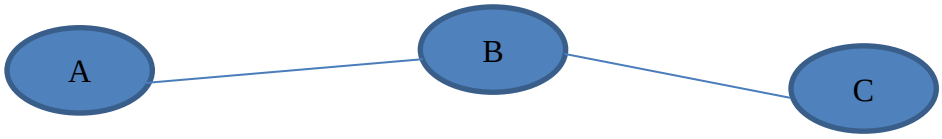
CPE 400/600: Computer Communication Networks

HW 1 (Total 20 points)

(Submission through WebCampus)
Submission format: Microsoft Word or pdf only
(file name: firstname_lastname_hw1)

Part 1

1.



Assume a node A is sending a packet to router B who is forwarding the packet to destination C. A – B link has transmission rate of 4 **Megabit**/second while B – C link has transmission rate of 6 **Megabit**/second. There is only one packet that needs to traverse from A – B – C. The packet size is 10000 **Bytes**.

a) [2] What are the transmission delays over link A – B and link B- C? Show your answer with clear explanation.

Ans.

b) [2] Assume nodal processing delay at B is 5 millisecond and queuing delay at B is 2 milliseconds. Then what is the total delay from node A to node C? Assume zero propagation delay. Show your answer with clear explanation.

Ans.

Part 2

This is the Wireshark part. When you capture network traffic, please make sure you are capturing network data in your home network only.

2. The Basic HTTP GET/response interaction

Let's begin our exploration of HTTP by downloading a very simple HTML file - one that is very short and contains no embedded objects. Do the following:
1. Start up your web browser.

2. Start up the Wireshark packet sniffer (but don't yet begin packet capture). 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, and then begin Wireshark packet capture.

4. Enter the following to your browser
https://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 (press the red square button)
Your Wireshark window may/should look like the window shown in Figure 1 (depending on your Wireshark download version).

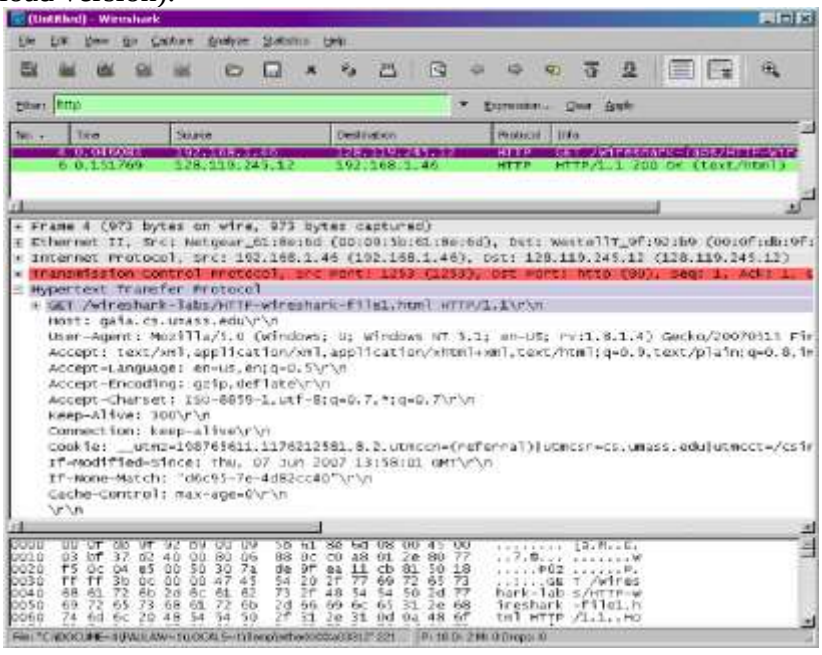


Figure 1: Wireshark Display

The example in Figure 1 shows 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 message, which is highlighted in the packet-listing window).

Recall the encapsulation conversation we have had, 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 in 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 information, and you can expand it), and the HTTP line has a minus sign (which means that all information about the HTTP message 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 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 must take screen shots of the GET and response messages and indicate where in the message you've found the information that answers the following questions. Answer clearly.

a) **[1]** Is your browser running HTTP version 1.0 or 1.1?
Ans.

b) **[1]** What is the IP address of the client? Of the gaia.cs.umass.edu server?
Ans.

c) **[1]** What is the status code returned from the server to your browser?
Ans.

d) **[1]** When was the HTML file that you are retrieving last modified at the server?
Ans.

e) **[1]** Which port is the request going to?
Ans.

f) **[1]** From which port, the client is making the request?
Ans.

Part 3

3. The HTTP CONDITIONAL GET/response interaction

Before performing the steps below, make sure your browser's cache is empty. Now do the following:

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

2. Start up the Wireshark packet sniffer

3. 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.

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

5. 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.
Then answer the following questions.

> (*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 file in this lab.).

Answer the following questions:

a) **[2]** 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?
Ans.

b**)** **[2]** 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? Explain.
Ans.

Part 4

Before beginning this part, review DNS from the class slides and textbook. In particular, review the material on **local DNS servers**, **DNS caching**, **DNS records and messages**, and the **TYPE field** in the DNS record.

4. nslookup

In this lab, we'll make extensive use of the *nslookup* utility, which is available in most Linux/Unix and Microsoft platforms today. To run *nslookup* in Linux/Unix, you just type the *nslookup* command on the command line. To run it in Windows, open the Command Prompt and run *nslookup* on the command line. To run on Mac, open Terminal, and type in *nslookup* command.

In it is most basic operation, *nslookup* utility allows the host running the utility to query any specified DNS server for a DNS record. The queried DNS server can be a root DNS server, a top-level-domain DNS server, an authoritative DNS server, or an intermediate DNS server (see the textbook & slides for definitions of these terms). To accomplish this task, *nslookup* sends a DNS query to the specified DNS server, receives a DNS reply from that same DNS server, and displays the result.

Just for example, consider the first command:

```
nslookup mit.edu
```

In words, this command is saying "please send me the IP address for the host mit.edu". The response from this command provides two pieces of information: (1) the name and IP address of the DNS server that provides the answer; and (2) the answer itself, which is the host name and IP address of mit.edu. Although the response came from the local DNS server at Polytechnic University, it is quite possible that this local DNS server iteratively contacted several other DNS servers to get the answer.

Now consider the second command:

```
nslookup –type=NS mit.edu
```

In this example, we have provided the option "-type=NS" and the domain "mit.edu". This causes *nslookup* to send a query for a type-NS record to the default local DNS server. In words, the query is saying, "please send me the host names of the authoritative DNS for mit.edu". (When the –type option is not used, *nslookup* uses the default, which is to query for type A records.) The answer first indicates the DNS server that is providing the answer (which is the default local DNS server) along with the MIT nameservers. Each of these servers is indeed an authoritative DNS server for the hosts on the MIT campus. However, *nslookup* also indicates that the answer is "non-authoritative," meaning that this answer came from the cache of some server rather than from an authoritative MIT DNS server. Finally, the answer should also include the IP addresses of the authoritative DNS servers at MIT. (Even though the type-NS query generated by *nslookup* did not explicitly ask for the IP addresses, the local DNS server returned these "for free" and *nslookup* displays the result.)

Let us assume bitsy.mit.edu is one of the authoritative servers.

Now finally consider the third command:

```
nslookup www.aiit.or.kr bitsy.mit.edu
```

In this example, we indicate that we want to the query sent to the DNS server bitsy.mit.edu rather than to the default DNS server. Thus, the query and reply transaction takes place directly between our querying host and bitsy.mit.edu. In this example, the DNS server bitsy.mit.edu provides the IP address of the host www.aiit.or.kr, which is a web server at the Advanced Institute of Information Technology (in Korea).

Now that we have gone through a few illustrative examples, you are perhaps wondering about the general syntax of *nslookup* commands. The syntax is:

```
nslookup –option1 –option2 host-to-find dns-server
```

In general, *nslookup* can be run with zero, one, two or more options. And as we have seen in the above examples, the dns-server is optional as well; if it is not supplied, the query is sent to the default local DNS server.

Now that we have provided an overview of *nslookup*, it is time for you to test drive it yourself. Do the following (and write down the results) with screenshots.

Run *nslookup* to obtain the IP address of a Web server in Twitter through the utility. Give the command. Run Wireshark in the background to analyze the traffic for this command the response obtained. Explain your findings from Wireshark. Provide screenshots.

a) **[3]** What is the IP address of the web server? Which server returned the answer? Give the IP address of this server. Is this authoritative answer or non-authoritative answer?

Ans.

Run *nslookup* to find the authoritative DNS servers for Twitter. Give the command. Run Wireshark in the background to analyze the traffic for this command the response obtained. Explain your findings from Wireshark. Provide screenshots.

b) **[3]** What is/are the IP address(s) of the authoritative DNS servers for Twitter?

Ans.