Laboratory 3 - Computer Systems & Architecture & Computer networking

Introduction

Todays Internet is arguably the largest engineered system ever created by mankind, with millions of communication links and connected computers with billions of users. Internet as an infrastructure that provide services to applications. The applications are said to be distributed applications since they involve multiple end system that exchange data with each other. All data exchanging in the Internet applications that involves two or more communicating remote entities is governed by a protocol. Protocols are running everywhere in the Internet. The Internet and computer networks in general make extensive use of protocols. Different protocols are used to accomplish different communication tasks. As Internet application developers you need to have good knowledge about the Internet protocols. The objective of the lab assignment is to provide an understanding of the what, why and how of Internet protocols. After you have completed this lab you should be able to identify and analysis some of the Internet protocols and services.

Literature for laboratory assignment

In addition to the course book (chapter 12) you can read about the protocols used during the lab at the following sites.
The TCP / IP Guide (http://www.tcpipguide.com/free/index.htm) by Charles M. Kozierok is good, but relatively long source. *Wikipedia* in this case is a good source to start with a (just only)short introduction.

HTTP - http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
TCP - http://en.wikipedia.org/wiki/Transmission_Control_Protocol
IP - http://en.wikipedia.org/wiki/IPv4
DNS - http://en.wikipedia.org/wiki/Domain_Name_System
SMTP - http://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol
Mailformat - http://en.wikipedia.org/wiki/Email#Message_format

Before you begin the lab you should have:
Read through this whole laboratory instruction .
Downloaded & install the latest version of Wireshark from :
http://www.wireshark.org/download.html

Lab Assignment 1:

This assignment preparing tasks which are supposed to be answered before the assignments 2 and 3 begin, and will provide a background to what occurs during the assignments 2 and 3.

Answer the following questions:

Protocol Layers

1.1 Which layers are included in the TCP / IP model?
1.2 What are the advantages and disadvantages of using a layer model?

Ports

1.3 What is the purpose of using ports?
1.4 In total there are 65536 ports and one usually divide them into three areas. What do these areas are called and the port numbers belong to which areas?
1.5 What is the default port for HTTP?
1.6 Which is the standard port for SMTP?

HTTP

1.7 What is the purpose of the HTTP protocol?
1.8 What response code is received from a web server if everything has gone well?
1.9 What response code is obtained if the web server can not find the resource that is in demand?
1.10 Which two response codes, you can get if you do not have rights to a requested resource?
1.11 What are the different requests can be done using HTTP?

SMTP

1.12 What is the purpose of the SMTP protocol?
1.13 How the command appears to indicate the sender's address to SMTP?
1.14 How the command appears to enter the destination address under SMTP?
1.15 How the command appears to indicate the actual text of an e-mail to SMTP?
1.16 Which command asks an SMTP server on which extensions it supports?
1.17 An e-mail message consists of two parts: Header and Body. Which two fields must always be in the header? Hint: Read about the format of e-mail Internet?
(hint: read more on the following link:
https://en.wikipedia.org/wiki/Special:Search/Email-Messageformat)

DNS

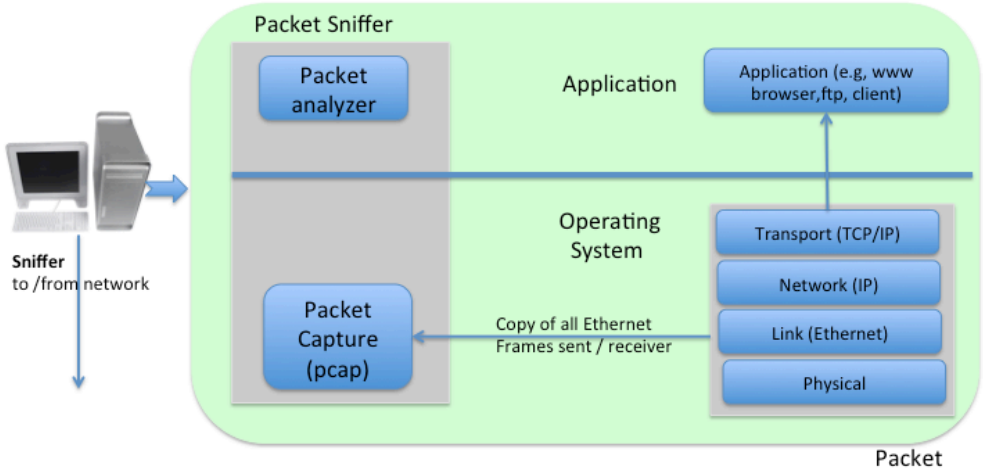
1.18 What is the purpose of DNS?
1.19 What type of DNS resource record should you ask for if you want to know the IPv4 address for a domain?
1.20 What type of DNS record should you ask for if you want to know the IPv6 address for a domain?

Wireshark

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 using your own computer (or you can borrow a friends; let me know if you don't have access to a computer where you can install/run Wireshark). You'll observe the network protocols in your computer "in action," interacting and exchanging messages with protocol entities executing elsewhere in the Internet. Thus, you and your computer will be an integral part of these "live" labs. You'll observe, and you'll learn, by doing.

In this first Wireshark test, you'll get acquainted with Wireshark, and make some simple packet captures and observations.

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 application and protocols executing on your machine.



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 in Figure 1. 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. Finally, 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 lecture slides.
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 – it is stable, has a large user base and well-documented support that includes a user-guide (http://www.wireshark.org/docs/wsug_html_chunked/), man pages (http://www.wireshark.org/docs/man-pages/), and a detailed FAQ (http://www.wireshark.org/faq.html), 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 (if the OS on which it's running allows Wireshark to do so).

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
Download and install the Wireshark software:
• Go to http://www.wireshark.org/download.html and 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.

Answer the following questions:

1.21 How can you define a filter in Wireshark to only see the traffic that contains the HTTP?
1.22 How to follow a specific TCP stream in Wireshark?

Lab Assignment 2:

The aim of this lab is to familiarize you with some common protocols by manually using them.

How to ask a question to a web server?
HTTP is a text-based protocol and can be ideally used "by hand", although of course it is much more common to use a browser that handles HTTP communication for the user. In this section of the lab you will communicate with a web server using
Telnet (https://en.wikipedia.org/wiki/Telnet) a protocol for sending data over TCP.
Telnet is often used just to test protocol that sends data in clear text, such as HTTP, SMTP and POP3.

Task 2.1
Open a terminal window and connect to the web server using telnet. The web server address is ***da.dsv.su.se*** and port is **80**. If you are running Windows, instead of command prompt use PuTTY
(http://www.chiark.greenend.org.uk/~sgtatham/putty/)

(do not forget to connect PuTTY in **Raw** mode).

2.1.1 Ask for the resource */art.txt*
2.1.2 Ask for resource */layers.html*
2.1.3 Ask for resource */grades.txt*

Please note , you need to reconnect after every question because the connection is closed by the Web server when it sent its reply.

Answer the following questions for each requested resource:

2.1.4 How did your conversation with the Web server look like?
2.1.5 What response codes you got?
2.1.6 What are response codes?
2.1.7 What type of content is in the response from the web server?

Note: If you get the answer *400 Bad Request* means you have sent an incorrect request. It is not the answer is request in the lab and you need to review your inquiry to see what is wrong and try again.

Task 2.2 (the application layer)
In the previous task, you looked at how HTTP works. In this task, we will look at how the protocols in the layers below the application layer looks with the help of *Wireshark.*

Just as in the first task, we use telnet / PuTTY to connect to a web server. This time, we will also start *Wireshark* to look at the packets of more than application level.

- Start *Wireshark* and start a new "capture" of the network interface you use to connect to the Internet. This is different for different operating systems (and Linux different between different distribution functions)

and connection means so it is best if you tried *Wireshark* before so you know which interface to use.

- Start PuTTY / telnet and connect to da.dsv.su.se on port 80 and ask for the resource /art.txt
- Stop your "capture" of Wireshark and apply a filter to only see HTTP packets.
- Look at the package in which you ask for */art.txt*

Answer the following Questions:

2.2.1 What are the IP addresses of both sender & receiver?
2.2.2 Between which ports the packet was sent?
2.2.3 Which transport protocol used?
2.2.4 How large is the package in bytes?
2.2.5 Between which MAC addresses the packet was send?

Lab Assignment 3:

The previous application layer protocols we looked at have sent data in plain text. In this task, we will look closely at DNS, a service that does not send data in a format that is readable outright.

In this task, you should find out the IP address that belongs to *da.dsv.su.se* . You should also use *Wireshark* to see more information about a package that contains the DNS information.

- Start Wireshark and start a new "capture" of the interface that you use to connect to the Internet.
- Start a terminal on MAC or Linux or on windows command prompt (http://windows.microsoft.com/en-us/windows/command-prompt-faq#1TC=windows-8)

  and enter the following:

  On windows: *ipconfig /flushdns*
  On Mac: *dscacheutil -flushcache*
  On Linux: */etc/rc.d/init.d/nscd restart*
  If you lack nscd you can continue to the next step.

- On **Linux / OS X (Mac),** type *dig da.dsv.su.se*
- On **Windows**, type *nslookup da.dsv.su.se*

- Turn off your "capture" of *Wireshark* and apply a filter to only see DNS traffic.
- Find the packet that contains the "*Standard query response*" for *dsv.su.se*

Answer the following Questions:

3.1which transport protocol was used by DNS?

3.2 Between which ports packets were sent?