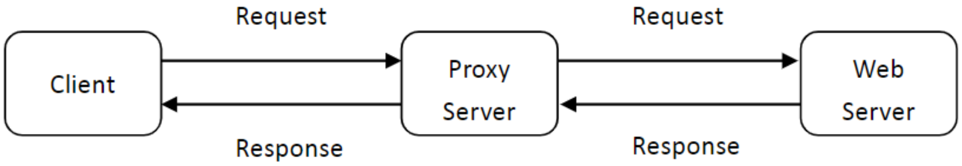
Computer Networks Project 1
A Simple HTTP Web Proxy

Note: Please carefully read through it!

A. Background

The web proxy sits between the web client and the web server to relay HTTP traffics. Generally, when the client makes a request, the request is sent to the web server. The web server then processes the request and sends back a response message to the requesting client. In order to improve the performance, we create a proxy server between the client and the web server. Now, both the request message sent by the client and the response message delivered by the web server pass through the proxy server. In other words, the client requests the objects via the proxy server. The proxy server will forward the client's request to the web server (if no cache hit). The web server will then generate a response message and deliver it to the proxy server, which in turn sends it to the client. You should have a clear understanding of how web proxy works before starting your implementation. You can review our textbook or find more information online.



B. Assignment Requirement

In this project, you will use Python to develop a small web proxy server that is able to cache web pages. It is a very simple proxy server that only understands simple GET requests of .html files.

1. Your proxy source code should be in one single .py file named *proxy.py*

Your proxy should expect a command-line argument that specifies a listening port number. Do NOT hard code your port number in your proxy. An example of execution of your proxy should be like:

```
python3 proxy.py 10001
```

2. Your proxy should support HTTP 1.1 and caching

Your proxy interacts with both clients and web servers over HTTP/1.1. Specifically, it only needs to support GET method. Your proxy is capable of accepting and parsing an HTTP request. If the requested file is in the cache, the proxy will respond to the user with the cached content; otherwise, the proxy will forward the request to the origin web server, cache the file sent by the server, and relay the response data to the client.

3. Listening and parsing HTTP requests.

After your proxy starts, it should listen to the port and wait for incoming client TCP connections. Once a client's connection is accepted, the proxy should receive a request from the client and then check for a properly-formatted HTTP request. Specifically, you will parse the client message to ensure that the proxy receives a request that contains a valid request line:

```
<METHOD>  <URL>  <HTTP VERSION>
```

In this project, client's requests to the proxy must be in their absolute URI form (see RFC 1945, Section 5.1.2), e.g.,

```
GET  http://zhiju.me/networks/valid.html HTTP/1.1
```

Your proxy needs to parse the requested URL to retrieve three pieces of information: the requested *host*, *port*, and *path*. If the hostname indicated in the absolute URL does not have a port specified, you should use the default HTTP port 80.

4. Respond client appropriately

a) No Cache Hit: getting data from the remote web server

After parsing the URL, if there is no cache hit of the requested file, your proxy will establish a TCP connection to the requested host and send the HTTP request for the file. The proxy should always send the request in the *relative URL + Host header* format. For instance:

A request sent from the client:

```
GET  http://zhiju.me/networks/valid.html HTTP/1.1
```

Then, the request sent from your proxy to remote server:

```
GET /networks/valid.html   HTTP/1.1
Host: zhiju.me
Connection: close
(Additional client specified headers, if any...)
```

Note that your proxy should send HTTP/1.1 flags and a Connection: close header to the web server, so that the server will close the connection after its response is fully transmitted, as opposed to keeping open a persistent connection. To add new headers or modify existing ones, do it when parsing the HTTP request.

If your proxy sends the request correctly, the remote web server will respond to the request. If the server responds with status code 200, your proxy should cache the content of the response as-is and then relay it to the client; if the server responds with status code 404, your proxy should relay this 404 message to the client; if the server responds with status code other than 200 and 404, your proxy should create a 500 'Internal Error' message and send it to the client. Since there is no cache hit in the last two scenarios, your proxy should also include a customized message field 'Cache-Hit: 0'.

Note that you should cache the files into the relative folder './cache/SUB_FOLDERS_IF_NEEDED'. Your proxy should be able to create such folder(s) if they do not exist. You can assume the maximum size of the requested file is 16Mb. Besides, if the HTTP request from the client is malformed or not supported or the server returned status other than 200 and 404, your proxy should return 500 'Internal Error' to the client.

b) Cache Hit: loading from cache and sending back to the client

After parsing the URL, if the requested file is in the cache, your proxy will directly load the cached file and send it back to the client. Since there is a cache hit, your proxy should also include a customized message field 'Cache-Hit: 1'.

5. Allowed libraries

With Python, there are various solutions and libraries for a web proxy. In this project, **only the following four libraries/functions are allowed.** If you believe you need more libraries, post a discussion on Slack or Canvas discussion (so that everyone can see it), I'll address case by case. Using unallowed libs would result in **zero points** on this assignment.

from socket import *	It's okay to use all the functions in the socket lib
from urllib.parse import urlparse	Only this urlparse function can be used
import sys	For parsing argument (port number) only
from pathlib import Path	For folder access (creating and reading)

C. Grading Rubrics

All the submissions will be graded on the CS1 server. Make sure your code can be executed on this server. Otherwise, it will result in zero points on this assignment. The detailed grading rubrics can be found on Canvas.

Label	Notes
Functionality	The proxy should behave exactly as described in this documentation. • It displays the received HTTP response on the terminal screen given a user GET request by using telnet. • Its cache system works correctly. • The status, e.g., Cache-Hit, is displayed correctly. • It prints out appropriate messages. Do not print out massive testing/debugging messages. • It should only use allowed libraries
Code style	• Code should have appropriate comments, consistent style, indentation, etc. • Code output should be well-organized
Overriding policy	If the code cannot be executed or can be barely executed (exceptions, for instance), it results in zero points on this assignment.
Late submission	Please refer to the late submission policy on Syllabus.
Academic Integrity	Strictly enforced. Please check more on Syllabus.

D. Development Tips

Your proxy supports only HTTP/1.1 and expects a request message with *only* a GET request line. Upon receiving a client's HTTP/1.1 request, your proxy either loads the requested file from cache or requests the file from the server, and then sends it back to the client.

Port Numbers

The port # (10000 - 10500) on cs1 can be used by other classes in the department. In order to reduce the risk of using the same port #, we develop a simple algorithm to calculate the port # your proxy will use.

Port # = 10000 + (your student ID) % 500

If you still believe this port is in use, then do a linear probing to try subsequent port numbers.

Testing

All the development is supposed to be on our cs1 server. You should test your program on the cs1 server before submission.

To start your proxy on port# 10000, run the command below on the cs1 server:

```
python3 proxy.py 10000
```

To test your proxy, use `telnet` on the cs1 server:

```
telnet 127.0.0.1 10000
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
GET http://zhiju.me/networks/valid.html HTTP/1.1
```

If your proxy is working correctly, the headers and HTML of the `valid.html` should be displayed on your terminal screen and this HTML file should be cached (if not cached before) in the correct cache folder.

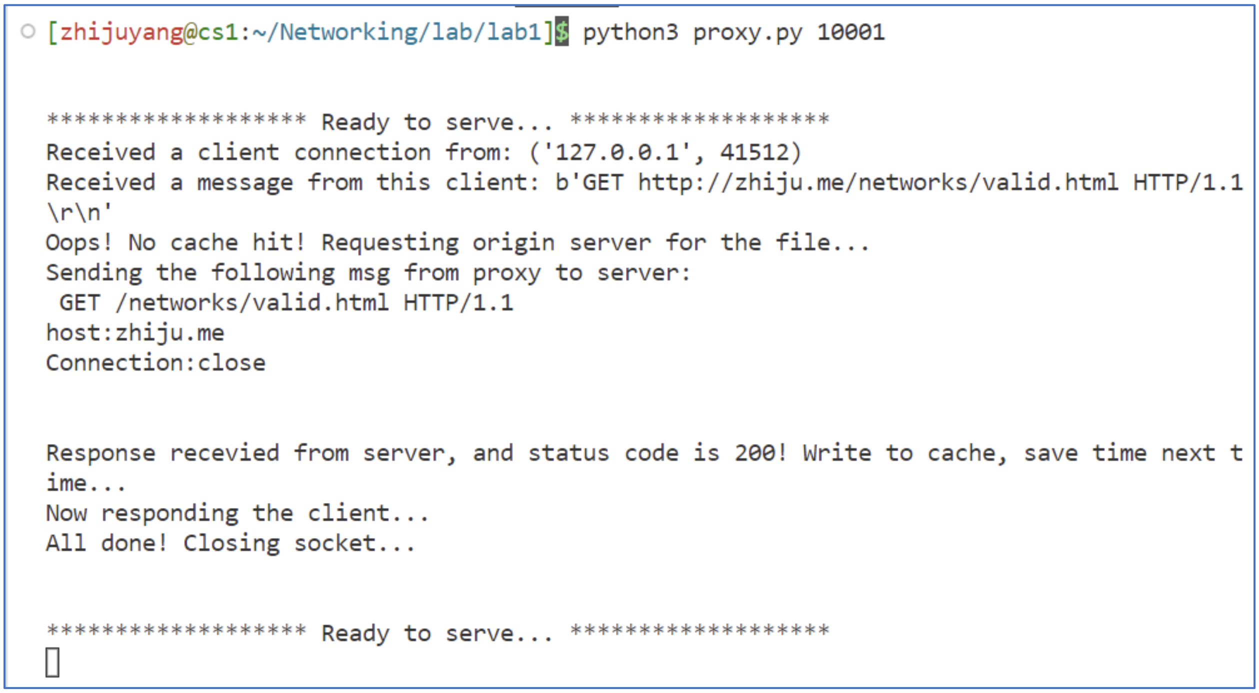
Sample URLs for testing

You may use the following URLs for testing. Note that these .html files are on my personal server. Please do not overwhelm this server. You are encouraged to find more pages on other websites to comprehensively test your code.

http://zhiju.me/networks/valid.html	For testing status 200
http://zhiju.me/networks/404.html	For testing status 404
http://zhiju.me/networks/500.html	For testing status 500

Sample Output (you should intensively test your code)

Requesting valid.html that is not in proxy cache



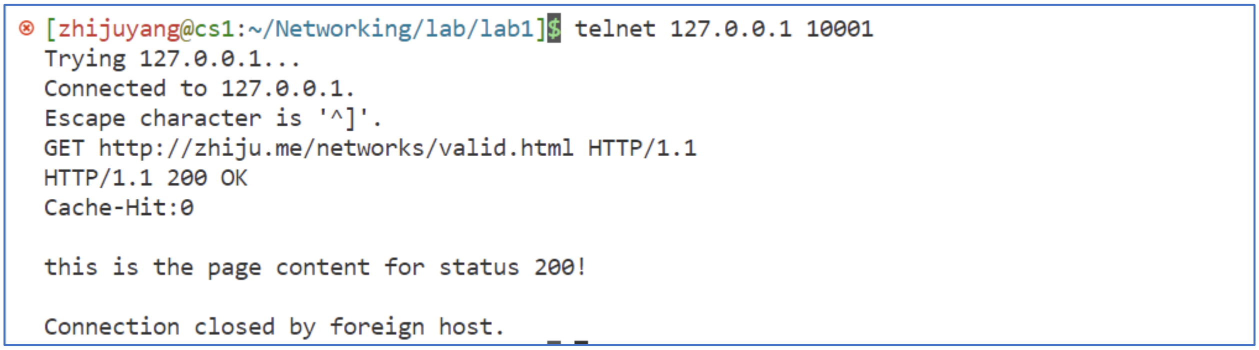
```
○ [zhijuyang@cs1:~/Networking/lab/lab1]$ python3 proxy.py 10001


  ******************* Ready to serve... *******************
  Received a client connection from: ('127.0.0.1', 41512)
  Received a message from this client: b'GET http://zhiju.me/networks/valid.html HTTP/1.1
  \r\n'
  Oops! No cache hit! Requesting origin server for the file...
  Sending the following msg from proxy to server:
   GET /networks/valid.html HTTP/1.1
  host:zhiju.me
  Connection:close


  Response recevied from server, and status code is 200! Write to cache, save time next t
  ime...
  Now responding the client...
  All done! Closing socket...


  ******************* Ready to serve... *******************
  ▯
```

Proxy output



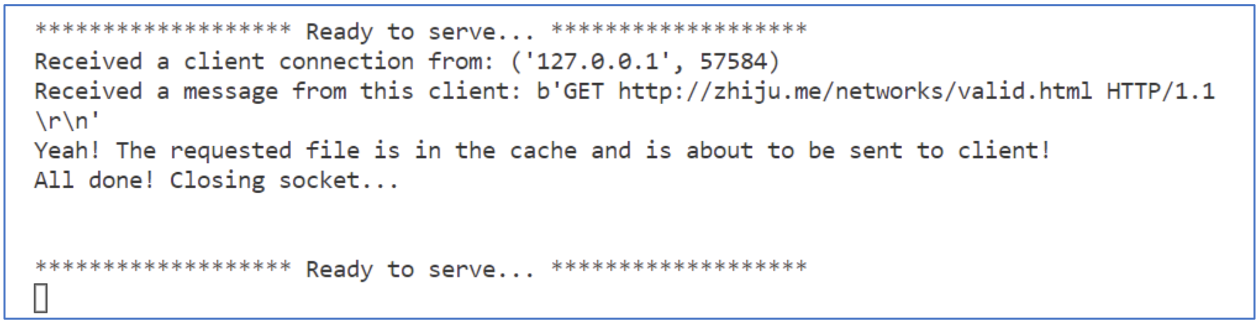
```
⊗ [zhijuyang@cs1:~/Networking/lab/lab1]$ telnet 127.0.0.1 10001
  Trying 127.0.0.1...
  Connected to 127.0.0.1.
  Escape character is '^]'.
  GET http://zhiju.me/networks/valid.html HTTP/1.1
  HTTP/1.1 200 OK
  Cache-Hit:0

  this is the page content for status 200!

  Connection closed by foreign host.
```

Client (using telnet) output

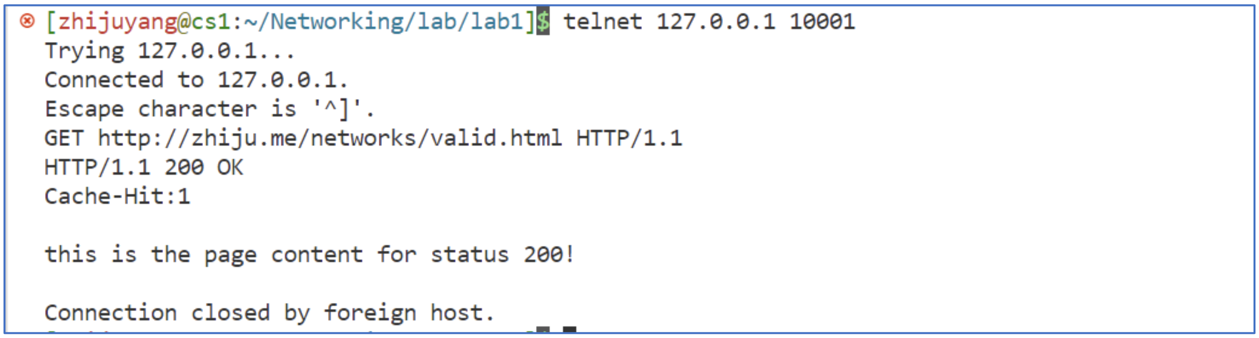
Requesting valid.html which is already in the proxy cache



```
******************* Ready to serve... *******************
Received a client connection from: ('127.0.0.1', 57584)
Received a message from this client: b'GET http://zhiju.me/networks/valid.html HTTP/1.1
\r\n'
Yeah! The requested file is in the cache and is about to be sent to client!
All done! Closing socket...


******************* Ready to serve... *******************
```

Proxy output



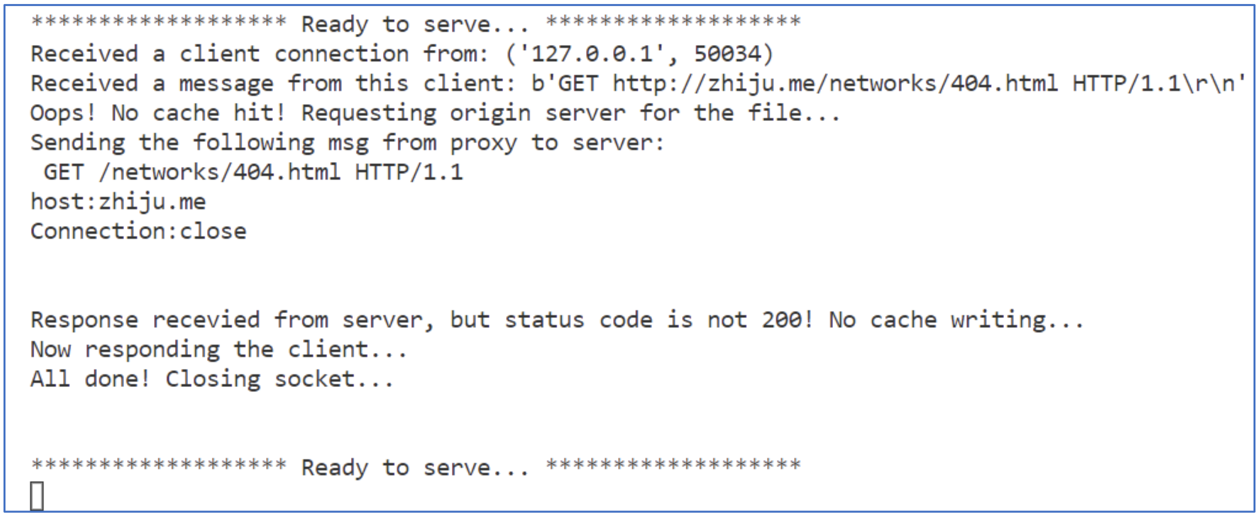
```
[zhijuyang@cs1:~/Networking/lab/lab1]$ telnet 127.0.0.1 10001
 Trying 127.0.0.1...
 Connected to 127.0.0.1.
 Escape character is '^]'.
 GET http://zhiju.me/networks/valid.html HTTP/1.1
 HTTP/1.1 200 OK
 Cache-Hit:1

 this is the page content for status 200!

 Connection closed by foreign host.
```

Client (using telnet) output

Requesting 404.html that does not exist on the server – 404 error

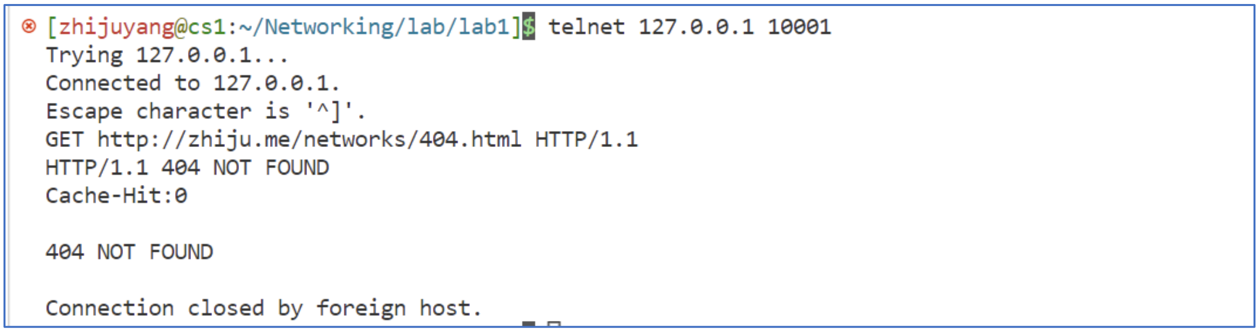


```
******************* Ready to serve... *******************
Received a client connection from: ('127.0.0.1', 50034)
Received a message from this client: b'GET http://zhiju.me/networks/404.html HTTP/1.1\r\n'
Oops! No cache hit! Requesting origin server for the file...
Sending the following msg from proxy to server:
 GET /networks/404.html HTTP/1.1
host:zhiju.me
Connection:close


Response recevied from server, but status code is not 200! No cache writing...
Now responding the client...
All done! Closing socket...


******************* Ready to serve... *******************
```

Proxy output



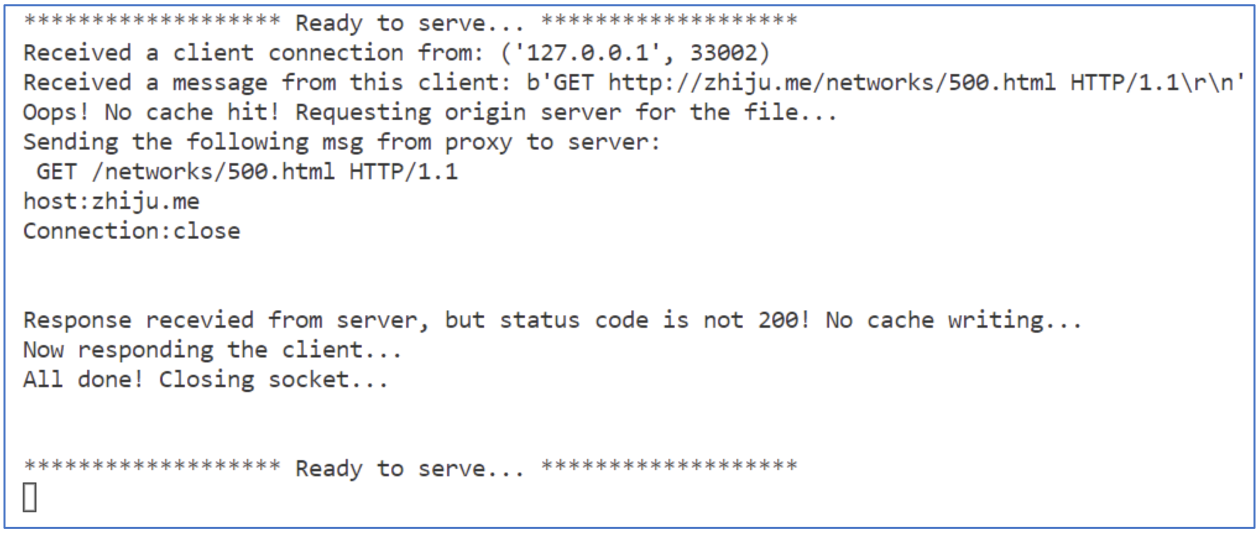
```
⊗ [zhijuyang@cs1:~/Networking/lab/lab1]$ telnet 127.0.0.1 10001
  Trying 127.0.0.1...
  Connected to 127.0.0.1.
  Escape character is '^]'.
  GET http://zhiju.me/networks/404.html HTTP/1.1
  HTTP/1.1 404 NOT FOUND
  Cache-Hit:0

  404 NOT FOUND

  Connection closed by foreign host.
```

Client (using telnet) output

Requesting a file but error occurs – 500 error

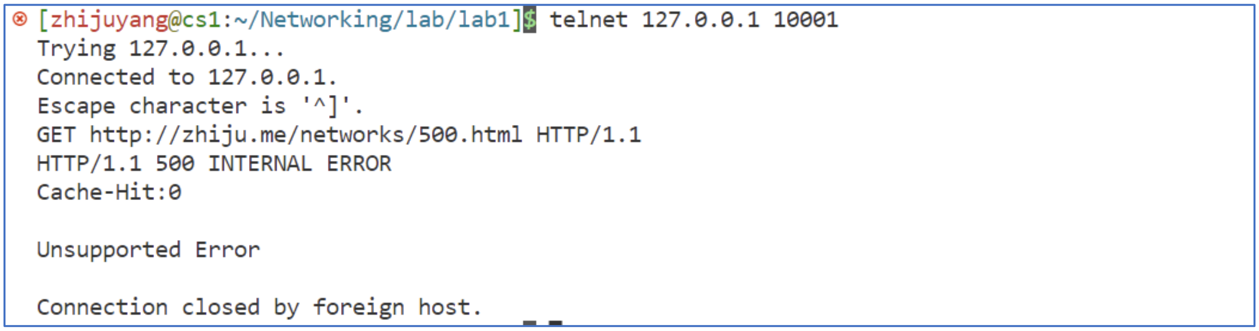


```
******************* Ready to serve... *******************
Received a client connection from: ('127.0.0.1', 33002)
Received a message from this client: b'GET http://zhiju.me/networks/500.html HTTP/1.1\r\n'
Oops! No cache hit! Requesting origin server for the file...
Sending the following msg from proxy to server:
 GET /networks/500.html HTTP/1.1
host:zhiju.me
Connection:close


Response recevied from server, but status code is not 200! No cache writing...
Now responding the client...
All done! Closing socket...


******************* Ready to serve... *******************
```

Proxy output



```
⊗ [zhijuyang@cs1:~/Networking/lab/lab1]$ telnet 127.0.0.1 10001
  Trying 127.0.0.1...
  Connected to 127.0.0.1.
  Escape character is '^]'.
  GET http://zhiju.me/networks/500.html HTTP/1.1
  HTTP/1.1 500 INTERNAL ERROR
  Cache-Hit:0

  Unsupported Error

  Connection closed by foreign host.
```

Client (using telnet) output