Computer Communication EDA344/DIT420

Lab 1a

Packet and Traffic Analysis Using Network Sniffing Software

1. The mandatory part of the course consists of one written (home) assignment and two practical sessions in the course labs. The assignment and both labs must be completed to pass the course.

2. You may work in a group of max. 2 students using a workplace in the lab room.

3. Completed lab work must be reported with answers of the questions embedded in the lab tasks. The report must be posted to Pingpong before the deadline.

Please read through this PM **before attending the lab session** and make sure that you are prepared to use **Wireshark**.

This lab is including selected parts from the **Wireshark Labs** which can be downloaded from the course book home page ("Student resources" / "Wireshark labs").

Wireshark ® is popular network protocol analyzer and it is available to download using http://www.wireshark.org/download.html under the GNU General Public License.

Purpose: To learn how to listen to local network traffic and analyze different protocols as well as learning to use some useful TCP/IP utilities.

Background reading:
Read related sections in Chapter 2 (Application) and Chapter 3 (Transport), please see below the outlined sections of the course book.

Preparation: Either at home using your own installation of the software or at campus using the computers of the lab rooms J232 (Gnistan) and J234 (Bryggan), floor 2 in Jupiter building; **prepare by yourself** the first part of the exercises "1. INTRO" (**Getting Started with Wireshark**) to be familiar with the program and its usage. Please refer to the "Wireshark Lab" at the end of Chapter 1 in the course book.

The following lab parts are the ones that should be performed fully during the lab session.

2. **HTTP** (section 2.2)
3. **DNS** (section 2.5)
4. **TCP** (sections 3.5 and 3.7)

For those who wish and who have some time, you may also perform by yourself the following optional labs at home:

(5. **IP**, 6. **ICMP**, 7. **Ethernet** and **ARP**, 8. **DHCP**, 9. **UDP**, 10. **SSL**, 11. **802.11,** 12**. NAT**)

Useful and practical information

You will be working in the Windows environment but the instructions include as well the Linux environment. To facilitate the work, a small list of things to think about and practical information is found next on this page.

Time estimation:

Estimate about little bit more than one hour for each part of lab (they are three; **HTTP**, **DNS** and **TCP**). Try to understand all topics before continuing with the next part!

Bring the book:

You will most likely have to consult the book about various topics, so make sure your group has at least one book available!

Using Wireshark on computers in lab room:

You are going to use the Chalmers studat-computers located in Jup232 (Gnistan) and Jup234 (Bryggan). When coming to the lab, you must log into the machine in front of you using operating system Windows. Use your Chalmers student account to log in. You are going to get some root privileges on these computers that allow you to perform the lab tasks using Wireshark. Normally studat-users are not allowed to capture network traffic.

Note: In any other Studat-computers, Wireshark must run with administrator (root) privileges. If you just start Wireshark, you will get an error message “permission denied” when trying to capture packets.

Using traces:

The book resources include traces files for the Wireshark labs. You may use these pre-recorded trace files **at home** but **not** during the lab session. Please note; **do not use** any of these trace files during the lab exercises, since you are able to run Wireshark on a live network connection in the lab room. You should use your own outworked traces.

Manuals and on-line help:

If you need help about a command, you can ask the system in Linux by giving for example the command “**man ifconfig**” in a terminal window. In Windows use the help parameter **/?** To get help, for example enter C:\> **ipconfig /?**. Also, note that the Linux **ifconfig** has different arguments than Windows **ipconfig**.

Web browser:

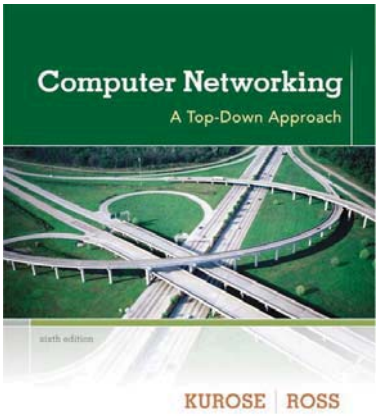
When using web browser you may need to clear the cache as outlined in the text of the instructions depending on which browser is in use.

DNS:

In addition to the **nslookup** command, you can experiment with the **host** command. In order to flush DNS-cache in Linux use the command “ **sudo /etc/init.d/nscd restart** “ and in Windows as it is outlined in the text of the instructions.

Adopted and Modified based on

Wireshark Lab: HTTP v6.1



Supplement to *Computer Networking: A Top-Down Approach, 6th ed.,* J.F. Kurose and K.W. Ross.

"Tell me and I forget. Show me and I remember. Involve me and I understand." Chinese proverb.

© 2005-21012, J.F Kurose and K.W. Ross, All Rights Reserved.

Preparation: Do the **Introductory lab** at home to be familiar with the Wireshark packet sniffer.

Reading: Learn carefully about "The Web and HTTP" in **Section 2.2** of the course book.

Main Task: Use Wireshark to investigate the HTTP protocol in operation.

You will explore several aspects of the HTTP protocol: the basic GET/response interaction, HTTP message format, retrieving large HTML file, retrieving HTML file with embedded objects.

The Basic HTTP GET/Response Interaction

Let's begin the exploration of HTTP by downloading a very simple HTML file, one that is very short, and contains no embedded objects.

Do the following:

- Start up your web browser.
- Start up the Wireshark packet sniffer, as described in the **Introductory lab**. Enter the letters "**http**" in the field of the display filter. Click the button **Apply** so that only captured HTTP messages will be displayed in the packet list. (The only interest here is in the HTTP protocol, and you don't want to see the clutter of all captured packets).
- Begin Wireshark packet capture.
- 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.
- Stop Wireshark packet capture.

The window-pane of the packet list will show 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.

***Note**: You should ignore any HTTP GET request and response for **favicon.ico**. If you see a request to this file, it is because of the browser which is automatically asking the server if it has a small icon file that would be displayed next to the URL in the address field.*

The next window-pane of the packet details shows details of the selected message (in this case the HTTP GET message, which is highlighted in the packet list).

Recall that since the HTTP message is carried inside a TCP segment, which is carried inside an IP datagram, which is carried within an Ethernet frame, Wireshark displays the Frame, Ethernet, IP, and TCP header information as well.

It is highly desired to minimize the amount of non-HTTP data displayed so make sure that the boxes at the far left of the Frame, Ethernet, IP and TCP information have a plus sign or a right-pointing triangle, and the HTTP line has a minus sign or a down-pointing triangle (which means that all information about the HTTP message is displayed).

By inspecting and looking at the information in the HTTP GET and response messages, answer the following questions.

Answer the following questions. Please explain and clarify how and where you can find answers.

1. What is the IP address of your computer? Of the **gaia.cs.umass.edu** server?

2. What HTTP version is your browser running? What version of HTTP is the server running?

3. What is the status code and phrase returned from the server to your browser?

4. What languages does your browser indicate to the server that it can accept? Which header line is used to indicate this information?

5. When was the HTML-file, that you have retrieved, last modified at the server? Which header line is used to indicate this information?

6. How many bytes of content (size of file) are returned to your browser? Which header line is used to indicate this information?

In your answer to question 5 above, you might have been surprised to find that the file you just retrieved has been last modified within a minute before you have downloaded the content. 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 and then retrieve again, the file will appear to have been recently modified, and hence your browser will download a "new" copy of it.

HTTP Conditional GET/Response Interaction

Recall from **Sections 2.2.5-2.2.6** of the course book, that most web browsers perform object caching and thus perform the conditional GET when retrieving HTTP objects.
Before performing the steps below, make sure that your browser's cache is empty. To do this under Firefox (25.0), select *Settings > Advanced > Network > Clear Now (Cached Web Content)*, or for Internet Explorer (11.0), select *Tools > Internet Options > Delete ..(Browsing history).* 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.
- Start up the Wireshark packet sniffer, and make sure that "http" is in the display-filter, so that only captured HTTP messages will be displayed in the packet-list pane.
- 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.

Answer the following questions. Please explain and clarify how you can find the answers.

7. Inspect the contents of the first HTTP GET **request** from your browser to the server. Is there an "IF-MODIFIED-SINCE" header line in the HTTP GET message? Why or why not?

8. Inspect the contents of the server **response**. Has the server explicitly returned the contents of the file? How can you tell?

9. Now inspect the contents of the second HTTP GET **request** from your browser to the server. Is there an "IF-MODIFIED-SINCE:" header line in the HTTP GET message? If so, what information follows the "IF-MODIFIED-SINCE:" header line?

10. What is the HTTP status code and phrase returned from the server in **response** to this second HTTP GET? Has the server explicitly returned the contents of the file? Explain.

Retrieving Long Documents

In the examples thus far, the files that are retrieved, have been simple and short HTML files. Let's next see what happens when you download a long HTML file. **Do the following:**

- Start up your web browser, and make sure that browser's cache is cleared.
- Start up the Wireshark and make sure that "http" is in the display-filter.
- 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.

In the packet-list pane, you should see the HTTP GET message, followed by a multiple-packet TCP response to the HTTP GET request. This multiple-packet response deserves a bit of explanation. Recall from **Section 2.2** of the course book (see Figure 2.9) 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 HTTP response, the entity body in the message is the entire requested HTML file. In this case, the HTML file is rather long, and is 4500 bytes such that it is too large to fit in one TCP segment. The single HTTP response

message is thus broken into several pieces by TCP; i.e. segmented, with each piece being contained within a separate TCP segment. In recent versions, Wireshark indicates each TCP segment carried in a separate packet, and the fact, that the single HTTP response is segmented across multiple TCP segments, is indicated by the "TCP segment of a reassembled PDU" in the Info column of the Wireshark display. Earlier versions of Wireshark use the "Continuation" phrase to indicate that the entire content of an HTTP message was broken across multiple TCP segments. It should be obvious here that there is no "Continuation" message in HTTP! Please check Wireshark menu **Edit → Preferences → Protocols → HTTP** to see what options to choose in order to suppress reassembly.

Answer the following questions with explanations.

11. How many HTTP GET request messages has your browser sent? Which packet in the trace contains the request for the **Bill of Rights**?
    Note: The packets here are only those containing HTTP messages and data.

12. Which packet 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?

13. How many TCP segments are needed to carry the single HTTP response and the text of the **Bill of Rights**? What is the number of bytes (of the text) in each segment?

HTML Documents with Embedded Objects

Now you can look at what happens when your browser downloads a file with embedded objects, i.e. a file with references to other objects (in the example below, image files) that are stored on another server(s).
Do the following and answer the questions given below:

- Start up your web browser, and make sure your browser's cache is cleared.
- 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. However the images themselves are not stored in the same server as the HTML file. As referenced objects, the browser will have to retrieve these images (logos) from the indicated web sites. The publisher's logo is retrieved from the Pearson Education **www.pearsonhighered.com** server. The image of the cover for the 5th edition of the course book (one of the authors favorite covers) is stored at the **manic.cs.umass.edu** server.

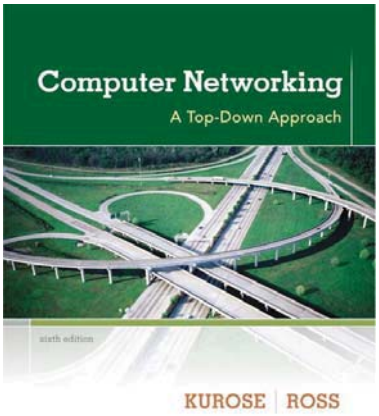
- Stop Wireshark packet capture.

14. How many HTTP GET request messages has your browser sent? To which IP-address is **each** of these GET requests sent?

15. How can you tell whether your browser has downloaded the two images serially, or whether they have been downloaded from the two web sites in parallel? Explain.

Adopted and Modified based on

Wireshark Lab: TCP v6.0



Supplement to *Computer Networking: A Top-Down Approach, 6th ed.*, J.F. Kurose and K.W. Ross.

"Tell me and I forget. Show me and I remember. Involve me and I understand." Chinese proverb.

© 2005-21012, J.F Kurose and K.W. Ross, All Rights Reserved.

Reading: Learn carefully about TCP in **Sections 3.5** and **3.7** of the course book.

Main Task: Analyze a trace of the TCP segments sent and received in transferring a large file (containing the text of Lewis Carrol's Alice's Adventures in Wonderland) from your computer to a remote server.

You will investigate the behavior of the celebrated TCP protocol in detail. You'll study TCP's use of sequence and acknowledgement numbers for providing reliable data transfer; you'll see TCP's congestion control algorithm, slow start and probably congestion avoidance in action; and you'll look at TCP's receiver-advertised flow control mechanism. You'll also briefly consider TCP connection setup and investigate the performance (throughput and round-trip time) of the TCP connection between your computer and a remote server.

Capturing a bulk TCP transfer from your computer to a remote server

In order to perform an exploration of TCP, you'll need to use Wireshark to obtain a packet trace of the TCP transfer of a large file from your computer to a remote server. You'll do so by accessing a Web page that will allow you to enter the name of a file stored on your computer (which contains the ASCII text of *Alice in Wonderland*), and then transfer the file to a Web server using the HTTP POST method (see **Section 2.2.3**).

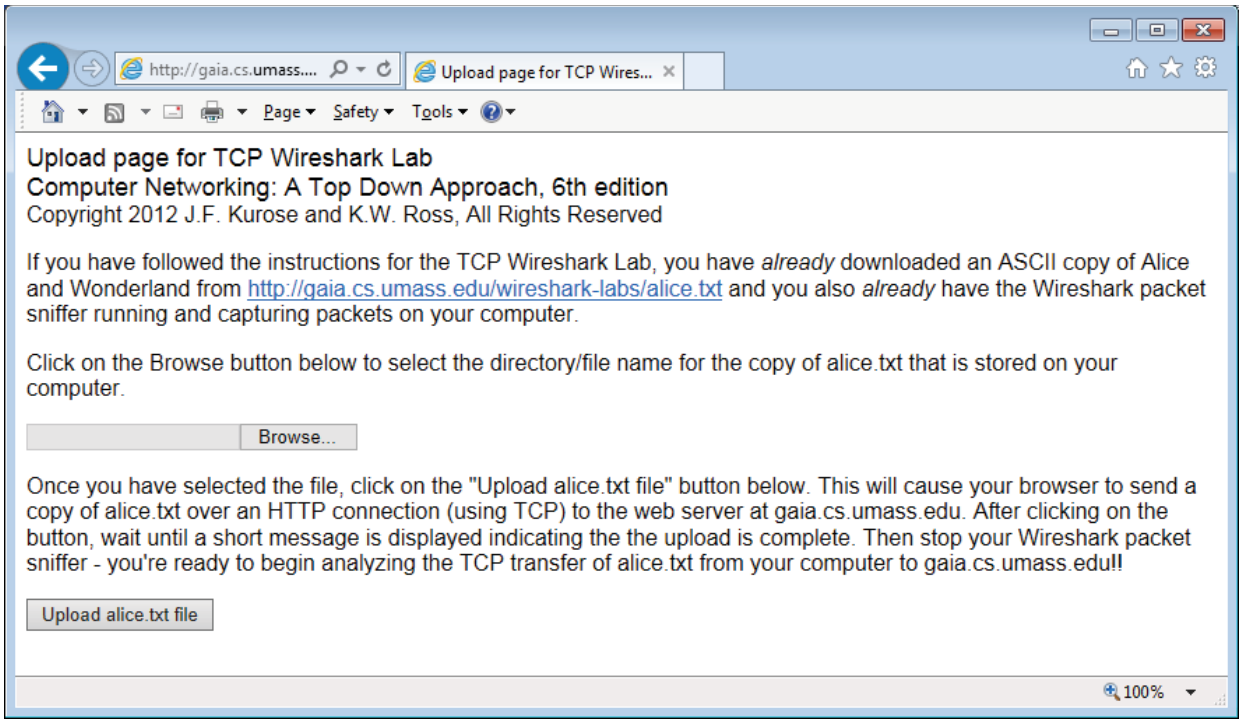
You're going to use the POST method rather than the GET method to transfer a large amount of data *from* your computer to another computer. Of course, you'll be running Wireshark during this time to obtain the trace of the TCP segments sent and received from your computer.

Do the following:

- Start up your web browser.
- Go to: http://gaia.cs.umass.edu/wireshark-labs/alice.txt Retrieve an ASCII copy of Alice in Wonderland. Store this file somewhere on your computer.
- Open the file **alice.txt** using the editing program **Notepad**, copy the whole text and append it to (paste after) the existing content so that the file will be twice as large as the original downloaded file. Save the file without changing the name.

- Next go to: http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html

- You should see a screen (on next page) that looks like:



- Use the *Browse* button in this form to enter the name of the file (full path name) on your computer containing *Alice in Wonderland* (or do so manually). **Don't yet press the "*Upload alice.txt file*" button**.
- Now start up Wireshark and begin packet capture with "tcp" as filter.
- Returning to your browser, press the "*Upload alice.txt file*" button to upload the file to the **gaia.cs.umass.edu** server. Once the file has been uploaded, a short congratulations message will be displayed in your browser window.
- Stop Wireshark packet capture.

A first look at the captured trace

Before analyzing the behavior of the TCP connection in detail, let's take a high level view of the trace. What you should see; is series of TCP segments between your computer and the server **gaia.cs.umass.edu**.

- You should see the initial three-way handshake achieved by SYN, SYN/ACK and ACK segments.
- You should see the first TCP **data**-segment containing the header of an HTTP POST message and probably the first part of the uploaded file.
- Depending on the version of Wireshark you are using, you might see a series of "HTTP Continuation" messages being sent from your computer to the server. Recall from the discussion in the earlier HTTP Wireshark lab, that there is no such thing as an HTTP Continuation message. These are TCP **data**-segments carrying the contents of the file.
- You should also see TCP **ACK**-segments being returned from **gaia.cs.umass.edu** to your computer.

Answer the following questions:

Hint: To answer these questions, it's probably easiest to select the segment containing POST message and explore the details of the packet used to carry this segment.

1. What is the source IP address and TCP port number used by your computer that is transferring the file to **gaia.cs.umass.edu**?

2. What is the destination IP address of **gaia.cs.umass.edu**? On what port number is it sending and receiving TCP segments for this connection?

TCP basics and behavior

Since this lab is about TCP rather than HTTP, make sure that Wireshark's packet list shows information **about the TCP segments** containing the HTTP messages, rather than about the HTTP messages. This will be shown as a series of TCP segments sent between your computer and the web server **gaia.cs.umass.edu**. You will use the packet trace that you have captured to study TCP behavior in the rest of this lab. **Save a file of the trace to your home directory so that you may make use of it at home**.

Note: The sequence and acknowledgement numbers in the following questions should be taken relatively to the initial sequence number (number 0). Wireshark will do this by default, displaying relative numbers.

Answer the following questions:

3. Identify the TCP segments that are used to initiate the TCP connection between the client computer and **gaia.cs.umass.edu**.

   - How many segments are used?
   - What is in the TCP header that identifies the segment as a handshaking segment?

4. Considering the TCP segment containing the HTTP POST, what are the sequence numbers of the **first six data-carrying** segments in the TCP connection (including the segment containing the HTTP POST)?

   Note: that in order to find the POST header field; you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.

5. What is the length of each of these six TCP segments? The length of the TCP segment is only the number of **data** bytes carried inside the segment (excluding the headers).

6. Record the time that each of the six segments (in question 4) has been sent. At which time has an acknowledgement (for each data-carrying segment) been received?

7. After the lab session, make the following calculations and include the results in your report. **Please make a table of the recorded and calculated values of time along with the segment and sequence numbers** (i.e. combining answers to questions 4, 5 and 6).

a) Given the difference between the time when each TCP segment was sent, and the time when its acknowledgement was received, calculate the RTT value for each of those six segments (in questions 4, 5 and 6).

b) Calculate the EstimatedRTT value using the measured RTT values in the answer of question 9. b). Assume that the initial value of the EstimatedRTT is equal to the measured RTT for the first (POST) segment, and then as described in **Section 3.5.3** in the course book (using the EstimatedRTT equation on page 256) calculate the EstimatedRTT for the subsequent segments (they are five!).

TCP Acknowledgements

10. What is it **specified** by the value of the Acknowledgement field in **any** received segment? How does **gaia.cs.umass.edu**; for example, determine this value?

11. How much data (number of bytes) does the receiver **typically** acknowledge in an ACK?

12. Are there cases where the receiver (the web server) is ACKing accumulatively? Please check the whole trace and describe the behavior of TCP when ACKing received data.

TCP congestion control in action

Let's now examine the amount of data (in bytes) sent in each **round** of the transfer session from the client to the server. Rather than (tediously!) calculating this from the trace in Wireshark, you will use one of Wireshark's TCP graphing utilities; **Time-Sequence-Graph** (Stevens) to plot out the amount of data versus time. This Stevens-graph tool is plotting the sequence number of each segment versus the time at which it has been sent.

- In Wireshark, highlight any TCP data-segment from the packet list for the trace that you have gathered when you transferred a file from the client to the server. Then select the menu: *Statistics > TCP Stream Graph > Time-Sequence-Graph (Stevens).*
- You should see a window with a graph of dots where each **dot** represents a TCP segment sent. Note that a set of dots stacked above each other represents a series of segments that were sent back-to-back (pipelined) by the TCP sender.

- **Save a copy of the window of the plot to be included in your report.**

Answer the following questions:

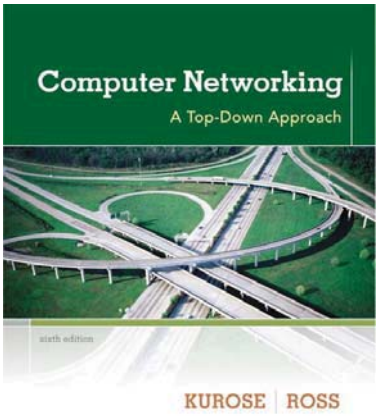
13. How can you identify the TCP's slow-start phase, when does it begin, and whether congestion avoidance takes over? Describe the behavior of the sending TCP during **each round** based on the plot that you get.

14. Where do you find the (advertised) amount of available buffer space at the receiver (the server)? Does the lack of receiver buffer space ever throttle the sender? Explain clearly.

15. What is the **overall throughput** in bit/s (bits transferred per unit time) for the **whole session** of transferring the file? Explain how you calculate this value.

Adopted and Modified based on

Wireshark Lab: DNS v6.0



Supplement to *Computer Networking: A Top-Down Approach, 6th ed.,* J.F. Kurose and K.W. Ross.

"Tell me and I forget. Show me and I remember. Involve me and I understand." Chinese proverb.

© 2005-21012, J.F Kurose and K.W. Ross, All Rights Reserved.

Preparation: Learn about the use and syntax of the command *nslookup* before the lab session.

Reading: Learn carefully about "DNS: The Internet's Directory Service" in **Section 2.5** of the course book. In particular, you may need to review the material on **local (cache-only)** DNS servers, DNS **caching**, DNS **resource records** (**RR**s), DNS **messages**, and the TYPE field in the DNS record.

Main Task: Make extensive use of the *nslookup* tool, which is normally available in most Linux/Unix and Microsoft platforms. Also use Wireshark to investigate the operation of DNS at the client side.

The Domain Name System (DNS) mainly translates hostnames to IP addresses, fulfilling a critical role in the Internet infrastructure. In this lab, you'll take a closer look at the client side of DNS. Recall that the client's role in the DNS is relatively simple; the client sends a *query* to its local DNS server, and receives a *response* back. As shown in Figures **2.21** and **2.22** in the course book, much can go on "under the covers," invisible to the DNS clients, as the hierarchical DNS servers communicate with each other to either recursively or iteratively resolve the client's DNS query.

Nslookup

You should have learned about the use of *nslookup* during the preparation work.

Do the following (and write down the results):

1. Run *nslookup* to obtain the IP address of a Web server in Africa. What is the hostname and IP address of that server? Provide the command and results.

2. Run *nslookup* to determine the authoritative DNS servers for a university in South America. Provide the command and results.

3. Run *nslookup* so that one of the DNS servers obtained in Question 2 is queried for the mail servers for the domain amazon.com. Provide the command and results. Do you get answer? Why or why not.

Tracing DNS with Wireshark

Now that you are familiar with *nslookup*, you'll investigate the DNS messages by capturing the DNS packets that are generated by ordinary Web-surfing activity.

- Use *ipconfig* to clear the DNS cache in your host.

- Open your browser and empty the browser cache.
- Open Wireshark and enter "dns && ip.addr = = host_IP_address" into the display filter, where you obtain host_IP_address with *ipconfig*. This filter removes all packets that neither originate nor are destined to your host.
- Start packet capture in Wireshark.
- With your browser, visit the Web page: http://www.ietf.org
- Stop packet capture.

Please answer the following questions with explanations.

Note: Each DNS message consists of a fixed-length **header** and **sections**. Please identify the different fields of any captured DNS message in the trace.

4. Locate the DNS **query** and **response** messages (resolving **www.ietf.org**). Are they transported using UDP or TCP? Explain why.

5. What is the destination port for the DNS **query** message?

6. What is the source port of DNS **response** message?

7. To what IP address is the DNS query message sent? Use *ipconfig /all* to determine the IP address of your local DNS server. Are these two IP addresses the same?

8. Examine the DNS **query** message. What "type" of query is it? What "sections" does the query message contain?

9. Examine the DNS **response** message. What "sections" does the response message contain? Are there "answers" provided? What does each of these answers contain?

10. This web page contains images. Before retrieving each image, does the host's DNS client issue new DNS queries? Why or why not?

Now let's capture DNS packets when executing the command *nslookup*.

- Start packet capture with Wireshark and use "dns && ip.addr = = host_address" filter.
- Do an *nslookup* on **www.tue.nl**.
- Stop packet capture.

For the purpose of this assignment, and in answering the following questions, ignore any sets of queries/responses rather than the query for (**www.tue.nl**), i.e. the hostname of the web server of the **Eindhoven University of Technology**. These extra queries are specific to the Windows DNS client (which appends primary and connection specific DNS suffixes) and are not normally generated by standard Internet applications. You should instead focus on the intended query/response messages that will normally appear at the end of the packet list.

11. Examine the DNS **response** message. Describe (with your own words) the content of this DNS message, its different parts "sections" and the format of the resource records in each section.