Basic Networking and Proxy Lab

15-213 Recitation #13

Connection Establishment Functions

- Server Sockets
  - socket(…)
  - bind(…)
  - listen(…)
  - accept(…)
  - close(…)
- Client Sockets
  - socket(…)
  - connect(…)
  - close(…)

socket(domain, type, protocol)

```
int sock_fd = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
```

- domain – Protocol Family to use
  - PF_INET is the IPv4 family of protocols
- type – Type of protocol to use
  - SOCK_STREAM suggests a steady data stream with guaranteed in-order delivery
- protocol – Specific protocol to use
  - IPPROTO_TCP suggests to use TCP (stream-based socket protocol)

bind(sock_fd, my_addr, addrlen)

```
struct sockaddr_in sockaddr;
memset(&sockaddr, 0, sizeof(sockaddr);
sockaddr.sin_family = AF_INET;
sockaddr.sin_addr.s_addr = INADDR_ANY;
sockaddr.sin_port = htons(listenPort)

err = bind(sock_fd, (struct sockaddr *) sockaddr, sizeof(sockaddr));
```

- sock_fd – file descriptor of socket
- my_addr – address to which to bind
- addrlen – size (in bytes) of address struct

listen(sock_fd, backlog)

```
err = listen(sock_fd, MAX_WAITING_CONNECTIONS);
```

- sock_fd – socket on which to listen
- backlog – Maximum size of list of waiting connections

accept(sock_fd, addr, addrlen)

```
struct sockaddr_in client_addr;
socklen_t my_addr_len = sizeof(client_addr);
client_fd = accept(listener_fd, &client_addr, &my_addr_len);
```

- sock_fd – listening socket from which to accept connection
- addr – pointer to sockaddr struct to hold client address
- Addrlen – pointer to length of addr that is overwritten with actual length of connection

connect(sock_fd, addr, addrlen)

```
struct sockaddr_in remote_addr;
/* initialize remote_addr */
err = connect(listener_fd, &remote_addr, sizeof(remote_addr));
```

- sock_fd – socket to connect to
- addr – pointer to sockaddr struct that holds remote address
- Addrlen –length of addr that is overwritten with actual length of connection

close(sock_fd)

```
err = close(sock_fd);
```

- sock_fd – socket to close

Socket Communication Functions

- send(…)
- recv(…)

send(sock_fd, buf, buf_len, flags)

```
total_sent = 0;
while(total_sent < buff_len) {
  sent = send(sock_fd, buff + total_sent, buff_len – total_sent, 0);
  if(sent <= 0) goto error;
  total_sent += sent;
}
```

- sock_fd – socket to send to
- buf – buffer to send from
- buf_len – max amount to send
- flags – additional flags

recv(sock_fd, buf, max_len, flags)

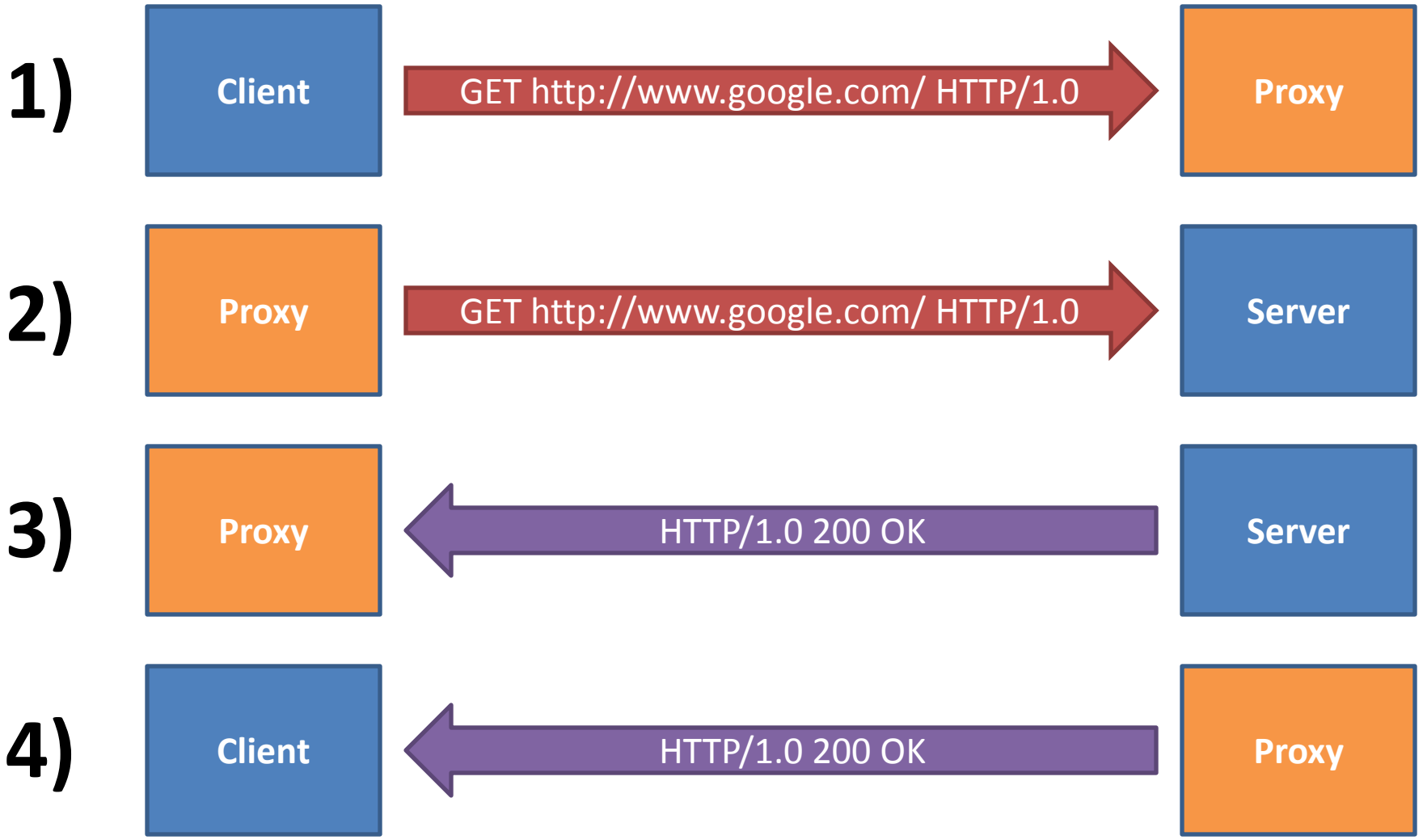
```
actual_size = recv(sock_fd, buf, sizeof(buf), 0);
if(actual_size <= 0) goto err;
```

- sock_fd – socket to receive from
- buf – buffer to receive into
- max_len – maximum number of bytes to receive
- flags – additional flags

What is Proxy Lab?

- Write an **IPv4 Caching HTTP Proxy**
- **Proxy** – Intermediate Request Router
- **IPv4** – Internet Addresses are 4 bytes (a.b.c.d)
- **HTTP** – Hypertext Transfer Protocol used by web servers and browsers
- **Caching** – Requests are stored to spare repeated network fetches

What is a Proxy?



What is a Caching Proxy?

1)

The Proxy has already serviced a request for http://www.google.com/ and has stored the result.

2)

The Proxy simply responds with the stored result for http://www.google.com/. The Client is unaware that it has not communicated with the google.com server directly.

On Testing Your Caching Proxy…

- ***Like the previous labs,*** your proxy will be graded with an autograder
- ***Unlike the previous labs,*** you will not have access to the autograder or traces
- ***This means*** that you must come up with your own set of tests that stress all parts of the specification
- ***Think like we think!*** Your testing should try to stress your proxy to the greatest extent possible. We will find your bugs!

Any Questions?