Instructions for virtual machine use

Markus Peuhkuri

2016-01-07

Contents

Virtual machine for ELEC-C7240 **2**

Installing system 3

Starting up system 3

Network and setup 4

Connecting to host 4

Monitoring network traffic 6

Configuring software 7

Corrections for virtual machine 7

Missing `nss-logs` directory 8

Invalid network settings on some hosts 8

No guest add-ons installed 8

Helpful tips 8

Installing software to a host without connectivity 8

Copying files from virtual machine 9

Phase 1 tasks **9**

Task: Monitor web traffic 9

Task: Set up DNS server 9

Task: Set up Web server 10

Task: Set up email server 11

Task: TCP congestion control 12

Task: iperf testing 12

Phase 2 tasks **13**

Task: Create and implement numbering plan for IPv4 13

Verification 14

Task: change authorative DNS and service addresses 14

Task: set up static IPv4 routing 15

Task: Set up IPv6 addreeses and routing 15

Alternative / additional approaches 15

Task: Set up dynamic routing for IPv4 15

Task: Set up dynamic routing for IPv6 16

Task: Link layer address resolution 17

Task: Set up chat server 17

Task: HTTPS inspection 17

Task: HTTP-streaming 18

Task: Wireless network performance 18

Virtual machine for ELEC-C7240

Laboratory assingment is done using virtual machine image. It is available in different formats depending on virtualization platform.

Refer to installation instructions below on Installing system. Required storage is approximately 10 GB. If you do not have own computer capable of running system, contact course assistants.

The system is build on Ubuntu 14.04 image (server with Lubuntu desktop) and each host and router in test network is a Linux container (lxc) interconnected with virtual ethernet interface to a bridge. All containers run the same version of Ubuntu. For the purposes of this work, term *host* is used to refer a *container* as it mostly works like a separate computer.

After system has booted up, you are provided with a graphical login screen. Log in with credentials provided. It is recommended that you change login password and prepare basic setup as soon as possible using command `set-up-cred` (in a terminal window). That changes passwords and sets up keys needed for ssh authentication. This is important as in later phases the system may be connected to the internet and a bad password may result someone to break in.

On second phase of exercise you need a **personal identifier**. That will be emailed you soon after those tasks are published. This is required to avoid any conflicts between your network and any other network in the latter phases.

In following assingments, your personal identifier is maked as **ID**.

For purposes of the second phase of course, address range 100.64.0.0/10 is regarded as *public* addresses although it is really shared address space.

The first phase the setup contains readily configured network that can be used to implement first tasks. On the second phase the virtual machines and network are reconfigured for more complex setup. To prepare for the phase two, run command `elec-c7240-2`. To return setup to the inital state, command `elec-c7240-1` can be used.[1] Both commands overwrite part of configuration files but do take backups that can be found under `backups` in the home directory.

Installing system

Install first virtualization software suitable for your system and liking. If software reports no virtualization support (at all or not for 64-bit guests), check BIOS settings: there may be an option to enable *Virtualization Technology (VT)*.

Download a virtual machine image suitable for your system from archive. Files are compressed with *gzip* compression so they are small enough to save on FAT file system USB drive. Before using, you need to extract those.

`elec-c7240-kvm.tar.gz` Raw image and xml configuration file suitable for Linux KVM virtualisation.

`elec-c7240.vmdk.gz` VMware disk image. Many non-vmware virtualization systems can also convert that to native format.

`elec-c7240.vdi.gz` VirtualBox 1.1 format.

`elec-c7240.ova` Open Virtualization Format Archive. Many virtualization systems can import it.

When installed, it should have one virtual ethernet interface. This can be in network behind NAT, one most virtual systems provide as default.

In addition, it should have video device so that it can provide graphical user interface. This is not strictly required: one can access host using SSH and X Windowing system.

Starting up system

Once virtual computer is running, one can log in using username `comnet` with password `Comnet2015`.

[1] Note that this only restores system partially. In many cases it may be needed do some manual reworking, so it is recommended that you complete the first phase. Also check backups.

Whan started, it allocates two IP address address using DHCP, so one need to find out the ip address it got by logging in to graphical interface. One is used by the management instance (main instance) and other is used as uplink for `r1`.

At the startup, virtual hosts are in started state. You can check state of hosts with `sudo lxc-ls -f`.[2]

Note that there are some issues on virtual machine images that need to be fixed before using. See Corrections for virtual machine below.

Network and setup

System is pre-configured to support basic networking in topology Figure 1.

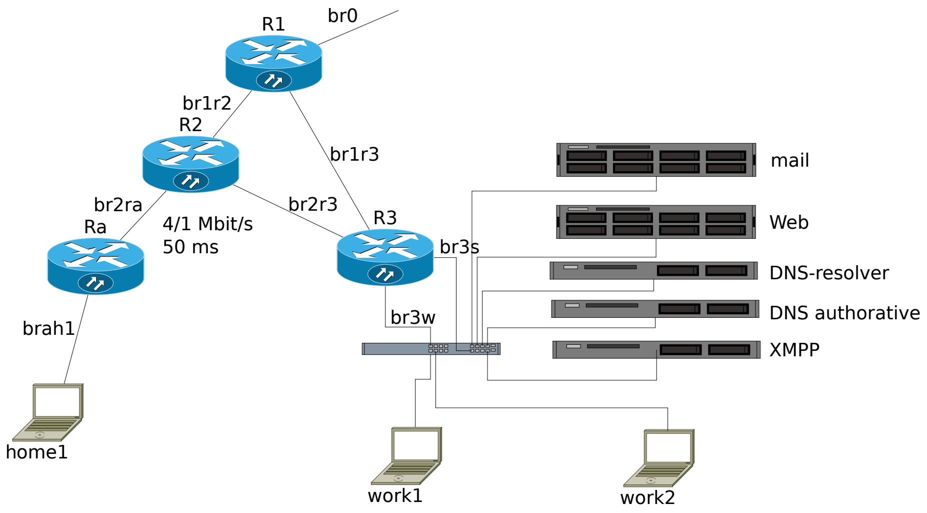

Figure 1: Network Topology

There is three client hosts: `home1`, `work1`, `work2`, and a set of servers and four routers. You can test connectivity by starting Firefox browser on `work1` with command `work1-firefox` and browsing to some internet site.

Connecting to host

As system is build utilizing Linux kernel namespaces – or containers – each router and host does have it own IP address and own view to file system. To access a host, you have two options:

[2]Unfortunately, the error message provided by `lxc-_*_` tools is a non-intuitive and misleading one if you try to run them as normal users.

- `lxc-attach` command runs a command (by default shell) in host specified with `-n` option: `lxc-attach -n work1`.

  It is also possible to run just one command by supplying `--` to indicate end of options and specifying command after that: `lxc-attach -n work1 -- ip route`. This can be usefull if you make a script to collect information from multiple hosts in coordinated manner.

- You can also make direct connect from your host machine to a container by running command like `ssh -t comnet@192.0.2.10 sudo lxc-attach -n r1`. If you have set up ssh authentication, you do not need to provide password.

- Use `ssh` command to log in remotely to a host as you would connect to any host. Note that you need to have IP addresses and routing set up so that the host can be reached.

Home directory for `comnet` user is shared accross different hosts and the main instance allowing easy transfer of files. This, however, needs to be taken into account when using applications that keep configuration or databases defined location. In these cases different hosts use different profiles or configuration files. See Table 1 for a list.

command	function
• `work1-firefox` • `work2-firefox` • `home1-firefox`	runs browser in that using profile name same as the host
• `work1-term` • `work2-term` • `home1-term`	runs terminal in that host as root user
• `work1-pidgin` • `work2-pidgin` • `home1-pidgin`	runs *pidgin* in that host with configuration in `~/.purple-`*host*

Table 1: Helper commands to run specific command in host

Part of commands – especially those that **modify** kernel states or access maybe confidential information – require root priviledges. A command can be run with *sudo* command, e.g. `sudo ip link set eth0 down`. Another alternative is to start root shell with `sudo -s` command. On default convention, prompt for user shell has `$` and superuser shell `#`. You can use command `id` to check effective user.

The root file system of each host is also visible to main system at `/var/lib/lxc/`*host*`/rootfs`. For example, to edit `/etc/network/interfaces` file in `r1`, you can open file `/var/lib/lxc/r1/rootfs/etc/network/interfaces`.[3]

There is an helper program *lxc-edit* to help editing files from main container using *leafpad* text editor. It uses same syntax as other lxc-tools. To edit above `/etc/network/interfaces` file in `r1`, you can give command: `lxc-edit -n r1 /etc/network/interfaces`.

> Remember always to verify which file you are editing not to unintentionally edit the main instance files.

Additional usefull commands include:

man Profides help for the command given as argument. The Example: `man lxc-ls`. This command is not available inside containers.

lxc-ls Shows state containers (running, stopped). Use `-f` option for fancy output including IP addresses.

lxc-start Starts container. Remember to specify `-d` option to start one in background. Example: `sudo lxc-start -n home1 -d`

lxc-stop Stops container.

lxc-attach Provides shell into running container or runs specified command. See examples above.

> Keep in mind that one **critical component of error** is missing from here: in a real world situation for example making even a simple error in configuration may lead the router unreachable over network and the only possibility is to *drive* (or fly!) for hours where router is located and access it via local console.

Monitoring network traffic

For many tasks you need to capture network traffic. For this purpose there is a *wireshark* program installed. There exists extensive documentation and tutorials for using it. Graphical UI is suitable for both analyzing live traffic real-time and for an easier analysis of recorded traffic.

In case you want to record from multiple locations at the same time, there are multiple options:

- use multi interface capability of *wireshark*
- run multple instance of *wireshark*, for each iterface

[3]This requires root proviledge.

- capture traffic to file using *dumpcap* (multiple instances or multi-interface)

Also there are text-based *tshark* and *tcpdump* programs if you are more comfortable using them. They are typically faster for quick debug checks if traffic goes via link you assume or similar checks.

All traffic between hosts goes via bridge devices visible at main main instance. For example `work1` host is connected to `r3` router via bridge `br3w`[4] as seen in Figure 1.

When monitoring traffic, consider on which network segment perform capture. When analyzing client traffic, it usually best is to have capture in the network close to client.

Configuring software

Each host has required applications installed but they are not completly configured. Note that in Ubuntu e.g. location of files may be different from application default.[5]

Some software packages can have initial configuration made in *debconf*. Before modifying configuration, run `dpkg-reconfigure`*package* to see and possibly modify default settings.

For most system software, the configuration is in `/etc/`*package*[6] file or directory. For many packages, example configurations can be found from `/usr/share/doc/`*package*`/examples`.

In addition, some software is controlled by a content of `/etc/default/`*package* also (e.g if the software is started at all).

After configuration has been modified, it needs to be activated. This can be performed by `service` *package* `reload` or `service` *package* `restart` commands.

To directly access each host, see Connecting to host.

Corrections for virtual machine

Unfortunately published virtual machine images contain few small issues that you need to fix them by yourself. These need to be done only once, i.e. change is permanent when done once.

[4]Bridge Router 3 Workstations

[5]One quick way is to list files included into package by running `dpkg -L`*package*.

[6]With a possible trailing `.conf`, `.d` or other extension.

Missing `nss-logs` directory

In Task: HTTPS inspection a directory `nss-logs` in user `comnet` home directory is needed. This can be created by opening terminal and issuing command `mkdir nss-logs` or from file manager.

Invalid network settings on some hosts

There are bugs in network configuration that may result issues for some client machines. These can be fixed by performing following set of commands in main instance:

```
wget http://www.netlab.tkk.fi/u/puhuri/elec-c7240/lxc-networking-patch.sh
sudo bash lxc-networking-patch.sh
```

This command fixes invalid lines on configuration. Reboot virtual machine after performing commands.

No guest add-ons installed

Depending on virtualization platform, it may be benefetical to install guest add-ons. These can improve graphical performance and overall management for virtual machine (e.g. shutting down works better). These depend on virtualization platform (VMWare, VirtualBox) and thus could not be installed in advance. Check documentation for the platform you are using.((Typically a menu exists with title like *Install guest add-on* or like.))

Helpful tips

Following instructions may save part of your day.

Installing software to a host without connectivity

In some case you may end with a host without connectivity to internet and you would need to install additional software. You can play with name spaces as follows to install `bastet` software on `r3`:

```
sudo lxc-attach -n r3 -s 'MOUNT|PID|UTSNAME|IPC' -- apt-get install --download-only bastet
sudo lxc-attach -n r3 -- apt-get install bastet
```

The first command uses host name space for networking to download needed packages and the second command installs downloaded packages.

Copying files from virtual machine

There are two easy ways to copy files from/to virtual machine.

1. Contact from your host to virtual machine using suitable ssh program (like *WinSCP* on Windows). You will find out the IP address for virtual machine e.g. with command `ip -4 a`. Use same login information as for console login.
2. Use `scp` program from virtual machine. For example, `scp output.txt account@kosh.org.aalto.fi:../data/Documents/` copies `output.txt` to `Documents` folder in Aalto computers. There are also graphical scp programs for linux (like `filezilla`).

Phase 1 tasks

In the first phase the setup is preconfigured on network level. No modifications are needed on routers, only servers and client hosts. Each host has already all required software packages installed. Of course, there are other software for similar functionality available, so you are free to replace installed software with another if you are more experienced with latter.[7]

Task: Monitor web traffic

Start a browser in `work1` host. Visit a website of your choise to verify network connectivity. If connectivity does not exists, resolve issue. Close browser.

Start packet capture on `br3w` interface. Enable **full-packet capture**. Start browser again and visit at different sites: at least two with plain `http` protocol and one over `https` protocol. Make note of wall clock time[8] when you visited each site. It may be easier to close browser and stop capture between each site to clearly isolate communication related to each site.

It may be helpful to activate *Developer tools* in browser and compare information you receive from *Network* tab and what you get from packet capture.

Task: Set up DNS server

The host `dns-a` has *bind* DNS server software installed. Configure it to work as authorative DNS server for domain `phase1.comnet-student.eu` and for reverse for addresses `172.16.40.0/24`.

[7]No support is guaranteed for the other software packages, so make sure you know what you are doing.

[8]Use the time provided by virtual machine. It should be syncronized over network.

Set up configuration according to Table 2. Table is available as text file from moodle.

name	IPv4 address
gw	172.16.40.1
dns-a	172.16.40.2
dns-r	172.16.40.3
smtp	172.16.40.4
imap	alias *smtp*
www	172.16.40.5
web	alias *www*
	172.16.40.5
chat	172.16.40.6
gw-ws	172.16.40.129
work1	172.16.40.130
work2	172.16.40.131
home-gw	172.16.41.1
home1	172.16.41.10
r1-r2	172.16.42.1
r2-r1	172.16.42.2
r1-r3	172.16.42.5
r3-r1	172.16.42.6
r2-r3	172.16.42.9
r3-r2	172.16.42.10
r2-ra	172.16.43.1
ra-r2	172.16.43.10

Table 2: IPv4 addresses for Phase 1

Check that both forward and reverse resolvation works. Use *host* tool on `work1`.

Task: Set up Web server

The web server has lighttpd software.

Configure a web server at address `http://www.phase1.comnet-student.eu`. Create few web pages that have some text and graphical content. Check that you can navigate between them.

Configure server also to respond to address `http://phase1.comnet-student.eu` and redirect any requests to canonical address `http://www.phase1.comnet-student.eu`.

Start packet capture and record traffic resulting from visit to own site. How traffic is different than a large site you visited earlier?

Capture few screenshots of those web pages you created at `work1`.[9]

Task: Set up email server

There is a host `smtp` that has mail server software preinstalled - *postfix* as a MTA and *dovecot* as IMAP server. However, there was some mistakes in configuration like having mail name became `phaser.lazy-student.eu` and not intended `phase1.comnet-student.eu`. Fix configutation so that user is able to send and receive email and restart server.

Create two new user account on email server with a different password so your password is not leaked in captures. Disable that user from other logins like *ssh*.

Create following aliases and have email to these addresses forwarded to account one of accounts you created:

- postmaster
- sales
- helpdesk

Configure DNS server so that email destined to `@phase1.comnet-student.eu` will be delivered to emails server you just configured.

Host `work1` and `home1` have *alpine* email program installed. Configure those so that they use `smtp.phase1.comnet-student.eu` as SMTP server and `imap.phase1.comnet-student.eu` as IMAP server. Use non-SSL[10] connections. Have different accounts on both.

Set up traffic capture on `br3s`. Send email to another account and read it from there. Have an picture (like JPEG-image) as attachment in one email. Send messages both with ASCII-characters only and including international characters. Send also email to your real email address.

[9]Host `work1` has also *cutycapt* program you can use to capture screenshots like `cutycapt --url=http://www.phase1.comnet-student.eu --out=www.png`

[10]This must not be done in real world, but easier to study.

Task: TCP congestion control

Linux kernel supports multiple TCP congestion control mechanisms. By default the kernel in Ubuntu 14.04 supports cubic and reno. All available modules can be found from `/lib/modules/`*kernelversion*`/kernel/net/ipv4/tcp_*`.

Additional ones can be activated by loading corresponding kernel module with command like `sudo modprobe tcp_vegas` on main instance. Normal users can not access new methords by default, but using command `dd if=/proc/sys/net/ipv4/tcp_available_congestion_control of=/proc/sys/net/ipv4/tcp_allowed_congestion_control` after loading modules those are activated. Note that even if the module load has to be performed in the main instance these must be activated separetly in every container you want to use them.

The default congestion control method is controlled by content of `/proc/sys/net/ipv4/tcp_congestion_control`. It will be used if application does not specify one. The *iperf* testing tool has an option `-Z` to activate specific method.

Link characteristics are controlled by linux traffic control *tc*. While there are accuracy shortcomings – in part by buffering – it is sufficient for this work.

By default the link between `ra` and `r3` is limited to 4 Mbit/s towards `ra` (*downlink*) and 1 Mbit/s in opposite. One-way delay is 40 ms. The main instance has following commands to modify that link:

command	Uplink Mbit/s	Downlink Mbit/s	Delay ms	Loss %
`tc-adsl`	1	4	40	0
`tc-oldcu`	1	4	40	2
`tc-ironsky`	8	8	1288	0.8
`tc-office`	8	8	10	0.02
`tc-edge`	0.056	0.3	500	1
`tc-satlink`	0.25	30	280	0.5

Wireshark has tools for TCP analysis. In addition, *tcptrace* tool provides extensive statistics, usefull options include `-l`, `-r` and `-o`*N*.

Task: iperf testing

Iperf is a TCP/UDP testing tool that allows sending test traffic. By default, the client sends traffic towards the server at port 5001. If using TCP, it tries to send as much traffic as allowed by TCP flow and congestion control. For UDP, maximum bit rate has to be specified.

Following are the most important options:

- `-u`: use UDP
- `-s`: this is a server
- `-c`*server*: this is client that contacts to *server*
- `-t`N: run test for N seconds (default 10 seconds)
- `-i`N: report statistics every N seconds
- `-w`Nk: request TCP flow window of N kibibytes
- `-b`Nk: set UDP bandwidth in kilobits/s
- `-Z`*name*: use congestion control algorihtm *name*

There are some public servers like `iperf.funet.fi` that can be used to test against. Iperf is also installed on all hosts and routers in test evironment.

Phase 2 tasks

In the second phase the network is reconfigured: run command `elec-c7240-2` to reconfigure all hosts. Note that this removes all IP address configuration from hosts. The existing configuration is backed up and stored in `backups` folder at `comnet` home directory as a tar archive.

Task: Create and implement numbering plan for IPv4

Following requrements are for the address allocation:

- all links between routers carry public[11] IP addresses
- each router (`r1-r3`) needs unique loopback address that must be a public one
- all servers use public IP addresses
- there must be possible to have at least 10 client computers with public IP address at company network *work*
- 50 ADSL customers with public address space: each will have one public IP address and has NAT in their router.

Calculating your address range:

- $a =$ **ID** / 2

[11]For purposes of this work, address range `100.64.0.0/10` is considered as *public* address space.

- b = **ID** mod 2 * 128

Address range: `100.64.`$a.b$`/25`. For example with **ID** 201 the address range would be `100.64.100.128/25`.

Configure IP addresses for every router and host. Router loopback addresses are configured later. You can first try to set up IP addresses using command `ip`. To set up IP address `192.0.2.2` with netmask 29 to interface `eth2` for example, you would issue a command:

```
ip addr add 192.0.2.2/29 dev eth2
```

To permanently store configuration, edit file `/etc/network/interfaces` (as root) and insert configuration there:

```
auto eth2
iface eth2 inet static
    address 192.0.2.2
    netmask 29
```

You can then set interface up using command `ifup eth2` or configure all interface marked with *auto* using `ifup -a`.

If the host is end system (one not routing) and the conneted network has only one router, then one can define default router. This is configured to interface file by adding line below if the router IP address is `192.0.2.1`.

```
    gateway 192.0.2.1
```

Make sure each router is configured to forward traffic with proper *sysctl*.

Verification

From each host, ping remote end at each link with a command like.

```
ping -c 3 192.0.2.1
```

Task: change authorative DNS and service addresses

As the DNS server IP address is changed, the addresses are not anymore resolved via it. Create a new zone for *dept*`.comnet-student.eu` and for the reverse zone. Include also router interface addresses for both forward and reverse zone.

Update all services (web, mail, chat, dns) according to the new names plan. Make needed modifications also for every host for them to be able to resolve addresses.

The host `dns-r` is a recursive name server. Reconfigure it to allow recurisive queries from new local network addresses and accept reverse resolving for addresses you configured.

Task: set up static IPv4 routing

It is not very useful to be only to access the host on the other end of link. To be able to access any host in network, a routing must be configured.

Configure routing for each router to be able to relay traffic with all routers in the network.

Also you need to configure IPv4 Network address translation on `r1`.

Verify that all services are reachable as in Phase 1.

Task: Set up IPv6 addreeses and routing

Your IPv6 address allocation is `2001:0708:40:9xx0::/60` where `xx` is your **ID** in hexadecimal number. For example if **ID** is 201 then allocation is `2001:0708:40:9c90::/60` (12*16+9=201).

Allocate addresses for different networks. Use longer masks for router point-to-point addresses. Define required static routes. Add required DNS records to your DNS configuration.

Verify routing by using ICMPv6 echo requests *ping6* command.

Alternative / additional approaches

- If you have a native IPv6 connectivity, you can also make configuration using addresses you have.
- It is possible to construct 6in4 tunnel from internet to your virtual machine. This requires however some additional configurations on your host system also and is subject for availability.

Task: Set up dynamic routing for IPv4

This work is conducted on routers `r1`, `r2` and `r3`.

Start *rtmon* program[12] in all routers to log routing changes in network. You may also want to initiate traffic capture on different links.

Remove all static IPv4 routes you defined earlier. Configure OSPF routing in routers `r1`, `r2` and `r3` using area `0.0.0.`*ID* using *quagga* software configured in each of those[13].

Make sure you are using IPv4 routing only. Disable IPv6 autoconfiguration and remove any IPv6 addresses from clients to make sure they use IPv4 only if needed.

Verify routing tables by listing routes in each router and pinging the same servers as in static routing case. Run *traceroute* command from `home1` to your `web` server.

Set up `ping` from `home1` to web server sending packets with 0.5 second intervals. Some other useful options include `-O` and `-D` among others. Shut down link between `r2` and `r3` by putting interface down `ip link set br2r3 down` on main instance. Wait until responses start to come back again. Stop *ping* with *Control-C* and record how many packets were lost. Run *traceroute* command from `home1` to your web server.

Record routing tables and ip neighbour tables (ARP). Request a web page from a `web` and outside site to verify end-to-end connectivity.

Restart `ping` as earlier and restore link back up between `r2` and `r3` by setting `br2r3` back up. Does traffic change back to original route? How you can identify that?

Repeat test three times.

From file generated from *rtmon* you can use `ip monitor route file`*file* to read routing changes with timestamps. Compare routers view to view resulted from `ping`. Are there any differences and what may have resulted them?

After tests, reactivate IPv6 addresses on clients.

Task: Set up dynamic routing for IPv6

This work is conducted on routers `r1`, `r2` and `r3`.

Start *rtmon* program[14] in all routers to log routing changes in network. You may also want to initiate traffic capture on different links.

Remove all static IPv4 routes you defined earlier. Configure OSPFv3 routing in routers `r1`, `r2` and `r3` using area `0.0.0.0` using *quagga* `ospf6` daemon.

[12] E.g. `rtmon file r1-rmon.log` at `r1`.

[13] If your **ID** is greater than 255, then use additional octets when needed like `0.0.1.10` for *ID* 266.

[14] E.g. `rtmon file r1-rmon.log` at `r1`.

Verify routing tables by listing routes in each router and pinging the same servers as in static routing case. Run *traceroute* command with option `-6` from `home1` to your `web` server.

Set up `ping6` from `home1` to web server sending packets with 0.5 second intervals. Some other useful options include `-O` and `-D` among others. Shut down link between `r2` and `r3` by putting interface down `ip link set br2r3 down` on main instance. Wait until responses start to come back again. Stop *ping6* with *Control-C* and record how many packets were lost. Run *traceroute* command with option `-6` from `home1` to your `web` server.

Record routing tables and IPv6 neighbour tables.

Restart `ping6` as earlier and restore link back up between `r2` and `r3`. Does traffic change back to original route? How you can identify that?

From file generated from *rtmon* you can use `ip monitor route file`*file* to read routing changes with timestamps. Compare routers view to view resulted from `ping`. Are there any differences and what may have resulted them?

Task: Link layer address resolution

Set up capture on `br3w`. Start *rtmon* on both `r3` and `work1`. Ping (IPv4) router `r3` from `work1` and wait until you see few ARP requests and responses between those. Keep ping running. On main instance side, set interface `br3eth3`[15] down.

Monitor how long `work1` is trying to send ICMP echo packets until it reports destination unreachable. Keep interface down for additional 30 seconds (approximately). Set interface back up. How long it takes to traffic to flow again?

Task: Set up chat server

Configure *prosody* server on `chat`. Create at least three user accounts and one conference room. Configure right DNS SRV records for the server.

Start capture on `br3s` and launch *pigdin* on `work1`, `work2` and `home1`. Chat with those both private chat and in conference room. Send messages of different lengths and including both ASCII-only and international characters and special glyphs.[16] Test differnet on-line statuses.

Task: HTTPS inspection

If web browser is started with one of helper commands, a https key logs are enabled. Those are stored in `nss-logs/`*host*`.txt` files. You can import these

[15]Check this is connected to `br3w` bridge.
[16]You can use Character Map tool (*gucharmap* on command line) to input those

to wireshark as explained in this document via *Preferences→Protocols→SSL* settings.

Make sure `nss-logs` directory exists. Set up packet capture. Visit a site that provides the same content both with HTTP and HTTPS, for example `www.netlab.tkk.fi`. Visit page with both methods. Compare amount of traffic both methods. Study and decode traffic to understand details.

Task: HTTP-streaming

Set up packet capture. Visit video streaming sites (like https://youtube.com, https://vimeo.com/, http://areena.yle.fi etc.) and pick few 5-10 minutes long videos from at least two different services. Play those videos. Calculate statistics: how many HTTP requests, TCP connections there are? How may bytes? As e.g. YouTube and Vimeo use HTTPS for streaming, you need to use method Task: HTTPS inspection described above.

Download then some large file with size approximately payload size of one of videos. Compare bandwidth used over time. Calculate statistics with *tcptrace*. Any significant differences?

Task: Wireless network performance

This work is not completed in virtual environment but in real world. You need either a laptop with WLAN capable to run *iperf* or a mobile phone with WLAN and *netradar*.

WLAN accesspoint with SSID `elec-c7240` is located in G220 laboratory in Otakaari 5. Connect to it and run *iperf* or *netradar* tests from different locations along main corridor of Otakaari 5. If weather allows, measure also outdoors. Mark your location on map and bandwidth received. If your device reports signal level and other connection quality figures, report those also. What is location furthest you were able to connect? Was there difference where you could (re)connect and where maintain connectivity? Mark also locations where attempt failed: indicate if WLAN assosiation was up or not. How distance affected on bandwidth and signal strength? Document device you used for measurements.