



KTH Electrical Engineering

# IK2218

## Protocols and Principles of the Internet

### Lab 1

## ARP, IP, UDP, TCP

September 2015

Department of Communication Networks  
School of Electrical Engineering  
KTH, Royal Institute of Technology  
Stockholm, Sweden

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Goals of the Lab . . . . .	4
1.2	Requirements and reporting . . . . .	4
<b>2</b>	<b>Preparation for the lab</b>	<b>5</b>
2.1	Software tools . . . . .	5
2.2	Reading list . . . . .	5
2.3	Preparation questions . . . . .	5
2.4	Lab equipment and software instructions . . . . .	6
2.4.1	Check-list for Linux users . . . . .	6
2.5	Operating system instructions . . . . .	7
2.6	Wireshark: A network packet sniffer . . . . .	8
2.6.1	Saving Wireshark output . . . . .	9
2.6.2	Plotting graphs in Wireshark . . . . .	9
<b>I</b>	<b>Network Configuration and Basic Diagnostic Tools</b>	<b>10</b>
<b>3</b>	<b>Configuring your host</b>	<b>10</b>
<b>4</b>	<b>Netstat</b>	<b>12</b>
<b>5</b>	<b>Ping</b>	<b>13</b>
<b>6</b>	<b>Traceroute</b>	<b>13</b>
<b>7</b>	<b>Arp</b>	<b>13</b>
<b>II</b>	<b>User Datagram Protocol (UDP)</b>	<b>14</b>
<b>8</b>	<b>Basic operation of UDP</b>	<b>14</b>
<b>9</b>	<b>UDP and fragmentation</b>	<b>15</b>
<b>III</b>	<b>Transmission Control Protocol (TCP)</b>	<b>16</b>
<b>10</b>	<b>Basic operation of TCP</b>	<b>16</b>
<b>11</b>	<b>TCP connection management</b>	<b>17</b>
11.1	Connection establishment and termination . . . . .	17
11.2	Connecting to a non-existing port . . . . .	18

<b>12 TCP data transfer</b>	<b>18</b>
12.1 Interactive application . . . . .	18
12.2 Bulk transfer . . . . .	19
<b>13 TCP retransmissions</b>	<b>20</b>
<b>14 TCP congestion control</b>	<b>21</b>
 <b>IV Application Layer Protocols</b>	 <b>22</b>
<b>15 Telnet</b>	<b>22</b>
<b>16 Anonymous ftp</b>	<b>22</b>
<b>17 Mail</b>	<b>23</b>
<b>18 Completing the lab</b>	<b>23</b>
<b>19 Preparation questions</b>	<b>25</b>

# 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:

- Do the preparations before the lab. Show your solutions to the lab assistant before starting 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 conduct in the second part of this lab. Each group should submit one report. 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.

## 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 Software tools

The following software tools will be used in the lab.

- `ttcp` - Test TCP and UDP performance.
- `telnet` - An application and a protocol to communicate with another host using TCP.
- `netstat` - Show statistics of the TCP/IP stack.
- `ifconfig` - Used to configure a network Interface.
- `route` - Used to view and to manipulate the routing table.
- `wireshark` - An interactive GUI based network traffic analyzer. Read more about Wireshark in Section 2.6.
- `tc` - Used to show/manipulate traffic control settings

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. It is even better if you install the above software on your own computer and learn to use them before the lab.

### 2.2 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

### 2.3 Preparation questions

Please print the preparation questions that you find in the Appendix of this syllabus and submit the solutions at the beginning of the lab.

## 2.4 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 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 than 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 3 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 2.4.1. 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 pages in Moodle 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/ep2120-ik2218-2015> (to access use the password provided in the [manual](#)).
3. 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.

In lack of expertise, we can unfortunately not provide much 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.

### 2.4.1 Check-list for Linux users

- Test which of the networking tools to be used in the lab are already installed in your system. The full list of commands includes: `ifconfig`, `route`, `tracert`, `arp`, `ttcp`, `telnet`, `tc`, `ftp`, `ping`, `ssh`, `sendmail`. You will probably

find most of the commands readily available; the tools that are not present in the system should be installed (for `ttcp` see the next check-list item). One of the ways to do it is through package managers, e.g. Ubuntu Software Center in Ubuntu or PackageKit in Fedora.

- 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](http://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
```

- 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](https://www.wireshark.org/docs/wsug_html_chunked/ChapterBuildInstall.html). Read carefully sections 2.1, 2.2, and follow the instructions from sections relating to UNIX operating systems (from Section 2.5 onwards).

## 2.5 Operating system instructions

*Note that these instructions are for equipment choices 2–4 only.*

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*.

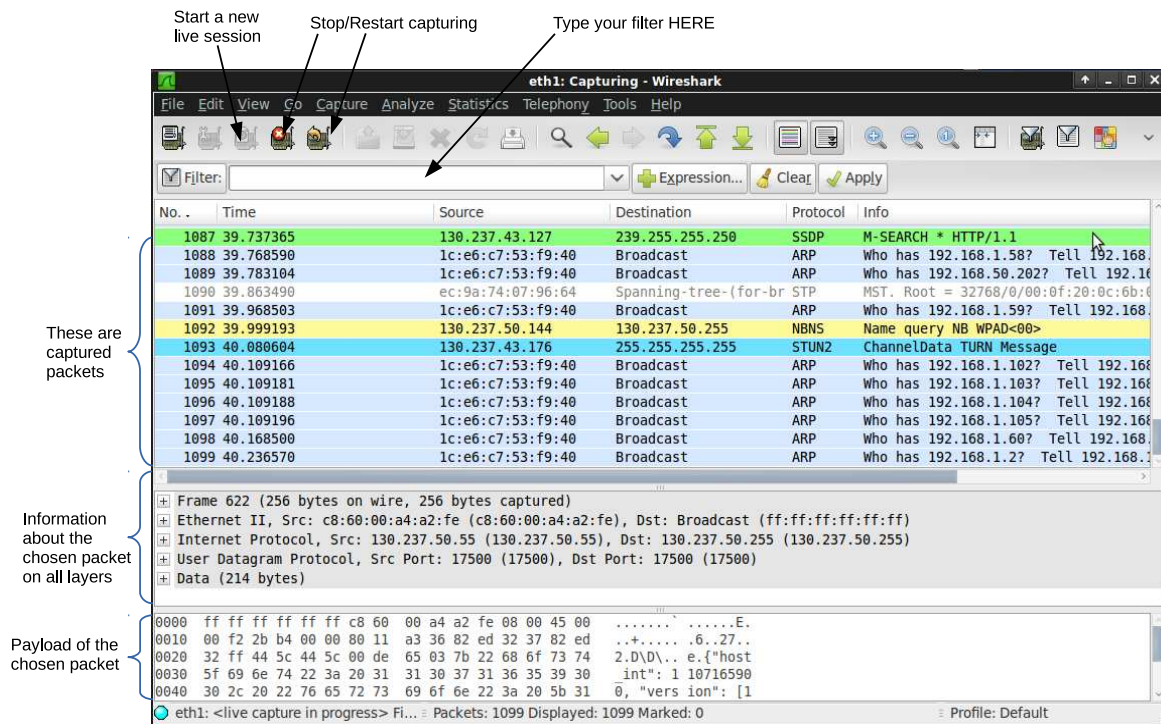


Figure 1: Main Wireshark window containing three frames.

## 2.6 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.



### 2.6.1 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*.

### 2.6.2 Plotting graphs in Wireshark

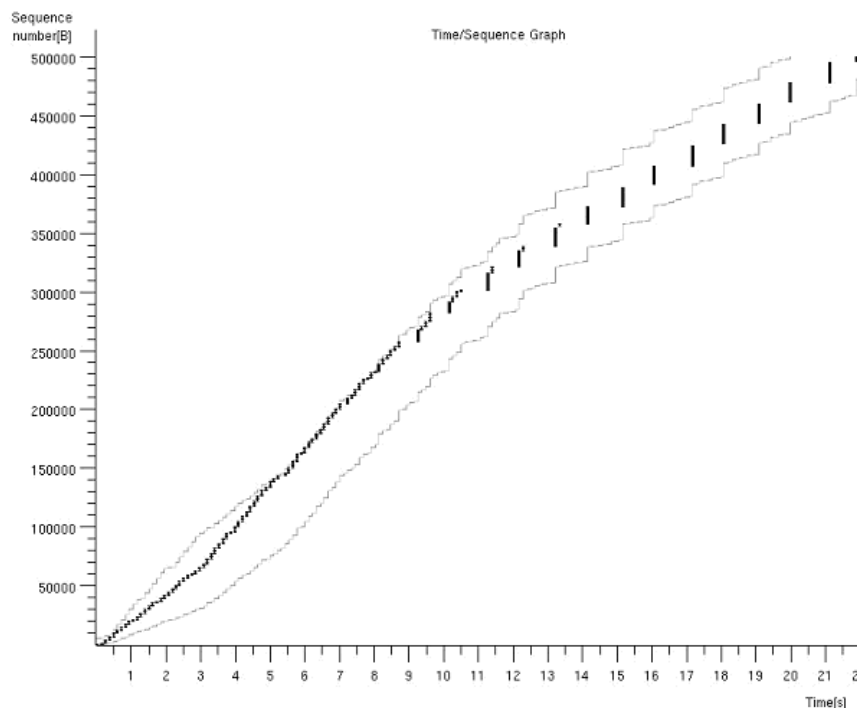


Figure 2: Example of a tcptrace graph.

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.

## 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.

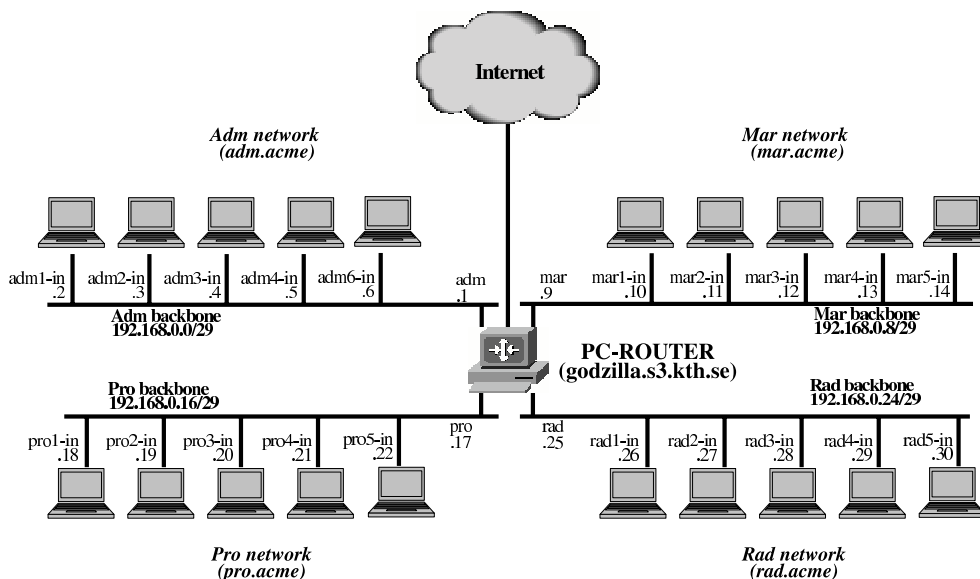


Figure 3: The network diagram.

## 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.
- 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?
- 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 `ttcp` 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 Stop 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 `ttcp` 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 `ttcp`.)
  - (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?

## 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 2.6](#).)
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 2.6](#) 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 `tcp`.

### 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 *B*. 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" occurred) 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 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.
2. Start capturing packets with Wireshark on Host A.
3. Start a `ttcp` receiver on Host B and a sender on Host A, send 200 packets with a length of 1000 bytes.
4. After the transmission is completed go carefully through the list of captured packets by Wireshark. What do you observe?
5. 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?
6. Save a summary Wireshark output to file `output_14_1_summary`.
  7. Save the tcptrace graph to file `output_14_1_tcptrace.jpg`.
  8. Configure the routes so the traffic from Host B to Host A goes through the fast link again. Go through steps 2 - 5 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.

## 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.



**IK2218 - Protocols and Principles of the Internet**  
**Lab 1 - Entry test questions**

Student name: \_\_\_\_\_

*The answers to the questions should be handed in at the beginning of the Lab session. Students who do not hand in correct answers will not be allowed to take the lab at this time but will need to attend the make-up lab session.*

1. How many valid Host IP addresses belong to the subnet 172.16.128.0/18?
  - Give the netmask for the subnet in dotted decimal format.
2. What is ARP and why is it needed?
3. Explain the role of the port numbers in the transport layer.
4. Write the tcp commands for both sender and receiver, which executes the following scenario: Send a TCP stream from host 10.2.3.4 to 10.4.5.6 on port 3333. The sender should send 4000 bytes with four datagrams of 1000 bytes in each datagram.
5. Suppose a TCP sender receives an ACK packet in which the acknowledgement number is set to 12345 and the window size is 2048. Which sequence numbers can the sender transmit?
6. Briefly describe the following algorithms and when they are used:
  - Nagle's algorithm.
  - Karn's algorithm.
  - Delayed acknowledgement.
  - Piggybacked acknowledgement.
7. How is the retransmission timeout (RTO) value computed in TCP?
8. Explain the following TCP mechanisms:
  - Sliding window flow control.
  - Slow start and congestion avoidance.
  - Fast retransmit and fast recovery.