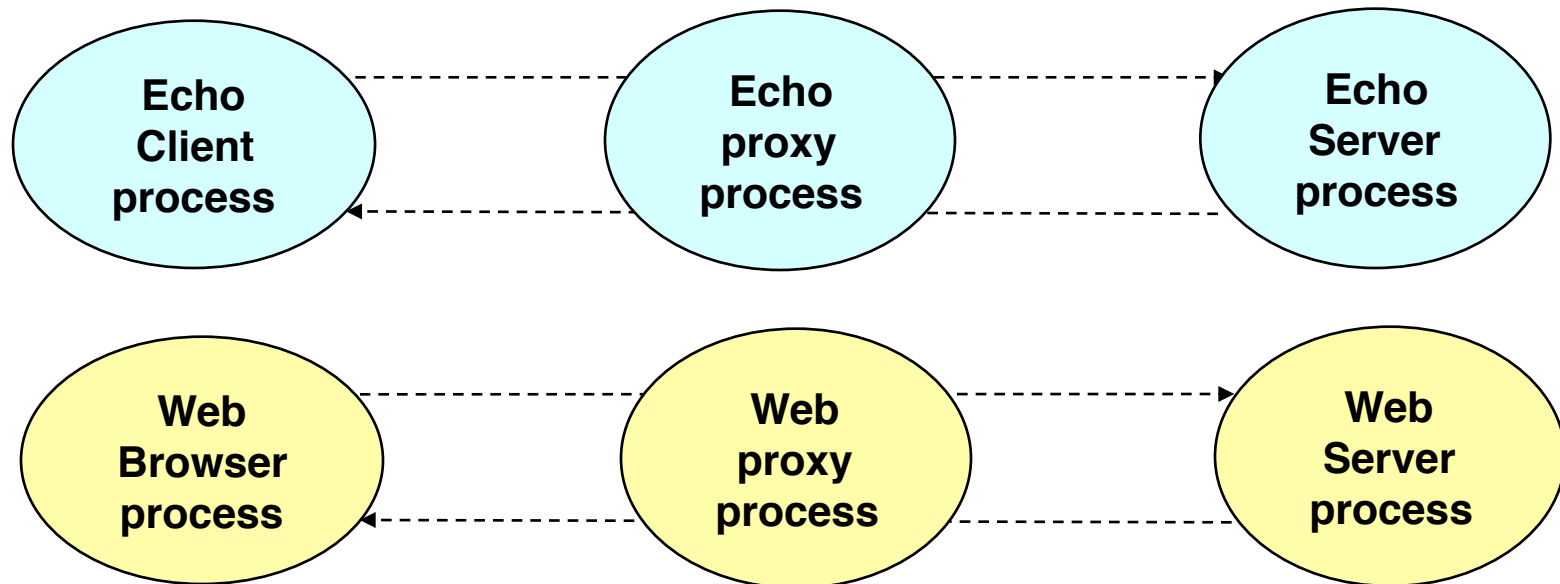


Sequential Web Proxy

L7 Proxy

- **Concurrency (next recitation)**
- **Step-by-step:**
 - Implement sequential web proxy first



Outline

- What information to parse from the HTTP headers
- What headers to suppress
- How to do data transfer
 - Browser -> Proxy -> Web Server
- Testing

HTTP Request

Request Type	Host	Path	Version
--------------	------	------	---------

```
GET http://csapp.cs.cmu.edu/simple.html HTTP/1.1
```

```
Host: csapp.cs.cmu.edu
```

```
User-Agent: Mozilla/5.0 ...
```

```
Accept: text/xml,application/xml ...
```

```
Accept-Language: en-us,en;q=0.5 ...
```

```
Accept-Encoding: gzip,deflate ...
```



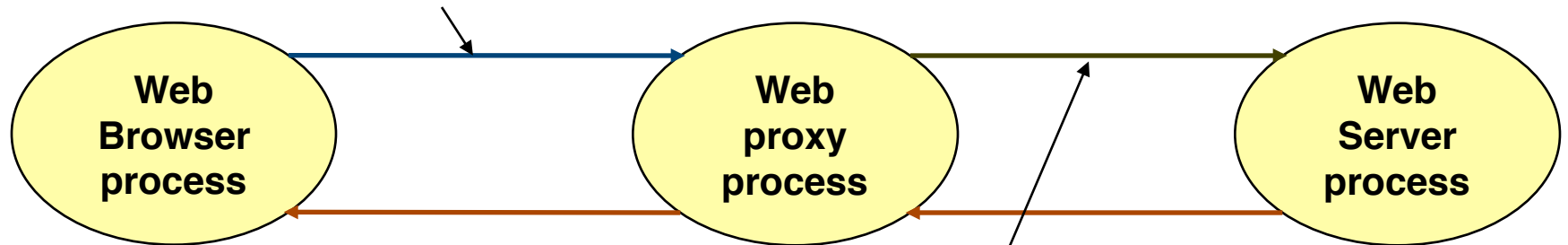
An empty line (“\r\n”) terminates a request.

What Headers to Parse

- First line of the HTTP request
 - Complete URL
 - Extract the URI for server HTTP request
 - Version
 - Change to HTTP 1.0 in the server request
 - Hostname
 - Needed for the Host: field in the server request
 - Port
 - Proxy needs to know the port of the server

Web proxy in Lab 7

```
GET http://www.cmu.edu:80/index.html HTTP/1.0  
<other information>
```



Connects to the target web server, sends request looking like this:

```
GET /index.html HTTP/1.0
```

```
<other information in the original request>
```

Lab 7 things to-do:

parse HTTP request (1st line): extract hostname & port number
port is not necessarily specified in the request if the default
number is used (80)

What Headers to Suppress

- Connection/Proxy-Connection
 - Change the field to **close**
- Keep-Alive
 - **Remove** the header
- Keep the rest
- Send an HTTP request to the server

HTTP Response

Status



```
HTTP/1.1 200 OK
```

```
Date: Mon, 20 Nov 2006 03:34:17 GMT
```

```
Server: Apache/1.3.19 (Unix) ...
```

```
Last-Modified: Mon, 28 Nov 2005 23:31:35 GMT
```

```
Content-Length: 129
```

```
Connection: Keep-Alive
```

```
Content-Type: text/html
```

Status indicates whether it was successful or not, if it is a “redirect”, etc.

Send the complete response back to the client.

How to Do Data Transfer

- Handle Broken Pipes
- Use Rio package
- `strcpy()` Vs `memcpy()`

Broken pipe error

- When writing to a socket whose connection has been closed prematurely at the other end
 - e.g. click "stop" on web browser
- For the first write, return normally. For subsequent writes
 - Kernel sends SIGPIPE signal, which terminates process by default
 - If SIGPIPE is blocked or caught, return -1 & set EPIPE.
- When reading from a socket whose connection has been closed
 - read returns a -1 with errno set to ECONNRESET

Handling Errors Gracefully

- We don't want to terminate the web proxy
 - Close the connection
 - Optionally, print an error message

Handling Client HTTP Request

- Use `rio_readlineb` to read the request
 - Consider the different return values
 - `<0, 0, >0`
 - "`\r\n`" signals end of the request
- `rio_writen` to send the request to the server

Handling Server HTTP Response

- `rio_readnb` to read the server response
 - binary data
 - `strcpy` Vs `memcpy`
- `rio_writen` to send the response to the client/browser

Testing Your Proxy

- Test the proxy on a variety of pages.
- Test the list given in the lab hand out.
- Test for both static and dynamic content.
- Test binary (e.g., images) file transfers.