COMPUTER NETWORKS

LAB LIST 5

1 General remarks

During today's classes the topology of network is irrelevant. To solve first batch of exercises you'll need an access to the Internet. Remember to begin your work by typing in `sudo netmode lab`.

2 DNS: Domain Name System

Exercise 1. Run `wireshark` tool. After that issue:

```
$> dhclient eth0
```

to configure network layer with DHCP protocol. Watch the packets being exchanged between the host and the server – it should take 4 packets to negotiate the lease. In particular, check source IP address in DHCP request. Note that DHCP is an application layer protocol, but we don't have an IP address yet! Check the address that was assigned to `eth0` interface. Display DNS resolution settings in `/etc/resolv.conf` and the gateway address.

With following command:

```
$> ps aux | grep dhclient
```

check that `dhclient` is actually running (it runs as a daemon and renews IP address lease from time to time).

Repeat steps described below to obtain IP address for our institute's web server `www.cs.uni.wroc.pl`. Start with one of the root DNS servers (e.g. `198.41.0.4`. The first command is:

```
$> dig www.cs.uni.wroc.pl @198.41.0.4
```

Consecutive commands should send requests to those DNS servers, that are authoritative for particular zone.

Use special syntax of `dig` arguments to perform work described above and run:

```
$> dig +trace www.cs.uni.wroc.pl @198.41.0.4
```

Which DNS servers are probed in this case? Check the IP address that is associated with following name: `cs.uni.wroc.pl`.

Exercise 2. In this exercises we'll familiarise ourselves with `host` command. We'll be recursively probing DNS servers, but we'll let our default server (i.e. listed in `/etc/resolv.conf` to do it on our behalf.

With:

```
#> host -a -l ii.uni.wroc.pl 156.17.4.1
```

fetch contents of `ii.uni.wroc.pl` zone. Then follow with:

```
#> host -t ns ii.uni.wroc.pl
```

to check what DNS server are assigned to this domain. The list of mail servers handling this domain will be revealed with:

```
#> host -t mx ii.uni.wroc.pl
```

And finally, check what domain is associated with the address in following command:

```
#> host -t ptr 156.17.4.1
```

Which particular subdomain of `in-addr.arpa` was questioned?

Exercise 3. In this exercise, we'll learn how to save requests sent by `dig` tool to a file. Our goal is to use the data in batch scripts. Note, that same technique can be used to intercept a request from arbitrary program like a web browser or instant messenger. Run following command:

```
$> nc.traditional -u -l -p 10053
```

in UDP server mode. It will listen for incoming datagrams on port number 10053. Note that binding it to standard DNS port number (53) would require you to have administrator's permissions. In second terminal run:

```
$> dig -p 10053 onet.pl @127.0.0.1
```

It'll send a request to our fake DNS server asking about `onet.pl`. The request will be dumped in binary (and quite incomprehensible) onto the display. Moreover the data is garbled and using copy-and-paste will fail to save it properly to a file. Hence, interrupt the server and run in again but dump its output to both a file and the terminal at the same time:

```
$> nc.traditional -u -l -p 10053 | tee dns_query
```

Subsequently you should resend the request to the server. As soon as the first datagram is received, both `dig` and `nc.traditional` should be stopped. [1] The contents of the file can be display in binary form with following command:

```
$> hexdump -C query_dns
```

`onet.pl` string should occur only once. Now... the request from the file can be sent to some DNS server (e.g. `8.8.8.8` or `8.8.4.4` – Google public DNS servers). To do it run:

```
$> nc.traditional -q 1 -u 8.8.8.8 53 < query_dns
```

Although the reply will be displayed in the terminal in a garbled form, you can actually check it with `wireshark` which will provide full interpretation of the packet.

[1] Otherwise, `dig` will send three datagrams that will be written to `dns_query` file.

3 WLAN: Wireless LAN

Before you proceed to exercises below, please deconfigure `eth0` interface with:

```
#> dhclient -r eth0
#> ifconfig eth0 0.0.0.0 down
```

First command stops `dhclient` daemon. Just before that, it relinquishes leased IP address (confirm that fact with wireshark!). This action is corteous to DHCP server though not compulsory.

Zadanie 4. Activate `wlan0` interface with:

```
#> ifconfig wlan0 up
```

After that use:

```
#> iwlist wlan0 scan
```

display available access points. There should be around two access points belonging to the institute network `e-stud-WiFi` and the lab access point `lab109`. On the basis of displayed information infer encryption type for each network. What is the type of the networks: `802.11a`, `b`, or `g`? On which channels do they operate?

Firstly, we configure the data link layer using `iwconfig` and associate with the lab access point.[2] Type in:

```
#> iwconfig wlan0 essid lab109
```

Using `iwconfig`, check that the network ESSID has been chosen, but the card is still `not associated` with the access point. It is because this network is WPA-PSK encrypted and we haven't provided the key yet.

WPA encryption and authentication is handled by `wpa_supplicant` daemon in a Linux-based system. In arbitrarily chosen directory create a configuration file `wpa.conf` with following contents:

```
ctrl_interface=DIR=/var/run/wpa_supplicant
network={
  ssid="lab109"
  scan_ssid=1
  key_mgmt=WPA-PSK
  psk="bardzotajne"
}
```

Then, run the daemon in the background:

```
#> wpa_supplicant -c wpa.conf -i wlan0 -B
```

From now on, the second (i.e. data link) layer is configured, which can be verified with `iwconfig` command. The only thing that is left to be done, is network layer configuration, which conveniently can be set with:

```
  #> dhclient wlan0
```

[2] Note, that in Ethernet case we didn't have to configure second layer.

After you configured both link and network layer, check the contents of `/etc/resolv.conf` file and routing tables (with `route` command). Check whether the connection works using `ping` and `traceroute` with some remote addresses.

Check in pairs the bandwidth of such wireless connection. To achieve that first person should run:

```
$> iperf -s
```

... and second one:

```
$> iperf -c ip_of_first_computer
```

How does the bandwidth change as more and more people are running these commands?

End the `wpa_supplicant` daemon with:

```
#> wpa_cli terminate
```

... and deconfigure `wlan0` interface:

```
#> dhclient -r wlan0
#> iwconfig wlan0 essid off
#> ifconfig wlan0 0.0.0.0 down
```

Zadanie 5. Repeat previous exercise for `e-stud-WiFi` network. You'll need a password which is `0123456789abc`. With `wireshark` program you can see incomprehensible array of connections. Haven't you learnt to filter packets yet? As in previous exercise, please check bandwidth of the network with `iperf`. Remember about turning off `wpa_supplicant` daemon i deconfiguring `wlan0` interface.

Zadanie 6 . In this task, we create an *ad-hoc* network (i.e. without an access point). Activate the `wlan0` interface.

```
#> ifconfig wlan0 up
```

In groups, choose a channel and a network identifier. It's best to use a channel not being currently used. You can display it with:

```
#> iwlist wlan0 scan
```

Disable the interface with:

```
#> ifconfig wlan0 down
```

... and switch it to the appropriate mode with:

```
#> iwconfig wlan0 mode ad-hoc
```

Set the network name with:

```
#> iwconfig wlan0 essid  name
```

... and the channel with:

```
#> iwconfig wlan0 freq  number
```

Activate the interface with:

```
#> ifconfig wlan0 up
```

Subsequently — inside each group — assign IP addresses (from the same network) to each wireless card, and check if computers can communicate with each other. Possible problems can be diagnosed with `iwconfig wlan0`: each of the stations should have the same ESSID name, same frequency and *cell* number. Check the connection with `ping` command and the bandwidth with `iperf`.

Marcin Bieńkowski
Translation: *Krystian Bacławski*