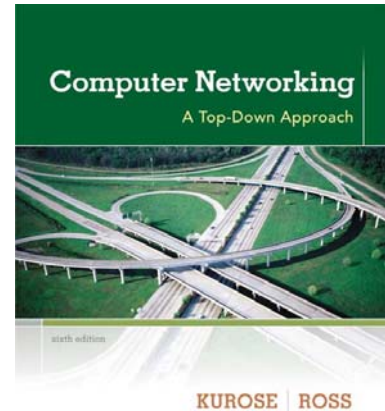


Adopted and Modified based on Wireshark Lab: INTRO v6.0

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: Download and install the Wireshark software. Go to <http://www.wireshark.org/download.html> download and install the Wireshark binary for your computer.

Main Task: In this first Wireshark lab, you'll get acquainted with Wireshark, and make some simple packet captures and observations. You will also learn using the commands **ipconfig/ifconfig** and **nslookup**.

Introduction

One's understanding of network protocols can often be greatly deepened by “seeing protocols in action” and by “playing around with protocols”, observing the sequence of messages exchanged between two protocol entities, delving down into the details of protocol operation, and causing protocols to perform certain actions and then observing these actions and their consequences. This can be done in simulated scenarios or in a “real” network environment such as the Internet. In the Wireshark labs you'll be doing in this course, you'll be running various network applications in different scenarios. You'll observe the network protocols “in action”, interacting and exchanging messages with protocol entities executing elsewhere in the Internet. Thus, you (and the computer that you use) will be an integral part of these “live” labs. You'll observe, and you'll learn, by doing.

The basic tool for observing the messages exchanged between executing protocol entities is called a **packet sniffer**. As the name suggests, a packet sniffer captures (“sniffs”) messages being sent/received from/by your computer; it will also typically store and/or display the contents of the various protocol fields in these captured messages. A packet sniffer itself is passive. It observes messages being sent and received by applications and protocols running on your computer, but never sends packets itself. Similarly, received packets are never explicitly addressed to the packet sniffer. Instead, a packet sniffer receives a *copy* of packets that are sent/received from/by applications and protocols executing on your computer.

Figure 1 shows the structure of a packet sniffer. At the right side of Figure 1 there are protocols (in this case, Internet protocols) and applications (such as a web browser or ftp client) that normally run on your computer. The packet sniffer, shown within the dashed rectangle in Figure 1 is an addition to the usual software in your computer, and consists of two parts. The **packet capture library** receives a copy of every link-layer frame that is sent from or received by your computer. Recall from the discussion from **Section 1.5.2** in the course book (**Figure 1.24**) that messages exchanged by higher layer

protocols such as HTTP, FTP, or DNS, are transported by TCP or UDP and sent as IP packets which are eventually encapsulated in link-layer frames that are transmitted over physical media such as an Ethernet cable. In Figure 1, the assumed physical media is an Ethernet, and so all upper-layer protocols are eventually encapsulated within Ethernet frames. Capturing all link-layer frames thus gives you all messages sent/received from/by all protocols and applications executing in your computer.

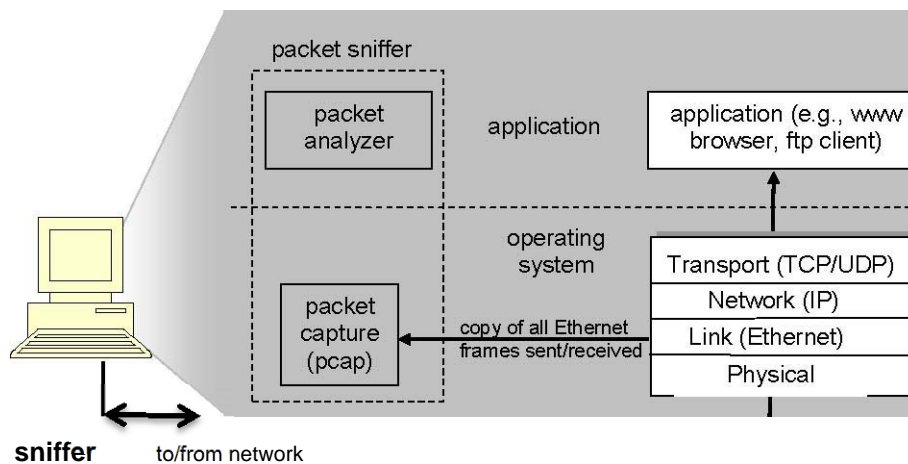


Figure 1: Packet sniffer structure

The second component of a packet sniffer is the **packet analyzer**, which displays the contents of all fields within a protocol message. In order to do so, the packet analyzer must “understand” the structure of all messages exchanged by protocols. For example, suppose we are interested in displaying the various fields in messages exchanged by the HTTP protocol. The packet analyzer understands the format of Ethernet frames, and so can identify the IP datagram within an Ethernet frame. It also understands the IP datagram format, so that it can extract the TCP segment within the IP datagram. Then, it understands the TCP segment structure, so it can extract the HTTP message contained in the TCP segment. Finally, it understands the HTTP protocol and so, for example, knows that the first bytes of an HTTP message will contain the string “GET”, “POST”, or “HEAD”, as shown in **Figure 2.8** in the course book.

We will be using the Wireshark packet sniffer [<http://www.wireshark.org/>] for these labs, allowing us to display the contents of messages being sent/received from/by protocols at different levels of the protocol stack. (Technically speaking, Wireshark is a packet analyzer that uses a packet capture library in your computer). Wireshark is a free network protocol analyzer that runs on Windows, Linux/Unix, and Mac computers. It’s an ideal packet analyzer for our labs. Wireshark is stable, has a large user base and well-documented support that includes:

- user-guide (http://www.wireshark.org/docs/wsug_html_chunked/),
- man pages (<http://www.wireshark.org/docs/man-pages/>), and
- detailed FAQ (<http://www.wireshark.org/faq.html>),

along with rich functionality that includes the capability to analyze hundreds of protocols, and a well-designed user interface. It operates in computers using Ethernet, serial (PPP and SLIP), 802.11 wireless LANs, and many other link-layer technologies.

Getting Wireshark

In order to run Wireshark, you will need to have access to a computer that supports both Wireshark and the *libpcap* or *WinPCap* packet capture library. The *libpcap* software will be installed for you, if it is not installed within your operating system, when you install Wireshark. See <http://www.wireshark.org/download.html> for a list of supported operating systems and download sites.

If you haven't done already, download and install the Wireshark software. Go to <http://www.wireshark.org/download.html>, download and install the Wireshark binary for your computer.

The Wireshark FAQ has a number of helpful hints and interesting tidbits of information, particularly if you have trouble installing or running Wireshark.

Running Wireshark

When you run the Wireshark program, you'll get a startup screen, as shown below:

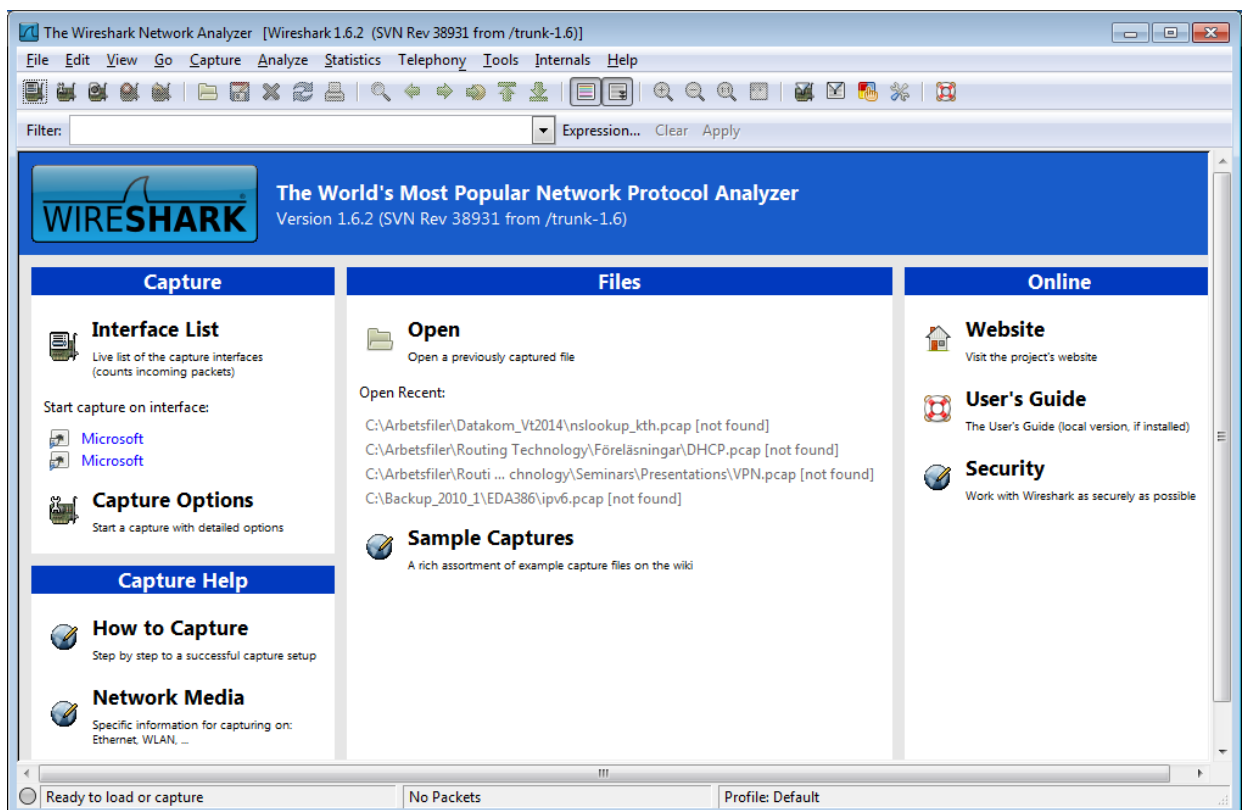


Figure 2: Initial Wireshark Screen

Take a look at the upper left hand side of the screen, you'll see an "Interface list". This is the list of network interfaces on your computer. Once you choose an interface, Wireshark will capture all packets on that interface. In the example above, there is an Ethernet interface and a wireless interface ("Microsoft").

If you click on the active one of these interfaces to start packet capture (i.e. for

Wireshark to begin capturing all packets being sent to/from that interface), a screen like the one below will be displayed, showing information about the packets being captured. Once you start packet capture, you can stop it by using the Capture pull-down menu and selecting Stop.

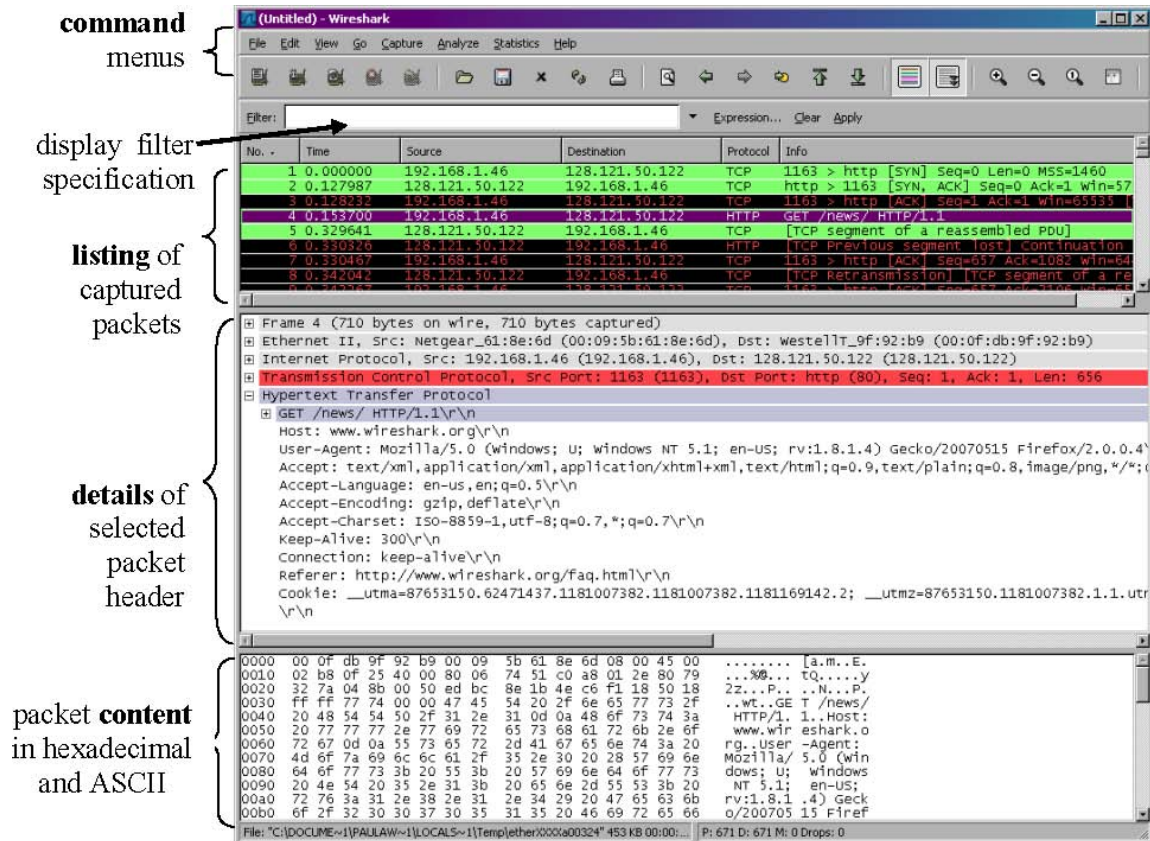


Figure 3: Wireshark Graphical User Interface, during packet capture and analysis

The Wireshark GUI has five major components:

- The **command menus** are standard pull-down menus located at the top of the window. Of interest to us now are the File and Capture menus. The File menu allows you to save captured packet data or open a file containing previously captured packet data, and exit the Wireshark application. The Capture menu allows you to begin packet capture.
- The **packet-listing pane (window)** displays a one-line summary for each packet captured, including the packet number (assigned by Wireshark; this is *not* a packet number contained in any protocol's header), the time at which the packet was captured, the packet's source and destination addresses, the protocol type, and protocol-specific information contained in the packet. The packet listing can be sorted according to any of these categories by clicking on a column name. The protocol type field lists the highest-level protocol that sent or received this packet, i.e., the protocol that is the source or ultimate sink for this packet.
- The **packet-header details pane** provides details about the packet selected (highlighted) in the packet-listing pane. (To select a packet in the packet-listing, place the cursor over the packet's one-line summary in the packet-listing pane and

click with the left mouse button.). These details include information about the Ethernet frame (assuming the packet was sent/received over an Ethernet interface) and IP datagram that contains this packet. The amount of Ethernet and IP-layer details displayed can be expanded or minimized by clicking on the plus minus boxes to the left of the Ethernet frame or IP datagram line in the packet-details pane. If the packet has been carried over TCP or UDP, TCP or UDP details will also be displayed, which can similarly be expanded or minimized. Finally, details about the highest-level protocol that sent or received this packet are also provided.

- The **packet-contents pane** displays the entire content of the captured frame, in both ASCII and hexadecimal formats.
- Towards the top of the Wireshark graphical user interface, is the **display filter** field, into which a protocol name or other information can be entered in order to filter the information displayed in the packet-listing (and hence the packet-header and packet-contents). In the example below, we'll use the display filter field to have Wireshark hide (not display) packets except those that correspond to HTTP messages.

Taking Wireshark for a Test Run

The best way to learn about any new piece of software is to try it out! We'll assume that your computer is connected to the Internet via a wired Ethernet interface. Indeed, it is recommended that you do this first lab on a computer that has a wired Ethernet connection, rather than just a wireless connection. **Do the following tasks in steps:**

1. Start up your favorite web browser, which will display your selected homepage.
2. Start up the Wireshark software. You will initially see a window similar to that shown in Figure 2. Wireshark has not yet begun capturing packets.
3. To begin packet capture, select the Capture pull-down menu and select *Interfaces*. This will cause the “Wireshark: Capture Interfaces” window to be displayed, as shown in Figure 4.

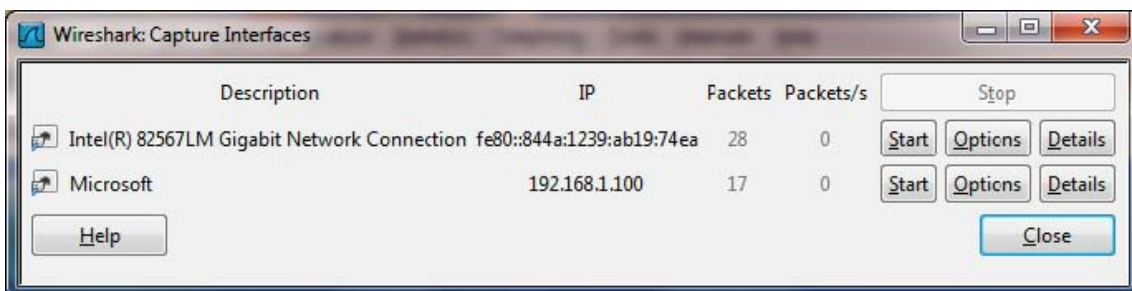


Figure 4: Wireshark Capture Interface Window

4. You'll see a list of the interfaces on your computer as well as a count of the packets that have been observed on that interface so far. Click on *Start* for the interface on which you want to begin packet capture. Packet capture will now begin. Wireshark is now capturing all packets being sent/received from/by your computer!
5. Once you begin packet capture, a window similar to that shown in Figure 3 will

appear. This window shows the packets being captured. By selecting *Capture* pull-down menu and selecting *Stop*, you can stop packet capture. But don't stop packet capture yet. Let's capture some interesting packets first. To do so, we'll need to generate some network traffic. Let's do so using a web browser, which will use the HTTP protocol (that we will study in detail later in lab) to download content from a website.

6. While Wireshark is running, enter the URL:

<http://gaia.cs.umass.edu/wireshark-labs/INTRO-wireshark-file1.html>, and have that page displayed in your browser. In order to display this page, your browser will contact the HTTP server at gaia.cs.umass.edu and exchange HTTP messages with the server in order to download this page, as discussed in **Section 2.2** of the course book. The Ethernet frames containing these HTTP messages (as well as all other frames passing through your Ethernet adapter) will be captured.

7. After your browser has displayed the `INTRO-wireshark-file1.html` page (it is a simple one line of congratulations), stop Wireshark packet capture by selecting stop in the Wireshark capture window. The main Wireshark window should now look similar to Figure 3. You now have live packet data that contains all protocol messages exchanged between your computer and other network entities! The HTTP messages exchanged with the gaia.cs.umass.edu web server should appear somewhere in the listing of packets captured. But there will be many other types of packets displayed as well (see for example the many different protocol types shown in the *Protocol* column). Even though the only action you have taken is to download a web page, there were evidently many other protocols running on your computer that are unseen by the user. You'll learn much more about these protocols as you progress through the course! For now, you should just be aware that there is often much more going on than "meets the eye"!
8. Type in "http" (without the quotes, and in lower case, because all protocol names are in lower case in Wireshark) into the display-filter field at the top of the main Wireshark window. Then select *Apply* (to the right of where you entered "http"). This will cause only HTTP messages to be displayed in the packet-listing pane.
9. Find the HTTP GET message that was sent from your computer to the gaia.cs.umass.edu web server. (Look for an HTTP GET message in the "listing of captured packets" portion of the Wireshark window (see Figure 3) that shows "GET" followed by the gaia.cs.umass.edu URL that you entered. When you select the HTTP GET message, the Ethernet frame, IP datagram, TCP segment, and HTTP message header information will be displayed in the packet details. By clicking on '+' and '-' right-pointing and down-pointing arrowheads to the left side of the packet details pane, you will *minimize* the amount of Frame, Ethernet, Internet Protocol, and Transmission Control Protocol information displayed, or *Maximize* the amount information displayed about the HTTP protocol. Your Wireshark window should now look roughly as shown in Figure 5. (Note, in particular, the minimized amount of protocol information for all protocols except HTTP, and the maximized amount of protocol information for HTTP in the packet details).

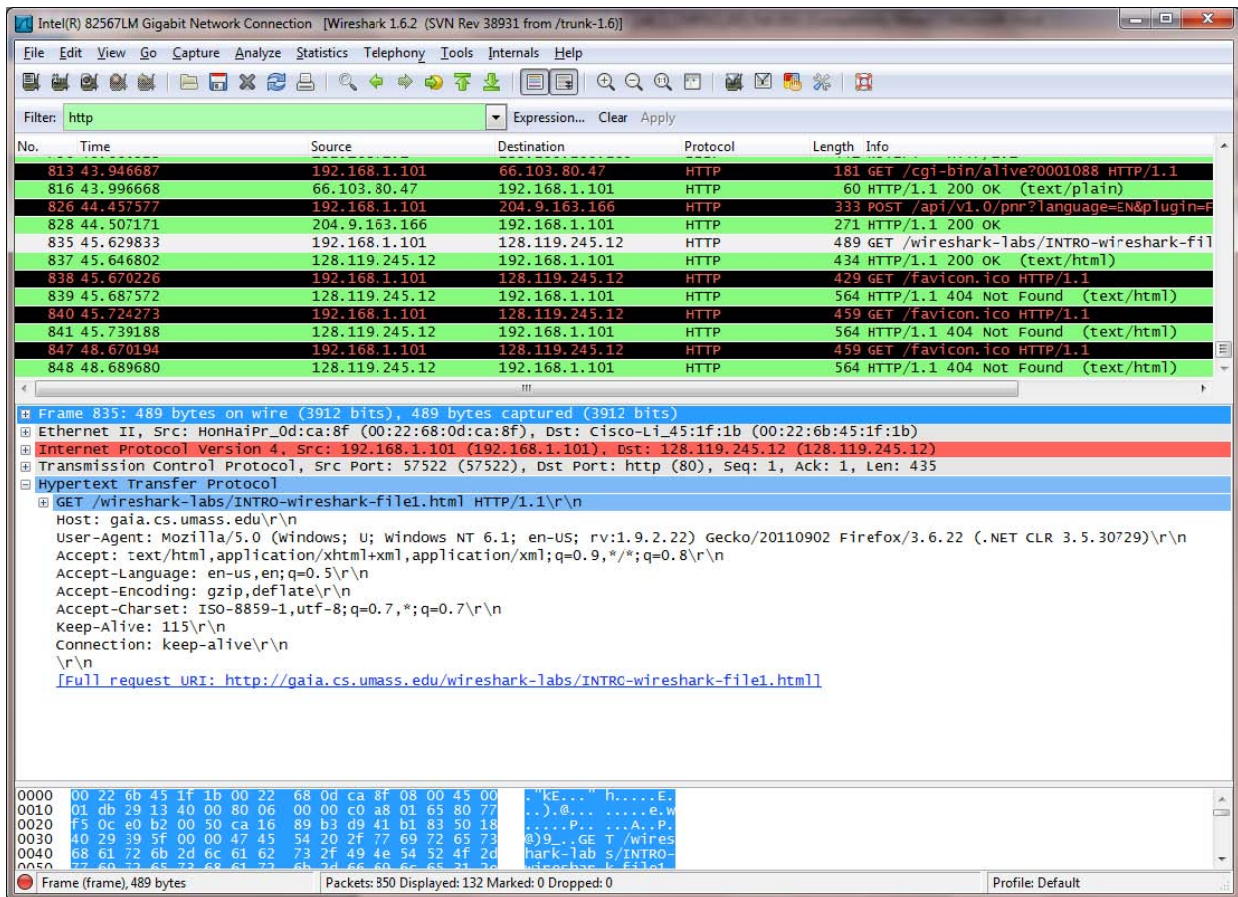


Figure 5: Wireshark window after step 9

To Handout: The goal of this introductory lab is primarily to introduce you to Wireshark. The following questions will demonstrate that you've been able to get Wireshark up and running, and have explored some of its capabilities.

Answer the following questions:

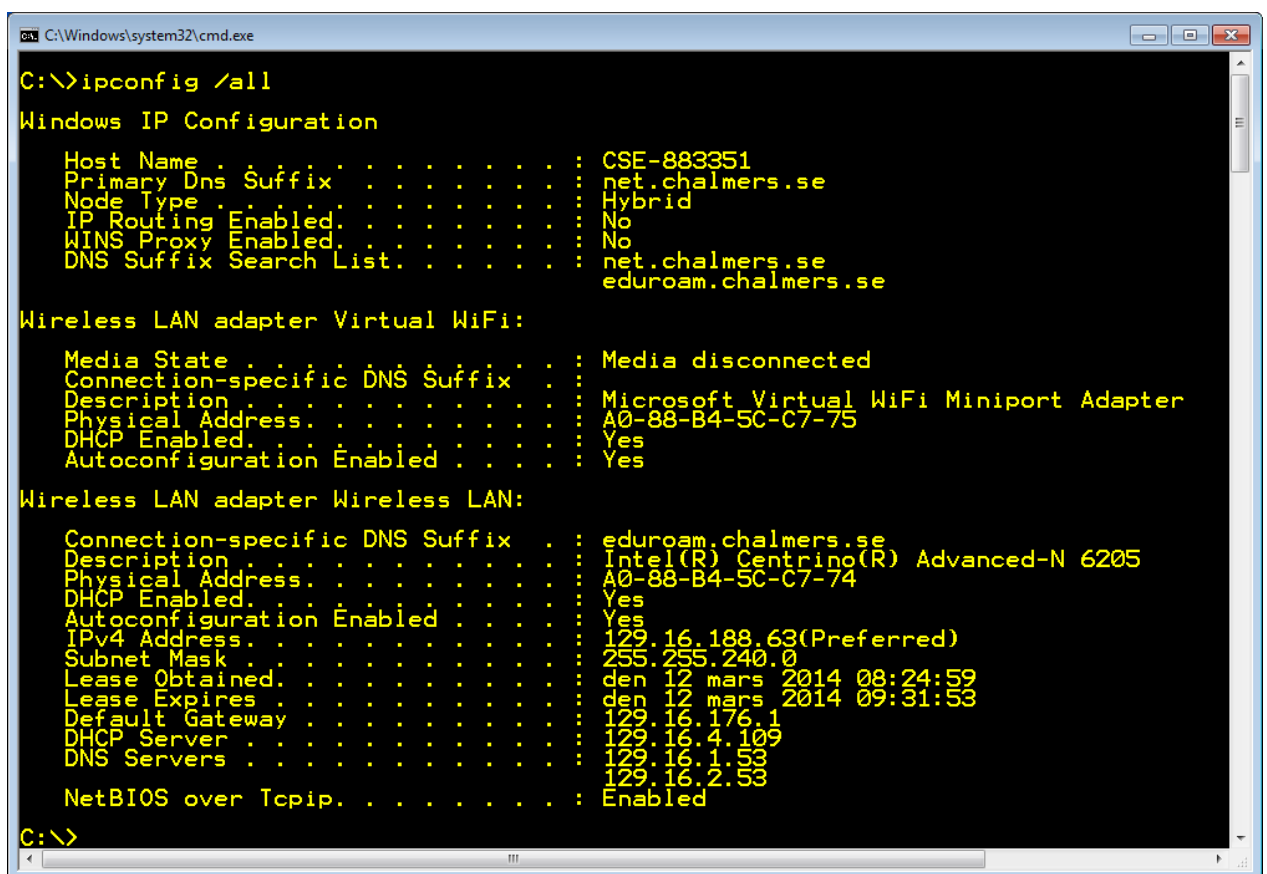
1. List 3 different protocols that appear in the protocol column in the unfiltered packet listing in step 7 above.
2. How long time did it take from when the HTTP GET message was sent until the HTTP OK response was received? (By default, the value of the Time column in the packet-listing pane is the amount of time, in seconds, since Wireshark tracing began. To display the Time field in time-of-day format, select the Wireshark *View* pull-down menu, then select *Time Display Format*, then select *Time-of-day*.)
3. What is the Internet IPv4 address of the gaia.cs.umass.edu? What is the Internet IPv4 address of your computer?
4. Run the command **ipconfig** (or **ifconfig**) on your computer to find out the IP configuration (please refer to Appendix A on the next page). Show the results and include in your preparation report.
5. How can you manage the cached DNS records on your computer?
6. Run the command **nslookup** with all the examples given (Appendix B on the next pages). Show your results and include in your preparation report.

Appendix A: ipconfig/ifconfig

ipconfig (for Windows) and *ifconfig* (for Linux/Unix) are among the most useful little utilities in your host, especially for debugging network issues. Here only *ipconfig* will be described, although the Linux/Unix *ifconfig* is very similar. *ipconfig* can be used to show your host current TCP/IP information, including the IP configuration (address and mask), DNS server addresses, adapter type and so on. For example, if you want all this information about your host, enter:

`ipconfig /all` into the Command Prompt, as shown in the following screenshot.

(`$/sbin/ifconfig` into a terminal, in Linux)



```
C:\Windows\system32\cmd.exe
C:\>ipconfig /all

Windows IP Configuration

Host Name . . . . . : CSE-883351
Primary Dns Suffix . . . . . : net.chalmers.se
Node Type . . . . . : Hybrid
IP Routing Enabled. . . . . : No
WINS Proxy Enabled. . . . . : No
DNS Suffix Search List. . . . . : net.chalmers.se
    eduroam.chalmers.se

Wireless LAN adapter Virtual WiFi:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . . . . . :
Description . . . . . : Microsoft Virtual WiFi Miniport Adapter
Physical Address. . . . . : A0-88-B4-5C-C7-75
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes

Wireless LAN adapter Wireless LAN:

Connection-specific DNS Suffix . . . . . : eduroam.chalmers.se
Description . . . . . : Intel(R) Centrino(R) Advanced-N 6205
Physical Address. . . . . : A0-88-B4-5C-C7-74
DHCP Enabled. . . . . : Yes
Autoconfiguration Enabled . . . . . : Yes
IPv4 Address. . . . . : 129.16.188.63(Preferred)
Subnet Mask . . . . . : 255.255.240.0
Lease Obtained. . . . . : den 12 mars 2014 08:24:59
Lease Expires . . . . . : den 12 mars 2014 09:31:53
Default Gateway . . . . . : 129.16.176.1
DHCP Server . . . . . : 129.16.4.109
DNS Servers . . . . . : 129.16.1.53
    129.16.2.53
NetBIOS over Tcpip. . . . . : Enabled

C:\>
```

ipconfig is also very useful for managing the DNS information stored in your host. In Section 2.5 we learned that a host can cache DNS records it recently obtained. To see these cached records, after the prompt `C:\>` provide the following command:

```
ipconfig /displaydns
```

Each entry of the output shows the remaining Time to Live (TTL) in seconds. In order to clear the cache, enter:

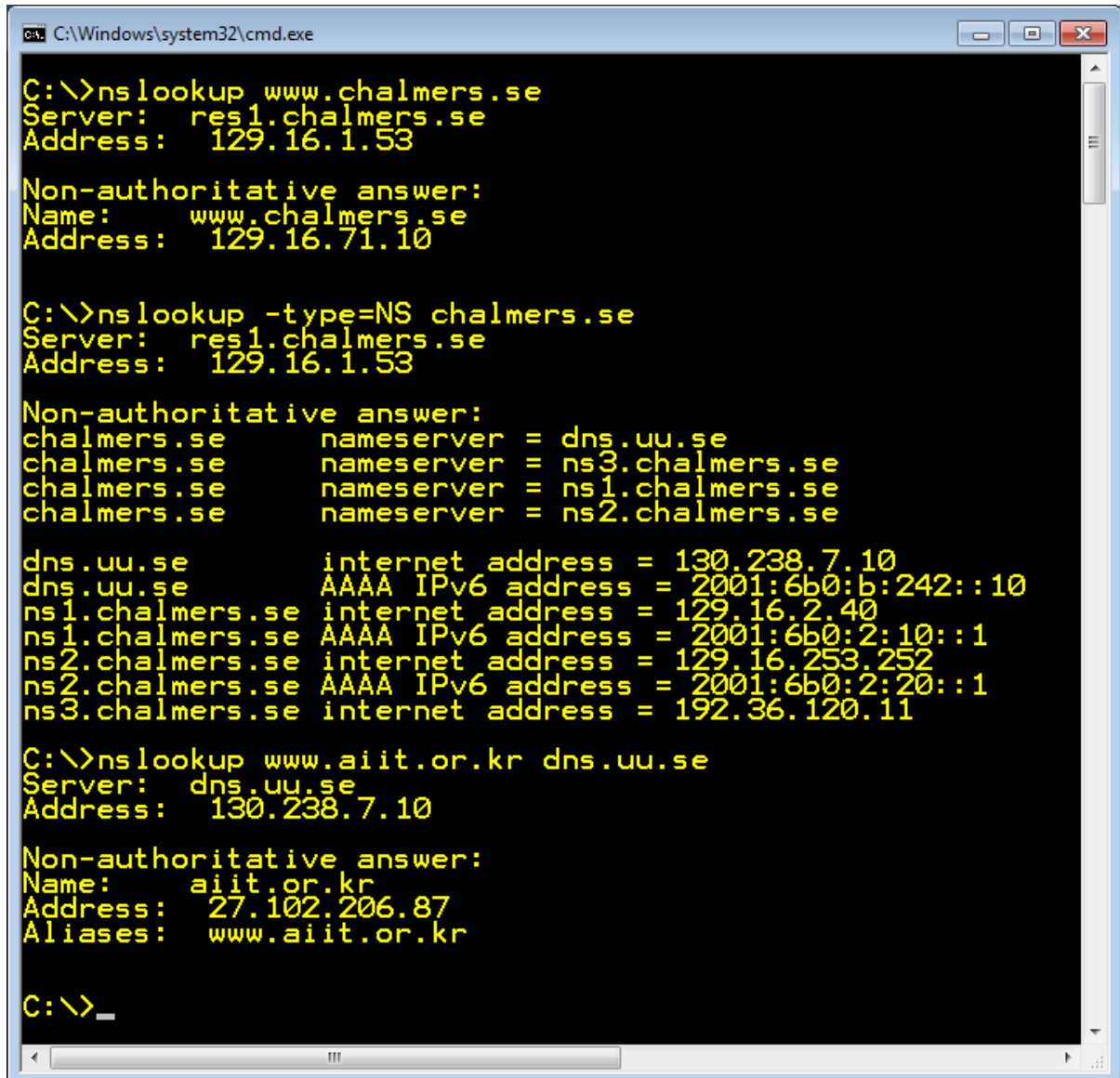
```
ipconfig /flushdns
```

(`$/sudo /etc/init.d/nscd restart` into a terminal, in Linux)

Flushing the DNS cache clears all entries and reloads the entries (if any) from the hosts file.

Appendix B: nslookup

In its most basic operation, *nslookup* tool allows the host running the tool 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 course book 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.



```
C:\Windows\system32\cmd.exe

C:\>nslookup www.chalmers.se
Server:  res1.chalmers.se
Address: 129.16.1.53

Non-authoritative answer:
Name:    www.chalmers.se
Address: 129.16.71.10

C:\>nslookup -type=NS chalmers.se
Server:  res1.chalmers.se
Address: 129.16.1.53

Non-authoritative answer:
chalmers.se      nameserver = dns.uu.se
chalmers.se      nameserver = ns3.chalmers.se
chalmers.se      nameserver = ns1.chalmers.se
chalmers.se      nameserver = ns2.chalmers.se

dns.uu.se        internet address = 130.238.7.10
dns.uu.se        AAAA IPv6 address = 2001:6b0:b:242::10
ns1.chalmers.se  internet address = 129.16.2.40
ns1.chalmers.se  AAAA IPv6 address = 2001:6b0:2:10::1
ns2.chalmers.se  internet address = 129.16.253.252
ns2.chalmers.se  AAAA IPv6 address = 2001:6b0:2:20::1
ns3.chalmers.se  internet address = 192.36.120.11

C:\>nslookup www.aiit.or.kr dns.uu.se
Server:  dns.uu.se
Address: 130.238.7.10

Non-authoritative answer:
Name:    aiit.or.kr
Address: 27.102.206.87
Aliases: www.aiit.or.kr

C:\>_
```

The screenshot above shows the results of three independent *nslookup* commands (displayed in the Windows Command Prompt). In this example, the client host is located at the campus of Chalmers University in Johanneberg, where one of the default local DNS servers is res1.chalmers.se.

When running *nslookup*, if no DNS server is specified, then *nslookup* sends the query to the default DNS server, which in this case is res1.chalmers.se.

Consider the first command:

```
nslookup www.chalmers.se
```

In words, this command is saying “please send me the IP address for the host (web server) www.chalmers.se”. As shown in the screenshot, the response for this command provides two pieces of information: (1) the hostname and IP address of the DNS server that provides the answer; and (2) the answer itself, which is the hostname and IP address of www.chalmers.se.

Although the response has come from the local DNS server at Chalmers University, it is quite possible that this local DNS server **iteratively** contacted several other DNS servers to get the answer, as described in **Section 2.5** of the course book.

Now consider the second command:

```
nslookup -type=NS chalmers.se
```

In this example, the user has provided the option “-type=NS” for the domain “chalmers.se”. 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 hostnames of the authoritative DNS servers for the domain chalmers.se”. (When the `-type` option is not used, *nslookup* uses the default, which is to query for **type A** records.) The answer, displayed in the above screenshot, first indicates the DNS server that is providing the answer (which is the default local DNS server) along with four Chalmers name servers. Each of these servers is indeed an authoritative DNS server for the hosts on the Chalmers campuses. However, *nslookup* also indicates that the answer is “non-authoritative,” meaning that this answer has come from the cache of the local server rather than directly from an authoritative DNS server.

Finally, the answer also includes the IP addresses of the authoritative DNS servers at Chalmers. (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” as Additional Information and *nslookup* displays the result.)

Finally consider the third command:

```
nslookup www.aiit.or.kr dns.uu.se
```

In this example, the user has indicated that the query should be sent to the DNS server dns.uu.se rather than to the default DNS server (res1.chalmers.se). Thus, the query and reply transaction takes place directly between the querying host and dns.uu.se. In this example, the DNS server dns.uu.se 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).

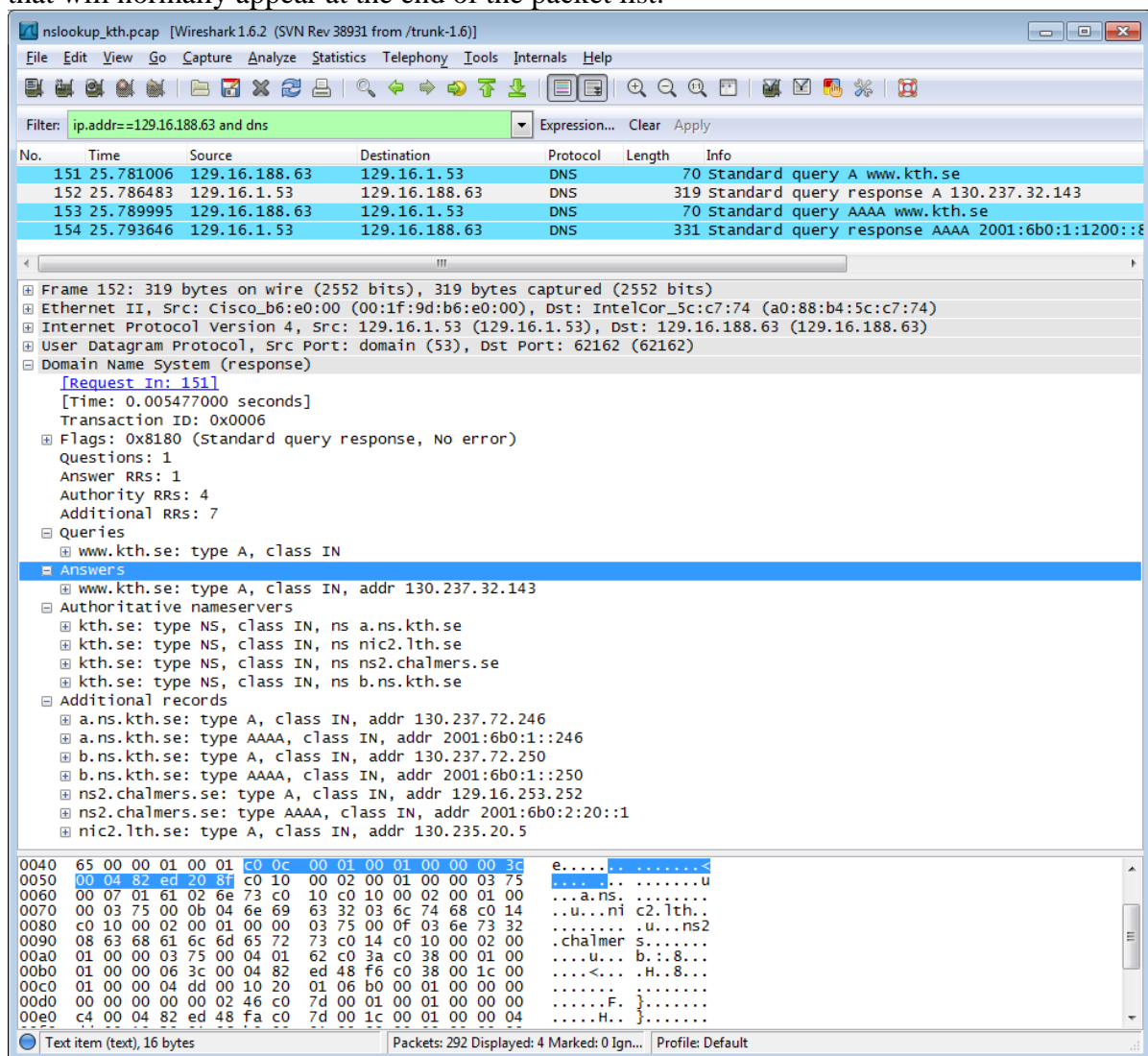
Now that you have gone through a few illustrative examples, you are perhaps wondering about the general syntax of *nslookup* command. The syntax is:

```
nslookup -option1 -option2 name-to-find dns-server
```

In general, *nslookup* can be run with zero, one, two or more options. And as you 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.

The screenshot; shown below, is obtained when using Wireshark to capture the packets during a similar example when *nslookup* is used to resolve www.kth.se. The DNS client (here in Windows) has actually sent many queries but you can see that only two DNS queries of type A and AAAA (and the corresponding two responses) are about www.kth.se.

These extra queries; if appeared 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.



Do a similar packet capturing when running *nslookup*. Highlight the response message and study the different parts, including the header and the DNS information in the message sections (Queries, Answers, Authoritative nameservers and Additional records). In each section there may be zero or a number of Resource Records. Learn about the format of the RR (specially the **name**, **type**, and **value**).