# Answers IaS-Exercise 2

**Tobias Hafner, Ruben Hutter**

## Exercise 1

*a)*

IP: 130.59.31.80

ADDRESS: CH-8021 Zürich

OWNER: SWITCH-LAN

*b)*

| Address | Ping | Browser | Comment |
|---|---|---|---|
| informatik.unibas.ch | yes | yes | |
| www.zurich.ibm.com | no | yes | The IBM-Server isn't pingable for security reasons as a ping can be used to test if a server exists and responds. |
| www.tik.ee.ethz.ch | yes | yes | |
| www.amazon.com | yes | yes | |

## Exercise 2

*i)*

| | min [ms] | average [ms] | max [ms] |
|---|---|---|---|
| 56 Byte | 8.204 | 8.246 | 8.176 |
| 112 Byte | 9.646 | 9.120 | 10.992 |
| 224 byte | 17.084 | 10.290 | 29.849 |

**Command and output for 56 byte ping:**

```
ping web.mit.edu -c 10
ING e9566.dscb.akamaiedge.net (23.37.44.254) 56(84) bytes of data.
64 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=1 ttl=59 time=9.10 ms
64 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=2 ttl=59 time=8.79 ms
64 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
```

```
icmp_seq=3 ttl=59 time=8.63 ms
64 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=4 ttl=59 time=8.20 ms
64 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=5 ttl=59 time=8.26 ms
64 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=6 ttl=59 time=9.06 ms
64 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=7 ttl=59 time=9.09 ms
64 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=8 ttl=59 time=8.52 ms
64 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=9 ttl=59 time=9.74 ms
64 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=10 ttl=59 time=17.1 ms

--- e9566.dscb.akamaiedge.net ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9016ms
rtt min/avg/max/mdev = 8.204/9.646/17.084/2.516 ms
```

**Command and output for 112 byte ping:**

```
ping web.mit.edu -c 10 -s 112
PING e9566.dscb.akamaiedge.net (23.37.44.254) 112(140) bytes of data.
120 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=1 ttl=59 time=10.1 ms
120 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=2 ttl=59 time=9.42 ms
120 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=3 ttl=59 time=9.80 ms
120 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=4 ttl=59 time=8.62 ms
120 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=5 ttl=59 time=10.3 ms
120 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=6 ttl=59 time=8.25 ms
120 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=7 ttl=59 time=8.50 ms
120 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=8 ttl=59 time=9.03 ms
120 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=9 ttl=59 time=8.70 ms
120 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=10 ttl=59 time=8.54 ms

--- e9566.dscb.akamaiedge.net ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9015ms
```

```
rtt min/avg/max/mdev = 8.246/9.120/10.290/0.685 ms
```

**Command and output for 224 byte ping:**

```
ping web.mit.edu -c 10 -s 224
PING e9566.dscb.akamaiedge.net (23.37.44.254) 224(252) bytes of data.
232 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=1 ttl=59 time=29.8 ms
232 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=2 ttl=59 time=8.69 ms
232 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=3 ttl=59 time=8.44 ms
232 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=4 ttl=59 time=8.53 ms
232 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=5 ttl=59 time=8.99 ms
232 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=6 ttl=59 time=8.67 ms
232 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=7 ttl=59 time=9.26 ms
232 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=8 ttl=59 time=9.02 ms
232 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=9 ttl=59 time=10.3 ms
232 bytes from a23-37-44-254.deploy.static.akamaitechnologies.com (23.37.44.254):
icmp_seq=10 ttl=59 time=8.18 ms

--- e9566.dscb.akamaiedge.net ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9013ms
rtt min/avg/max/mdev = 8.176/10.992/29.845/6.309 ms
```

**Conclusion:**

The packagesize doesn't seem to have a significant effect on the Round-Trip-Time.

*ii)*

As "Sheldon's Office" is at CALTEC we use traceroute to find that there are 11 hops from unibas to www.caltec.edu.

**Command and output:**

```
ruben@debian:~$ traceroute www.caltech.edu
traceroute to www.caltech.edu (104.18.14.60), 30 hops max, 60 byte packets
 1  _gateway (192.168.122.1)  0.405 ms  0.337 ms  0.311 ms
 2  10.172.255.254 (10.172.255.254)  5.635 ms  5.606 ms  5.584 ms
 3  10.36.253.6 (10.36.253.6)  5.560 ms  5.540 ms  5.505 ms
 4  * * *
 5  192.43.192.196 (192.43.192.196)  7.118 ms  7.096 ms  7.021 ms
 6  swiBS1-100GE-0-0-0-0.switch.ch (130.59.37.34)  8.176 ms  4.013 ms  3.927 ms
```

```
 7  swiPS1-100GE-0-0-1-3.switch.ch (130.59.37.190)  5.530 ms  4.536 ms  4.418 ms
 8  swiPS2-100GE-0-0-1-4.switch.ch (130.59.37.58)  4.374 ms  4.727 ms  4.682 ms
 9  swiZH3-100GE-0-0-0-2.switch.ch (130.59.36.170)  5.302 ms  5.237 ms  5.194 ms
10  as13335.swissix.ch (91.206.52.192)  6.055 ms  6.529 ms  6.378 ms
11  104.18.14.60 (104.18.14.60)  5.724 ms  6.185 ms  5.262 ms
```

*iii)*

- Los Angeles - CA - US
- Los Angeles - CA - US
- Los Angeles - CA - US
- Amsterdam - NL
- Basel, Switzerland - CH
- Zurich, Switzerland - CH
- Zurich, Switzerland - CH
- Zurich, Switzerland - CH
- Zurich, Switzerland - CH
- San Francisco - CA - US

The signal travels through the United States, the Netherlands and Switzerland.

*iv)*

The three routes have the following hops in common:

- 10.172.255.254
- 10.36.253.6
- 130.59.36.114
- 130.59.36.69
- 130.59.39.74
- 192.168.122.1
- 192.43.192.196
- 62.40.124.21

## Exercise 3

*a)*

**Circuit Switching:**

As an exclusive connection between each party is established the full individual bandwith of 2 MB/s is allocated for each user. Therefore the maximum number of users is (10 MB/s)/(2 MB/s) = 5.

**Packet Switching:**

In the worst case, all users are active simultaneously. To ensure that our shared connection is never overloaded we can't support more than 5 users as 5*(2 MB/s) = (10 MB/s). We can however allow a small chance of our connection to get overloaded, leading to delay and package drop. The chance that more than 5

users are active simultaneously can be calculated by the following formula with N being the amount of users in the network:

$$P(n \geq 5) = \sum_{i=5}^{N} \binom{N}{i} * 0.35^i * (1 - 0.35)^{N-i}$$

The number of supported users lies in our choice of the chance with wich we allow our shared connection to be overloaded.

*b)*

*i)*

The maximum achievable bandwith on the specified route is limited by the proxy with the lowest traffic limit. As proxy D has a traffic limit of 512 KB/s this is equal to the maximum achievable bandwith. As 2 GB = 2'000 MB = 2'000'000 KB, we need (2'000'000 KB / 512 KB) = 3906.25 s = aprox 65 min 6s = 1 h 5 min 6s to transfer the 2 GB of data.

*ii)*

- processing delay
- queueing delay
- transmission delay
- propagation delay

## Exercise 4

*a)*

The following steps were necessary to setup ssh servers on both vms:

- install openssh-server using `sudo apt install openssh-server` on Debian and `doas apk add openssh-server` on Alpine.
- enable the ssh service at boot time using `doas rc-update add sshd`. On Debian, the service is enabled on boot by default so this step isn't required (in case it would be enabled by running `sudo systemctl enable --now sshd`).
- start the service using `sudo service ssh start` on Debian and `doas service sshd start` on Alpine.

*b)*

Both VMs can be accessed using the command `ssh username@x.x.x.x` with x.x.x.x being the servers ip address.

## Exercise 5

*a)*

With our setups, VMs hypervised by QEMU on Arch Linux host, no changes to the config where necessary for an SSH connection from one VM to the other. Btw we use Arch ;)

*b)*

To setup ssh key-pair based login we have to execute the following steps:

- create a new key-pair using `ssh-keygen`.
- enter a name for the new key-pair when prompted
- transfer the public key to the server using `ssh-copy-id -i .ssh/keyname.pub user@x.x.x.x`.

*c)*

For Tobi everything worked out of the box. Ruben had to create the "~/.ssh/config" file and in that file specify parameters like User, HostName, IdentityFile, etc.

*d)*

- SCP/sFTP the text-file to either VM: `scp file.txt user@x.x.x.x:/home/user/`
- use RPC to rename the file by executing `ssh user@x.x.x.x "mv /home/user/Documents/test_file.txt /home/user/Documents/renamed_file.txt"`.
- login by SSH with `ssh user@x.x.x.x`. Then use the commands `cd directory` and `nano filename` to edit the file.
- SCP/sFTP from the client to pull back the file using the command `scp file user@x.x.x.x:/home/user/Documents/renamed_file.txt /home/user/uni_basel/internet_and_security/ias-exercise-2/`.
- copy files directly to the other one by using the command `scp user@ipaddress1:/path user@ipadress2:/path`.
- The -3 has the effect that "Copies between two remote hosts are transferred through the local host. Without this option the data is copied directly between the two remote hosts. Note that, when using the original SCP protocol (the default), this option selects batch mode for the second host as scp cannot ask for passwords or passphrases for both hosts. This mode is the default." (source: man scr)

## Exercise 6

*a)*

Arch (host):

- Run the command `sudo pacman -Syu` to synchronize and update the packages database.
- Install wireshark using `sudo pacman -S wireshark-cli wireshark-qt`.
- Grant execution permission to non-root users by adding your user to the wireshark group, with `sudo usermod -aG wireshark USERNAME`.

Debian:

- Run the command `sudo apt update` to synchronize and update the packages database.
- Install wireshark using `sudo apt install wireshark`.
- During the installation you are prompted to select if you want also non-root users to execute certain commands; there select 'yes'.

Alpine:

- Enable community repositories by uncommenting ' "https:///alpine//community" ' in the config file '/etc/apk/repositories', if not done during Alpine installation.
- Run `doas apk update` to update the index of the available packages.
- Install wireshark with `doas apk add wireshark`.
- Add user to group wireshark, to execute commands without root privileges: `doas adduser USERNAME wireshark`.

*b)*

Desktop environments on Linux use the X-Server (if not on Wayland) to communicate with the GUI. As this communication uses the internal network of the machine, the server can also be located on an entirely different machine like a ssh client accessing a server over a network that runs a gui application. The graphics are then rendered on the ssh client's X-Server according to the data of the ssh server. To use this X-forwarding the following steps are necessary:

- In the config-file '/etc/ssh/sshd_config' on the ssh server, set the parameter 'X11Forwarding' to 'yes'
- Now we can connect to the server by running 'ssh -X user@serveraddress' on the ssh client
- Then we can launch a GUI application over the shell resulting in the gui being displayed in a window on our ssh client.
- Alternatively we can directly connect and launch an application by using 'ssh -X user@serveraddress path_to_application'.

## Exercise 7

*a)*

Running wireshark as root is dangerous as every exploit in the sofware ir running with administrator priviledges and can therefor acces and modify nearly everything on the machine. Accordingly to the wireshark wiki, under Security chapter: *"... That's not a good idea, as using a root account makes any exploit far more dangerous: a successful exploit will have immediate control of the whole system, compromising it completely. ... "*.

*b)*

Like already explained in the installation steps, we had to add our user to the wireshark group, so that he is able to record traffic by using "/usr/bin/dumpcap".

In Debian the packet manager opens an interactive shell, were you can choose this option during the installation.

Arch and Debian:

- `sudo usermod -aG wireshark USERNAME`

Alpine:

- `doas adduser USERNAME whireshark`

## Exercise 8