PA 2 Report

2015121088 강소영

[Introduction/Reference]

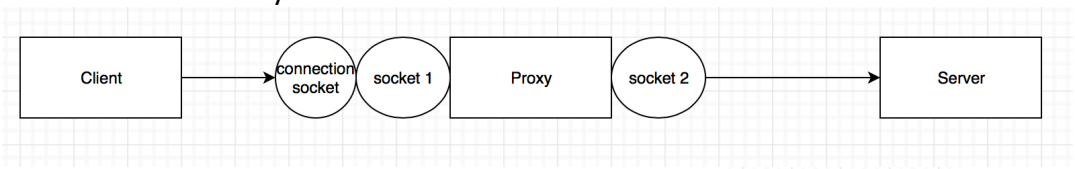
Python3, Mac&Ubuntu

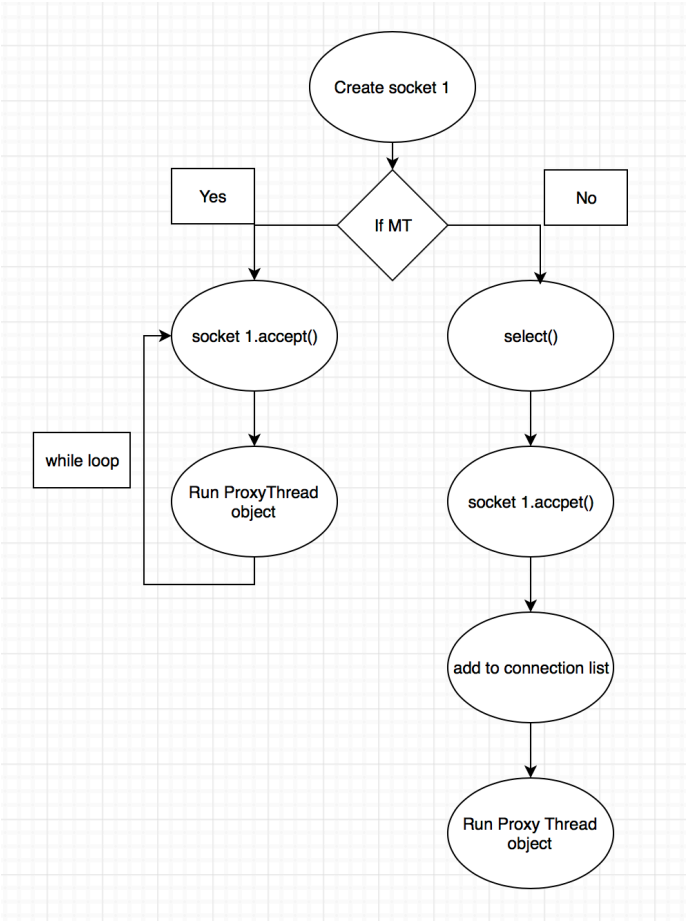
Measurement method: Used Developer Tools in network tab of chrome browser F12 key, created timestamp with python datetime module for logging.

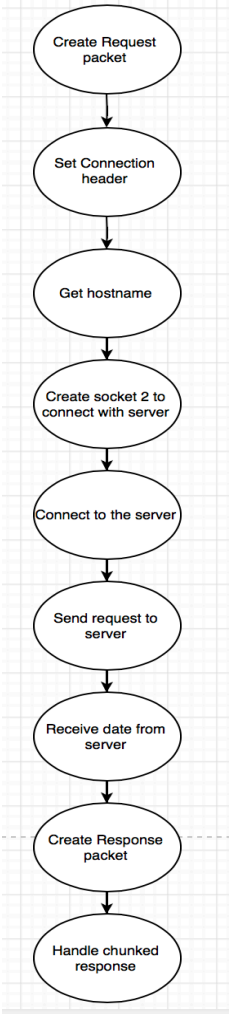
Reference: Forward/Reverse proxy image: https://www.incapsula.com/cdn-guide/glossary/reverse-proxy.html

[Flow chart]

Two sockets for Proxy

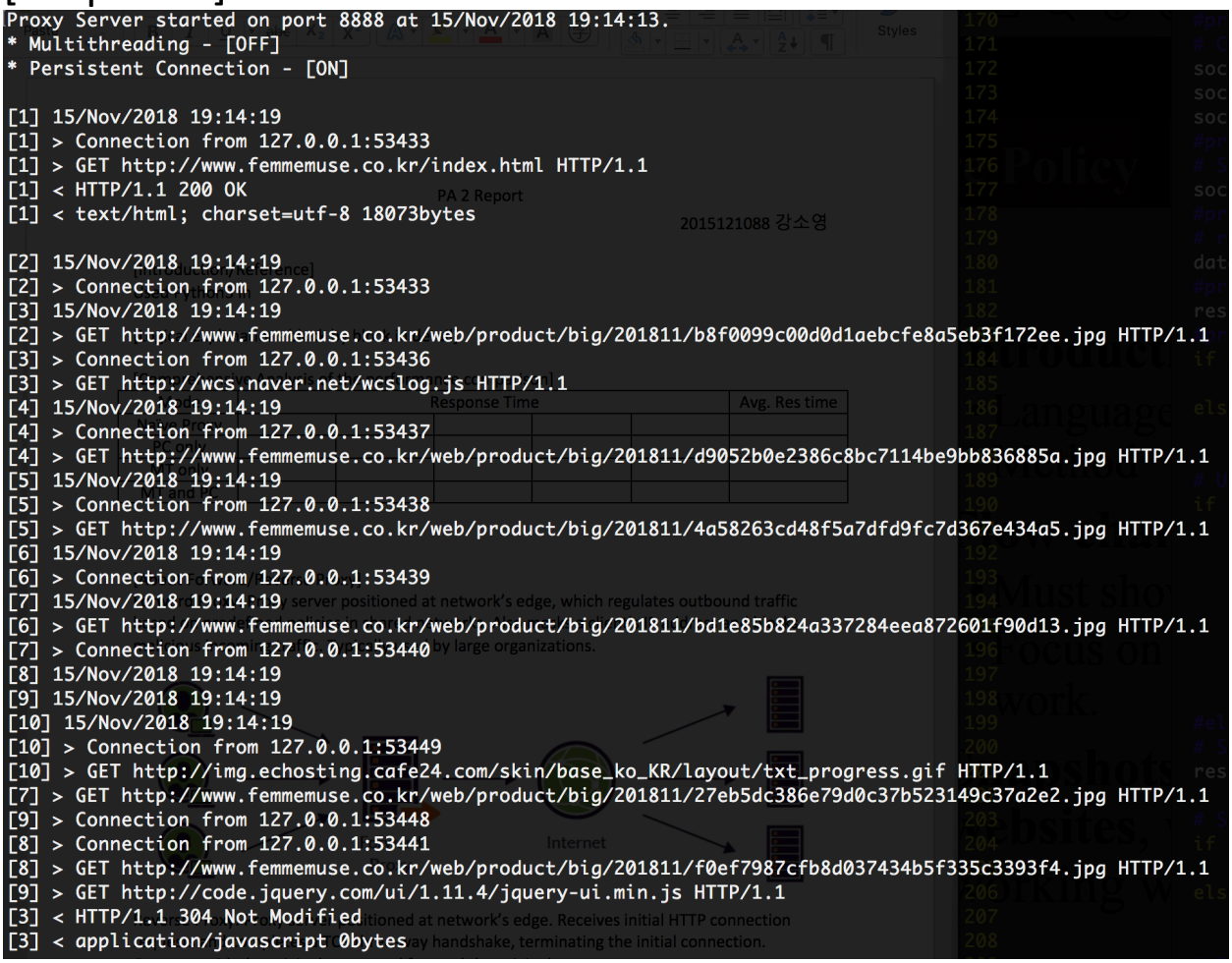


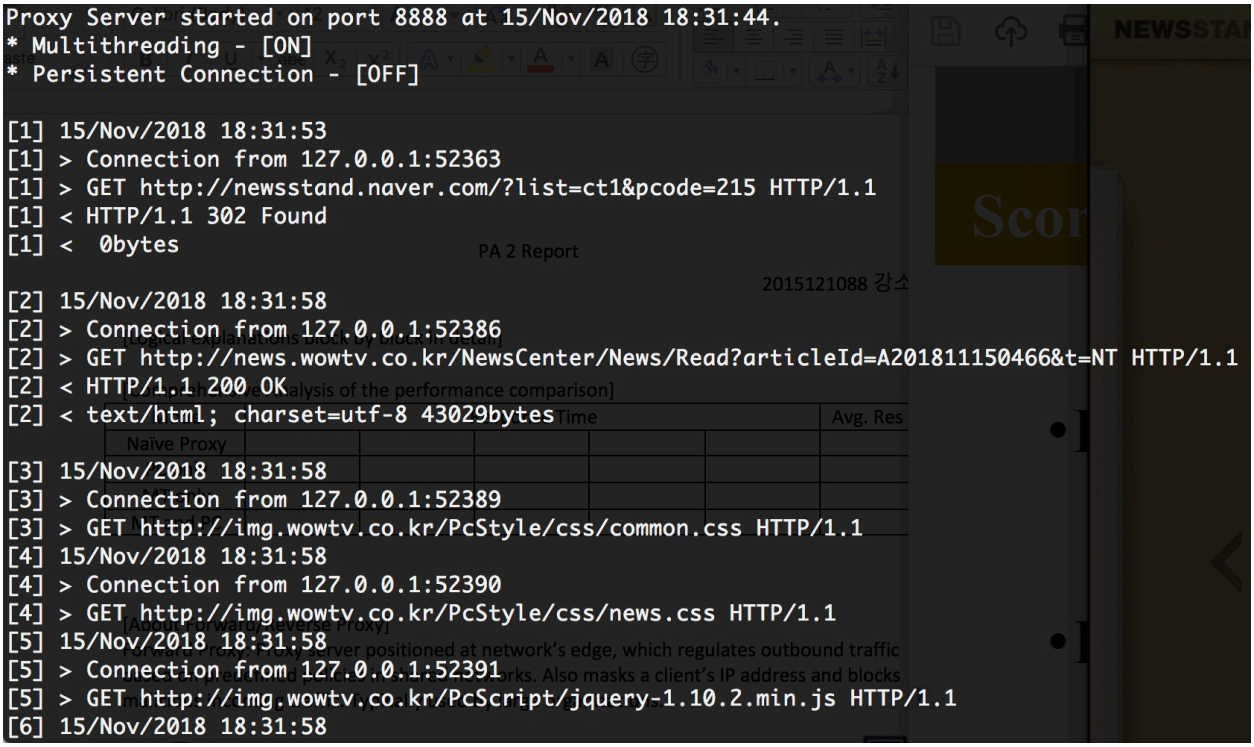


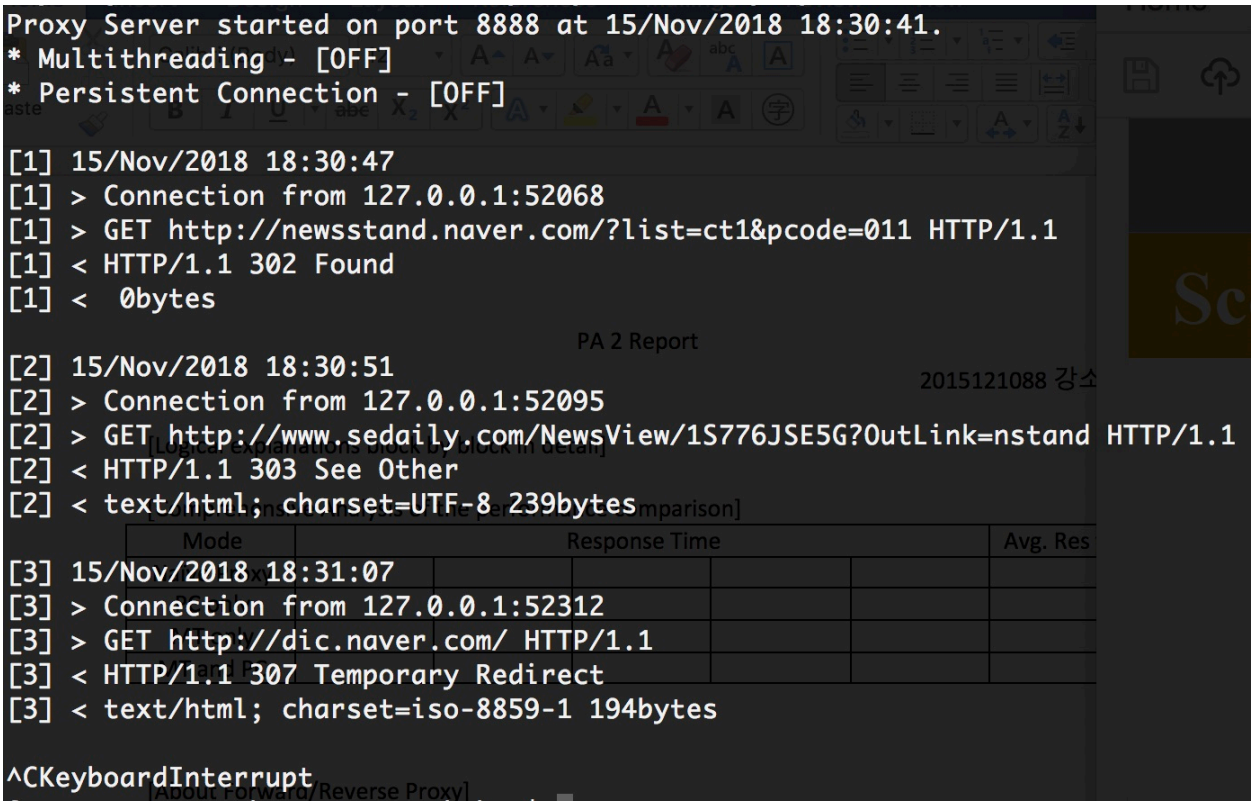


Main, Run

[Snapshots]


Proxy Server started on port 8888 at 15/Nov/2018 19:14:13.
* Multithreading - [OFF]
* Persistent Connection - [ON]
[1] 15/Nov/2018 19:14:19
[1] > Connection from 127.0.0.1:53433
[1] > GET http://www.femmemuse.co.kr/index.html HTTP/1.1
[1] < HTTP/1.1 200 OK
[1] < text/html; charset=utf-8 18073bytes
[2] 15/Nov/2018 19:14:19
[2] > Connection from 127.0.0.1:53433
[3] 15/Nov/2018 19:14:19
[2] > GET http://www.femmemuse.co.kr/web/product/big/201811/b8f0099c00d0d1aebcfe8a5eb3f172ee.jpg HTTP/1.1
[3] > Connection from 127.0.0.1:53436
[3] > GET http://wcs.naver.net/wcslog.js HTTP/1.1
[4] 15/Nov/2018 19:14:19
[4] > Connection from 127.0.0.1:53437
[4] > GET http://www.femmemuse.co.kr/web/product/big/201811/d9052b0e2386c8bc7114be9bb836885a.jpg HTTP/1.1
[5] 15/Nov/2018 19:14:19
[5] > Connection from 127.0.0.1:53438
[5] > GET http://www.femmemuse.co.kr/web/product/big/201811/4a58263cd48f5a7dfd9fc7d367e434a5.jpg HTTP/1.1
[6] 15/Nov/2018 19:14:19
[6] > Connection from 127.0.0.1:53439
[7] 15/Nov/2018 19:14:19
[6] > GET http://www.femmemuse.co.kr/web/product/big/201811/bd1e85b824a337284eea872601f90d13.jpg HTTP/1.1
[7] > Connection from 127.0.0.1:53440
[8] 15/Nov/2018 19:14:19
[9] 15/Nov/2018 19:14:19
[10] 15/Nov/2018 19:14:19
[10] > Connection from 127.0.0.1:53449
[10] > GET http://img.echosting.cafe24.com/skin/base_ko_KR/layout/txt_progress.gif HTTP/1.1
[7] > GET http://www.femmemuse.co.kr/web/product/big/201811/27eb5dc386e79d0c37b523149c37a2e2.jpg HTTP/1.1
[9] > Connection from 127.0.0.1:53448
[8] > Connection from 127.0.0.1:53441
[8] > GET http://www.femmemuse.co.kr/web/product/big/201811/f0ef7987cfb8d037434b5f335c3393f4.jpg HTTP/1.1
[9] > GET http://code.jquery.com/ui/1.11.4/jquery-ui.min.js HTTP/1.1
[3] < HTTP/1.1 304 Not Modified
[3] < application/javascript 0bytes


Proxy Server started on port 8888 at 15/Nov/2018 18:31:44.
* Multithreading - [ON]
* Persistent Connection - [OFF]
[1] 15/Nov/2018 18:31:53
[1] > Connection from 127.0.0.1:52363
[1] > GET http://newsstand.naver.com/?list=ct1&pcode=215 HTTP/1.1
[1] < HTTP/1.1 302 Found
[1] < 0bytes
[2] 15/Nov/2018 18:31:58
[2] > Connection from 127.0.0.1:52386
[2] > GET http://news.wowtv.co.kr/NewsCenter/News/Read?articleId=A201811150466&t=NT HTTP/1.1
[2] < HTTP/1.1 200 OK
[2] < text/html; charset=utf-8 43029bytes
[3] 15/Nov/2018 18:31:58
[3] > Connection from 127.0.0.1:52389
[3] > GET http://img.wowtv.co.kr/PcStyle/css/common.css HTTP/1.1
[4] 15/Nov/2018 18:31:58
[4] > Connection from 127.0.0.1:52390
[4] > GET http://img.wowtv.co.kr/PcStyle/css/news.css HTTP/1.1
[5] 15/Nov/2018 18:31:58
[5] > Connection from 127.0.0.1:52391
[5] > GET http://img.wowtv.co.kr/PcScript/jquery-1.10.2.min.js HTTP/1.1
[6] 15/Nov/2018 18:31:58


Proxy Server started on port 8888 at 15/Nov/2018 18:30:41.
* Multithreading - [OFF]
* Persistent Connection - [OFF]
[1] 15/Nov/2018 18:30:47
[1] > Connection from 127.0.0.1:52068
[1] > GET http://newsstand.naver.com/?list=ct1&pcode=011 HTTP/1.1
[1] < HTTP/1.1 302 Found
[1] < 0bytes
[2] 15/Nov/2018 18:30:51
[2] > Connection from 127.0.0.1:52095
[2] > GET http://www.sedaily.com/NewsView/1S776JSE5G?OutLink=nstand HTTP/1.1
[2] < HTTP/1.1 303 See Other
[2] < text/html; charset=UTF-8 239bytes
[3] 15/Nov/2018 18:31:07
[3] > Connection from 127.0.0.1:52312
[3] > GET http://dic.naver.com/ HTTP/1.1
[3] < HTTP/1.1 307 Temporary Redirect
[3] < text/html; charset=iso-8859-1 194bytes
^CKeyboardInterrupt

[Logical explanations block by block in detail]

Function parseHTTP(data):
Parse data into First line, Header lines, Body field of packet, delimited by CRLF. Note to create a dictionary for Header lines for future reference.

Function recvData(conn):
Create a packet from data received through the socket. Note to handle chunked and non-chunked packets differently, unchunking the chunks and make a body field to be a whole.

Class HTTPPacket
Function getHeader: If the packet does not have a specified header, return empty string, otherwise return the matching value.

Function setHeader: If the given value is empty string, delete the header from the packet. Otherwise, add new header with the specified value.

Function getURL: Note that URL comes in the second field of the first line in the packet. Split the line by space' ', then take the value in index 1.

Class ProxyThread
__init__: Create a ProxyThread object, taking connection socket, client address, and connection type-whether it's a persistent connection or not.

Function run: Make a global variable conn_no, indicating a connection number. Note that this variable is a global variable in order to keep track of all the connections throughout all the threads and all the sockets. This function receives data from connection socket of its own member variable and make a request packet. Based on the connection type, this function sets Connection header to be keep-alive(persistent connection) or close(not a persistent connection). For the sake of accessibility, remove the proxy information by calling setHeader function with header Proxy-Connection. Extract a host information from request packet, and create a socket(which is the second socket) in this case as a client. Connect this socket to the host server and send all the requests received from the client. Receive response data from the host. Again, set the connection header, and print out all the information so far. Send the response packet back to the original client and close the sockets based on the connection type.

Function main: Start with creating a server-side socket, binding with 0.0.0.0 and user input port number. Mark current time using datetime module, print it out. Make two flags mt and pc, each indicates multithreading and persistent connection. Initiate global variable conn_no(connection number) to be 0. In case of multithreading, keep iterate through the while loop. For each iteration of this while loop, accept client connection with the socket previously created, then create a proxyThread object with this connection. Calling start() on this proxyThread object will automatically call run() function defined beforehand. On the other hand if it is not a multithreading mode, use a select() function to handle multiple connections in a single thread.

List variable connections and dictionary variable util are helping this single threading with select function. Connections is a list of connection sockets, util stores a client address of each connection socket, having connection socket as a key and client address as a value. What remains is to iteratively create proxyThread object and run it, for each connection in connections.

[Comprehensive Analysis of the performance comparison]

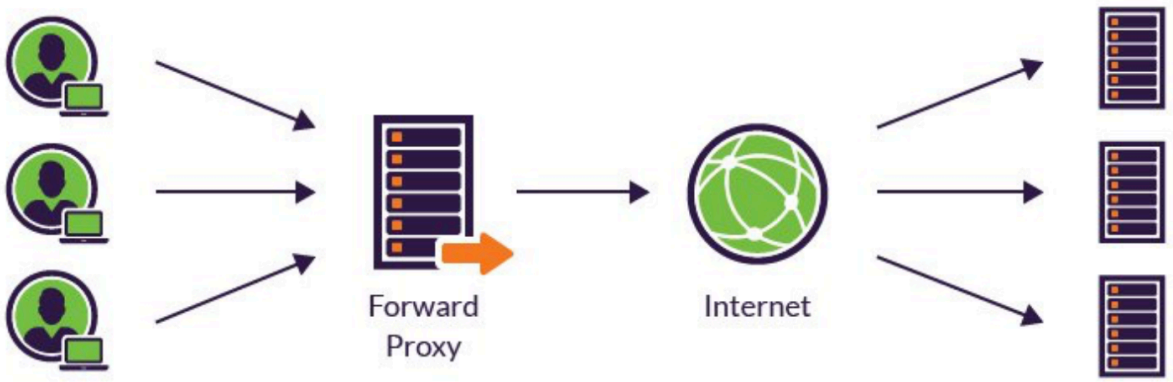
Mode	Website	Response time (5 times repetition)					Avg. Response time
Naïve proxy	Yonsei CS department	1.35s	2.44s	2.22s	1.32s	2.19s	1.904s
	Naver dictionary:search for proxy	1.6s	1.31s	1.32s	1.43s	1.26s	1.384s
	www.bbc.com	4.71	2.81	2.45	3.25	2.81	3.206s
Total average response time in Naïve proxy mode							2.165s
PC only	Yonsei CS department	1.26s	2.05s	1.82s	2.25s	1.97s	1.87s
	Naver dictionary:search for proxy	1.32s	1.32s	1.35s	1.23s	1.33s	1.31s
	www.bbc.com	2.71	2.56	2.55	3.65	2.92	2.878s
Total average response time in PC only mode							2.019s
MT only	Yonsei CS department	1.23s	1.82s	1.99s	2.13s	2.15s	1.858s
	Naver dictionary:search for proxy	1.45s	1.29s	1.37s	1.44s	1.35s	1.38s
	www.bbc.com	2.51	3.29	2.89	3.76	2.70	3.03s
Total average response time in MT only mode							2.089s
PC and MT	Yonsei CS department	1.24	1.28	1.24	1.25	1.17	1.236
	Naver dictionary:search for proxy	1.39	1.36	1.31	1.41	1.30	1.354
	www.bbc.com	3.22	2.68	2.92	2.43	2.70	2.79
Total average response time in PC and MT mode							1.793s

Conclusion: PC&MT < PC < MT < Naïve (Fastest to slowest)

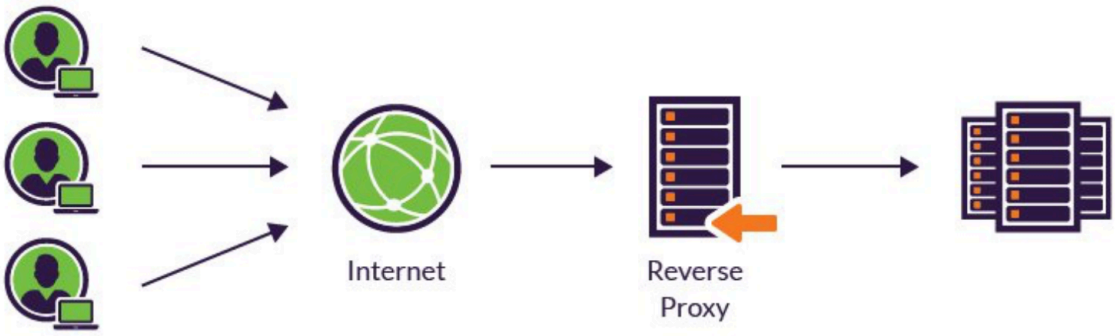
Reasoning: The conclusion driven above can be reasoned simply by referring to the concepts of each mode. Persistent connection keeps the connection socket open for a certain time rather than closing the socket after every connection. This reduces time spent for setting up new socket, thereby making proxy with PC faster compared with the one without PC. Multithreading by definition creates multiple threads and let them run at the same time. It is obvious that in this way we can save some time by distributing tasks to multiple threads, balancing the load of each thread. Among PC and MT, seems like PC makes the connection much faster than MT does. Therefore we can support the conclusion above driven by the experiment.

[About Forward/Reverse Proxy]

Forward Proxy: Proxy server positioned at network's edge, which regulates outbound traffic based on predefined policies in shared networks. Also masks a client's IP address and blocks malicious incoming traffic. Typically used by large organizations.



Reverse Proxy: Proxy server positioned at network's edge. Receives initial HTTP connection requests and completes a TCP three-way handshake, terminating the initial connection. Connects with the original server and forward the original request.
In a nutshell, the main difference between Forward and Reverse proxy is that Reverse Proxy is an intermediary for its associated servers to be contacted by any client, while Forward Proxy is an intermediary for its associated clients to contact any server.
'Reverse' in reverse proxy means 'backside'. We can say the forward proxy works as a client side proxy and reverse proxy as server side proxy.



Pros/Cons: Pro side of reverse proxy is that since it presents internal resources without directly allowing traffic to pass between two, it is often more simpler. The pro side of forward proxy is that it can prevent clients from malicious traffic incoming by disguising their IP addresses. Con side of both forward and reverse proxies in common is that there always exists an issue of security due to the nature of caching.
When they are used: Forward Proxy is usually used for the purpose of content filtering, email security, NAT'ing, and compliance reporting.
Reverse Proxy is usually used for load balancing(TCP Multiplexing), compression etc.