

# COMP4621 Programming Project

Due Date: 23:59:00 May 5, 2015 (Tuesday)

## 1. Project requirements

In this project, we will implement a simple web server, which is able to:

- **Handle basic request of web page**

This means that when client requests a webpage, say the default one “/”, this web server should return a simple web page which shows your name and student ID (for the basic HTML programming, in tutorial# 3), as shown in Figure 1.

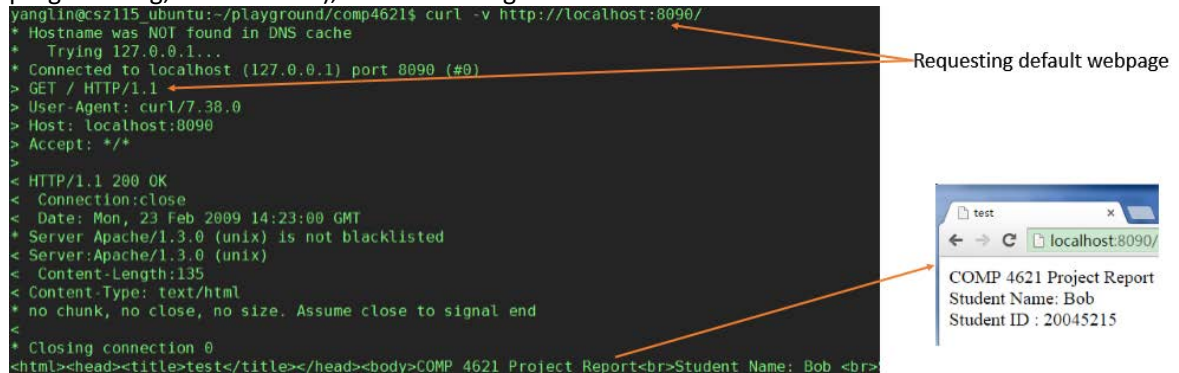


Figure 1. Response to web page request

- **Handle request of downloading large file with HTTP Range**

When client requests a file, *i.e.*, the “test.mp4” in our project, start a HTTP range download. Figure 2 demonstrates an example of HTTP range request/response.

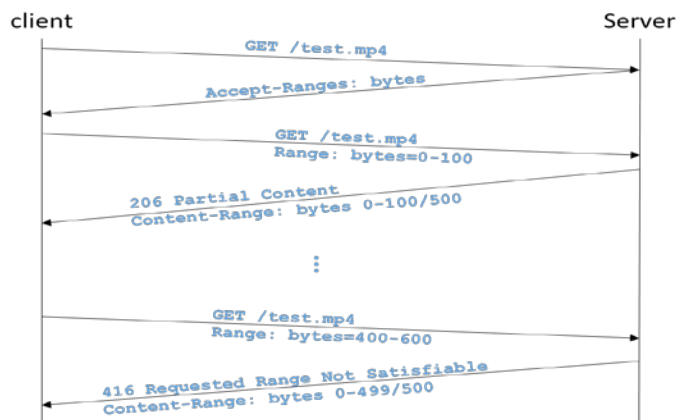


Figure 2. HTTP Range

The detailed behavior of webserver should be:

- a) If the client requests “test.mp4” (hardcode this name in your code), then show support of HTTP range.

```
yanglin@csz115_ubuntu:~/playground/comp4621$ curl -v http://localhost:8090/test.mp4
* Hostname was NOT found in DNS cache
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 8090 (#0)
> GET /test.mp4 HTTP/1.1
> User-Agent: curl/7.38.0
> Host: localhost:8090
> Accept: */*
>
< HTTP/1.1 200 OK
* Server Apache/1.3.0 (unix) is not blacklisted
< Server: Apache/1.3.0 (unix)
< Content-Length: 46858280
< Content-disposition: attachment; filename=test.mp4
< Content-Type: video/mp4
< Accept-Ranges: bytes
<
* transfer closed with 46858279 bytes remaining to read
* Closing connection 0
curl: (18) transfer closed with 46858279 bytes remaining to read
```

Requesting a video file

Indicating support of HTTP range

- b) If the client requests “test.mp4” with a valid range request, then replies specified data range.

```
yanglin@csz115_ubuntu:~/playground/comp4621$ curl -v -r 0-1000 http://localhost:8090/test.mp4 -o part_1
* Hostname was NOT found in DNS cache
*   Trying 127.0.0.1...
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
   0      0     0     0         0      0      0      0      0* Connected to localhost
> GET /test.mp4 HTTP/1.1
> Range: bytes=0-1000
> User-Agent: curl/7.38.0
> Host: localhost:8090
> Accept: */*
>
< HTTP/1.1 206 Partial Content
< Content-Type: video/mp4
< Content-Length: 1001
< Connection: keep-alive
* Server Apache/1.3.0 (unix) is not blacklisted
< Server: Apache/1.3.0 (unix)
< Content-disposition: attachment; filename=test.mp4
< Accept-Ranges: bytes
< Content-Range: bytes 0-1000/46858280
<
{ [data not shown]
100 1001 100 1001  0     0  276k    0 --:--:-- --:--:-- --:--:--  488k
```

Requesting a range

Return specified range

- c) If client request an invalid range, then return 416 Requested Range Not Satisfiable, use Content-Range header to indicate valid range

```
yanglin@csz115_ubuntu:~/playground/comp4621$ curl -v -r 1001-46858299 http://localhost:8090/test.mp4 -o part_2
* Hostname was NOT found in DNS cache
*   Trying 127.0.0.1...
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
   0      0     0     0         0      0      0      0      0* Connected to localhost (127.0.0.1)
> GET /test.mp4 HTTP/1.1
> Range: bytes=1001-46858299
> User-Agent: curl/7.38.0
> Host: localhost:8090
> Accept: */*
>
< HTTP/1.1 416 Requested Range Not Satisfiable
< Content-Type: video/mp4
< Connection: keep-alive
* Server Apache/1.3.0 (unix) is not blacklisted
< Server: Apache/1.3.0 (unix)
< Content-disposition: attachment; filename=test.mp4
< Accept-Ranges: bytes
< Content-Range: bytes 0-46858279
* no chunk, no close, no size. Assume close to signal end
<
{ [data not shown]
0     0  0     0  0     0  0 --:--:-- --:--:-- --:--:--  0
* Closing connection 0
```

Requesting a range

Return error and valid range

- **Handle multiple requests simultaneously**

This webserver should be able to handle multiple requests at the same time: while a client is downloading a large file, another client should be able to get a webpage from this web server.

## 2. Implementation details

You may start with the webserver example we have implemented in lab tutorial 3, some more functions need to implement:

- Parse HTTP Request.
- Parse Range-related header.
- Implement HTTP range.

Note:

- When you access the test.mp4, please use a relative path: `“./test.mp4”`.
- Do not package your code.

## 3. Marking Scheme

Your code will be compiled with Java 7, and tested by the curl command on a Linux server, the marking scheme is:

- Handle basic request of web page correctly (20%)
- Handle multiple requests simultaneously (20%)
- Handle request of downloading large file with Range support appropriately (30%)
  - Support of range (10%)
  - Handle valid range request (10%)
  - Handle invalid range request (10%)
- Project report (30%)

## 4. Submission

Please submit a folder via CASS, with folder name like `“studentName_studentID”`, which includes:

- All your source code in Java.
- A two-page project report (Times New Roman, 11pts, single column, in PDF), which explains:
  - a) Structure of your code, give explanation for import functions
  - b) How to handle multiple HTTP requests simultaneously?
  - c) How to handle HTTP request of webpage? Please paste your test result with curl
  - d) How to handle HTTP Range requests? Please paste your test result with curl
- **Do NOT submit test.mp4, just hardcode this name in your code.**