File Uploads - Buffers

File Uploads

- When parsing file upload requests
  - Parse multipart/form-data
  - Never decode the bytes of the file as a String

```
POST /form-path HTTP/1.1
Content-Length: 746
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarycriD3u6M0UuPR1ia

------WebKitFormBoundarycriD3u6M0UuPR1ia
Content-Disposition: form-data; name="commenter"

Jesse
------WebKitFormBoundarycriD3u6M0UuPR1ia
Content-Disposition: form-data; name="upload"; filename="discord.png"
Content-Type: image/png

<bytes_of_the_file>
------WebKitFormBoundarycriD3u6M0UuPR1ia--
```

File Uploads

- First, read data from the TCP socket

```
socket.on("data",function (data) {});
```

```
received_data = self.request.recv(2048).strip()
```

```
POST /form-path HTTP/1.1
Content-Length: 746
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarycriD3u6M0UuPR1ia

------WebKitFormBoundarycriD3u6M0UuPR1ia
Content-Disposition: form-data; name="commenter"

Jesse
------WebKitFormBoundarycriD3u6M0UuPR1ia
Content-Disposition: form-data; name="upload"; filename="discord.png"
Content-Type: image/png

<bytes_of_the_file>
------WebKitFormBoundarycriD3u6M0UuPR1ia--
```

TCP Reminder

- TCP creates a persistent connection
- Bytes are streamed over this connection
- Data can be sent and received until one side closes the connection

- With small GET requests
  - Read from the TCP socket once to read the entire request

Large File Uploads

- What if we receive a fairly large POST request?
  - Might not be able to read the entire request in one read from the socket

```
socket.on("data",function (data) {});
```

```
received_data = self.request.recv(2048).strip()
```

```
POST /form-path HTTP/1.1
Content-Length: 91320
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarycriD3u6M0UuPR1ia

------WebKitFormBoundarycriD3u6M0UuPR1ia
Content-Disposition: form-data; name="commenter"

Jesse
------WebKitFormBoundarycriD3u6M0UuPR1ia
Content-Disposition: form-data; name="upload"; filename="flamingo.jpg"
Content-Type: image/jpeg

<bytes_of_the_file>
------WebKitFormBoundarycriD3u6M0UuPR1ia--
```

Large File Uploads

- What if a very large file is uploaded
  - **Must** read from the socket multiple times!

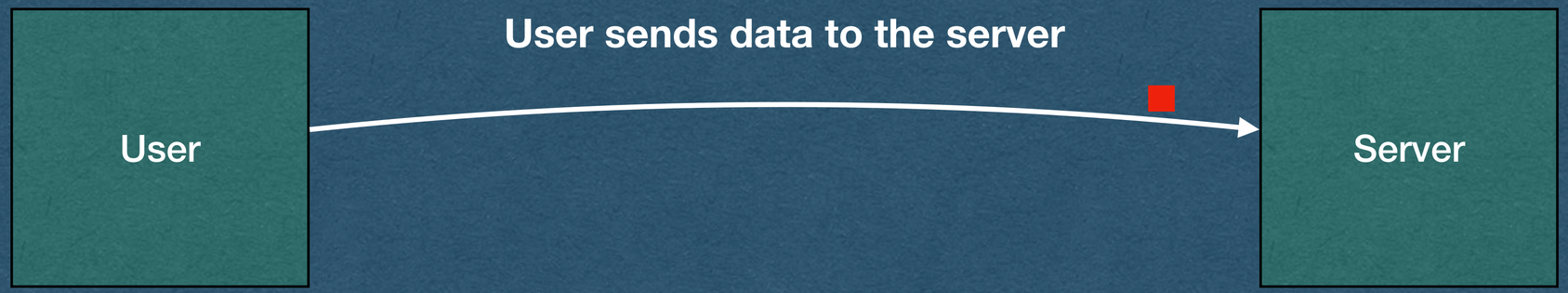
```
socket.on("data",function (data) {});
```

```
received_data = self.request.recv(2048).strip()
```

```
POST /form-path HTTP/1.1
Content-Length: 1884206
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarycriD3u6M0UuPR1ia

------WebKitFormBoundarycriD3u6M0UuPR1ia
Content-Disposition: form-data; name="commenter"

Jesse
------WebKitFormBoundarycriD3u6M0UuPR1ia
Content-Disposition: form-data; name="upload"; filename="hq_image.png"
Content-Type: image/png

<bytes_of_the_file>
------WebKitFormBoundarycriD3u6M0UuPR1ia--
```

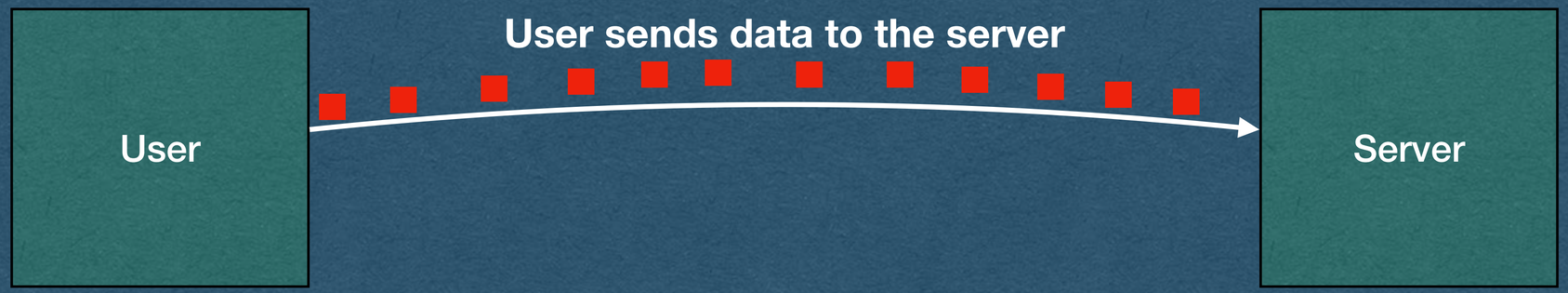
Buffers

- TCP socket libraries will use buffers
- No matter your language/library you will have a method/ function that reads bytes from the socket
  - Called when there are bytes that arrive over the socket
  - Returns *some* bytes of the request



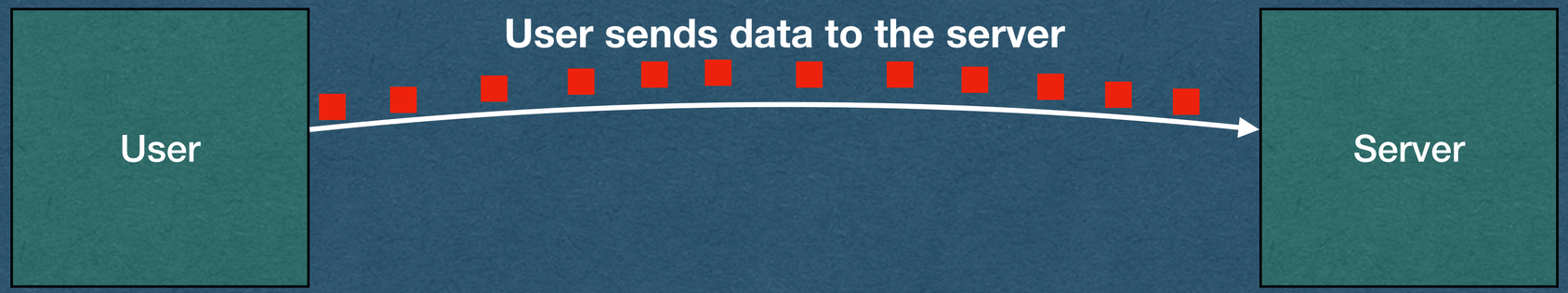
Buffer Questions

- What happens when the user has a lot of data to send?
- What if the user has a slow connection?
- Does the socket server wait for all of the data to be received before calling your code?
- What if the data takes an hour to send?
- What if the data contains streaming video that never ends?



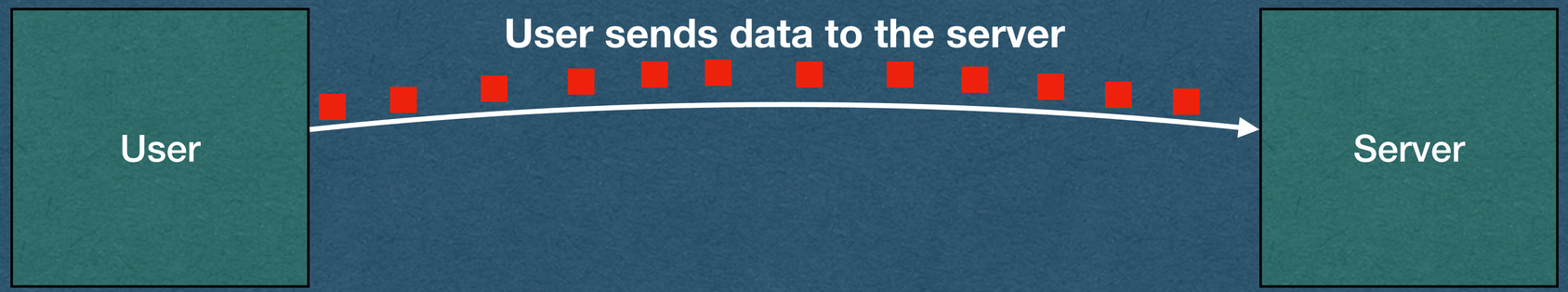
Buffer Answer

- The socket notifies your code when there is data to read - even if it's not the entire request
- The socket server will have a buffer size, typically a few kB, and will read **at most** that many bytes in a single call
  - For GET requests the entire request is smaller than the buffer (Safe assumption *in this course*)



Buffers

- Now that we're handling file uploads, we must be aware of these buffers
- The server will need data that persists across multiple calls that read bytes from a socket
  - Create data structures that store the bytes read from a request
  - Combine the bytes from multiple calls to receive the entire file



Buffers

- When receiving a large request:
  - Read bytes from the socket
  - Parse the headers
  - Find the Content-Length header and store this value
  - Keep reading bytes from the stream until you have Content-Length number of bytes in your data structure
  - Process the request

Assumptions

- Assumptions you may make in HW2:
- Only one user will be uploading a file at any given time
  - You don't have to support multiple simultaneous TCP connections, *yet* (you will in HW3)
- The first read from your buffer will contain all the headers
  - If you're not currently buffering, you can safely parse the first bytes as the headers of the request
  - This allows you to read the Content-Length
- We will test with files larger than your TCP buffer size
  - ie. Do not do this: `received_data = self.request.recv(1048576).strip()`

Demos