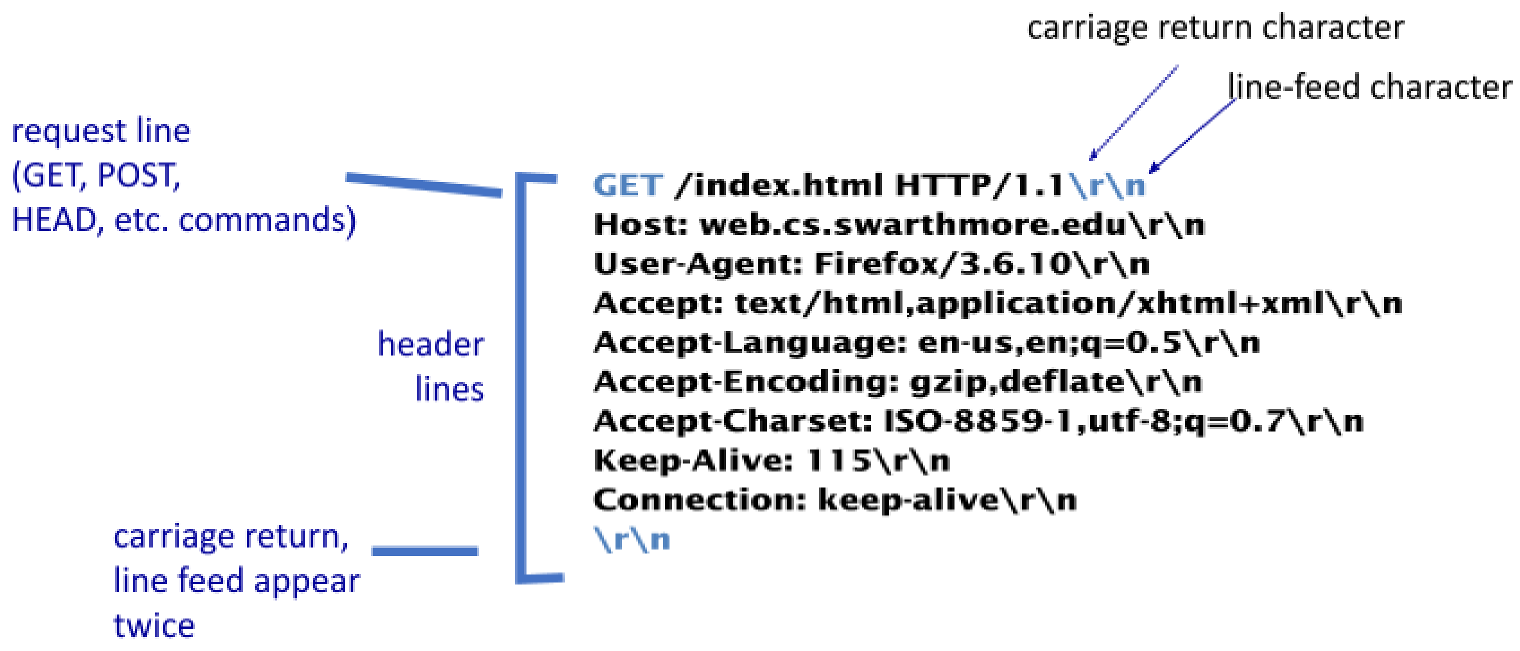
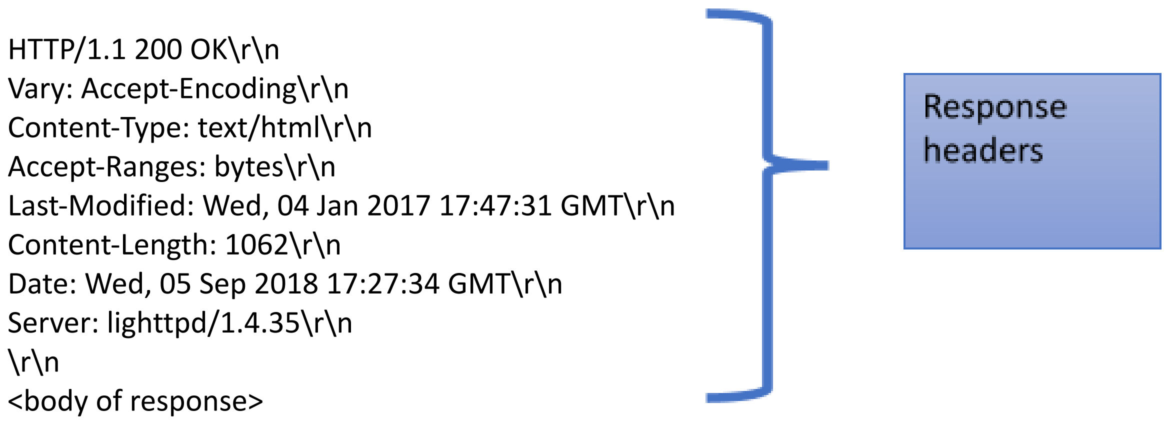
# Worksheet Class 2-3: HTTP and Socket Programming



```
GET /index.html HTTP/1.1\r\n
Host: web.cs.swarthmore.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

carriage return character
line-feed character
request line (GET, POST, HEAD, etc. commands)
header lines
carriage return, line feed appear twice



```
HTTP/1.1 200 OK\r\n
Vary: Accept-Encoding\r\n
Content-Type: text/html\r\n
Accept-Ranges: bytes\r\n
Last-Modified: Wed, 04 Jan 2017 17:47:31 GMT\r\n
Content-Length: 1062\r\n
Date: Wed, 05 Sep 2018 17:27:34 GMT\r\n
Server: lighttpd/1.4.35\r\n
\r\n
<body of response>
```

Q1. We have these \r\n (CRLF) things all over the place.

(a) Are all of them necessary? What would happen if we didn't have any of them?

(b) How might we delineate messages in HTTP? Discuss the pros and cons of each protocol design

a. There's no way to delineate messages

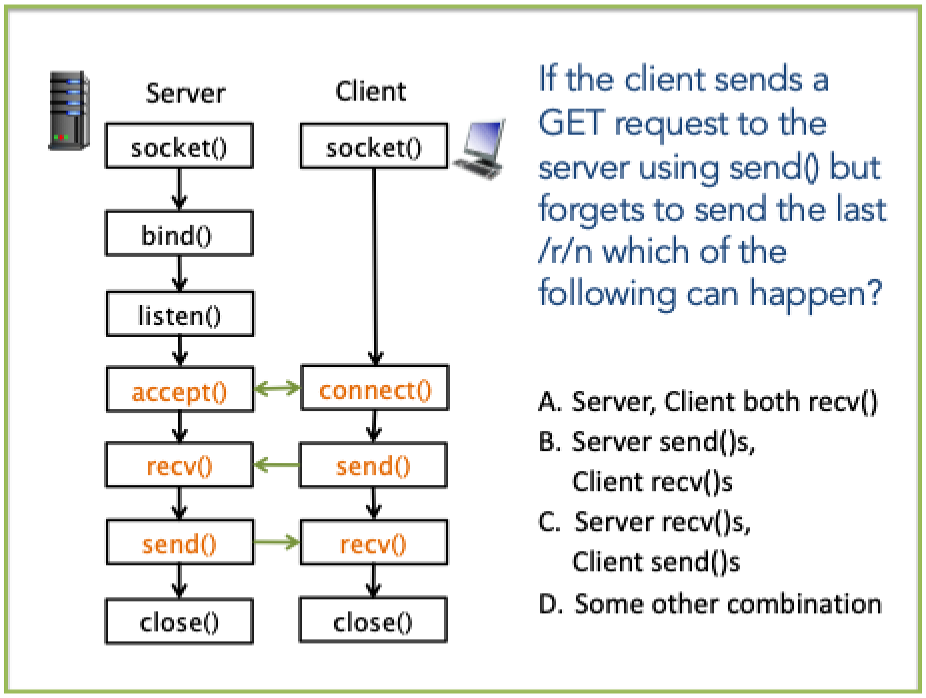
b. The way it's currently done is using _____

c. Force all messages to be the same size
d. Send the message size prior to the message
e. Some other way (discuss)

Q2. Let's say HTTP was not a text-based protocol, but a binary protocol.

(a) Would we still be able to use CRLFs? Why or why not?
(b) We talked about header sizes in relation to the payload size of a packet last week. Do you see any advantage of a binary protocol

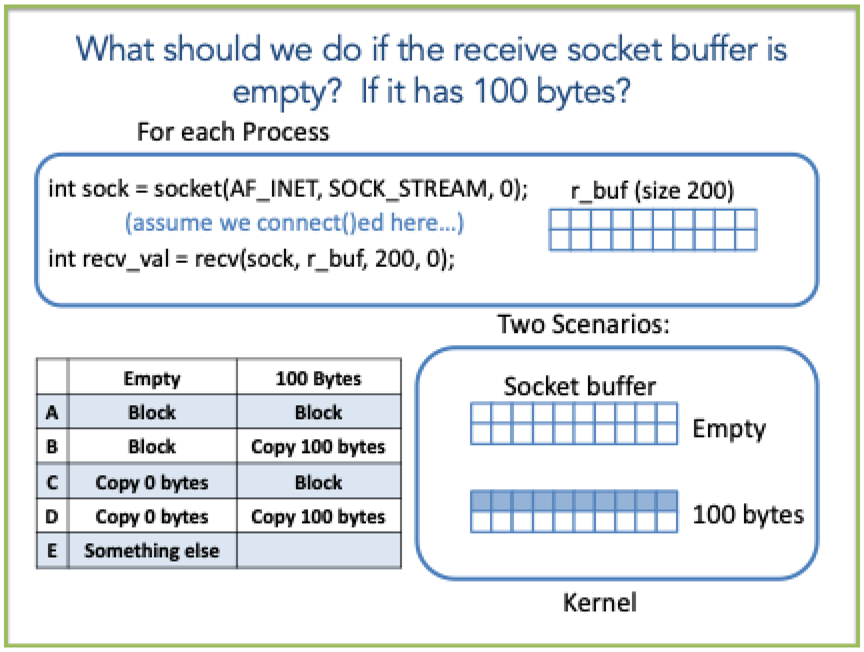
Socket Programming Questions



If the client sends a GET request to the server using send() but forgets to send the last /r/n which of the following can happen?

A. Server, Client both recv()
B. Server send()s, Client recv()s
C. Server recv()s, Client send()s
D. Some other combination

Discuss your choice of answer here:



## What should we do if the receive socket buffer is empty? If it has 100 bytes?

For each Process

```
int sock = socket(AF_INET, SOCK_STREAM, 0);
        (assume we connect()ed here...)
int recv_val = recv(sock, r_buf, 200, 0);
```

r_buf (size 200)

Two Scenarios:

|   | Empty | 100 Bytes |
|---|---|---|
| A | Block | Block |
| B | Block | Copy 100 bytes |
| C | Copy 0 bytes | Block |
| D | Copy 0 bytes | Copy 100 bytes |
| E | Something else |  |

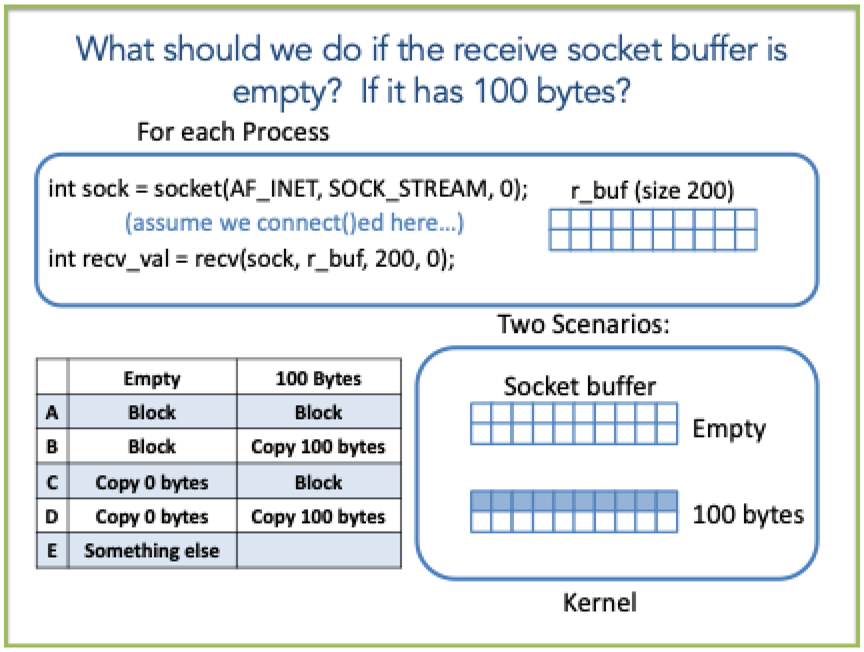
Socket buffer

Empty

100 bytes

Kernel

Your answer here:



## What should we do if the receive socket buffer is empty? If it has 100 bytes?

For each Process

```
int sock = socket(AF_INET, SOCK_STREAM, 0);
        (assume we connect()ed here...)
int recv_val = recv(sock, r_buf, 200, 0);
```

r_buf (size 200)

Two Scenarios:

|   | Empty | 100 Bytes |
|---|---|---|
| A | Block | Block |
| B | Block | Copy 100 bytes |
| C | Copy 0 bytes | Block |
| D | Copy 0 bytes | Copy 100 bytes |
| E | Something else |  |

Socket buffer

Empty

100 bytes

Kernel

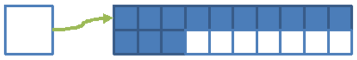
Your answer here:

## ALWAYS check send() and recv()'s return value!

- When send() /recv() return value is less than the data size, **you are responsible for sending/receiving the rest**.

Data sent: 0

Data to send: 130

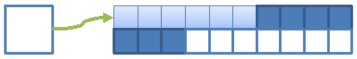
Data: 

60 send(sock, data, 130, 0);

---

Data sent: 60

Data to send: 130

Data: 

// what should your next send call look like?
send(...)

Write in your next send() call here assuming that the first call to send() has successfully sent 60 out of 130 bytes.