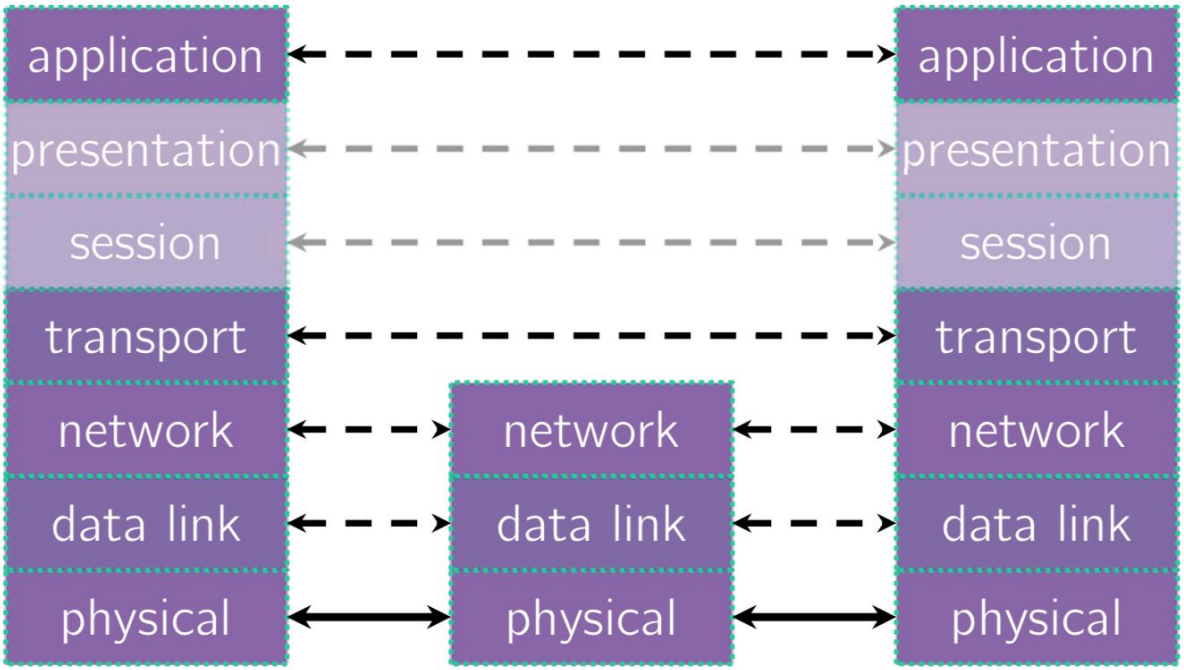
CSE 333 Section 8 - Client-Side Networking

Welcome back to section! We're glad that you're here :)

Networking Quick Review

What are the following protocols used for? (bonus: what *layer* of the networking stack is it found?)

- DNS: Translating between IP addresses and host names. (Application Layer)
- IP: Routing packets across the Internet. (Network Layer)
- TCP: Reliable, stream-based networking on top of IP. (Transport Layer)
- UDP: Unreliable, packet-based networking on top of IP. (Transport Layer)
- HTTP: Sending websites and data over the Internet. (Application Layer)



Why would you want to use TCP over UDP?

TCP is reliable and has simpler semantics than UDP, so it's easier to use for a lot of applications.

Why would you want to use UDP over TCP?

Some applications can't tolerate delays from resending lost packets and/or don't mind losing a few packets, so UDP is a better choice for these.

Exercise:

For each of the following, identify the layer in the network stack that performs the described operation. Be sure you identify the layer, not the protocol.

a). Use WiFi or Bluetooth to transmit data to other hosts.

Physical Layer

b). use media access control(MAC) to figure out when and where to send the packet.

Link Layer

c). Forward a packet from the local wired or wireless network to a different local network if its destination address is not on the same local network.

Network Layer

d). If two packets that make up a message arrive out of order, rearrange them into the correct order before they are transmitted to the process reading the data.

Transport Layer (TCP)

e). Send a 200 OK message to a web browser in reply to a request for something from a web server.

Application Layer (HTTP)

f). Transmit an Ethernet packet on the local network from one host machine's NIC interface address to another's.

Link Layer

g). Resolve what IP address is www.youtube.com pointing to.

Application Layer (DNS)

Step-by-step Client-Side Networking

1) Figure out what IP address and port to talk to. (getaddrinfo())

```
// returns 0 on success, negative number on failure
int getaddrinfo(const char *hostname,    // hostname to lookup
           const char *servname,         // service name
           const struct addrinfo *hints, // desired output (optional)
           const addrinfo **res);        // results structure
```

```
struct addrinfo {
     int ai_flags;              // additional flags
     int ai_family;             // AF_INET, AF_INET6, AF_UNSPEC
     int ai_socktype;           // SOCK_STREAM, SOCK_DGRAM, 0
     int ai_protocol;           // IPPROTO_TCP, IPPROTO_UDP, 0
     size_t ai_addrlen;         // length of socket addr in bytes
     struct sockaddr* ai_addr;  // pointer to socket addr
     char* ai_canonname;        // canonical name
     struct addrinfo* ai_next;  // can have linked list of records
}
```

2) Create a socket. (socket())

```
// returns file descriptor on success, -1 on failure (errno set)
int socket(int domain,     // AF_INET, AF_INET6, etc.
           int type,       // SOCK_STREAM, SOCK_DGRAM, etc.
           int protocol);  // usually 0
```

3) Connect to the server. (connect())

```
// returns 0 on success, -1 on failure (errno set)
int connect(int sockfd,                    // fd from step 2
            struct sockaddr *serv_addr,    // server info from step 1
            int addrlen);                  // size of serv_addr
```

4) Transfer data through the socket. (read() and write())

```
// returns amount read, 0 for EOF, -1 on failure (errno set)
ssize_t read(int fd, void *buf, size_t count);

// returns amount written, -1 on failure (errno set)
ssize_t write(int fd, void *buf, size_t count);
```

Notice that these are the same POSIX calls you used for files! Remember you have to deal with partial reads/writes!

5) Close the socket when done. (close())

```
// returns 0 for success, -1 on failure (errno set)
int close(int fd);
```

Exercise:

A bit of networking. When we were describing how a network server works, we listed 5 steps that need to be done to establish communication with a server, exchange data, and shut down. In the list below, fill in the name of the function that is used at each step , then give a 1-sentence description of the purpose of that step.

1. Function: getaddrinfo — Purpose: Figure out what IP address and port to talk to.
2. Function: socket — Purpose: Create a socket
3. Function: connect — Purpose: connect socket created in step 2 to the server's IP address and port
4. Function: read/write — Purpose: exchange data with the server using the socket
5. Function: close — Purpose: shutdown socket and free resources

Exercise:

Write a program that reads a file (passed as the first argument to the program) and writes it *in reverse* to a server (hostname passed as the second argument, port passed as the third argument).

See section8.cc (available Monday).