INDIAN INSTITUTE OF INFORMATION TECHNOLOGY KALYANI
Autonomous institution under MHRD, Govt. Of India
&
Department of Information Technology & Electronics, Govt. of West Bengal
WEBEL IT Park Campus (Near Buddha Park), Kalyani -741235, West Bengal
Tel : 033 2582 2240, website : www.iiitkalyani.ac.in

Lab Assignment #03
Submit on or before 31/01/2018

Weekly contact : **0 – 0 – 3 (L – T – P)**
Course No. : **CS 612**
Course Title : **Networking**
Instructor-In-Charge : **Dr. SK Hafizul Islam (hafi786@gmail.com)**

Aim

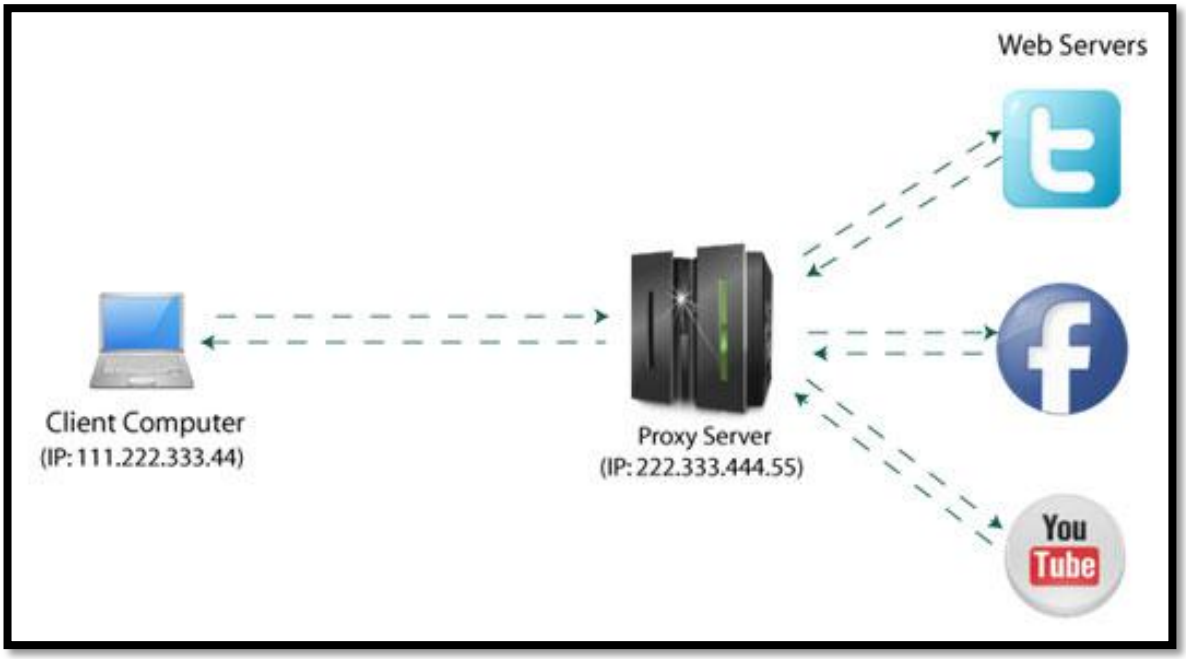
- Explaining SQUID Proxy Server configuration and analysis of HTTP traffic using Wireshark.

Objectives

- **To learn how to install** SQUID Proxy Server
- **To learn how to configure a** Proxy Server

Proxy server:

A proxy server is a dedicated computer or a software system running on a computer that acts as an intermediary between an endpoint device, such as a computer, and another server from which a user or client is requesting a service. The proxy server may exist in the same machine as a firewall server or it may be on a separate server, which forwards requests through the firewall.



SQUID Proxy Server

Squid is a full-featured web proxy cache server application which provides proxy and cache services for HTTP, FTP, and other popular network protocols.

A Squid proxy server is generally installed on a separate server than the web server with the original files.

Squid works by tracking object use over the network. Squid will initially act as an intermediary, simply passing the client's request on to the server and saving a copy of the requested object.

Installation of SQUID at Client PC

Step 1: Type the command

sudo apt-get install squid

Server side configuration

Step 1: Keep the original copy of configuration file for reference

sudo cp /etc/squid/squid.conf /etc/squid/squid.conf.original

Protect it from writing

sudo chmod a-w /etc/squid/squid.conf.original

Step 2: Edit the configuration file

sudo –i

gedit /etc/squid/squid.conf

To set your Squid server to listen on TCP port 8888 instead of the default TCP port 3128

```
# Squid normally listens to port 3128
http_port 8888

#  TAG: https_port
# Note: This option is only available if Squid is rebuilt with the
#       --enable-ssl
#
#       Usage:  [ip:]port cert=certificate.pem [key=key.pem] [mode] [options...]
#
#       The socket address where Squid will listen for client requests made
#       over TLS or SSL connections. Commonly referred to as HTTPS.
#
#       This is most useful for situations where you are running squid in
#       accelerator mode and you want to do the SSL work at the accelerator level.
#
#       You may specify multiple socket addresses on multiple lines,
#       each with their own SSL certificate and/or options.
#
#       Modes:
```

- **Make http allow**

```
# And finally deny all other access to this proxy
http_access allow all

#  TAG: adapted_http_access
#       Allowing or Denying access based on defined access lists
#
#       Essentially identical to http_access, but runs after redirectors
#       and ICAP/eCAP adaptation. Allowing access control based on their
#       output.
#
#       If not set then only http_access is used.
#Default:
# Allow, unless rules exist in squid.conf.

#  TAG: http_reply_access
#       Allow replies to client requests. This is complementary to http_access.
#
#       http_reply_access allow|deny [!] aclname ...
#
```

- **Modify Access Control List to block a site -- Define a acl**

```
acl aclname srcdomain   .foo.com ...
  # reverse lookup, from client IP [slow]
acl y dstdomain   yahoo.com
  # Destination server from URL [fast]
acl aclname srcdom_regex [-i] \.foo\.com ...
  # regex matching client name [slow]
acl aclname dstdom_regex [-i] \.foo\.com ...
  # regex matching server [fast]
  #
  # For dstdomain and dstdom_regex a reverse lookup is tried if a IP
  # based URL is used and no match is found. The name "none" is used
  # if the reverse lookup fails.
```

- **Block the site**

```
# Only allow cachemgr access from localhost
http_access allow localhost manager
http_access deny manager

# We strongly recommend the following be uncommented to protect innocent
# web applications running on the proxy server who think the only
# one who can access services on "localhost" is a local user
 http_access deny y

#
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
#

# Example rule allowing access from your local networks.
# Adapt localnet in the ACL section to list your (internal) IP networks
# from where browsing should be allowed
#http_access allow localnet
http_access allow localhost
```

Step 3: Restart service
service squid3 restart

Client Side Configuration
Firefox and go to Edit–>Preferences–>Advanced–>Network–>Settings->Manual proxy configuration->enter your proxy server IP address and Port to be used for all connection

Step 4: Open the site www.yahoo.com
It will not open.

Step 5: After all experiments

sudo cp /etc/squid/squid.conf.original /etc/squid/squid.conf

Reference
1) https://help.ubuntu.com/lts/serverguide/squid.html

Assignment 1

Do the following:

- Both the client and proxy server start the “Wireshark packet sniffer”.
- The client enter the following URL into the browser: iiitkalyani.ac.in
- Stop Wireshark packet capture, and enter “http” in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.

- The proxy server also stops the Wireshark packet capture, and enter “http” in the display-filter-specification window.

Answer the following question:

- Write the observations of client and server machine

Assignment 2

Do the following:

- Modify the acl to block the multiple sites
- Write the observations of client and server machine

Assignment 3

The HTTP CONDITIONAL GET/response interaction

First, clear your browser’s cache (empty).

(For Firefox, select Tools->Clear Recent History and check the Cache box, or for Internet Explorer, select Tools->Internet Options->Delete File; these actions will remove cached files from your browser’s cache.)

Now do the following:

- Start up your web browser, and make sure your browser’s cache is cleared, as discussed above.
- Start up the Wireshark packet sniffer
- Enter the following URL into your browser

http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file2.html

Your browser should display a very simple five-line HTML file.

• Quickly enter the same URL into your browser again (or simply select the refresh button on your browser)

• Stop Wireshark packet capture, and enter “http” in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window.

Answer the following questions:

- Inspect the contents of the first HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE” line in the HTTP GET?
- Inspect the contents of the server response. Did the server explicitly return the contents of the file? How can you tell?
- Now inspect the contents of the second HTTP GET request from your browser to the server. Do you see an “IF-MODIFIED-SINCE:” line in the HTTP GET? If so, what information follows the “IF-MODIFIED-SINCE:” header?
- What is the HTTP status code and phrase returned from the server in response to this second HTTP GET? Did the server explicitly return the contents of the file? Explain.

Retrieving Long Documents

In our examples thus far, the documents retrieved have been simple and short HTML files. Let’s next see what happens when we download a long HTML file. Do the following:

• Start up your web browser, and make sure your browser’s cache is cleared, as discussed above.

• Start up the Wireshark packet sniffer

• Enter the following URL into your browser

http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file3.html

Your browser should display the rather lengthy US Bill of Rights.

• Stop Wireshark packet capture, and enter “http” in the display-filter-specification window, so that only captured HTTP messages will be displayed.

In the packet-listing window, you should see your HTTP GET message, followed by a multiple-packet TCP response to your HTTP GET request. This multiple-packet response deserves a bit of explanation. Note that the HTTP response message consists of a status line, followed by header lines, followed by a blank line, followed by the entity body. In the case of our HTTP GET, the entity body in the response is the entire requested HTML file. In our case here, the HTML file is rather long, and at 4500 bytes is too large to fit in one TCP packet. The single HTTP response message is thus broken into several pieces by TCP, with each piece being contained within a separate TCP segment. In recent versions of Wireshark, Wireshark indicates each TCP segment as a separate packet, and the fact that the single HTTP response was fragmented across multiple TCP packets is indicated by the "TCP segment of a reassembled PDU" in the Info column of the Wireshark display. Earlier versions of Wireshark used the "Continuation" phrase to indicated that the entire content of an HTTP message was broken across multiple TCP segments.. We stress here that there is no "Continuation" message in HTTP!

Answer the following questions:

- How many HTTP GET request messages did your browser send? Which packet number in the trace contains the GET message for the Bill or Rights?
- Which packet number in the trace contains the status code and phrase associated with the response to the HTTP GET request?
- What is the status code and phrase in the response?
- How many data-containing TCP segments were needed to carry the single HTTP response and the text of the Bill of Rights?

HTML Documents with Embedded Objects

Now that we've seen how Wireshark displays the captured packet traffic for large HTML files, we can look at what happens when your browser downloads a file with embedded objects, i.e., a file that includes other objects (in the example below, image files) that are stored on another server(s).

Do the following:

- Start up your web browser, and make sure your browser's cache is cleared, as discussed above.
- Start up the Wireshark packet sniffer
- Enter the following URL into your browser

http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file4.html

Your browser should display a short HTML file with two images. These two images are referenced in the base HTML file. That is, the images themselves are not contained in the HTML; instead the URLs for the images are contained in the downloaded HTML file. As discussed in the textbook, your browser will have to retrieve these logos from the indicated web sites. Our publisher's logo is retrieved from the gaia.cs.umass.edu web site. The image of the cover for our 5th edition (one of our favourite covers) is stored at the caite.cs.umass.edu server. (These are two different web servers inside cs.umass.edu).

• Stop Wireshark packet capture, and enter "http" in the display-filter-specification window, so that only captured HTTP messages will be displayed.

Answer the following questions:

- How many HTTP GET request messages did your browser send? To which Internet addresses were these GET requests sent?
- Can you tell whether your browser downloaded the two images serially, or whether they were downloaded from the two web sites in parallel? Explain.