

Step 3: Connect to the Server

The connect () system call establishes a connection to a remote host

int connect(int sockfd, const struct sockaddr* addr, socklen_t addrlen);

sockfd: Socket file description from Step 2

addr and addrlen: Usually from one of the address structures returned by getaddrinfo in Step 1 (DNS lookup)

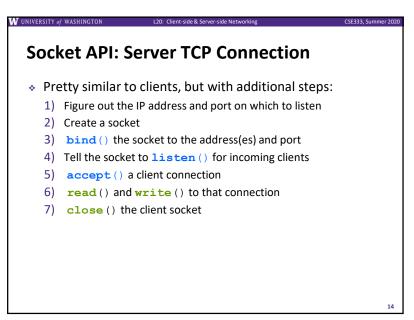
Returns 0 on success and -1 on error

connect() may take some time to return

It is a blocking call by default

The network stack within the OS will communicate with the remote host to establish a TCP connection to it

This involves ~2 round trips across the network



Step 3: Bind the socket

int bind(int sockfd, const struct sockaddr* addr, socklen_t addrlen);

Looks nearly identical to connect()!

Returns 0 on success, -1 on error

Some specifics for addr:

Address family: AF_INET or AF_INET6

What type of IP connections can we accept?

POSIX systems can handle IPv4 clients via IPv6 ☺

Port: port in network byte order (htons() is handy)

Address: specify particular IP address or any IP address

"Wildcard address" - INADDR ANY (IPv4), in6addr any (IPv6)

Step 4: Listen for Incoming Clients
* int listen(int sockfd, int backlog);
• Tells the OS that the socket is a listening socket that clients can connect to
• backlog: maximum length of connection queue
• Gets truncated, if necessary, to defined constant SOMAXCONN

- Returns 0 on success, -1 on error
- Clients can start connecting to the socket as soon as listen()
 returns

• The OS will refuse new connections once queue is full until server

• Server can't use a connection until you accept() it

accept()s them (removing them from the queue)

19

18

19