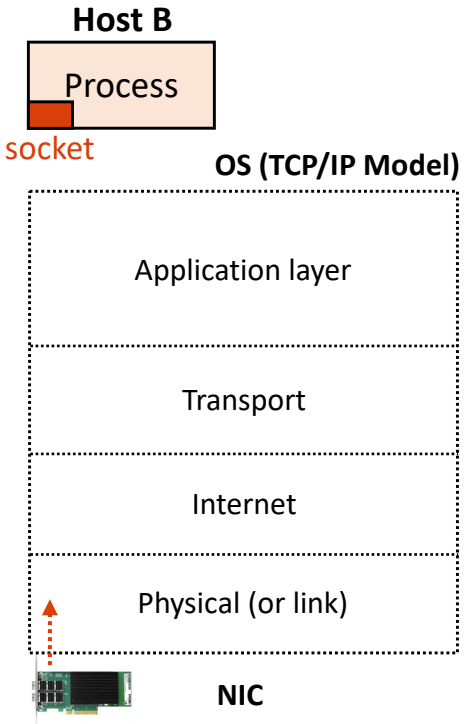
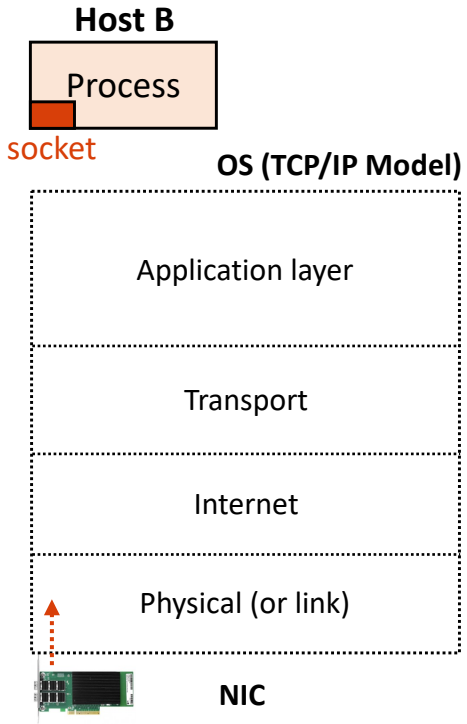


Illustration of a LAN
(Hosts are located quite nearby)



Packet encapsulation: physical (or link) layer

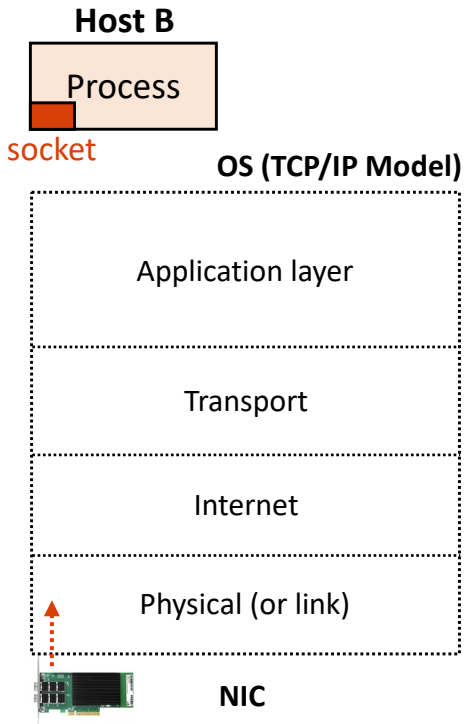
- Physical (or link) layer
  - Ethernet Protocol (ETH)
    - Developed in 80s
    - Used for local area networks (LANs)
    - Use 48-bit MAC addresses; NIC has it
      - DIY check: ifconfig (Linux) and ipconfig (Windows)
  - Ethernet packet contains:
    - Preamble (0x111111111… or a unique 64-bit data)
    - Destination MAC address (48-bit)
    - Source MAC address (48-bit)
    - Type (16-bit)
    - Data (variable ~1500 bytes)
    - CRC (32-bit; for error correcting)



Packet encapsulation: physical (or link) layer

- Physical (or link) layer
  - Ethernet Protocol (ETH)
    - Developed in 80s
    - Used for local area networks (LANs)
    - Use 48-bit MAC addresses; NIC has it
      - DIY check: ifconfig (Linux) and ipconfig (Windows)
  - Ethernet packet contains:
    - Preamble (0x111111111… or unique 64-bit data)
    - Destination MAC address (48-bit)
    - Source MAC address (48-bit)
    - Type (16-bit)
    - Data (variable ~1500 bytes)
    - CRC (32-bit; for error correcting)

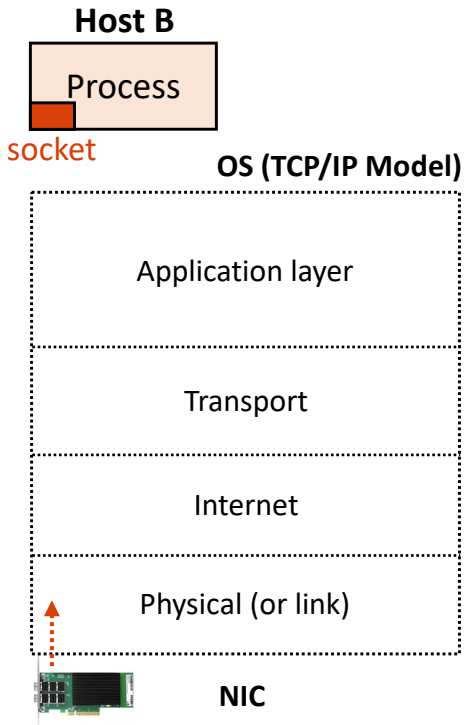
NIC Manages



Packet encapsulation: physical (or link) layer

- Physical (or link) layer
  - Ethernet Protocol (ETH)
    - Developed in 80s
    - Used for local area networks (LANs)
    - Use 48-bit MAC addresses; NIC has it
      - DIY check: ifconfig (Linux) and ipconfig (Windows)
  - Ethernet packet contains:
    - Preamble (0x111111111… or unique 64-bit data)
    - Destination MAC address (48-bit)
    - Source MAC address (48-bit)
    - Type (16-bit)
    - Data (variable ~1500 bytes)
    - CRC (32-bit; for error correcting)

OS Manages



Packet encapsulation: physical (or link) layer

- Physical (or link) layer
  - Ethernet Protocol (ETH)
    - Developed in 80s
    - Used for local area networks (LANs)
    - Use 48-bit MAC addresses; NIC has it
      - DIY check: ifconfig (Linux) and ipconfig (Win
  - Ethernet packet contains:
    - Preamble (unique 64-bit data)
    - Destination MAC address (48-bit)
    - Source MAC address (48-bit)
    - Type (16-bit)
    - Data (variable ~1500 bytes)
    - CRC (32-bit; for error correcting)

```
/*
 *      This is an Ethernet frame header.
 */

/* allow libcs like musl to deactivate this, glibc does not implement this. */
#ifndef __UAPI_DEF_ETHHDR
#define __UAPI_DEF_ETHHDR               1
#endif

#if __UAPI_DEF_ETHHDR
struct ethhdr {
        unsigned char   h_dest[ETH_ALEN];       /* destination eth addr */
        unsigned char   h_source[ETH_ALEN];     /* source ether addr    */
        __be16          h_proto;                /* packet type ID field */
} __attribute__((packed));
#endif
```

h_dest: Destination MAC address
h_source: Source MAC address
h_proto: Packet type ID field

ETH_ALEN: 6 bytes (48 bits)
(see the kernel code: include/uapi/linux/if_ether.h)

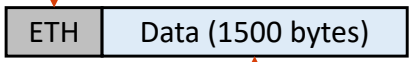
Packet encapsulation: physical (or link) layer

- Physical (or link) layer
  - Ethernet Protocol (ETH)
    - Developed in 80s
    - Used for local area networks (LANs)
    - Use 48-bit MAC addresses; NIC has it
      - DIY check: ifconfig (Linux) and ipconfig (Win
  - Ethernet packet contains:
    - Preamble (unique 64-bit data)
    - Destination MAC address (48-bit)
    - Source MAC address (48-bit)
    - Type (16-bit)
    - Data (variable ~1500 bytes)
    - CRC (32-bit; for error correcting)

```
/*
 *      This is an Ethernet frame header.
 */

/* allow libcs like musl to deactivate this, glibc does not implement this. */
#ifndef __UAPI_DEF_ETHHDR
#define __UAPI_DEF_ETHHDR               1
#endif

#if __UAPI_DEF_ETHHDR
struct ethhdr {
        unsigned char   h_dest[ETH_ALEN];       /* destination eth addr */
        unsigned char   h_source[ETH_ALEN];     /* source ether addr    */
        __be16          h_proto;                /* packet type ID field */
} __attribute__((packed));
#endif
```



ETH_ETH_DATA_LEN: 1500

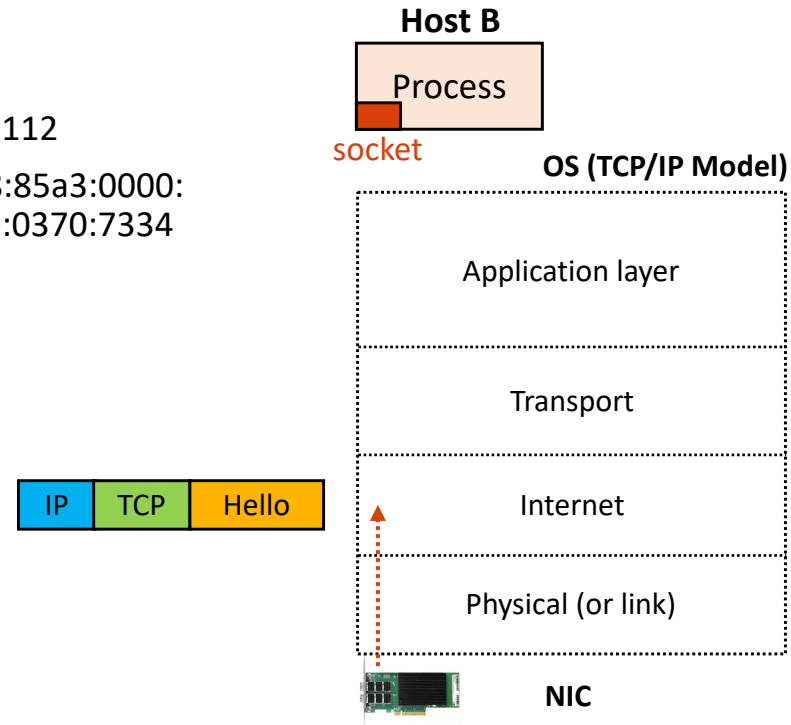
(see include/uapi/linux/if_ether.h)

Packet encapsulation in detail

- Dive into the encapsulation
  - Hardware: NIC (and the network driver)
  - Physical: MAC address-based communication
  - Internet: IP address-based communication
  - Transport: Define protocols, *e.g.*, TCP or UDP
  - Application: Define custom protocols, *e.g.*, HTTP, HTTPS, FTP, etc.

Packet encapsulation: internet layer

- Internet Layer
  - Internet Protocol (IP)
    - Use to connect multiple LANs
    - Use IP addresses to locate a host(s)
      - IPv4: 32-bit address, *e.g.*, 54.189.37.112
      - IPv6: 64-bit address, *e.g.*, 2001:0db8:85a3:0000:0000:8a2e:0370:7334



Packet encapsulation: internet layer

- Internet Layer
  - Internet Protocol (IP)
    - Use to connect multiple LANs
    - Use IP addresses to locate a host(s)
      - IPv4: 32-bit address, *e.g.*, 54.189.37.112
      - IPv6: 64-bit address, *e.g.*, 2001:0db8:85a3:0000:0000:8a2e:0370:7334

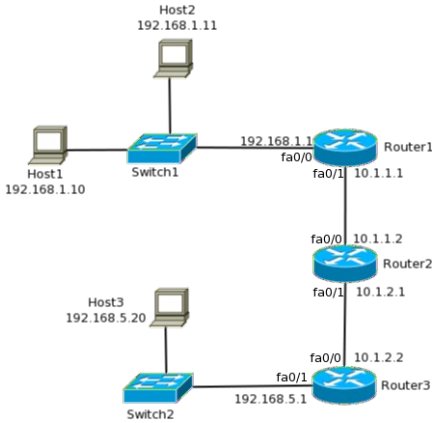
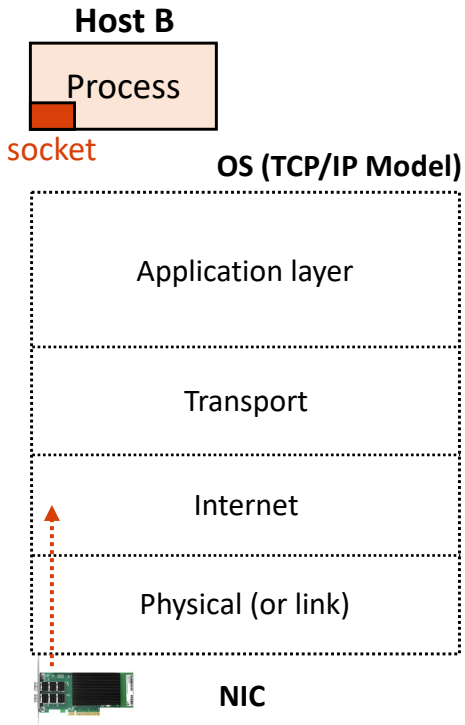
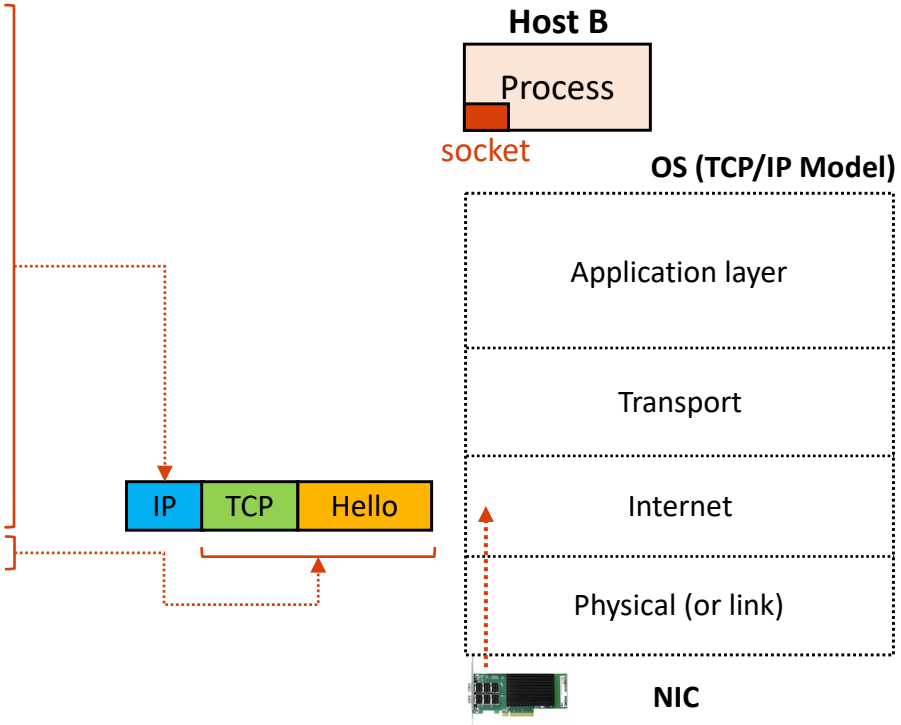


Illustration of the Internet
(Hosts are located remotely)



Packet encapsulation: internet layer

- Internet Layer
  - IP packet contains:
    - Version v4/v6 (8-bit)
    - Type of service (8-bit)
    - Total length (16-bit)
    - Identification (16-bit)
    - Frame offset (16-bit)
    - Time to live (8-bit)
    - Protocol (8-bit)
    - CRC (checksum) (16-bit)
    - Source IP address (32-bit)
    - Destination IP address (32-bit)
    - Data (1 ~ 65515 bytes)



Packet encapsulation: internet layer

- Internet Layer
  - IP packet header contains:
    - Version v4 (8-bit)
    - Type of service (8-bit)
    - Total length (16-bit)
    - Identification (16-bit)
    - Frame offset (16-bit)
    - Time to live (8-bit)
    - Protocol (8-bit)
    - CRC (checksum) (16-bit)
    - Source IP address (32-bit)
    - Destination IP address (32-bit)

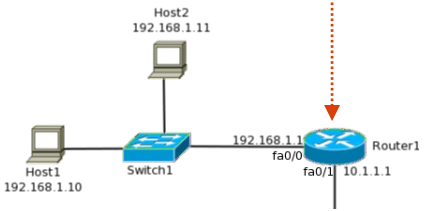
IPv4 Header
(source code)

```
struct iphdr {
#if defined(__LITTLE_ENDIAN_BITFIELD)
        __u8    ihl:4,
                version:4;
#elif defined (__BIG_ENDIAN_BITFIELD)
        __u8    version:4,
                ihl:4;
#else
#error  "Please fix <asm/byteorder.h>"
#endif
        __u8    tos;
        __be16  tot_len;
        __be16  id;
        __be16  frag_off;
        __u8    ttl;
        __u8    protocol;
        __sum16 check;
        __be32  saddr;
        __be32  daddr;
        /*The options start here. */
};
```

Packet encapsulation: internet layer

- Internet Layer
  - IP packet header contains:
    - Version v4 (8-bit)
    - Type of service (8-bit: reference)
    - Total length (16-bit)
    - Identification (16-bit)
    - Frame offset (16-bit)
    - Time to live (8-bit)
    - Protocol (8-bit)
    - CRC (checksum) (16-bit)
    - Source IP address (32-bit)
    - Destination IP address (32-bit)

Priority of IP Packet
Important at the router
(CS 372 Topic)



IPv4 Header

```
struct iphdr {
#if defined(__LITTLE_ENDIAN_BITFIELD)
        __u8    ihl:4,
                version:4;
#elif defined (__BIG_ENDIAN_BITFIELD)
        __u8    version:4,
                ihl:4;
#else
#error  "Please fix <asm/byteorder.h>"
#endif
        __u8    tos;
        __be16  tot_len;
        __be16  id;
        __be16  frag_off;
        __u8    ttl;
        __u8    protocol;
        __sum16 check;
        __be32  saddr;
        __be32  daddr;
        /*The options start here. */
};
```

Packet encapsulation: internet layer

- Internet Layer
  - IP packet header contains:
    - Version v4 (8-bit)
    - Type of service (8-bit: reference)
    - Total length (16-bit)
    - Identification (16-bit)
    - Frame offset (16-bit)
    - Time to live (8-bit)
    - Protocol (8-bit)
    - CRC (checksum) (16-bit)
    - Source IP address (32-bit)
    - Destination IP address (32-bit)

Total packet length
Min : 21 bytes
Max: 65535 bytes

(20 bytes header +
1 ~ 65515 bytes data)

IPv4 Header

```
struct iphdr {
#if defined(__LITTLE_ENDIAN_BITFIELD)
        __u8    ihl:4,
                version:4;
#elif defined (__BIG_ENDIAN_BITFIELD)
        __u8    version:4,
                ihl:4;
#else
#error  "Please fix <asm/byteorder.h>"
#endif
        __u8    tos;
        __be16  tot_len;
        __be16  id;
        __be16  frag_off;
        __u8    ttl;
        __u8    protocol;
        __sum16 check;
        __be32  saddr;
        __be32  daddr;
        /*The options start here. */
};
```

Packet encapsulation: internet layer

- Internet Layer
  - IP packet header contains:
    - Version v4 (8-bit)
    - Type of service (8-bit: reference)
    - Total length (16-bit)
    - Identification (16-bit)
    - Frame offset (16-bit)
    - Time to live (8-bit)
    - Protocol (8-bit)
    - CRC (checksum) (16-bit)
    - Source IP address (32-bit)
    - Destination IP address (32-bit)

Fragmentation

Oftentimes, we send multiple IP packets that sum up as a whole (ex. the entire data 4000 bytes, we send 4 1000-byte packets)

For example, 3rd IP packet's
ID: 3
Offset: 2000

IPv4 Header

```
struct iphdr {
#if defined(__LITTLE_ENDIAN_BITFIELD)
        __u8    ihl:4,
                version:4;
#elif defined (__BIG_ENDIAN_BITFIELD)
        __u8    version:4,
                ihl:4;
#else
#error  "Please fix <asm/byteorder.h>"
#endif
        __u8    tos;
        __be16  tot_len;
        __be16  id;
        __be16  frag_off;
        __u8    ttl;
        __u8    protocol;
        __sum16 check;
        __be32  saddr;
        __be32  daddr;
        /*The options start here. */
};
```

Packet encapsulation: internet layer

- Internet Layer
  - IP packet header contains:
    - Version v4 (8-bit)
    - Type of service (8-bit: reference)
    - Total length (16-bit)
    - Identification (16-bit)
    - Frame offset (16-bit)
    - Time to live (8-bit)
    - Protocol (8-bit)
    - CRC (checksum) (16-bit)
    - Source IP address (32-bit)
    - Destination IP address (32-bit)

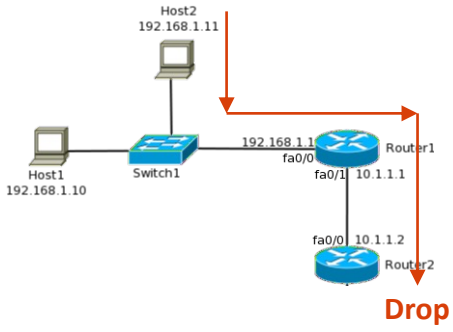
Time-to-live (TTL)
The number of routers a packet can maximally pass (# of "hops")

Reason: drop packets that live too long in a network infrastructure

Example: on the right, setting TTL to 2 -> a packet drops at router 2

IPv4 Header

```
struct iphdr {
#if defined(__LITTLE_ENDIAN_BITFIELD)
        __u8    ihl:4,
                version:4;
#elif defined (__BIG_ENDIAN_BITFIELD)
        __u8    version:4,
                ihl:4;
#else
#error  "Please fix <asm/byteorder.h>"
#endif
        __u8    tos;
        __be16  tot_len;
        __be16  id;
        __be16  frag_off;
        __u8    ttl;
        __u8    protocol;
        __sum16 check;
        __be32  saddr;
        __be32  daddr;
        /*The options start here. */
};
```



Packet encapsulation: internet layer

- Internet Layer
  - IP packet header contains:
    - Version v4 (8-bit)
    - Type of service (8-bit: reference)
    - Total length (16-bit)
    - Identification (16-bit)
    - Frame offset (16-bit)
    - Time to live (8-bit)
    - Protocol (8-bit)
    - CRC (checksum) (16-bit)
    - Source IP address (32-bit)
    - Destination IP address (32-bit)

IP Protocols

Examples:

TCP	6
UDP	7
ETHERNET	143

(see the list of IP protocols: link)

IPv4 Header

```
struct iphdr {
#if defined(__LITTLE_ENDIAN_BITFIELD)
        __u8    ihl:4,
                version:4;
#elif defined (__BIG_ENDIAN_BITFIELD)
        __u8    version:4,
                ihl:4;
#else
#error  "Please fix <asm/byteorder.h>"
#endif
        __u8    tos;
        __be16  tot_len;
        __be16  id;
        __be16  frag_off;
        __u8    ttl;
        __u8    protocol;
        __sum16 check;
        __be32  saddr;
        __be32  daddr;
        /*The options start here. */
};
```

Packet encapsulation: internet layer

- Internet Layer
  - IP packet header contains:
    - Version v4 (8-bit)
    - Type of service (8-bit: reference)
    - Total length (16-bit)
    - Identification (16-bit)
    - Frame offset (16-bit)
    - Time to live (8-bit)
    - Protocol (8-bit)
    - CRC (checksum) (16-bit)
    - Source IP address (32-bit)
    - Destination IP address (32-bit)

IPv4 Header

```
struct iphdr {
#if defined(__LITTLE_ENDIAN_BITFIELD)
        __u8    ihl:4,
                version:4;
#elif defined (__BIG_ENDIAN_BITFIELD)
        __u8    version:4,
                ihl:4;
#else
#error  "Please fix <asm/byteorder.h>"
#endif
        __u8    tos;
        __be16  tot_len;
        __be16  id;
        __be16  frag_off;
        __u8    ttl;
        __u8    protocol;
        __sum16 check;
        __be32  saddr;
        __be32  daddr;
        /*The options start here. */
};
```

Checksum
The number to check the *integrity* of a packet while routing.

To make sure the packet is not manipulated, each router re-computes it and compares the checksum value with the stored one. (**Q:** what's the issue?)

Packet encapsulation: internet layer

- Internet Layer
  - IP packet header contains:
    - Version v4 (8-bit)
    - Type of service (8-bit: reference)
    - Total length (16-bit)
    - Identification (16-bit)
    - Frame offset (16-bit)
    - Time to live (8-bit)
    - Protocol (8-bit)
    - CRC (checksum) (16-bit)
    - Source IP address (32-bit)
    - Destination IP address (32-bit)

IPv4 Header

```
struct iphdr {
#if defined(__LITTLE_ENDIAN_BITFIELD)
        __u8    ihl:4,
                version:4;
#elif defined (__BIG_ENDIAN_BITFIELD)
        __u8    version:4,
                ihl:4;
#else
#error  "Please fix <asm/byteorder.h>"
#endif
        __u8    tos;
        __be16  tot_len;
        __be16  id;
        __be16  frag_off;
        __u8    ttl;
        __u8    protocol;
        __sum16 check;
        __be32  saddr;
        __be32  daddr;
        /*The options start here. */
};
```

Source / Destination IPs

Example:
Source : 53.59.125.98
Destination: 22.156.71.44

Router: each router has a routing table that stores the subnetwork addresses. It uses the subnetwork addr and source/destination IPs to decide where to send the packet

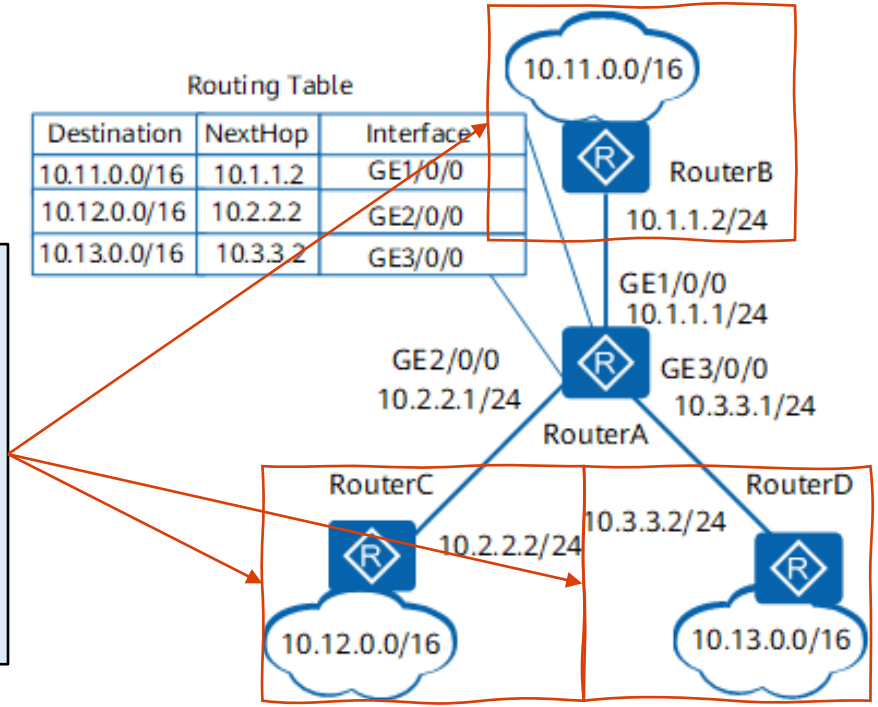
PACKET ENCAPSULATION: INTERNET LAYER

- Internet Layer
  - Router mechanism:
    - Router connects subnetworks
    - **Subnet:** a logical *division* of an IP net

Subnetwork examples:

Router B: 10.11.0.0/16 (65536 addr.)
Router C: 10.12.0.0/16 (same)
Router D: 10.13.0.0/16 (same)

CIDR Notations:
10.11.0.0 : base address
16 : # of leading bits we fix
10.11.0.0/16: 10.11.0.0 – 10.11.255.255



Packet encapsulation: internet layer

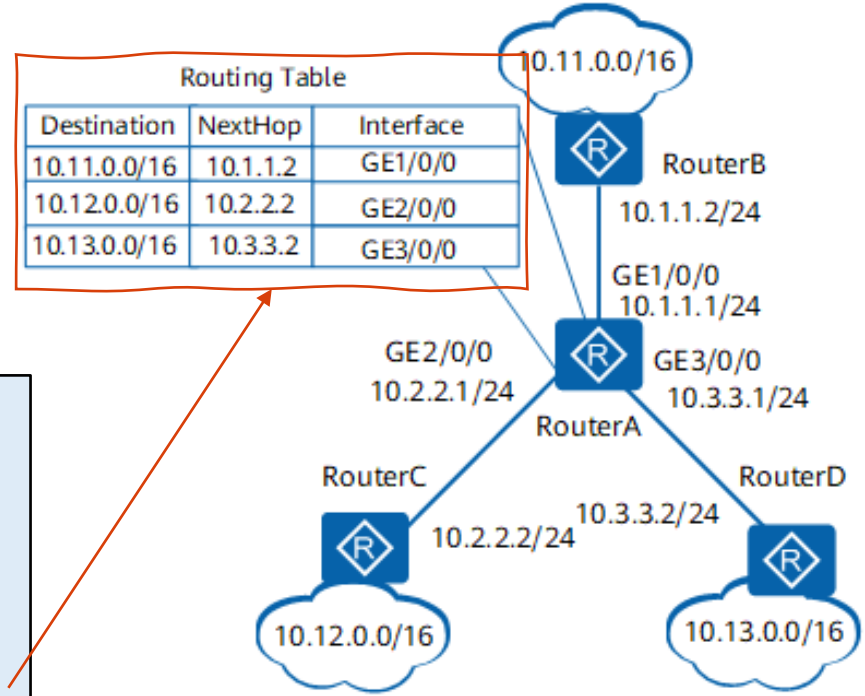
- Internet Layer
  - Router mechanism:
    - Router connects subnetworks
    - **Subnet:** a logical *division* of an IP net
  - Routing table:
    - Describe the network destinations

In Router A: Suppose a packet
comes from the source (10.12.1.45)
moves to the destination (10.11.5.97)

Routing:
Host (src.) -> Router C
Router C -> Router A
Router A -> Router B (subnet: 10.11.0.0/16)
Router B -> Host (dest.)

Routing Table

Destination	NextHop	Interface
10.11.0.0/16	10.1.1.2	GE1/0/0
10.12.0.0/16	10.2.2.2	GE2/0/0
10.13.0.0/16	10.3.3.2	GE3/0/0

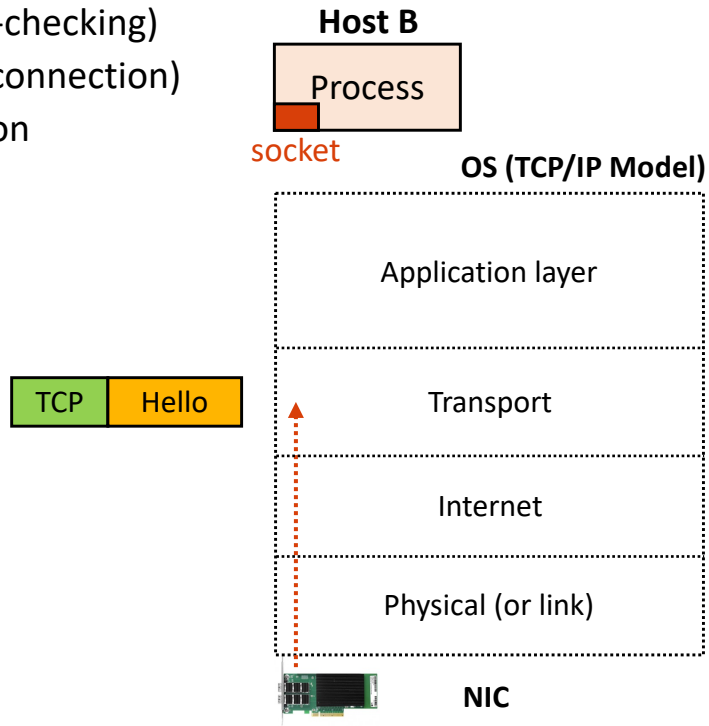


Topics for Today

- Dive into the encapsulation
  - Hardware: NIC (and the network driver)
  - Physical: MAC address-based communication
  - Internet: IP address-based communication
  - Transport: Define protocols, *e.g.*, TCP or UDP
  - Application: Define custom protocols, *e.g.*, HTTP, HTTPS, FTP, etc.

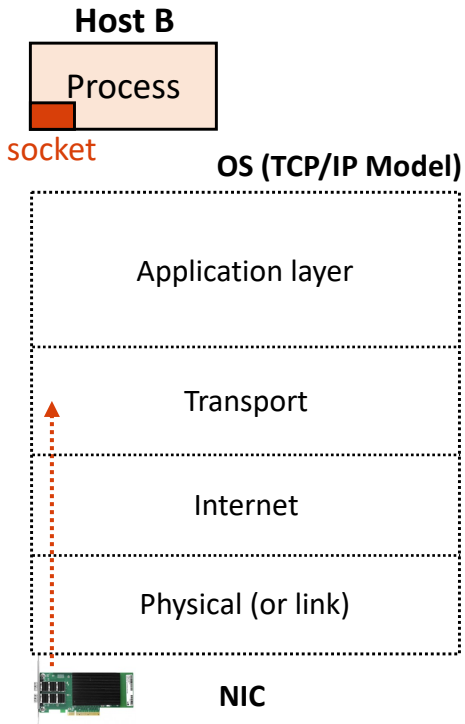
Packet encapsulation: transport layer

- Transport layer
  - **T**ransmission **C**ontrol **P**rotocol (**TCP**)
    - Reliable communications (ordered and error-checking)
    - Connection oriented (first need to establish connection)
    - 3-way handshake for establishing a connection



Packet encapsulation: transport layer

- Transport layer
  - Transmission Control Protocol (**TCP**)
    - Reliable communication (ordered and error-checking)
    - Connection oriented (first need to establish connection)
    - 3-way handshake for establishing a connection
  - User Datagram Protocol (**UDP**)
    - Simple connectionless communication (no handshake)
    - Less reliable communication (no order)
    - Useful for video/audio streaming [where losing packets is acceptable]



Packet encapsulation: transport layer

- Transport layer
  - Transmission Control Protocol (TCP)
    - Reliable communications (ordered and error-checking)
    - Connection oriented (first need to establish connection)
    - 3-way handshake for establishing a connection
  - TCP packet header contains:
    - Source ports (16-bit)
    - Destination ports (16-bits)
    - Sequence number (32-bit)
    - Acknowledgement number (32-bit)
    - Others (flags, checksums, window-size, pointer, …)

TCP Header

(source code)

```
struct tcphdr {
        __be16  source;
        __be16  dest;
        __be32  seq;
        __be32  ack_seq;
#if defined(__LITTLE_ENDIAN_BITFIELD)
        __u16   res1:4,
                doff:4,
                fin:1,
                syn:1,
                rst:1,
                psh:1,
                ack:1,
                urg:1,
                ece:1,
                cwr:1;
#elif defined(__BIG_ENDIAN_BITFIELD)
        __u16   doff:4,
                res1:4,
                cwr:1,
                ece:1,
                urg:1,
                ack:1,
                psh:1,
                rst:1,
                syn:1,
                fin:1;
#else
#error  "Adjust your <asm/byteorder.h> defines"
#endif
        __be16  window;
        __sum16 check;
        __be16  urg_ptr;
};
```

Packet encapsulation: transport layer

- Transport layer
  - Transmission Control Protocol (TCP)
    - Reliable communications (ordered and error-checking)
    - Connection oriented (first need to establish connection)
    - 3-way handshake for establishing a connection
  - TCP packet header contains:
    - Source ports (16-bit)
    - Destination ports (16-bits)
    -
    -
    -

Source / Destination Ports

IP addresses are in the IP packet header
TCP header only contains the port numbers (src/dest)

TCP Header

(source code)

```
struct tcphdr {
        __be16  source;
        __be16  dest;
        __be32  seq;
        __be32  ack_seq;
#if defined(__LITTLE_ENDIAN_BITFIELD)
        __u16   res1:4,
                doff:4,
                fin:1,
                syn:1,
                rst:1,
                psh:1,
                ack:1,
                urg:1,
                ece:1,
                cwr:1;
#elif defined(__BIG_ENDIAN_BITFIELD)
        __u16   doff:4,
                res1:4,
                cwr:1,
                ece:1,
                urg:1,
                ack:1,
                psh:1,
                rst:1,
                syn:1,
                fin:1;
#else
#error  "Adjust your <asm/byteorder.h> defines"
#endif
        __be16  window;
        __sum16 check;
        __be16  urg_ptr;
};
```

Packet encapsulation: transport layer

- Transport layer
  - Transmission Control Protocol (TCP)
    - Reliable communications (ordered and error-checking)
    - Connection oriented (first need to establish connection)
    - 3-way handshake for establishing a connection
  - TCP packet header contains:
    - Source ports (16-bit)
    - Destination ports (16-bits)
    - Sequence number (32-bit)
    - Acknowledgement number (32-bit)
    - ...

Sequence number
The **byte index** of the data a packet has (**4103** / 11945)

Ack number
Receiver: # of packets remaining to receive
Sender : packet # to send next (to the receiver)

TCP Header
(source code)

```
struct tcphdr {
        __be16  source;
        __be16  dest;
        __be32  seq;
        __be32  ack_seq;
#if defined(__LITTLE_ENDIAN_BITFIELD)
        __u16   res1:4,
                doff:4,
                fin:1,
                syn:1,
                rst:1,
                psh:1,
                ack:1,
                urg:1,
                ece:1,
                cwr:1;
#elif defined(__BIG_ENDIAN_BITFIELD)
        __u16   doff:4,
                res1:4,
                cwr:1,
                ece:1,
                urg:1,
                ack:1,
                psh:1,
                rst:1,
                syn:1,
                fin:1;
#else
#error  "Adjust your <asm/byteorder.h> defines"
#endif
        __be16  window;
        __sum16 check;
        __be16  urg_ptr;
};
```

Packet encapsulation: transport layer

- Transport layer
  - Transmission Control Protocol (TCP)
    - Reliable communications (ordered and error-checking)
    - Connection oriented (first need to establish connection)
    - 3-way handshake for establishing a connection
  - TCP packet header contains:
    - Source ports (16-bit)
    - Destination ports (16-bits)
    - Sequence number (32-bit)
    - Acknowledgement number (32-bit)
    - Others (flags, checksums, window-size, pointer, …)

Other fields
Refer to this Wikipedia article (link)

TCP Header
(source code)

```
struct tcphdr {
        __be16  source;
        __be16  dest;
        __be32  seq;
        __be32  ack_seq;
#if defined(__LITTLE_ENDIAN_BITFIELD)
        __u16   res1:4,
                doff:4,
                fin:1,
                syn:1,
                rst:1,
                psh:1,
                ack:1,
                urg:1,
                ece:1,
                cwr:1;
#elif defined(__BIG_ENDIAN_BITFIELD)
        __u16   doff:4,
                res1:4,
                cwr:1,
                ece:1,
                urg:1,
                ack:1,
                psh:1,
                rst:1,
                syn:1,
                fin:1;
#else
#error  "Adjust your <asm/byteorder.h> defines"
#endif
        __be16  window;
        __sum16 check;
        __be16  urg_ptr;
};
```

Packet encapsulation: transport layer

- Transport layer
  - User Datagram Protocol (UDP)
    - Simple connectionless communication (no handshake)
    - Less reliable communication (no order)
    - Useful for video/audio streaming
  - UDP packet header contains:
    - Source port (16-bit)
    - Destination port (16-bits)
    -
    -

Source / Destination Ports

IP addresses are in the IP packet header
UDP header only contains the port numbers like TCP

```
struct udphdr {
        __be16  source;
        __be16  dest;
        __be16  len;
        __sum16 check;
};
```

Packet encapsulation: transport layer

- Transport layer
  - User Datagram Protocol (UDP)
    - Simple connectionless communication (no handshake)
    - Less reliable communication (no order)
    - Useful for video/audio streaming
  - UDP packet header contains:
    - Source port (16-bit)
    - Destination port (16-bits)
    - Packet length (16-bit)
    - Checksum (16-bits)

Length
Total UDP packet size (max. 65515 bytes)
Note: 65535 – 8-byte UDP header – 20-byte IP header

Checksum
Optional field as IP header already contains this data

```
struct udphdr {
        __be16  source;
        __be16  dest;
        __be16  len;
        __sum16 check;
};
```

Topics for Today

- Dive into the encapsulation
  - Hardware: NIC (and the network driver)
  - Physical: MAC address-based communication
  - Internet: IP address-based communication
  - Transport: Define protocols, *e.g.*, TCP or UDP
  - Application: Define custom protocols, *e.g.*, HTTP, HTTPS, FTP, etc. (**CS 372 Topic – Have more fun in this class**)

Topics for today

- Part III: Networking
  - Provide abstraction
    - OSI models
    - Packet encapsulation
  - Offer standard interface
    - RPC mechanisms (e.g., sockets)
  - Manage resources
    - Packet encapsulation in detail

Thank You!

M/W 12:00 – 1:50 PM (LINC #200)

Sanghyun Hong

sanghyun.hong@oregonstate.edu

Oregon State University

SAIL

Secure AI Systems Lab