Networking Exercises

Experiential Learning Workshop
BITE-EWIT

Mar 26, 2018

1 General Guidelines

1. Make a team of two unless stated otherwise.
2. For each exercise, use wireshark capture to verify contents
3. Ensure to use proper capture filter and don't capture unnecessary traffic
4. Where appropriate or applicable, use `wget` or `nc` to access the web server.
5. The default client for accessing web server is assumed to be browser, preferably firefox. You can use Chrome or any other browse as well.
6. The webserver in the example below is taken as 'mywww.com'. Please use your hostname or corresponding IP address instead in your exercise.

2 Hands-on 1: Tools

2.1 Using wget

1. Open terminal
2. Mimic your college website www.ewit.edu, and access locally (turn off your internet).
3. Download a large file e.g. https://www.dropbox.com/s/89nzt3tbhz2q2v8/TerryMoore_2005-480p.mp4?dl=0, break the download by pressing Ctrl-C and then download with resume option (-c).
4. Explore other options.

2.2 Using nc

1. Open terminal on two machines.
2. Identify each other's IP address
3. Run as TCP server in one terminal and UDP server in another terminal.
4. Connect using clients (from another machine) and do chat.
5. Transfer some files across machines.
6. Login in to remote machine without authentication

2.3 Using Wireshark

1. Open wireshark
2. Select the applicable interface e.g. `enp0s1`. If capturing traffic for your own web server on the same machine, use the Loopback interface e.g. 'lo'.
3. Specify the capture filter e.g. 'host a.b.c.d' for the other hosts.
4. Click start
5. Do some communication with each other e.g. nc. After the actual capture has been seen, Stop the capture.
6. Browse your college website and capture traffic. Analyse the capture and understand headers. What is the difference between this and nc capture.

3 Hands-on 2: Network Delays

3.1 Transmission delay

1. Connect two systems directly via a RJ45 cable.
2. Ping other systems with packet size of 200B, 400B, 600B, 800B, 1000B and note down the response time (avg). Compute the transmission delay for 200B.
3. Connect two systems via one switch. Repeat the ping exercise. Analyze the increase in response time w.r.t. expected time.
4. Connect two systems via two switches. Repeat the ping exercise. Analyze the increase in response time w.r.t. expected time.

3.2 Processing delay

1. Run the server program nwdelay_server.py with simulating processing delay of equal to transmission delay(T ms) of previous experiments.
2. Run the client program nwdelay_client.py and connect to server. Notice the delay.
3. Change the simulated processing time and repeat the experiment. Redo the exercise and study impact of processing delay in end-to-end delay.

3.3 Queuing delay

1. Run the server program nwdelay_server.py with simulating processing delay of T on port number **P** (e.g. 9999).
2. Run two client programs connecting to server on port **P.** Analyze the response time of each client. Do you notice any difference in response time.
3. Now run 3 or more clients connecting to server on port P. Analyse the response time. Does it change significantly from previous response time. Analyze why?

3.4 Overcoming Queuing delay

1. Run N (e.g. 3) server programs nwdelay_server.py (concurrently) with simulating processing delay of T on different ports e.g. 9996, 9997, 9998.
2. Run N clients, each client communicating with separate server.
3. Do you notice any queueing delay. Analyze.
4. In general, server program is expected to run on a well known port. So, publishing multiple ports is not a choice. Thus, all clients should connect to same server (e.g. same IP and port e.g. 9999). So to overcome queueing delay on account of processing (1 CPU is being used to deal network traffic on one port), we need to do load balancing. This load balancing needs to be agnostic to clients. At a very basic level, do the following load balancing using iptables on server m/c.
    a. sudo iptables -t nat -A PREROUTING -p udp --dport 9999 -m statistic --mode nth --every 3 --packet 0 -j REDIRECT --to-port 9996

    b. sudo iptables -t nat -A PREROUTING -p udp --dport 9999 -m statistic --mode nth --every 3 --packet 0 -j REDIRECT --to-port 9997

    c. sudo iptables -t nat -A PREROUTING -p udp --dport 9999 -j REDIRECT --to-port 9998
5. Now connect 3 concurrent clients to same port number P (i.e. 9999) and analyze the response time.

4 Hands-on 3: Basics of HTTP

4.1 HTTP/0.9

1. You need to use 'nc' since by default browser and wget use the protocol HTTP/1.1. In the request line, do not specify the HTTP protocol version
2. Just send the request line without protocol version and notice if you get any HTTP headers.
    a. echo "GET /"; echo "" | nc www.yahoo.com

4.2 Status Code 200

1. Create a simple webpage e.g. welcome.html
2. Access the web page in the browser e.g. welcome.html
3. Verify the status code 200 and other required headers.
4. Access the same webpage using 'wget –d' and verify the status code.
5. Access the same webpage using 'nc' and verify the status code

```
nc myweb.com 80
GET /welcome.html HTTP/1.1
Host:  myweb.com
```

4.3 Content-Type

1. Copy welcome.html file as welcome.txt
2. Access the url welcome.txt.
3. Look at the content displayed on browser.
4. Analyze the wireshark capture to study the header `Content-Type`:
5. Repeat the exercise with `wget`.
6. Study the headers.

4.4 Using Accept-Language.

1. Change the preferred language setting in firefox browser.
2. Access google.com.
3. The browser should display the content of web page.

4.5 Status code 404

1. Access a non-existence webpage e.g `nonexist.html`
2. Check the status code in wireshark.
3. Verify this status code using `wget` as well.

4.6 Status code 403

1. Create file restricted.html with some content.
2. Access this file using browser. Browser should display the contents
3. Set the file permission to 400 (`chmod 400 restricted.html`).
4. Access this file using browser. Browser should display the error message 'Forbidden'.

4.7 Status code 400

1. To experience this access code, we need to use `nc`. By default, both browser and `wget` send the proper HTTP header.
2. This exercise requires that client should send invalid header e.g. '`Host mywww.com`' instead of '`Host: mywww.com`'. Please note that HTTP header field name should be separated by its value by Colon (:) character.

3. Using terminal, do the following

```
nc webserverhost 80
GET /welcome.html HTTP/1.1
Host  mywww.com
```

   a. Analyze the response given by web server and verify that it corresponds to '400 Bad Request'.
   b. Make another access with different header with syntax error.
   c. Verify the Bad Request error

4.8 Status code 301 (or 302)

1. Access google.com using wget –d. Identify the first response.
2. Make following changes in Apache config file
   `Redirect /oldwelcome.html /welcome.html`
3. Restart Apache webserver (sudo service apache2 restart)
4. Access the url http://mywww.com/oldwelcome.html
5. Verify that content is served from the file `welcome.html`
6. Verify the status code 301 being returned and second access to new url in the wireshark
7. Repeat the exercise using '`wget`' and verify HTTP redirect.
8. Access `google.com` using `wget` and verify HTTP redirect.

5 HTTP Persistent Connections.

5.1 HTTP Non-Persistent Connections.

1. Configure Apache web server with `KeepAlive Off` and restart.
2. Create a web page (e.g. pictures.html) with multiple embedded images (say 10) images as in http://rprustagi.com/accs/pictures.html
3. Access the web page from your local web server with `KeepAlive Off` in firefox browser and do a wireshark capture. How many TCP connections you notice. There should be as many connection as number of embedded objects plus 1.

5.2 HTTP Persistent Connections.

1. Configure Apache web server with following configurations.
   a. `KeepAlive On.`
   b. `MaxKeepAliveRequests 10`
   c. `KeepAliveTimeout 5`
2. Restart Apache.
3. Access the web page again with firefox browser. Analyze the number of TCP Connections. By default, firefox makes 6 concurrent TCP connections. You should see similar number and on some connections you should see two or more HTTP requests (e.g. images).
4. Configure Firefox to setup only 3 persistent connection.
   a. Type about:config in firefox browser.
   b. search the field *`max-persistent-connections-per-server`*
   c. set the value to 3.
5. Access the page again and analyze the number of TCP Connections.
6. Refresh the page after 5 seconds. Analyze setup of new TCP connections.

7. Refresh the web page within 4 seconds multiple times e.g. 10 times. Analyze the wireshark capture on when does a new TCP connection is made.
8. Tweak (or reconfigure) the value `MaxKeepAliveRequests` to your other values.
9. Continue to refresh the page multiple times less than configured timeout value (e.g. 5s). Analyze and understand when does a browser makes a new TCP Connection.

← end of exercises handout →