HTTP Proxy

CPSC 441 - TUTORIAL 3

RACHEL MCLEAN

WINTER 2020

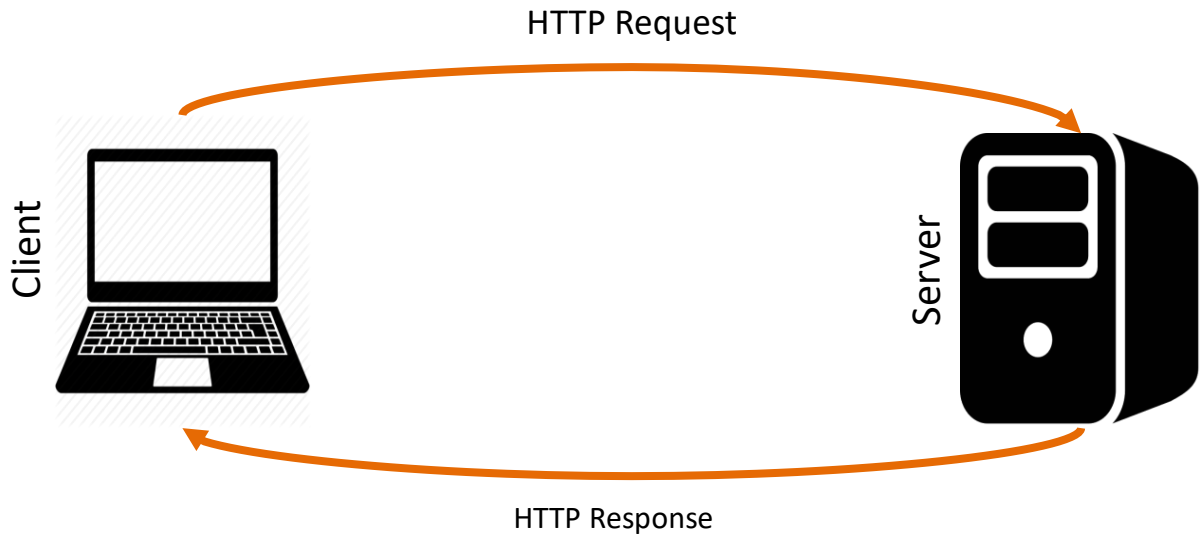
*Modified from slides by Reza Gholizadeh

What is HTTP

HyperText Transfer Protocol

- An application layer protocol used for transferring data between a client and server

Client-Server Model:



HTTP Request

HTTP Client sends request message and waits for response.

Method |space| **URL** |space| **Version** |new line|

Header-field-name: |space| **value** |new line|

…

|new line|

Body (data)

Ex:

```
GET /page.html HTTP/1.1
Host: www.website.com
Connection: close
User-agent: Mozilla/57.0.4
Accept-language: en
```

HTTP Request Methods

HTTP Clients use these methods to interact with an HTTP server

- GET
- POST
- HEAD
- PUT
- DELETE

HTTP **GET** and **POST**

GET:

- Most useful HTTP request
- Retrieve the information on requested url.

POST:

- For sending data to server (You don't know the actual URI)
- Data can be updated version of the same resource or a subordinate of it

```
GET /page.html HTTP/1.1
Host: www.website.com
Connection: close
Accept-language: en

POST /page.html HTTP/1.1
Host: www.website.com
User-agent: Mozilla/57.0.4
Accept-language: en
Connection: Keep-Alive
|new line|
(HTML data)
```

Other Methods

HEAD:

- For debugging purpose – acts like get but the response does not contain any data

PUT:

- For uploading an object to a specific path on the server (you know the exact URI)

DELETE:

- For deleting an object on the server

```
HEAD /page.html HTTP/1.1
Host: www.website.com
User-agent: Mozilla/57.0.4
Connection: keep-Alive
Accept-language: en-us

PUT /hi.html HTTP/1.1
Host: www.website.com
Connection: close
Content-Type: text/html
Content-Length: 182

|Blank Line|

<html>
<body>
<h1>Hi There</h1>
</body>
</html>

DELETE /textfiles/page.txt HTTP/1.1
Connection: close
Host: www.website.com
User-agent: Mozilla/52.0.01
```

HTTP Response

Status-Codes:

- 1xx: Informational
- 2XX: Success notifications
- 3XX: Redirections Notifications
- 4XX: Client Errors
- 5XX: Server Errors

Version |space| **Status-Code** |space| **Method** |new line|

Header-field-name: |space| **value** |new line|

…

|new line|

Body (data)

Ex:

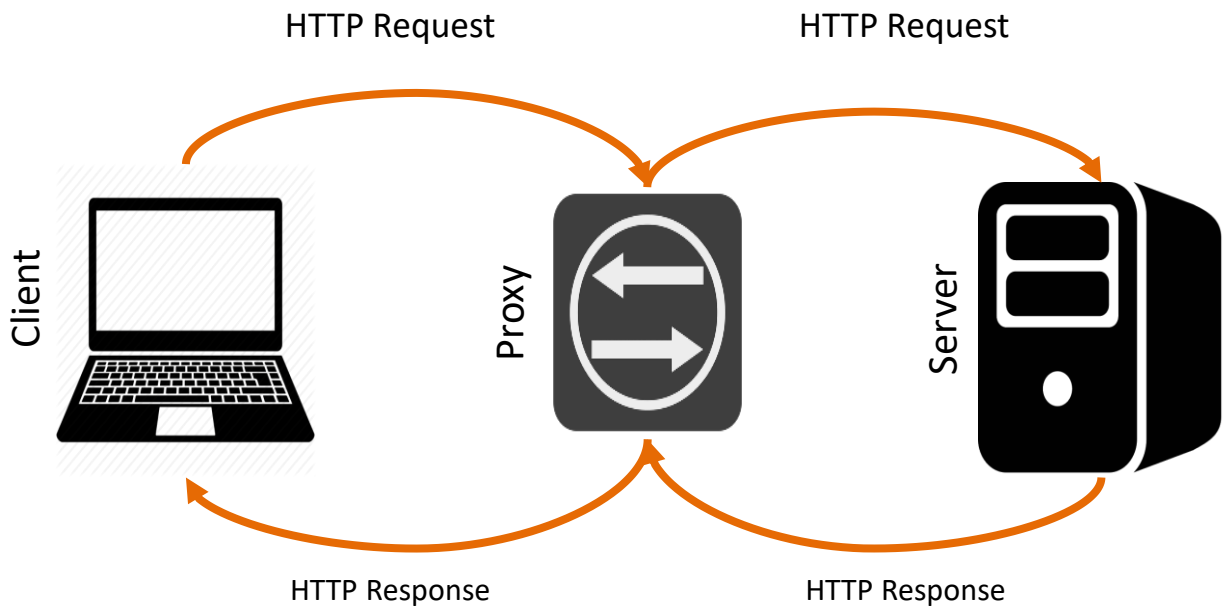
```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Jan 2020 15:44:04 GMT
Server: Apache/2.4.29 (CentOS)
Last-Modified: Tue, 18 Jan 2019 10:20:04 GMT
Content-Length: 6821
Content-Type: text/html
<html>…</html>
```

UNIVERSITY OF CALGARY

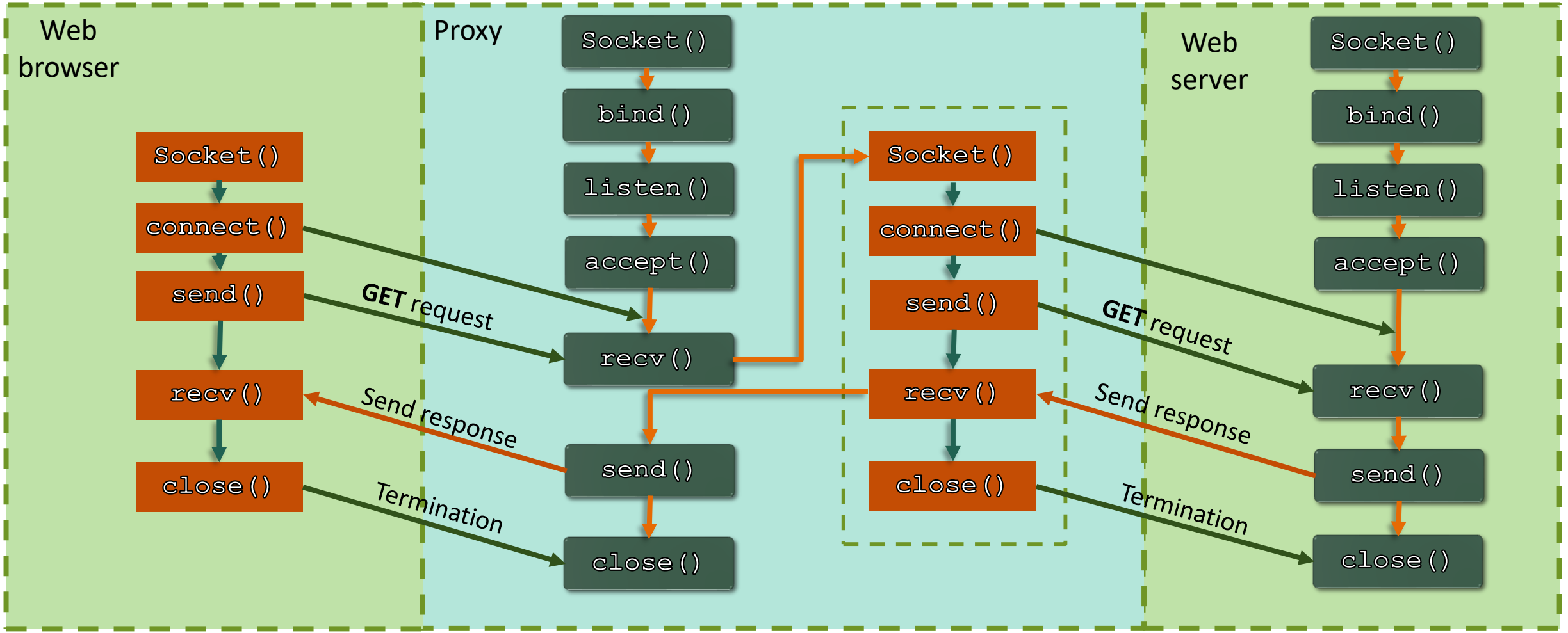
What is an HTTP PROXY?

For client: Proxy is a Server

For server: Proxy is a Client



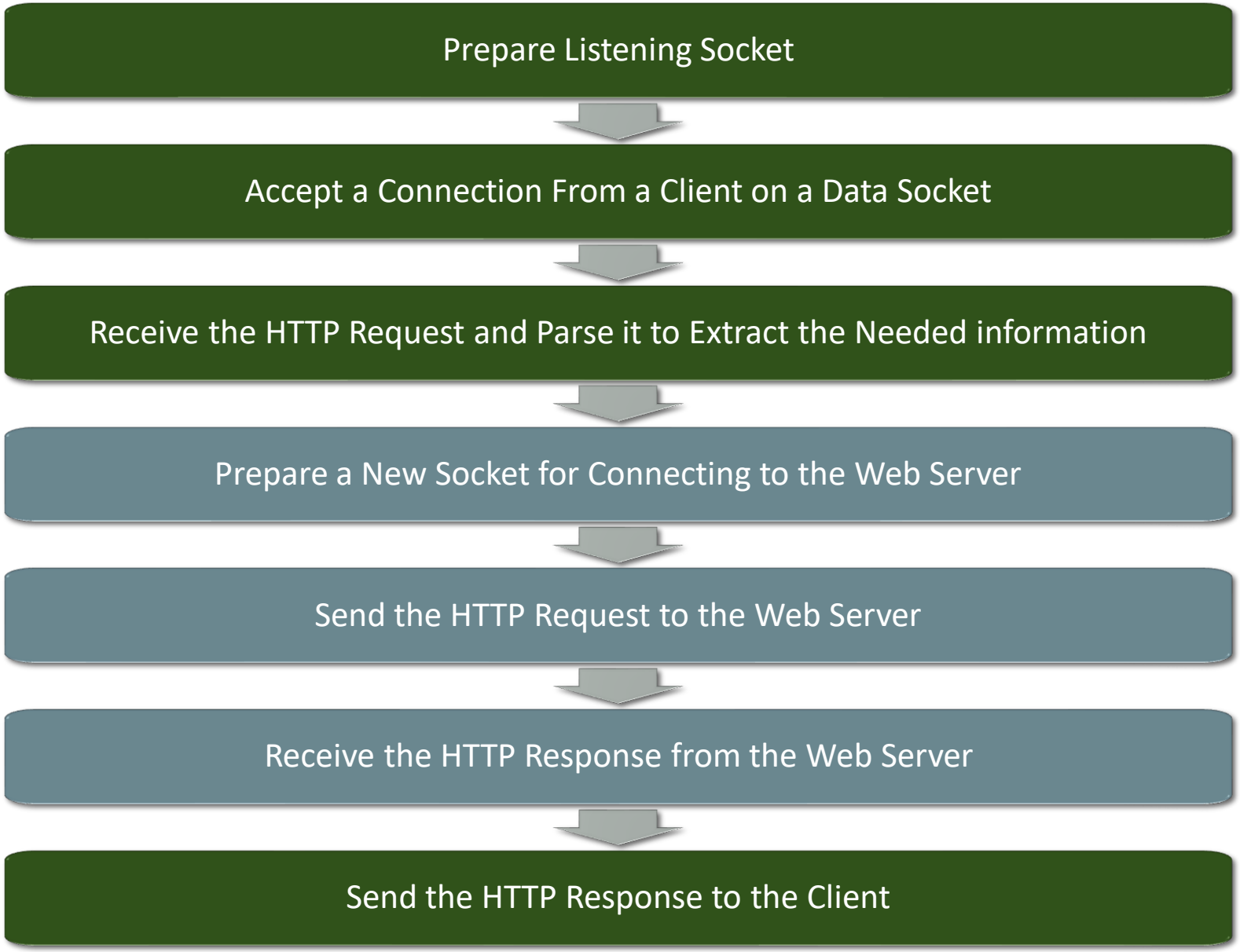
HTTP Proxy Architecture



HTTP Proxy in C

The main procedure of a proxy

- Some steps acts as a Server
- Some steps acts as a Client



Parse HTTP Request

Need to capture:

- URL: pages.cpsc.ucalgary.ca/~carey/CPSC441/testpage0.txt
- Host: pages.cpsc.ucalgary.ca
- Path: /~carey/CPSC441/testpage0.txt

The request is a string → use string functions for parsing

- `char *strcpy(char *dest, char *source, int num):`

  copies chars from *source* to *dest* and stopped after *num* element

- `int strlen(const char *source):`

  returns number of chars, excluding NULL

- `char *strchr(const char *source, const char ch):`

  returns pointer to first occurrence of *ch* in source; NULL if none

- `char *strstr(const char *source, const char *search):`

  return pointer to first occurrence of search in *source*

- `Char *strtok(char *str, const char *delim)`

  returns a pointer to the last token found in the string.

You can find All useful functions with examples here: https://www.tutorialspoint.com/c_standard_library/string_h.htm

Converting Hostname to IP Address

After extracting the value of Host field from the client request → Convert it to IP address

- Easiest solution → gethostbyname()

gethostbyname(char *address):

- Returns a structure of type **hostent** for the given ***address***.

```
1.  struct sockaddr_in server_addr;
2.  struct hostent address;
3.  address = gethostbyname("www.website.com");
4.  …
5.  bcopy((char *) address->h_addr, (char *)
    &server_addr.sin_addr.s_addr, address->h_length);
```

address: the instance of hostent structure that holds the address.

h_addr: the first element of address vector in hostent struct which holds the first IP address of the server

h_length: the length of the first IP address in bytes