Homework 4

Network Architecture I, Spring 2016

Due: ***Friday May 6th Noon***

(Submit both paper to Rm 453 FH and on Blackboard)

1. Suppose within your Web browser you click on a link to obtain a web page. The IP address for the associated URL is cached in your local host, so a DNS look-up is not necessary to obtain the IP address. Further suppose that the Web page associated with the link references ten very small objects on the same server. Let RTT_0 denote the RTTs between the local host and one of the objects. Assuming zero transmission time of the object, how much time elapses from when the client clinks on the link until the client receives the full web page with

    a. Nonpersistent HTTP?
    b. Persistent HTTP?

2. Describe in detail i) what information should be added in which DNS servers for your own start-up company (say 'networkguru.com') that has a webserver and email service to its employees. ii) What are companies you can contact for domain name registration and how much are the fees?

Laboratory Homework

Part 1: Telnet experiments

1. Try HTTP request (GET, HEAD, or POST) without using a web-browser. You can do this on command line using '> telnet *webserver* 80'. (for example, www.umkc.edu) Record the HTTP responses from the server – retrieve at least two different response status from the server.

Part 2: Wireshark experiments

In this part of the homework, you will use Wireshark to investigate HTTP protocol in operation. In this lab, you'll explore a couple of aspects of the HTTP protocol: the basic GET/response interaction, and retrieving large HTML files.

Download a packet trace file (http-wireshark-trace-1) of this homework from the Blackboard, Assignment Section. Or create a trace file on your own following the instruction at the footnote[1] from you home (not at University!).

Then, open the trace file on the Wireshark, and answer to the questions below.

Part 2-1: The Basic HTTP GET/response interaction

Once you open the trace file, it will show in the packet-listing window that two HTTP messages were captured: the GET message (from your browser to the gaia.cs.umass.edu web server) and the response message from the server to your browser. The packet-contents window shows details of the selected message (in this case the HTTP GET

[1] *FYI, below are the steps taken for the packet capture in the given trace. You don't*
1. Start up your web browser.
2. Start up the Wireshark packet sniffer. Enter "http" (just the letters, not the quotation marks) in the display-filter-specification window, so that only captured HTTP messages will be displayed later in the packet-listing window. (We're only interested in the HTTP protocol here, and don't want to see the clutter of all captured packets).
3. Wait a bit more than one minute (we'll see why shortly), and then begin Wireshark packet capture.
4. Enter the following to your browser
   http://gaia.cs.umass.edu/wireshark-labs/HTTP-wireshark-file1.html
   Your browser should display the very simple, one-line HTML file.
5. Stop Wireshark packet capture.

message, which is highlighted in the packet-listing window). Recall that since the HTTP message was carried inside a TCP segment, which was carried inside an IP datagram, which was carried within an Ethernet frame, Wireshark displays the Frame, Ethernet, IP, and TCP packet information as well. We want to minimize the amount of non-HTTP data displayed (we're interested in HTTP here, and will be investigating these other protocols is later labs), so make sure the boxes at the far left of the Frame, Ethernet, IP and TCP information have a plus sign (which means there is hidden, undisplayed information), and the HTTP line has a minus sign (which means that all information about the HTTPmessage is displayed).

(*Note:* You should ignore any HTTP GET and response for favicon.ico. If you see a reference to this file, it is your browser automatically asking the server if it (the server) has a small icon file that should be displayed next to the displayed URL in your browser. We'll ignore references to this pesky file in this lab.).

By looking at the information in the HTTP GET and response messages, answer the following questions. When answering the following questions, you should print out the GET and response messages (see the introductory Wireshark lab for an explanation of how to do this) and indicate where in the message you've found the information that answers the following questions.

1. Is your browser running HTTP version 1.0 or 1.1? What version of HTTP is the server running?
2. What languages (if any) does your browser indicate that it can accept to the server?
3. What is the IP address of your computer? Of the gaia.cs.umass.edu server?
4. What is the status code returned from the server to your browser?
5. When was the HTML file that you are retrieving last modified at the server?
6. How many bytes of content are being returned to your browser?

(In your answer to question 5 above, if you have created your own trace file, you might have been surprised to find that the document you just retrieved was last modified within a minute before you downloaded the document. That's because (for this particular file), the gaia.cs.umass.edu server is setting the file's last-modified time to be the current time, and is doing so once per minute. Thus, if you wait a minute between accesses, the file will appear to have been recently modified, and hence your browser will download a "new" copy of the document.)