# Networking: Using Sockets

CS 241

April 21, 2014

University of Illinois

# TCP vs UDP

TCP

- Reliable Delivery
- Flow Control
- Slower / More Overhead

- Requires a 3-way handshake on connect

- Ideal for applications where data integrity is critical.

- UDP
  - Fast / Low Overhead
  - No delivery guarantees

  - "Connectionless": no setup required

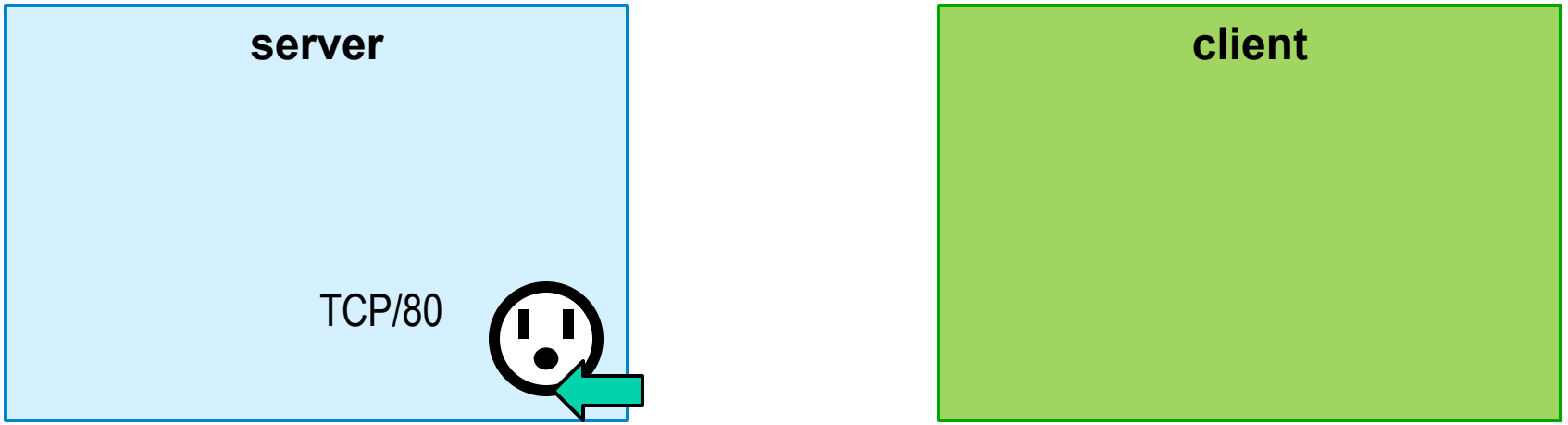
  - Ideal for applications where speed is most important.

*…and both provide port numbers.*

# Creating a TCP session

Server:

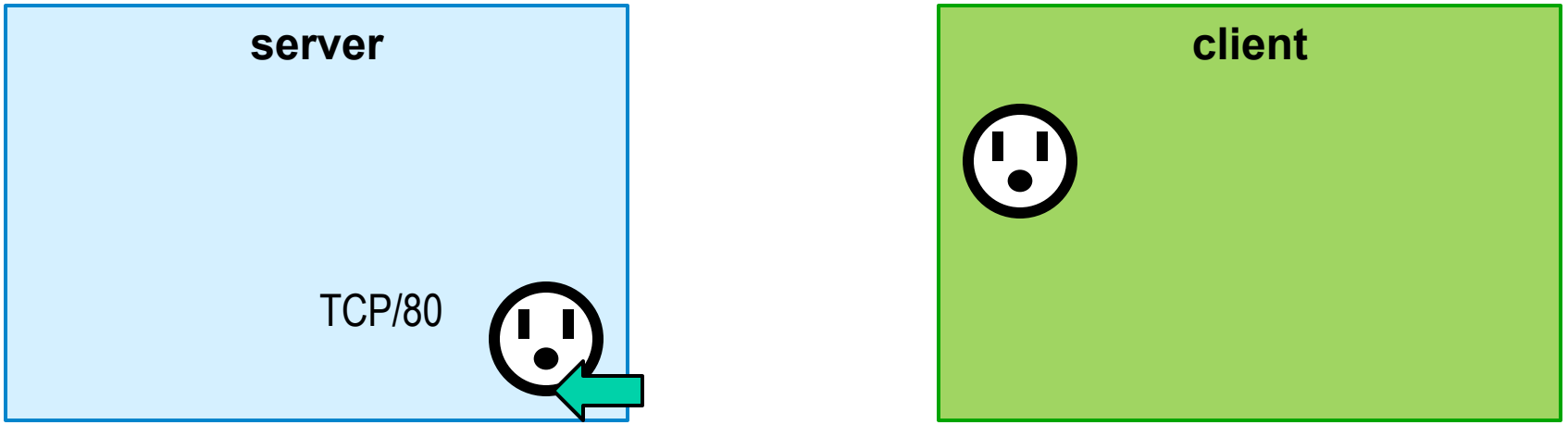
- Creates a socket to listen for incoming connections.
- Must listen on a specific protocol/port.



# Creating a TCP session

Client:

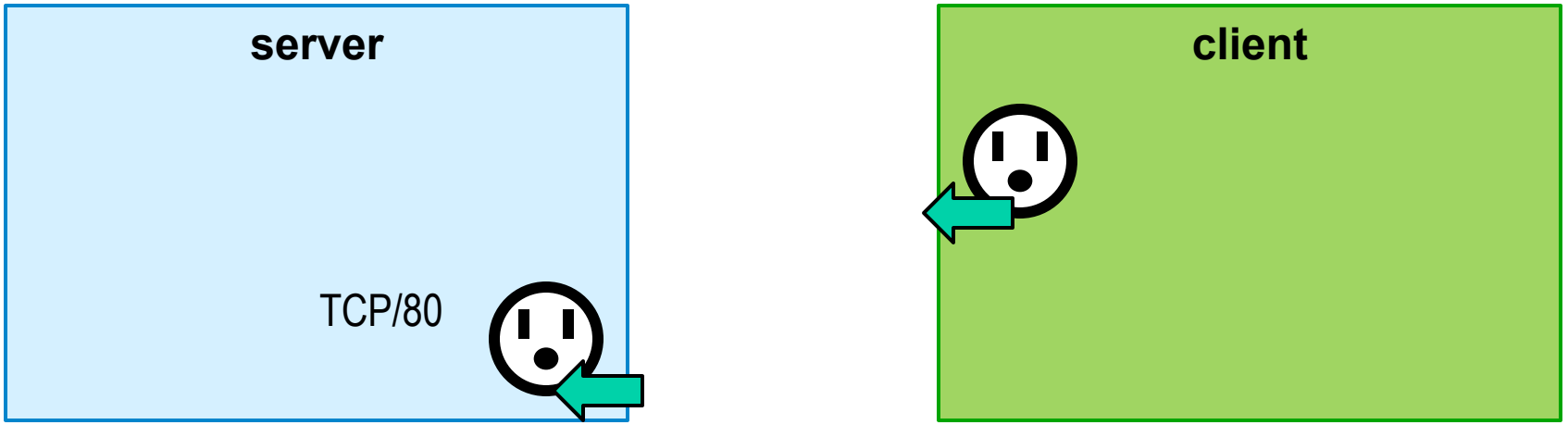
- Creates a socket to connect to a remote computer.



# Creating a TCP session

Client:

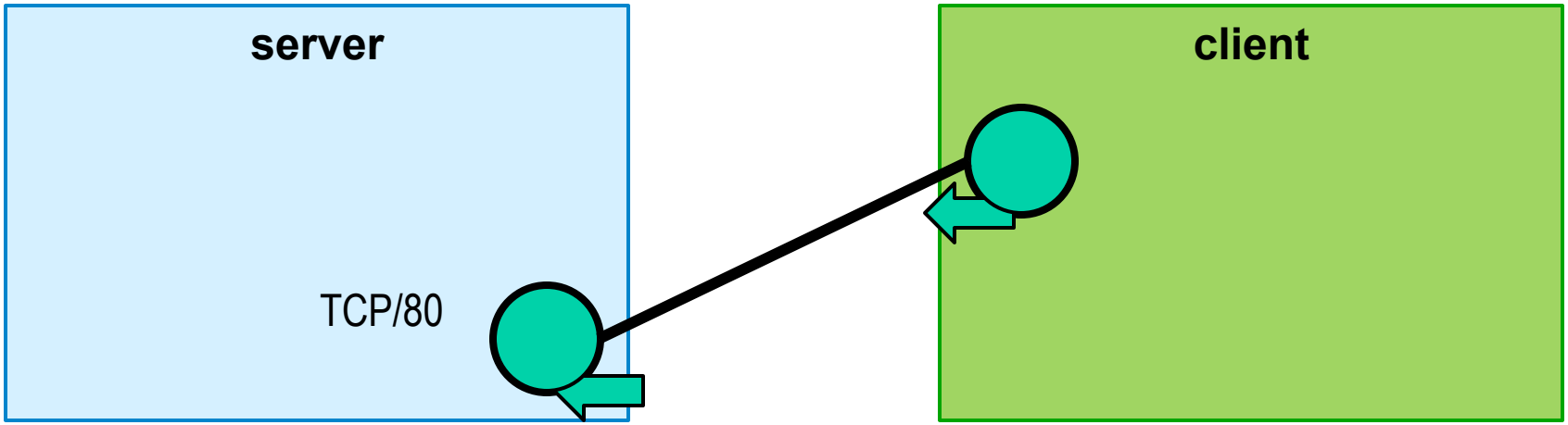
- Requests a connection to TCP port 80 on 74.125.225.70.



# Creating a TCP session

Server:

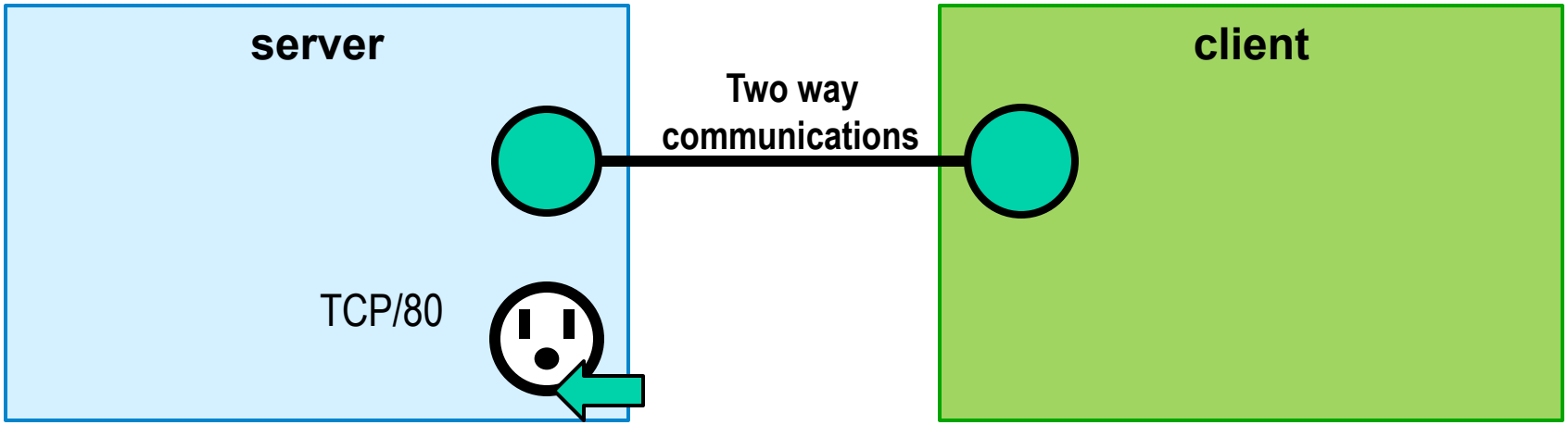
- Accepts the connection.



# Creating a TCP session

Server:

- Spawns a new socket to communicate directly with the newly connected client.
- Allows other clients to connect.



# Creating a network socket (client and server)

**`socket()`**: Create an endpoint for communication

```
int socket(int network_protocol,
           int transport_protocol,
           int sub_protocol)
```

```
IP: AF_INET          IPv6: AF_INET6
TCP: SOCK_STREAM     UDP: SOCK_DGRAM
```

# Setting up a server socket

`getaddrinfo()`: network address translation

- Translates a hostname (IP address or domain name), port, and protocol into a socket address struct.

`bind()`: binds an socket address to a socket

- Required in order to know what port number your socket will be listening for new connections

`listen()`: places the socket in a listening state

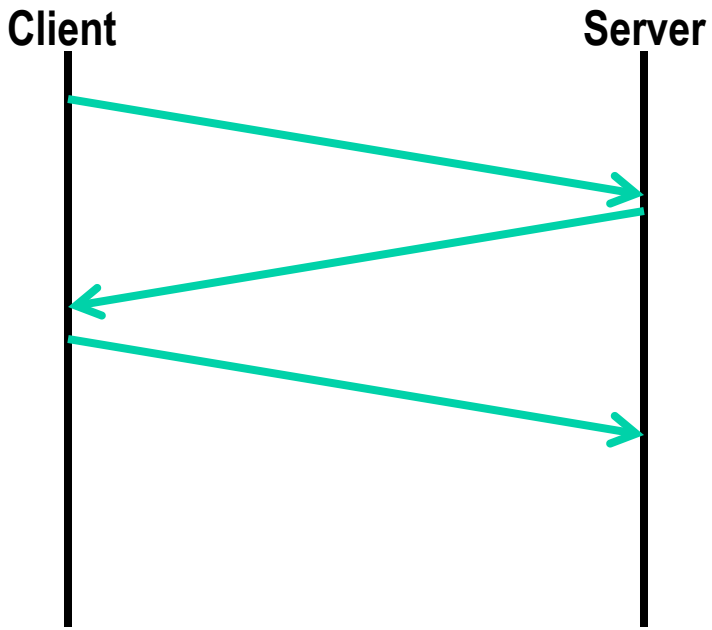
`accept()`: accept a communication on a socket

- ```
  int accept(int sockfd,
             struct sockaddr *addr,
             socklen_t *addrlen);
  ```
  ```

# Setting up a client socket

getaddrinfo(): network address translation (as before)

connect(): initiate a connection on a socket

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

# Behind the scenes in connect()

When a client connects to a host on TCP, a "**TCP session**" is initiated.

- Requires a **three-way handshake** before any data can be sent on the TCP socket.



# HTTP:
# Hypertext Transfer Protocol

# HTTP Request

Sent from a client (eg: web browser) to a server.

```
GET /index.html HTTP/1.1
Host: linux4.ews.illinois.edu
User-Agent: Mozilla/5.0 (Windows NT[...]
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

# Network Frame

In networking, you must identify when a packet ends.

- **Network frame**: The region of data that consists of one request for a given protocol.

In HTTP:

- **Header**: Always ends with `\r\n\r\n`
- **Body**: If a body exists, the header will always specify a `Content-Length` field that specifies the number of bytes in the body

# HTTP Response

Sent in response to an HTTP request.

```
GET /cs241/ HTTP/1.1
Content-Length: 23774
Content-Type: text/html
Server: Microsoft-IIS/7.5
Set-Cookie: ASPSESSIONIDAEEESRAB=PN[...]
X-Powered-By: ASP.NET
Date: Fri, 15 Nov 2013[...]
Connection: close
```

# Optimizing HTTP

Consider loading a website:

- You request a single HTML page
- The HTML page contains 5 images
- (Since the HTML page contains the 5 images, you don't know about them before you receive the images.)

In terms of RTTs (assume it takes no time to transmit the actual data), how long would it take to completely load the webpage given:

- You had to make a new connection for every request,
  **and**
- You can only have one active connection at any time

# Optimizing HTTP

# Optimizing HTTP

In HTTP, the **`Connection`** header requests the server to keep the TCP connection active.

- **`Connection: keep-alive`**
- **`Connection: close`**

Using **`Connection: keep-alive`**, how many RTTs would it take to load the same website?

# Optimizing HTTP

# Optimizing HTTP

Nearly all modern browsers make at least 2 connections to every web server. This allows for requests to be made in parallel.

Using **`Connection: keep-alive`** and three (3) parallel connections, how many RTTs would it take to load the same website?

# Optimizing HTTP

# Optimizing HTTP

In HTTP/1.1, a new feature called HTTP Pipelining allows multiple requests to be made in one header.

- Request all five images at once!

Using **`Connection: keep-alive`**, HTTP pipelining, and only one connection, how many RTTs would it take to load the same website?

# Optimizing HTTP