KTH Electrical Engineering

IK2218
Protocols and Principles of the Internet

Lab 1

ARP, IP, UDP, TCP

September 2017

Department of Network and Systems Engineering
School of Electrical Engineering
KTH, Royal Institute of Technology
Stockholm, Sweden

Contents

1 Introduction **4**
1.1 Goals of the Lab 4
1.2 Requirements and reporting 4

2 Preparation for the lab **5**
2.1 Lab equipment and software instructions 5
2.2 Software tools 6
2.3 Reading list 7
2.4 Operating system instructions 7
2.5 Wireshark: A network packet sniffer 8
2.5.1 Saving Wireshark output 8
2.5.2 Plotting graphs in Wireshark 9

I Network Configuration and Basic Diagnostic Tools **11**

3 Configuring your host **11**

4 Netstat **13**

5 Ping **14**

6 Traceroute **14**

7 Arp **15**

II User Datagram Protocol (UDP) **16**

8 Basic operation of UDP **16**

9 UDP and fragmentation **17**

III Transmission Control Protocol (TCP) **18**

10 Basic operation of TCP **18**

11 TCP connection management **19**
11.1 Connection establishment and termination 19
11.2 Connecting to a non-existing port 20

12 TCP data transfer **20**
12.1 Interactive application 20
12.2 Bulk transfer 21

13 TCP retransmissions 22

14 TCP congestion control 23

IV Application Layer Protocols 24

15 Telnet 24

16 Anonymous ftp 24

17 Mail 25

18 Completing the lab 25

1 Introduction

This syllabus contains the information about Lab 1. The topic of Lab 1 is basic network configuration, study of transport layer protocols UDP and TCP, and the introduction to some basic application layer protocols. Please read the syllabus carefully.

1.1 Goals of the Lab

The objective of this lab is the following:

- Give you an overview of the most important network diagnosis tools, such as ping, traceroute, netstat.
- Introduce you to `Wireshark`, a tool often used to capture network traffic.
- Make you familiar with the operation of the ARP table.
- Introduce you to common applications such as `telnet`, `ftp` and `smtp`.
- Give you insight into the operation of UDP, and fragmentation for UDP.
- Demonstrate the difference between UDP and TCP data transmission.
- Give insight into the TCP connection establishment and termination.
- Expose the difference between TCP bulk and interactive data transfer.
- Help understanding TCP congestion and flow control.

1.2 Requirements and reporting

In order to pass the lab, you need to fulfill the following:

- Pass the preparation quiz on Canvas before the lab starts. The preparation quiz is a prerequisite for performing the lab.
- Perform the lab. You need to be present during the lab session.
- Show your solutions to the questions to one of the lab assistants.
- Submit a group report with your answers as well as the output data from the measurements that you will perform in the second part of this lab. The lab report shall be written in English and contain the following:
  - Names, social security numbers (personnummer) and e-mail addresses of each participant.
  - Answers to the questions in the exercises in Sections 10-14.
  - `Wireshark` output from the exercises in Sections 10-14, including the summary data, the detailed data, and the graphs. It is explicitly stated in the exercises what data you should save.

Each group shall submit one lab report via Canvas as a group assignment, within one week after the lab session. Depending on the quality of the lab report, further revision might be required. The Canvas groups for labs will be created by the lab assistants during the lab sessions.

2 Preparation for the lab

In order to prepare for the lab, read the items in the reading list and answer the questions below. The lab is limited in time, so it is necessary that preparations are made in advance. The schedule is tight.

2.1 Lab equipment and software instructions

For this lab, every group of two persons will use the following equipment:

- One end-host equipped with one Ethernet interface, running Linux and the software required to perform the lab.
- One straight Ethernet cable.

Ethernet cables will be provided in the lab. As for the end-host, you can either bring your own laptop or use one of the laptops available in the lab. We recommend that you use your own laptop for the lab, for two reasons. First, you will have better opportunity to get familiar with the tools used during the lab which leaves you more time for solving the lab problems at the lab session. You also get a number of useful networking tools installed on your computer that you can use later. Second, your laptop is presumably more recent that the machines that we can provide, hence you will experience less delays (startup, etc).

If you would like to use your own laptop for performing the lab, please ensure that you have an Ethernet port (either built in, or via a USB adapter). You can then choose from 4 options:

1. If your computer is running Linux, you only need to ensure that your system supports all networking functionalities required to perform the lab. The check-list is given in Section 2.2. Please install all tools listed there.
2. If you are a Windows user (or use Mac OS X), you can set up a virtual machine running Linux from a bootable CD. For installing Linux with VirtualBox, please refer to the installation manual available on the course web pages under Labs. (If you are already familiar with the installation procedure, you can just proceed to download and install the ISO image which we will use in the lab—it contains all the necessary tools to perform the lab. The image is available at `https://kth.box.com/s/ns9fi06nimhp18qjtuqhyghvdr6bf08r` (to access use the password lab).
3. You can also try to use a Raspberry Pi to perform the lab. Raspberry Pis will be provided in the lab, and you only need to connect your computer to a Raspberry Pi to use it remotely. For detailed information about how to connect to the Raspberry Pis in the lab, please check the Raspberry Pi tutorial available in the Lab module on Canvas.

4. There is a certain number of USB sticks that the teaching assistants will have at disposal to distribute during the lab session. These USB sticks contain the same version of Linux as the images provided for VirtualBox, and can be used for booting the operating system. Please verify that your laptop can boot from a (bootable) USB stick.

Note also that we can not provide full troubleshooting support for Mac OS X users. As mentioned above, one host is sufficient for a group of two students. Thus, if you know who will be your lab partner, you can decide together and choose the optimal option for your group.

2.2 Software tools

Please make sure that the TCP congestion control algorithm running on your computer is Reno (instead of cubic). To check the congestion control algorithm running on your computer, please run the following command.

```
sudo sysctl net.ipv4.tcp_congestion_control
```

If your computer is not running Reno as the congestion control algorithm, you can try to switch to Reno by running the following command as the root user and restart your computer.

```
echo reno > /proc/sys/net/ipv4/tcp_congestion_control
```

The following software tools will be used in the lab. Manual pages of these tools are installed on all Linux computers (man command) and they can also be found on the web (for example `http://linux.die.net`). Read the manual pages of these commands. If you plan to use your own Linux computer during the lab, ensure that you have a working installation of all of the listed tools.

- `arp` - Manipulate the system ARP cache.
- `ftp` - Internet file transfer program.
- `ifconfig` - Configure a network interface.
- `netstat` - Print network connections, routing tables, interface statistics, etc.
- `ping` - Send ICMP ECHO_REQUEST to network hosts.
- `route` - Show and manipulate the IP routing table.
- `sendmail` - An electronic mail transport agent.
- `ssh` - Remote login program.
- `tc` - Show and manipulate traffic control settings.
- `telnet` - User interface to the TELNET protocol, for interactive communication with another host.
- `traceroute` - Print the route packets trace to network host.
- `ttcp` - Test TCP and UDP performance.

- `wireshark` - Interactively dump and analyze network traffic. Read more about Wireshark in Section 2.5.

Some Linux distributions (e.g. Fedora) come with `ttcp` installed. If this is not the case with the distribution you use, you will need to install it yourself. You can do this by downloading the source file, e.g. from `www.netcore.fi/pekkas/linux/ipv6/ttcp.c`. and compiling it with

```
gcc ttcp.c -o ttcp
```

Then, copy the file to your */usr/bin* folder:

```
sudo cp ttcp /usr/bin
```

Note that certain distributions may have alternative versions of `ttcp` available, e.g. `nttcp` in Ubuntu. Please do NOT use these versions, as different versions may be incompatible with one another.

Wireshark is another tool that you need to install if you do not have it. Guidelines are given here:
`https://www.wireshark.org/docs/wsug_html_chunked/ChapterBuildInstall.html`.

In the online Wireshark manual, read carefully the sections 2.1, 2.2, and follow the instructions from sections relating to UNIX operating systems (from Section 2.5 onwards).

2.3 Reading list

Forouzan, "TCP/IP Protocol Suite", 3rd ed., Chapters 4, 5, 6, 7, 8, 9, 11, 12.
Forouzan, "TCP/IP Protocol Suite", 4th ed., Chapters 4, 5, 6, 7, 8, 9, 13, 14, 15.

- Fragmentation
- UDP, TCP
- EP2120/IK2218 Lecture Notes: Addressing, IP, Routing, ARP, UDP, TCP

2.4 Operating system instructions

Note that these instructions apply only for those using the provided ISO image, either with VirtualBox or, booting from the provided USB sticks.
The username and password that you will use to log in to the system are

username:	*user*
password:	*1234*

Several commands (e.g., `ifconfig` and `route`) used during the lab can only be executed with superuser (*root*) privileges. In order to be able to execute these commands as *root* you will have to precede them with the command `sudo`. As an example, to configure an interface using `ifconfig` you have to type

```
sudo ifconfig eth1 192.168.0.25 netmask 255.255.255.0 broadcast 192.168.0.255
```

The first time that you execute the `sudo` command you will be asked to provide your password. An alternative to using `sudo` is to start a shell as `root`, and to enter the commands in this shell. You can do this by typing:

```
sudo -s
```

Since during this lab you are primarily going to use system commands, we recommend you to start a shell as `root`.

2.5 Wireshark: A network packet sniffer

`Wireshark` is a packet sniffer that enables you to inspect the packets on the network. You are going to use Wireshark in the upcoming sections to study various protocols in the Internet protocol stack. This is a brief introduction to some basic functions of Wireshark that you will need to complete the exercises.

In order to start `Wireshark` do the following:

- wireshark & (make sure you do this as root user - use `sudo` or run it in a shell as `root`)
- Start a capture and set it to update list in real-time.

The `Wireshark` window consists of three frames. A list of the captured packets is shown in the top frame. When a packet in the list is marked, the content is shown in the middle and lower frames. The middle frame shows the packet in symbolic form, while the lower frame shows the packet in raw, hexadecimal format. In the middle frame, the fields may be expanded to show a more detailed view.

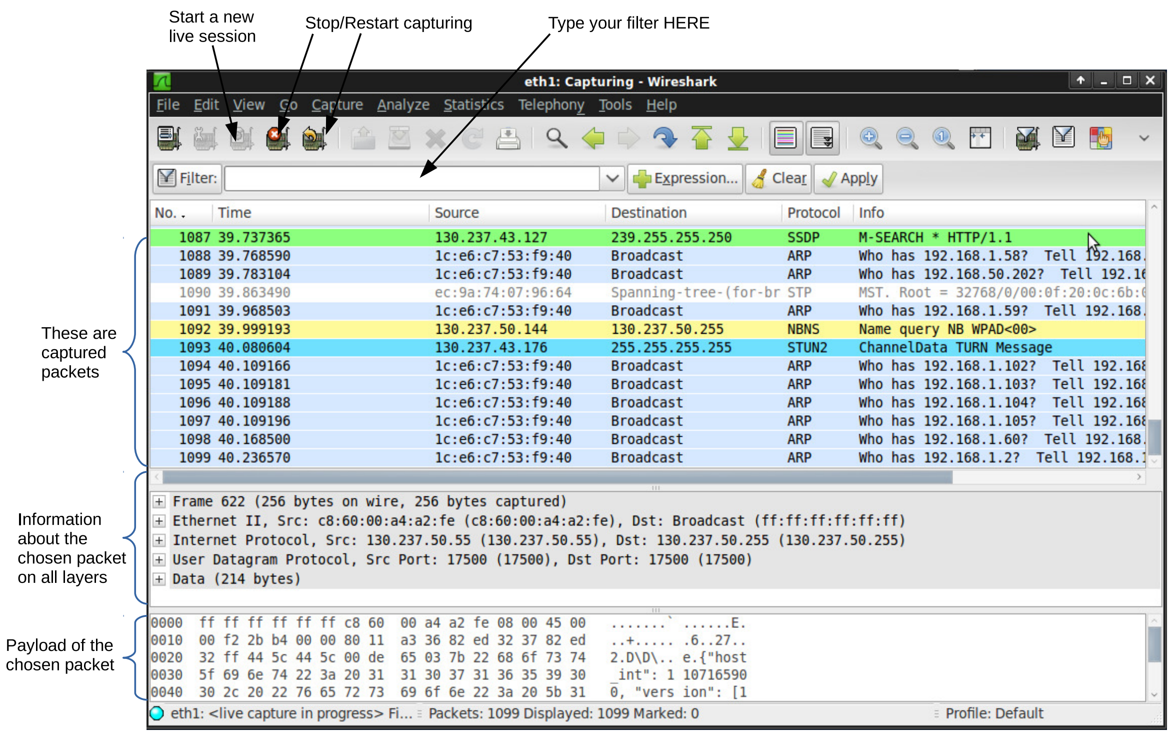
A session is started by clicking on Capture»Start. This brings up a capture options window. Typically, the display option *Update list of packets in real time* and *Automatic Scrolling in live capture* should be selected. When started, the packets are shown in the main area, and a capture window is shown. To stop the capture, click on the *stop* button in this window.

Some statistics can be shown by choosing Tools»Summary.

2.5.1 Saving Wireshark output

Data can be saved in two formats: summary or detailed, as follows:

- Click the *File»Print* on the menu bar.
- Choose *Plain Text format*.
- Choose *Output to file*.
- Name a file.
- For saving a summary, choose *Packet summary line* in the *Packet Format* frame.
- For saving detailed data, choose *Packet details*.


Start a new live session
Stop/Restart capturing
Type your filter HERE
These are captured packets
Information about the chosen packet on all layers
Payload of the chosen packet
eth1: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression...
Clear
Apply

```
No. . Time          Source              Destination          Protocol Info
 1087 39.737365     130.237.43.127      239.255.255.250      SSDP     M-SEARCH * HTTP/1.1
 1088 39.768590     1c:e6:c7:53:f9:40   Broadcast            ARP      Who has 192.168.1.58?  Tell 192.168.
 1089 39.783104     1c:e6:c7:53:f9:40   Broadcast            ARP      Who has 192.168.50.202?  Tell 192.16
 1090 39.863490     ec:9a:74:07:96:64   Spanning-tree-(for-br STP     MST. Root = 32768/0/00:0f:20:0c:6b:
 1091 39.968503     1c:e6:c7:53:f9:40   Broadcast            ARP      Who has 192.168.1.59?  Tell 192.168.
 1092 39.999193     130.237.50.144      130.237.50.255       NBNS     Name query NB WPAD<00>
 1093 40.080604     130.237.43.176      255.255.255.255      STUN2    ChannelData TURN Message
 1094 40.109166     1c:e6:c7:53:f9:40   Broadcast            ARP      Who has 192.168.1.102?  Tell 192.168
 1095 40.109181     1c:e6:c7:53:f9:40   Broadcast            ARP      Who has 192.168.1.103?  Tell 192.168
 1096 40.109188     1c:e6:c7:53:f9:40   Broadcast            ARP      Who has 192.168.1.104?  Tell 192.168
 1097 40.109196     1c:e6:c7:53:f9:40   Broadcast            ARP      Who has 192.168.1.105?  Tell 192.168
 1098 40.168500     1c:e6:c7:53:f9:40   Broadcast            ARP      Who has 192.168.1.60?  Tell 192.168.
 1099 40.236570     1c:e6:c7:53:f9:40   Broadcast            ARP      Who has 192.168.1.2?  Tell 192.168.1
```

```
+ Frame 622 (256 bytes on wire, 256 bytes captured)
+ Ethernet II, Src: c8:60:00:a4:a2:fe (c8:60:00:a4:a2:fe), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
+ Internet Protocol, Src: 130.237.50.55 (130.237.50.55), Dst: 130.237.50.255 (130.237.50.255)
+ User Datagram Protocol, Src Port: 17500 (17500), Dst Port: 17500 (17500)
+ Data (214 bytes)
```

```
0000  ff ff ff ff ff ff c8 60  00 a4 a2 fe 08 00 45 00   .......` ......E.
0010  00 f2 2b b4 00 00 80 11  a3 36 82 ed 32 37 82 ed   ..+..... .6..27..
0020  32 ff 44 5c 44 5c 00 de  65 03 7b 22 68 6f 73 74   2.D\D\.. e.{"host
0030  5f 69 6e 74 22 3a 20 31  31 30 37 31 36 35 39 30   _int": 1 10716590
0040  30 2c 20 22 76 65 72 73  69 6f 6e 22 3a 20 5b 31   0, "vers ion": [1
```

```
eth1: <live capture in progress> Fi...  Packets: 1099 Displayed: 1099 Marked: 0      Profile: Default
```

Figure 1: Main `Wireshark` window containing three frames.

2.5.2 Plotting graphs in Wireshark

Figure 2 illustrates a TCP trace graph. After a TCP experiment, this graph is shown by choosing *Statistics»TCP Stream Graph»tcptrace* on the menu bar. The figure shows the time in seconds on the x-axis, and the sequence number on the y-axis. The plots illustrate the segments, and the advertised window.

Unfortunately, `Wireshark` does not allow you to save the graphs to a file. However, by using the import utility, the graph can be dumped to a bitmap. In order to dump the graph to a bitmap do the following. Type

```
# import graph.jpg
```

in a terminal window, then click on the window containing the TCP graph. This saves the graph to a file with the corresponding image format.

`Wireshark` can also be used to create other useful graphs, including the throughput plotted as a function of the time.

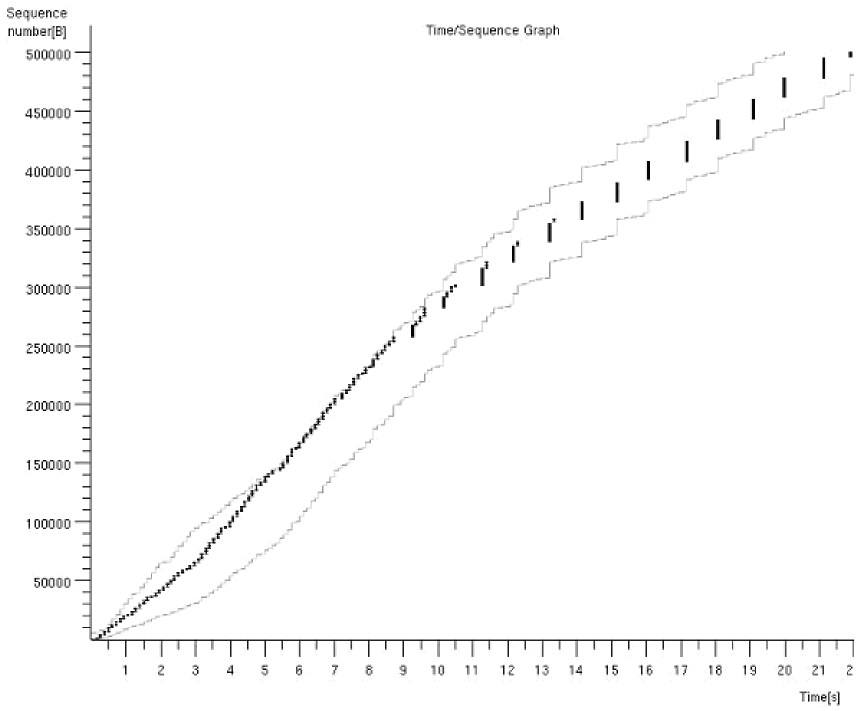


Figure 2: Example of a tcptrace graph.

Part I
Network Configuration and Basic Diagnostic Tools

During this lab session you will work with what could be a corporate network of a company with several hundreds of users. The name of the fictitious company is Acme. It has four departments: administration, production, marketing as well as research and development. The network has four backbones, one per department. The backbones are connected to a single router as shown in Figure 3. Before you start the exercises, you must *manually* configure your host's network connectivity, i.e. statically configure the IP address and possibly also DNS information. You must select an IP address from one of the four subnets, connect your laptop to the corresponding subnet and configure your IP address, netmask and broadcast address.

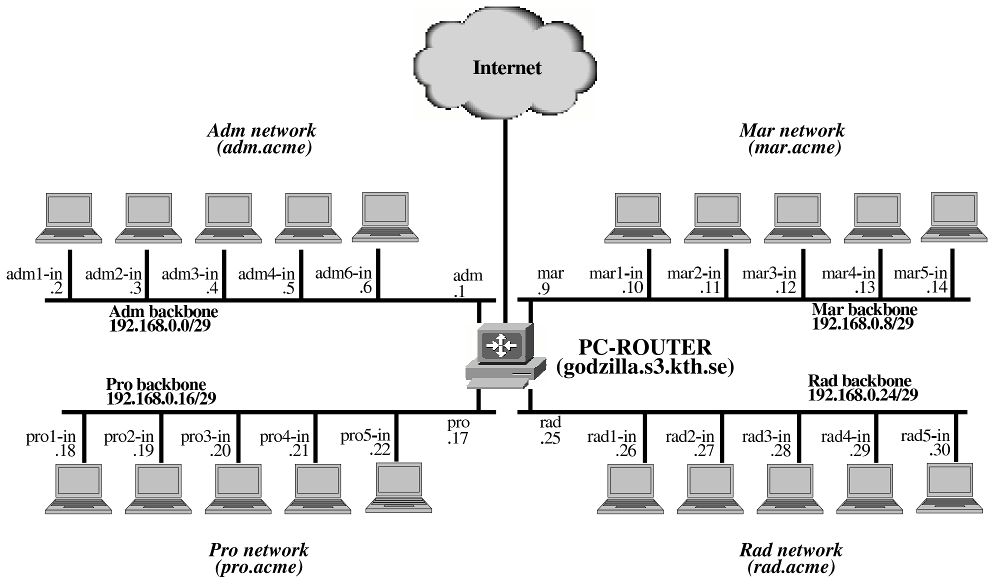


Figure 3: The network diagram.

3 Configuring your host

After you have read the network diagram follow the steps below to configure you host.

- At the beginning of the lab session, each group (of two) will be given a label that marks their host in the network diagram above. Based on your label, configure your host with the parameters given in the diagram. For example, if your label is MAR1, your computer belongs to the Marketing subnet (`192.168.0.8/29`) and the computer's IP address is `192.168.0.10`.

- In the patch panel make sure that your machine is connected to the switch for the subnet that contains your IP address. The socket where you plug in the network cable on the table in front of you has a number. The corresponding number on the patch panel in the rack has to be connected to the correct switch.

- If you are performing the lab on your own Linux machine (i.e. not using VirtualBox or booting from one of the USB sticks provided during the lab) you will have to ensure that the systems network manager is disabled. In Ubuntu, this can be done with the command

  ```
  service network-manager stop
  ```

  At the end of the lab you may restart the network manager with the command

  ```
  service network-manager start
  ```

- The `ifconfig` command is used to configure network interfaces on your computer.

  You can find information about the arguments and options of the `ifconfig` command by typing `man ifconfig` in the terminal window:

  - Which interfaces are defined on your computer?
  - What IP addresses are assigned to them?
  - On the interfaces with IP addresses: What is the interfaces' broadcast address?
  - What other information can you see on the interfaces?

  Now use the `ifconfig` command to bring up the interface *`eth1`* and configure it with your IP parameters (address, netmask and broadcast address for the subnet).
  To bring the network interface up manually, the syntax of the command is:

  ```
  ifconfig Interface_name up
  ```

  The syntax of the command when you would like to assign IP parameters is

  ```
  ifconfig <iface> <ipaddr> netmask <netmask> broadcast <bcast>
  ```

  After configuring the interface try to ping the PC-router's interface that belongs to your subnet. Then, try to ping one of the other three interfaces of the PC-router or try to ping one of the KTH DNS servers with address 130.237.72.200.

  - You will see that you are unable to reach addresses outside your subnet.
    If you cannot find the explanation for this, the next step will lead you to it.

- Add a default route to the gateway with the `route` command.

  After you have checked that your interface is configured properly, but you still do not have any response from ping or traceroute, it is a good time to check that the routing information in your PC is correct. You can check the content of the routing table by typing `route` in the terminal window. You will see the output of this command like the one below.

```
root@live:~# route
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.213.0      *               255.255.255.0   U     0      0        0 eth1
127.0.0.0       *               255.0.0.0       U     0      0        0 lo
default         itguest-gw.gues 0.0.0.0         UG    0      0        0 eth1
```

Or, if you execute this command with the option “-n”:

```
root@live:~# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.0.213.0      *               255.255.255.0   U     0      0        0 eth1
127.0.0.0       *               255.0.0.0       U     0      0        0 lo
0.0.0.0         10.0.213.1      0.0.0.0         UG    0      0        0 eth1
```

If your PC has one network card the routing table will consist of three records: the route to your network, the route to the 127.0.0.0 network, and the default route. When sending packets to an IP address that is inside of your own network, your PC will use the first record; for the packets whose destination is outside your network the PC will use the third record, and send them to the default gateway. Check the entry corresponding to the **default** route (the network address for default route is 0.0.0.0), it should point to the first router in your network. If you do not have this record or it does not point to the first router configure the routing table with the command

```
route add default gw router_addr
```

Before you try this command, make sure that you can ping the router.

- If name resolution does not work you might have to edit the file `/etc/resolv.conf` to configure the correct nameserver (DNS). Set the nameservers to `130.237.72.200` and `130.237.72.201`.

  Note: If you cannot edit the /etc/resolv.conf file manually using an editor, try typing instead:

  ```
  echo "nameserver 130.237.72.200" | sudo tee - a /etc/resolv.conf
  ```

Your host is now configured and is able to reach any other host on the Internet.

4 Netstat

Netstat is a command that shows you statistics of the TCP/IP stack, such as which ports are open in your system as well as the processes associated with the ports used.

- What TCP connections are established on your system?
- Is your system listening on any TCP ports? Which ports and what are the related processes?
- Open a second terminal window and start an FTP connection to ftp.sunet.se by typing

  ```
  ftp ftp.sunet.se
  ```

Which TCP connections are established now? Close the connection and exit ftp with the command `bye`.

- What flags should you use so that `netstat` avoids resolving IP addresses to host names?
- Take a look at the routing table. Are routing tables really necessary on a host?

5 Ping

Ping is used for network maintenance and allows to check whether the network is well configured and there is connectivity among the hosts.

- Ping your neighbour's computer. Study the output of "ping" and be ready to describe what everything means.
- Ping `www.google.com`. How is the output different from the previous ping?
- Ping `130.237.32.51`. How is the output different from the previous pings?
- Ping `ftp.sunet.se` with TTL set to 1. What happens? Increment the TTL and retry the command until you stop getting errors. Describe what happened.
- Ping some hosts and study the output in `Wireshark`. Investigate all the fields of a ping packet. What protocol headers can you identify?

6 Traceroute

The command `traceroute` is used to discover the IP level path of packets travelling between two hosts.

- How many router hops are there between you and the destinations below? Also: What do the hostnames (in the `traceroute` output) mean?
  - `rtfm.mit.edu`
  - `acs.ucalgary.ca`
  - `www.yahoo.se`
  - `210.155.130.190`
- What does `traceroute` do? What kind of conclusions can you draw from the output of `traceroute`? What do the times mean?
- How can you get `traceroute` to send max two packets, jump a maximum of ten hops and aim at `www.nada.kth.se`, port 80?

7 Arp

`Arp` is a network protocol used for the discovery of a host's physical address (MAC), when its IP address is known.

Take a look at your `arp` table.

- How many tuples does your `arp` table contain?
- How is the `arp` table built up?
- Why is an `arp` table necessary on your computer?
- Investigate the meaning of the flags in the `arp` table.

Part II
User Datagram Protocol (UDP)

In this part of the lab you will send data via UDP between two hosts and you will observe the traffic. You will use `ttcp` to send the data. Please consult the man page of `ttcp` (`man ttcp`) for the command line parameters of `ttcp` if you have not done so yet.

8 Basic operation of UDP

To perform the exercise follow the instructions below.

1. On your host, start `Wireshark` to capture traffic on interface *eth1*, add a filter so it will only capture the UDP traffic from and to host "TA-host".
2. Click Capture»Start on the menu bar to start capturing the traffic.
3. On your host: start a `ttcp` receiver to receive UDP traffic at port number 1234.
4. Login to host "TA-Host" using ssh. Start a `t`tcp sender to send 10 UDP packets with the length of 1000 bytes to your host at port number 1234.
5. When the transfer is completed, stop capturing the packets by pressing `S`top in the "Wireshark Capture" window. You should now have a packet trace in the main window. Make sure that you recognize the Ethernet header fields, the IP header fields, and the UDP header fields in the mid section.
6. Observe the traffic and answer the following questions:
   (a) How many packets were transferred in the two directions?
   Observe that `t`tcp always transfers six extra UDP packets when testing UDP transfer. One of the six packets is at the beginning of the transmission and the other five packets are at the end of the transmission. You should always exclude these packets when doing measurements.
   (b) How many bytes were transmitted in the recorded transfer in total including the Ethernet, IP, and UDP headers as well as the application data? (Do not include the six extra UDP packets in the calculation, which constitute 360 bytes.)
   Hint: Check the statistics summary by choosing Tools»Summary.
   (c) How many bytes of user data were transmitted?
   (Calculate it or find it out by looking at the statistics from `t`tcp.)
   (d) Inspect the UDP header fields. Which fields do not change in the different packets? (Note that you should disregard the 6 extra packets).
   (e) What is the UDP port number used in host "TA-Host"?

9 UDP and fragmentation

In this exercise you will observe the effects of fragmentation in UDP. Fragmentation occurs in the network layer, when the transport layer sends so much data to the IP layer that the data together with the network layer header exceeds the Maximum Transmission Unit (MTU) of the underlying link.
In the following you will investigate how IP fragmentation works for UDP traffic. You will use `ttcp` to send data.

1. Start capturing packets with `Wireshark` on your host. For this exercise it is convenient to set Display options “Update list of packets in real time” and “Automatic scrolling in live capture”.

2. Use ssh to login to Host “TA-Host” and use `ttcp` to send packets to your host. Increase the size of the UDP datagrams (the `-l` option) until fragmentation occurs. Note that when `ttcp` runs UDP, it is not necessary to start a receiving `ttcp`. You can work solely on the sending `ttcp` in this exercise and increase the packet size gradually. A hint is to start with sending datagrams of size 1470 bytes and to work upwards.

   (a) What is the largest UDP data length you can send without fragmentation?

   (b) Can you explain this length (given an MTU of 1500)?

   (c) What is the largest UDP data length the system can send, regardless of fragmentation?
   Hint: the largest possible IP packet is $2^{16}-1$ bytes long. `ttcp` will return “Message too long” for over-sized packets.

   (d) What is the explanation for this length?

Please ask one of the lab assistants to check your progress before you continue the lab.

Part III
Transmission Control Protocol (TCP)

For part III of the lab new groups have to be formed. Each group should consist of 4 persons. Each new group will be formed by the merging of two groups from the previous part. You will not need any extra equipment to perform the exercises.

You are going to measure TCP traffic between the two computers of the group and in order to differentiate them you can assign them the names *Host A* and *Host B*. To make it easy to remember which host you are working on, set the prompt to reflect the name of the host.

```
root@live:~# cat » /root/.bashrc
export PS1="Host A# "
ctrl-D
root@live:~# source /root/.bashrc
Host A#
```

It does not matter which of the two is Host A and Host B, it is just a convention in order to distinguish between the two computers.

10 Basic operation of TCP

This exercise is an introduction to measuring TCP traffic. You will use `ttcp` to send TCP traffic from host A to host B, and you will use `Wireshark` to capture the traffic on Host B. This exercise is very similar to the exercise that you performed in the first part of this lab for UDP. Note however that `ttcp` does not send any extra packets when running TCP. (Can you explain why there are no extra packets when using TCP to send data with `ttcp`?)

1. On Host B, start `Wireshark` to capture traffic on interface *eth1*, add a filter so it will only measure the traffic from and to host A.

2. On Host B, start capturing packets with `Wireshark`. (Choose *Capture»Start* on the menu bar to start capturing the traffic (see Section 2.5).)

3. On Host B, start a `ttcp` receiver to receive TCP traffic at port number 1234.

4. On Host A, start a `ttcp` sender to send 10 packets with the length of 1000 bytes using TCP to Host B, at port number 1234.

5. When the transfer is completed, stop capturing. You should now have a packet trace in the main window. Make sure that you recognize the Ethernet header fields, the IP header fields, and the TCP header fields in the mid section.

6. Save the output (detailed and summary) to file output_10_detail and

   output_10_summary. (Section 2.5 shows in detail how to do it.)

7. Observe the traffic and answer the following questions:

   (a) How many packets are transmitted in total (count both directions)?

(b) What is the range of the sequence numbers used by the sender (Host A)?

(c) How many packets do not carry a data payload?

(d) What is the total number of bytes transmitted in the recorded transfer? (Calculate the amount of user data that was transmitted!)

(e) Compare the total amount of data transmitted in the TCP data transfer to that of a UDP data transfer. Which of the protocols is more efficient in terms of overhead? What is the efficiency in percentage for these two protocols?
(Recall the UDP measurements from the previous lab. How many bytes were sent in total using UDP?)

Lab report: Use the captured data to answer the questions above. Support your answers with the saved `Wireshark` data. Append the file output_10_summary to the report.

11 TCP connection management

This exercise gives an insight in TCP connection establishment and termination. In this exercise we use the `telnet` application to establish and terminate TCP connections.

11.1 Connection establishment and termination

1. Start capturing packets on Host B with `Wireshark`.

2. Establish a `telnet` connection from Host A to Host B.

3. On Host A, terminate the connection. Type `ctrl-] (ctrl+AltGr+9)` at the `telnet` prompt and then type "`quit`".

4. Stop the `Wireshark` capturing.

5. Save a summary `Wireshark` output to the file output_11_1_summary.

6. Study the list of captured packets. Observe the TCP connection establishment and answer the following questions:

   (a) Which packets constitute the three-way handshake? Which flags are set in the headers of these packets?

   (b) What are the initial sequence numbers used by the client and the server, respectively?

   (c) Which packet contains the first application data?

   (d) What are the initial window sizes for the client and for the server?

   (e) How long does it roughly take to open the TCP connection?

7. Study the list of captured packets. Observe the TCP connection termination and answer the following questions.

(a) Which packets are involved in closing the connection?

(b) Which flags are set in these packets?

Lab report: Use the captured data to answer the questions above. Support your answers with the saved `Wireshark` data. Append the file output_11_1_summary.

11.2 Connecting to a non-existing port

In the following exercise you will see what happens when you try to establish a TCP connection to a non-existing port. This could be the case when you try to load a home page from a host that does not have a web server.

1. Start capturing packets with `Wireshark` on Host B.
2. Try to make a `telnet` connection from Host A to a port on Host B without listeners.
3. Stop capturing packets.
4. Save a summary `Wireshark` output to file output_11_2_summary.
5. Study the captured list of packets and observe the TCP segments that are transmitted. Answer the following questions.
   (a) How does the server host (Host B) close the connection?
   (b) How long does the process of ending the connection take?

Lab report: Use the captured data to answer the questions above. Support your answers with the saved `Wireshark` data. Append the file output_11_2_summary.

Note: What happens when you try to establish a TCP connection to a non-existing host? This could be the case when you try to load a home page from a host given with its IP address that does not exist. Think about this case and provide an answer.

12 TCP data transfer

In this exercise you will study TCP flow control and some properties of TCP data transfer. TCP consists of several heuristics to cope with different network and traffic conditions. In particular, TCP has different behaviour for interactive applications and for bulk transfer. The interactive application in this exercise is `telnet`, the bulk data transfer is done with `ttcp`.

12.1 Interactive application

In order to make the interactive application scenario more realistic you can add some delay on the link between the two hosts. You do so by using the `tc` command.

1. Add a delay of 150 ms at interface *eth1*of Host A. You can do that by typing

```
    tc qdisc add dev eth1 root netem delay 150ms
```

2. Establish a `telnet` connection from Host A to Host B. Log in as user *user* with the credentials mentioned above.

3. Start capturing packets with `Wireshark` on Host A.

4. Type a few characters in the `telnet` application. The `telnet` client (Host A) sends each character in a separate TCP segment to the server (Host B) which in turn echoes the character back to the client. Including echoes, we would therefore expect to see four TCP segments for each typed character.

   (a) How many segments can be seen?

   (b) Describe the payload of each packet.

   (c) Explain why you do not see four packets per typed character.

   (d) When the client receives the echo, it waits a certain time before sending the ACK. Why? How long is the delay?

   (e) In the segments that carry characters, what window size is advertised by the `telnet` client and by the server? Does the window size vary as the connection proceeds?

5. Type quickly a lot of characters in the `telnet` application, such as by pressing a key continuously.

   (a) Do you observe a difference in the transmission of segment payloads and ACKs?

6. Stop the capturing of packets.

7. Save the `Wireshark` output to file output_12_1_summary.

Lab report: Use the captured data to answer the questions above. Support your answers with the saved `Wireshark` data. Append the file output_12_1_summary to the report.

12.2 Bulk transfer

In this exercise, the behavior of TCP is examined when large amounts of data are transmitted. TCP uses the acknowledgements to limit the sending rate; this together with the receiver window size constitute the basis of flow control.

1. Start capturing packets with `Wireshark` on Host A.

2. Start a `ttcp` receiver on B and a sender on A, send 1000 packets of length 1000 bytes each.

3. Stop capturing packets.

4. Draw a tcptrace graph. Study the graph carefully.

5. Observe the sliding window protocol from the output of `Wireshark`. The sender transmits data up to the window size of the receiver. Answer the following questions.

(a) How often does the receiver send ACKs? Can you see a rule on how TCP sends ACKs?

(b) How many bytes of data does a receiver acknowledge in a typical ACK?

(c) How does the window size vary during the session?

(d) Select any ACK packet in the `Wireshark` trace and note its acknowledgement number. Find the original segment in the `Wireshark` output. How long did it take from the transmission until it was ACKed?

(e) Does the TCP sender generally transmit the maximum number of bytes as allowed by the receiver?

6. Save the tcptrace graph to file output_12_2_tcptrace.jpg and save the `Wireshark` packet trace to file output_12_2_summary.

Lab report: Use the captured data to answer the questions above. Support your answers with the saved `Wireshark` data. Append the output_12_2_tcptrace.jpg to the report.

13 TCP retransmissions

In this exercise you will study TCP retransmissions. TCP uses ACKs and timers to trigger retransmissions of lost segments. In order to cause retransmissions, you will have to artificially introduce errors on some of the links. To do so, you are going to use `tc`, a kernel module for manipulating traffic control settings. In this exercise you are going to introduce losses on the interface *eth1*of host A. The command to introduce losses is:

```
tc qdisc add dev eth1 root netem loss 1%
```

The above command tells the kernel to introduce a 1% loss at the interface $eth1$, namely to drop on average 1 every 100 packets going out from that interface. To restore zero losses you can simply type

```
tc qdisc change dev eth1 root netem loss 0%
```

or you can cancel outright the queueing discipline between the network layer and the network adapter if you are not going to use it later on. Cancelling the queueing discipline is done by invoking the command

```
tc qdisc del dev eth1 root
```

In order for you to be able to introduce losses in the middle of the data transmission, you should add some delay in the link between the two hosts, otherwise the transmission will be too fast. You can add a delay of 500 ms at interface *eth1*of Host A by typing in the terminal

```
tc qdisc add dev eth1 root netem delay 500ms
```

After having set the delay at the link, follow the steps below.

1. Start a second terminal window (Terminal 2) as `root` with `sudo -s`. You are going to use this terminal to switch on and then off losses at step 4

2. Start capturing packets with `Wireshark` on Host A.

3. Start a `ttcp` receiver on host B and a sender on host A, send 300 packets of length 1000 bytes each.

4. When a few dozen packets have been captured by `Wireshark`, in Terminal 2 introduce 100% losses at the interface *eth1*of Host B. Wait for one minute (watch `Wireshark` and wait for a few packets to be transmitted) then reinstate zero losses at $eth1$.

5. When the transfer is completed, stop capturing packets.

6. Study the list of captured packets. Draw a tcptrace graph. Observe when retransmissions take place (during the period where the ”disconnection” occured) and answer the following questions.

   (a) How many packets are transmitted at retransmission timeout?

   (b) Do the retransmissions end at some point?

7. Save a summary `Wireshark` output to file output_13_1_summary.

8. Save the tcptrace graph to file output_13_1_tcptrace.jpg.

Lab report: Use the captured data to answer the questions above. Support your answers with the saved `Wireshark` data. Append the `Wireshark` data (output_13_1_summary) and the tcptrace graph (output_13_1_tcptrace.jpg).

14 TCP congestion control

In this exercise, the behaviour of TCP congestion control is examined. TCP congestion control operates in two phases, called slow start and congestion avoidance. Congestion over the link between A and B is emulated by introducing losses at interface *eth1*of Host A.

1. On the terminal of both hosts run the following command to check and make sure the congestion control algorithm used is Reno.

```
sudo sysctl net.ipv4.tcp_congestion_control
```

2. On the terminal of Host A and introduce a 1.5% loss rate and a delay of 100 ms for the outgoing traffic at interface eth1 using the `tc` command as in the previous exercises.

3. Start capturing packets with `Wireshark` on Host A.

4. Start a `ttcp` receiver on Host B and a sender on Host A, send 200 packets with a length of 1000 bytes.

5. After the transmission is completed go carefully through the list of captured packets by Wireshark. What do you observe?

6. Draw a tcptrace graph in `Wireshark`.

   (a) Try to observe periods when TCP sender is in slow start phase and when the sender switches to congestion control. Verify if the congestion window follows the rule of the slow-start phase.
   (b) Can you find occurrences of fast recovery?

7. Save a summary `Wireshark` output to file output_14_1_summary.
8. Save the tcptrace graph to file output_14_1_tcptrace.jpg.
9. Configure the routes so the traffic from Host B to Host A goes through the fast link again. Go through steps 3 - 6 again. Include `Wireshark` data and the tcptrace graph to support your answer. Annotate the events in the tcptrace graph, and explain the events that you observe.

Please ask one of the lab assistants to check your progress before you continue the lab.

Part IV
Application Layer Protocols

15 Telnet

`Telnet` is a network protocol used for bi-directional text-based communication between two hosts over a network.

- Use telnet to connect to the HTTP daemon (httpd) at `www.nada.kth.se`. To download the main web page (index.html), type: `GET ./`
- Why is TCP and not UDP used for HTTP traffic?
- What are the advantages/disadvantages?

16 Anonymous ftp

`ftp` is a network protocol used to transfer a file from one computer to another over a network.

- How do you log onto an ftp server anonymously?
- What are the main differences between normal ftp and anonymous ftp?
- Why are both necessary?
- Start capturing packets with Wireshark and add a filter to display only FTP or TCP traffic.

- Get the file `rfc0959.txt` from `ftp://ftp.ietf.org/rfc`. Download this file using the command line interface, and not netscape etc.
- Study the captured list of packets. What source and destination ports were used to transfer the data? Based on the previous, can you tell which FTP mode was used?

17 Mail

Check if the smtp server on your machine is running. This can be done with the command `service exim4 status` or alternatively `ps ax|grep exim4`. If it is not running you can start it with `sudo /etc/init.d/exim4 start`.

- Use the command `sendmail -v email@address.com` to send an email to your regular email account. The "-v" flag makes the tcp session verbose, so you can see what protocol messages were used. When using `sendmail`, you have to manually specify the headers of the e-mail message as well as the body. An example of execution of `sendmail` is

  ```
  user@live:~$ sendmail user@domain.com
  From:  John Doe <johnd@somedomain.com>
  Reply-to:  johnd@somedomain.com
  Subject:  Hello
  This is a test message!

  .
  ```

  After typing in the headers and the body of the mail message press `ctrl-d` or "." to send.

  - Which lines are sent by your computer to the mail server?
  - Which lines are returned by the remote host?
  - How did you see if the email was successfully sent?

- Find out the domain name of your smtp server using the command: `dig domain.name mx`, where *domain.name* is the domain name part of your favourite email address. If this doesn't give you an domain name, try `dig domain.name a` to get an IP address instead.

18 Completing the lab

In order to complete the lab, you will have to put together all the output of Wireshark (packet data and graphs). If you used your own computer running Linux then you are done; if you used a USB stick or VirtualBox, you need to save all your data in one place and collect them.

1. Collect the output into one common archive. One way to do this is to copy all files over to Host A, and then build an archive.
   ```
   Host A# scp IP_of_Host_A:/home/user/* .
   Host A# tar czf lab_report.tgz *
   ```

2. If your output is not directly saved on a USB stick you can transfer the archive to your personal mailbox or cloud storage by connecting to the Internet.

3. Transfer the archive to a remote Internet site by using ftp, scp, browser, or some other tool.

4. Power down the hosts.
   ```
   Host # halt -p
   ```

5. Disconnect the cables and ensure that the equipment is in the same state (or better) as when you started.