Miika Valkonen

Upgrading MetroIX Ubiquiti routers

Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Thesis

18 April 2018

Metropolia

Author	Miika Valkonen
Title	Upgrading MetroIX Ubiquiti routers
Number of Pages	33 pages + 1 appendix
Date	18 April 2018
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialization	Computer Networks
Instructors	Marko Uusitalo, Senior Lecturer

The purpose of this thesis was to upgrade the Ubiquiti routers in Metropolia's lab network called MetroIX and to create a basis for documentation of the network. The thesis was divided into three main parts.

Firstly, a previous software upgrade attempt had failed for an unknown reason and this needed to be troubleshooted. Once the problem had been resolved the software was to be upgraded to the newest release available at the time. The problem turned out to be a memory leak in BGP multipathing and was resolved by disabling it until it gets patched in a newer release.

Secondly, the lab network is missing any kind of documentation of the network, making it very hard for anyone new to start making changes. As a result, two network diagrams were drawn, both before and after the changes made in this thesis. Additionally, other crucial elements of documentation were considered, but none are currently implemented. These include login credentials and setting up an external log server, which could also hold backups of the configurations.

Lastly, once the software had been upgraded and a network diagram had been drawn, the network was to be upgraded to use IPv6 side-by-side its existing IPv4 configuration. This meant subnetting an IPv6 global routing prefix assigned to Metropolia, configuring the interfaces to use the IPv6 addresses, and setting up BGP and OSPFv3 sessions between the Ubiquiti routers using their IPv6 addresses. Once the internal network was up and running, the ISP was contacted to set up BGP peering with their gateway router so that the whole IPv6 routing table of the internet could be added to the Ubiquiti routers' routing table.

Keywords	Software upgrade, documentation, router, IPv6, BGP

Metropolia

Contents

Metropolia

List of Abbreviations

ACL	Access Control List, a list that may be used to grant or block access from certain hosts, networks, or protocols
AS	Autonomous System, a set of addresses assigned to an organization or a company
BGP	Border Gateway Protocol, a dynamic routing protocol
CLI	Command Line Interface, a text based user interface to issue commands to a system
DHCP	Dynamic Host Configuration Protocol, a server that assigns an IP address, gateway, and DNS to hosts so they can communicate with other networks
DNS	Domain Name System, a system that translates more easily memorized domain names into IP addresses
FTP	File Transfer Protocol, a protocol used to transfer files between a client computer and a server
IANA	Internet Assigned Numbers Authority, a nonprofit corporation in charge of IP address allocation among other tasks
IP	Internet Protocol, a protocol that defines how devices on the internet communicate with each other
IPv4	Internet Protocol version 4, which uses a 32-bit addresses
IPv6	Internet Protocol version 6, which uses a 128-bit addresses
IoT	Internet of Things, a network consisting of home appliances, cars, and other items with embedded electronics allowing them to send and receive data over the internet

ISP	Internet Service Provider, an organization that provides access to the internet as a service
IX(P)	Internet Exchange (Point), a location for organizations and companies to exchange traffic between their networks
KiB	Kibibyte, a more accurate unit than kilobyte to denote 1024 bytes
LAN	Local Area Network, a network in a limited area such as a school, a home, or an office
NAT	Network Address Translation, remapping an IP address into another allowing private addresses to communicate with public addresses, sometimes also used when talking about IP masquerading
OSI	Open Systems Interconnect, a layered model for network design
OSPF	Open Shortest Path First, a dynamic routing protocol
QA	Quality Assurance, a process during which an organization tests its product or service to ensure they’re of high quality and defect free before a release to the public
RFC	Request for Comments, a publication from the internet governing organizations defining standards to be used on internet-connected applications and systems
RIP	Routing Information Protocol, one of the first popular dynamic routing protocols
RIR	Regional Internet Registry, an organization in charge of assigning internet number resources, such as IP addresses, to its region’s internet service providers and other organizations
SFTP	SSH File Transfer Protocol, a way to access or transfer files between systems over a secure SSH connection

SSH	Secure Shell, a protocol to encrypt connections over an unsecure network
TAC	Technical Assistance Center, a device manufacturer's department helping customers with technical support cases
TFTP	Trivial File Transfer Protocol, same as SFTP but over an unsecure connection
TOP	Table of Processes, a program on Unix-systems that displays information about CPU and memory usage
URL	Uniform Resource Locator, also often called a web address, tells the computer system how to locate and access a resource
USB	Universal Serial Bus, a standard developed to standardize how computers and their accessories should connect and communicate with each other
VLAN	Virtual Local Area Network, a virtual LAN that appears as if it was a single network but may be separated to multiple different ones
WAN	Wide Area Network, a network comprising large geographical distances such as linking a home to the internet

1 Introduction

The world is quickly running out of IPv4 addresses. Several tools have been developed and used over the years to mitigate the issue, such as DHCP, NAT, private address ranges, and more. None of these, however, are a permanent solution to the ever-increasing number of devices connected to the internet – and the pace is accelerating. While it was understood in the late 1980s and early 1990s that the world would run out of IPv4 addresses and a solution was needed, no one could have anticipated how quickly it would happen.

The solution was to develop a new version of Internet Protocol, namely IPv6. As with any new feature, changes are required on the software side, and sometimes on hardware as well. New changes are applied on devices via upgrading to newer software versions. Commonly changes include new features, bug fixes, or security patches, and the changes are listed in release notes published by the manufacturer.

The software version currently being used on network devices is a detail that is often included in documentation. Documentation for a network should include all the relevant information and needs to be kept up-to-date. However, what is relevant depends on the network setup and who maintains it. There is no one correct answer, but some of the major components that should be included in any network are network diagrams and information on how to access the devices. Other key information varies based on the size of the network in question; the bigger the network, the more information should be included. A good rule of thumb to sufficient documentation is to imagine if the current documentation would be enough for a new person to take over managing and maintaining the network.

2 Software upgrade

2.1 Why upgrade a network device’s software?

There are multiple reasons for upgrading a network device’s software, which can also be called firmware – a type of software. The most important reason is patching security vulnerabilities. Security vulnerabilities are vulnerabilities that leave a product open for an

unauthorized third party who could potentially have malicious goals. On the topic of network devices, this most often means an unauthorized person could gain access to the device remotely and read its configuration, change its configuration, or use it to at-tack resources that are only accessible once you are inside the LAN.

Earlier this year, a security vulnerability was found on modern processors that made the headlines all over the world. The vulnerabilities, nicknamed Spectre and Meltdown, exploit something intended to speed up processors called speculative execution and caching. What the vulnerability essentially does is it allows an attacker to read information from the processor's cache that could contain passwords or other sensitive data. [1.]

Security vulnerabilities, especially ones where an attacker can bypass other security measures, should generally be patched soon after a fix has been released. However, if the software upgrade causes a network outage, for example if the network devices need to be rebooted for the upgrade to come into effect and there is no redundant connection available, the upgrade should be planned in advance to minimize downtime and to inform anyone using that connection of the upcoming outage so they can plan their work around it. Depending on how critical the vulnerability is, sometimes patching it as soon as possible should be a first priority and planning and informing users comes second. Vulnerabilities like this could include situations where an attacker can login to de-vices or capture unencrypted traffic between devices from an external connection.

Another reason for software upgrades are bug fixes. Bugs that can be fixed or circumvented with a software upgrade are unintended behavior in a software caused by erroneous code. Bugs can result in just about anything from a broken feature with minimal overall impact to a device crash leaving the network impaired or completely down in worst case scenarios. A network device's code is complicated and thousands and thousands of rows in length, and as such bugs are almost unavoidably going to occur in any given manufacturer's code. New features are also being added which may have un-foreseen consequences with much older features, and while the manufacturers do ex-tensive testing it is logistically impossible to account for every possible scenario. Some bugs may only occur if a very specific set of configuration and network setup is being used making them very hard to identify beforehand.

Updated software can also add new features to old hardware. While they may not be a necessity for the network to work, they may be desirable for other reasons such as easier

maintenance and management. For example, being able to configure an automatic export to a separate server of the configuration on the device whenever a change is applied. New features however are rarely so necessary that a software upgrade needs to be scheduled immediately, and instead can wait for the next available maintenance window or when a security vulnerability or a software bug has been fixed.

Software versions are normally numbered incrementally, though not all versions are released publicly so from time to time users may see certain numbers skipped in between versions. Some are tested internally and due to problems with the version or other reasons are not published for users. A common version numbering has three different numbers separated by a dot telling the major, minor, and patch version. Major versions include big changes that may not be backwards compatible with older software releases. Minor versions add smaller functionalities that are compatible with older soft-ware of the same major version. Patches are mostly used for bug fixes and security vulnerabilities, and other small changes. [2.] Some manufacturers also use hotfixes in addition to the numbering system to indicate more urgent updates, such as critical security vulnerabilities or bugs causing major issues. Hotfixes are not as well tested due to time being a factor and the need to get the fix out to users as soon as possible, which may cause additional bugs in the code that are then fixed with future hotfixes or patches depending on severity. For example, a device running software version 1.9.7+hf4 would be on the first major version, ninth minor version, seventh patch, and fourth hotfix.

2.2 Release notes and upgrade path

Release notes are an important part of a software upgrade. They are produced by the manufacturer and tell the user what has changed between the previous released software and the new one. They list all the bug fixes, security patches, new features, and any other possible changes made to the code. Depending on the manufacturer, the release notes may even list any known issues that they had noticed during testing. Usually these issues do not have significant impact as the release would not have happened otherwise. They may be isolated to a specific feature with no impact elsewhere and anyone using said feature should avoid upgrading to the newest software until the issues have been fixed in a future release.

The notes also contain another key piece of information for anyone planning a software upgrade. If the software version being upgraded is several major or minor versions behind the newest release, and the plan is to upgrade all the way to the newest version, the upgrade path needs to be checked. As mentioned earlier, not all changes are backwards compatible, and as such may require a very specific software version to have been installed in between the much older version and the newest version. For example, if version 3.3.0 uses something introduced in 3.2.0 and the device is currently running version 3.0.0, you may have to upgrade the device first to version 3.2.0 and then to the desired 3.3.0 afterwards. The upgrade path refers to the path of versions you must install for all the new features to be added correctly.

Some systems are clever enough to be able to use the code from version 3.2.0 in the previous example as long as the file is on the device's storage so that it's not a necessity to install it first. Instead you can go straight from 3.1.0 to 3.3.0 and the installation of 3.3.0 uses the version 3.2.0 files to install the necessary components. Installing 3.2.0 in between would cause an additional reboot and thus a brief network impairment or outage depending on the topology. However, this kind of behavior is not common when upgrading over multiple major versions.

There are also devices, which have their software built in such a way that an upgrade path isn't needed. Every release contains all the necessary information for that software version, and no prior extra releases are needed on the device. Release notes from manufacturers building their software in this manner hardly ever include any mention of an upgrade path in their notes or may simply state that it is not needed with their soft-ware.

2.3 Upgrading Metropolia lab network's software

2.3.1 Problems with a previous upgrade attempt

A software upgrade for the lab network was attempted in an earlier thesis. For an unknown reason, the BGP daemon, a process run by the system in the background for the dynamic routing protocol BGP, would crash sometime after the latest reload and the daemon was unable to recover on its own or with manual input, thus requiring a reboot of the device.

The four routers were running on software version 1.6.0 and being upgraded to 1.9.0, which was the newest release at the time. First step was to go through the release notes to see if anything had been posted under the known issues or if any major changes to how BGP works on the router had been made, which could've invalidated the current configuration on the devices. While multiple bug fixes and other improvements had been made to routing in general, BGP, and many other things, none were noticed that could have been a potential cause for the BGP daemon crashing.

When going through release notes did not resolve the issue, the next step was to attempt upgrading the software to the newest version, which was 1.9.7+hf4 at the time. None of the release notes between versions 1.9.0 and 1.9.7+hf4 seemed to be directly related to the problem experienced with the lab network either. After upgrading the software to the newest version, the BGP process was checked and deemed to be operating normally with no signs of problems. The next time the routers were checked, the BGP daemon had crashed and due to the limited log buffer, no system log messages about the crash were visible anymore.

Now that it had been verified the problem still existed in the newest software and due to limited capability to troubleshoot the issue at the time, the change had to be rollbacked. Rollbacking is explained in more detail in chapter 2.3.3, but it basically means the situation was returned to the previous stable setup. In this case, the software was rollbacked to version 1.6.0, which had not had issues with BGP daemon crashing.

Due to not knowing how much time had elapsed between the reboot after the upgrade and the daemon crashing, the next time the situation needed to be monitored more closely. As the routers are accessible remotely and the firewall rules allowed pings and SSH-connections from Metropolia's public and private networks, a ping could be run from Metropolia's shell to see if the router was still replying to pings. If it did not respond, the daemon had likely crashed meaning the router wasn't being advertised to the service provider's router anymore and the route for the ping packets was missing from the internet's routing table.

The software upgrade could be done remotely, but once the process crashed physical access to the router was required. To recover from the crash and to bring the BGP daemon back online, the router needed to be rebooted, but rebooting also wipes all temporary files such as system log from the device. System log and any error messages it

contains are instrumental in solving any kind of crash related issue, so before a reboot the system log needed to be recovered and saved externally. A computer with a console connection to the crashed router was needed.

Once the system log had been saved the router could be rebooted and then rollbacked once again to version 1.6.0 so that remote access to it was possible via shell. Unfortunately, once the text file containing the system log outputs was being analyzed it was found that the file had been corrupted and most of the text was illegible. The procedure needed to be repeated one more time.

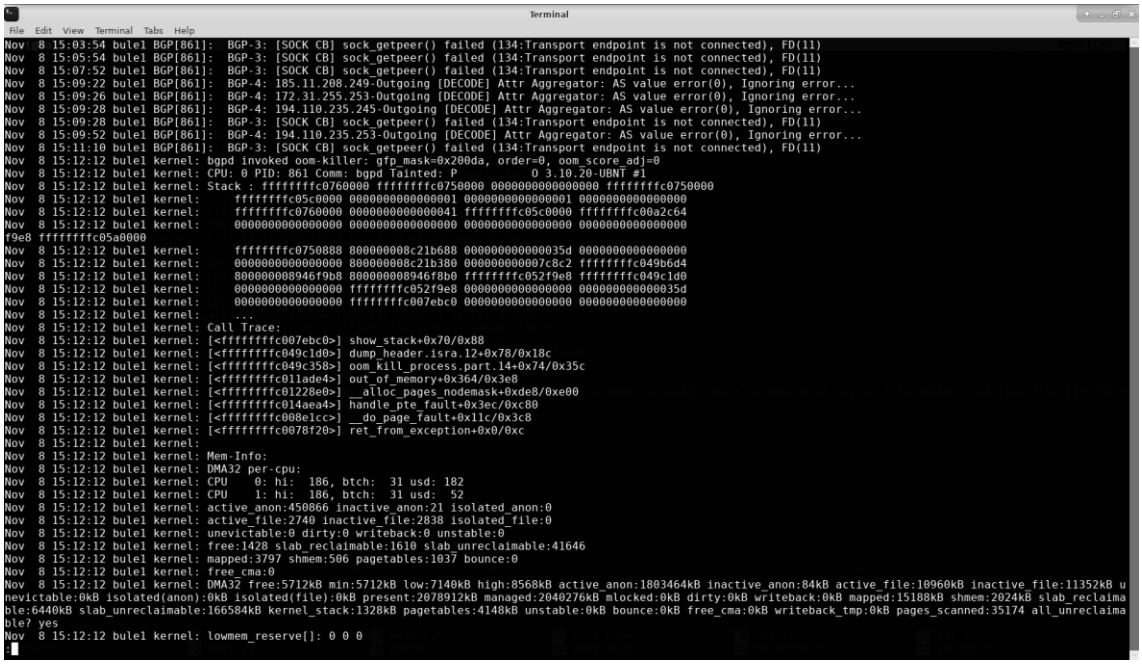


Figure 1. System log output right after the BGP daemon has crashed showing a stack trace with the line “out of memory” in the middle of it.

Once the post-crash system log was finally analyzed, it revealed that the crash was caused by the device running out of memory as seen in image 1. The software had a memory leak and the memory usage kept going up until it reached 100% and the BGP daemon crashed freeing all the used-up memory in the process. Afterwards the router was back to operating apart from the BGP process.

```
top - 14:13:11 up 6 min,  1 user,  load average: 1.87, 1.19, 0.53
Tasks:  66 total,   2 running,  64 sleeping,   0 stopped,   0 zombie
%Cpu(s): 13.9 us, 42.7 sy,  0.0 ni, 42.5 id,  0.0 wa,  0.0 hi,  0.8 si,  0.0
st
KiB Mem:   2040584 total,  1044756 used,   995828 free,    24656 buffers
KiB Swap:        0 total,        0 used,        0 free,    95436 cached
```

```
  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
  614 root      20   0 72916  67m 1520 R  43.4  3.4   0:57.89 ribd
 2372 ntp       20   0  6732 2128 1716 S  40.8  0.1   0:45.31 ntpd
  589 root      20   0  7328 2864 1908 S  17.9  0.1   0:23.80 nsm
  861 root      20   0  703m 699m 2468 S   6.6 35.1   2:22.27 bgpd
```

Listing 1. An output of top-command 6 minutes after a reload, with the BGP process showing 35.1 % memory usage.

To verify and to see it in action, the routers were reloaded once again with the new software and memory usage on the device was monitored with the top-command. As BGP peering with the neighbor is being established and the routes are calculated it takes a significant amount of CPU power to perform the necessary calculations. After the BGP session has stabilized, CPU usage goes down quite a lot and memory usage was around 35 % as seen in listing 1. Relevant information has been bolded for easier readability.

The memory usage of the bgpd process kept increasing at a relatively steady pace averaging a little below 20 % for every 10 minutes. The time it took for the process to use up all available memory and crash seemed to vary from an hour up to four hours between the several tests done to troubleshoot the issue.

```
top - 15:00:18 up 53 min,  1 user,  load average: 1.72, 1.81, 1.77
Tasks:  65 total,   3 running,  62 sleeping,   0 stopped,   0 zombie
%Cpu(s): 13.3 us, 41.0 sy,  0.0 ni, 45.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0
st
KiB Mem:   2040584 total,  1946528 used,    94056 free,    24672 buffers
KiB Swap:        0 total,        0 used,        0 free,    95528 cached

  PID USER      PR  NI  VIRT  RES  SHR S  %CPU %MEM    TIME+  COMMAND
  614 root      20   0  150m 146m 1520 R  45.4  7.4  20:46.58 ribd
 2372 ntp       20   0  6732 2128 1716 S  42.8  0.1  19:00.30 ntpd
  589 root      20   0  7328 2864 1908 R  18.6  0.1   8:28.60 nsm
    3 root      20   0     0    0    0 S   1.7  0.0   0:48.65 ksoftirqd/0
   13 root      20   0     0    0    0 S   1.7  0.0   0:42.16 ksoftirqd/1
  564 root      20   0  121m  10m 3376 S   0.7  0.5   0:17.16 ubnt-util
  534 root      20   0  1948  276  220 S   0.3  0.0   0:01.74 rngd
  861 root      20   0 1420m 1.4g 2468 S   0.3 71.1   7:14.70 bgpd
```

Listing 2. An output of top-command 53 minutes after the last reload with the BGP process showing 71.1 % memory usage.

As seen in listing 2, the BGP daemon, or bgpd, has already taken up over 70 % of the router's total memory of 2 040 584 KiB.

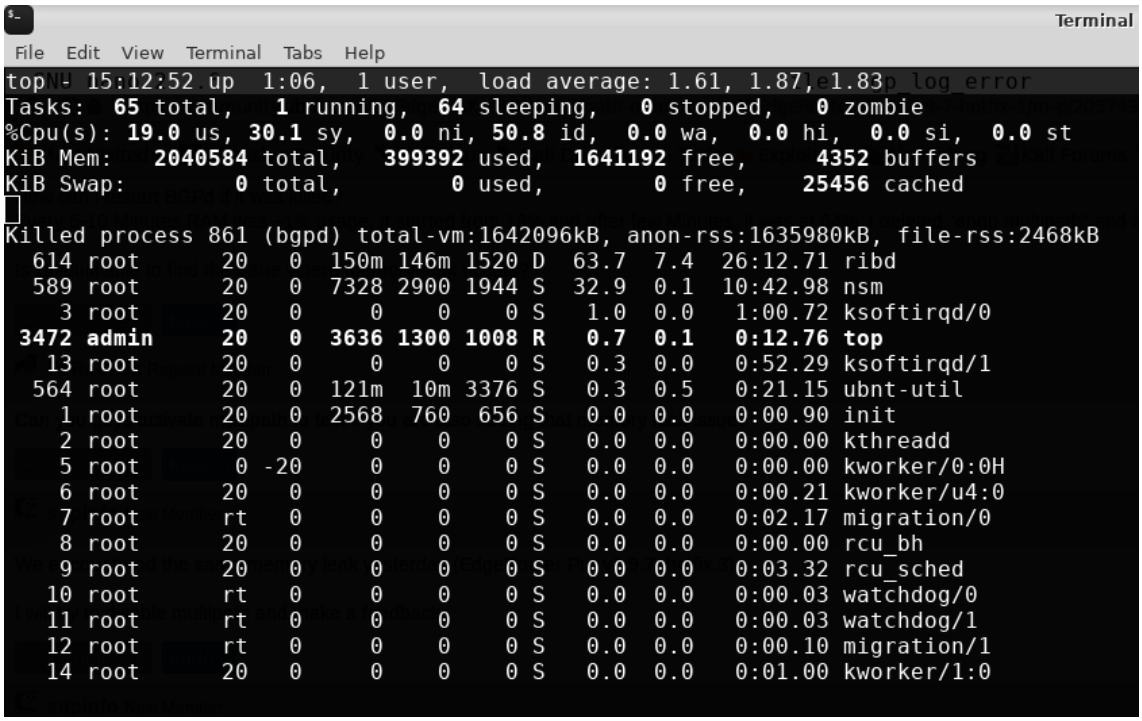


Figure 2. A screenshot of the output from top-command showing the bgpd process has crashed.

Figure 2 shows that the bgpd process has crashed with an uptime of 1 hour and 6 minutes on the device. At this point the system log has the same outputs that are seen in figure 1 telling that the device has run out of memory and bgpd has crashed because of it.

With the knowledge of the daemon crashing due to memory related issues, the release notes were checked again for any BGP and memory related notes. In version 1.8.5 there was a mention of a fix having been implemented.

> [BGP] Fix possible memory corruption that was causing the routing daemon to crash in some cases. Reported by michaeldale who also helped run a debug build and provide detailed information which made the fix possible (see here). Thanks! [3.]

However, the issue was clearly reoccurring in the 1.9 versions or it was a completely new, possibly unrelated memory fault in BGP. Searching on Ubiquiti's forums for any other users experiencing similar issues, a post by the username RcRaCk2k mentioned he had noticed BGP daemon crashing due to running out of memory after enabling BGP multipathing on version 1.9.7+hf1. A few other users reported seeing similar results after multipathing was enabled on 1.9.7 software versions, and disabling had stabilized

memory usage for them. At the time there were no official comments from the manufacturer about the issue. [4.]

Due to this information, BGP multipathing was disabled on both Bulevardi's routers to test if it was causing the crashes.

```
admin@bule1# show protocols bgp 65300 maximum-paths
 ebgp 2
 ibgp 2
[edit]
admin@bule1# delete protocols bgp 65300 maximum-paths
[edit]
admin@bule1# commit
[edit]
admin@bule1# show protocols bgp 65300 maximum-paths
Configuration under specified path is empty
[edit]
admin@bule1#
```

Listing 3. The CLI commands used to disable multipathing and to verify it's off on Ubiquiti's routers.

The routers were reloaded to ensure a fresh start and that multipathing had indeed been properly disabled, and the memory usage was monitored over the next couple of hours. Listing 3 shows the commands used to disable multipathing, which can eventually be used to enable it should a fix be released by Ubiquiti. The same process was repeated for all four routers, with the exception that Leppävaara's routers use BGP AS 65 400 instead of 65 300.

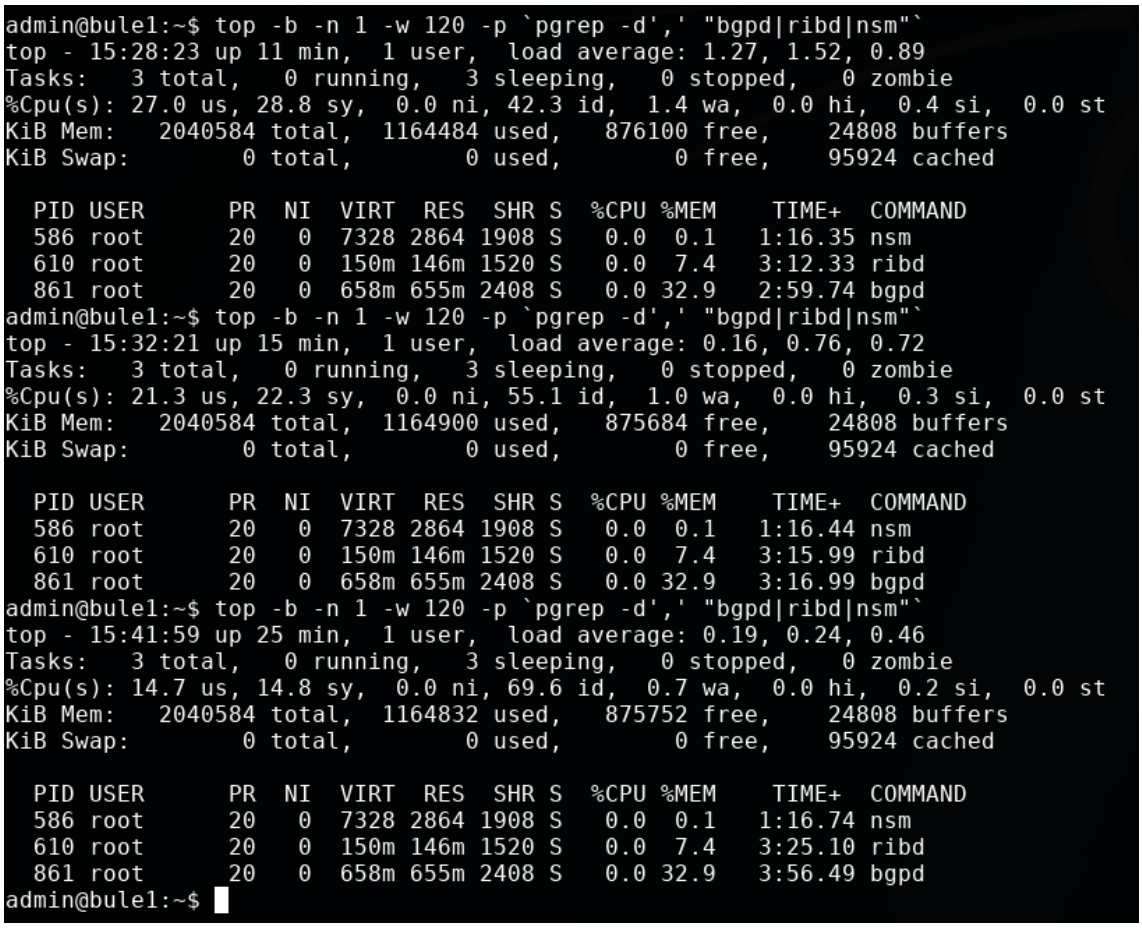


Figure 3. The bgpd process is stable at 32.9 % memory usage over the first 25 minutes after a reload.

As seen in figure 3, the bgpd process was staying at a stable memory usage and not climbing as it had behaved previously with the multipathing enabled. With no significant changes to memory usage noticed over the next couple of weeks the fault had been located.

A couple of months later the thread was checked again for any updates. Ubiquiti's employee with the username UBNT-afomins had posted on update in December mentioning the issue had been fixed in a previous software but had to be rollbacked due to other issues. At the time of UBNT-afomins' post the development team was working on a release for version 1.10.0, and UBNT-afomins commented that they would work on a new resolution for the issues after the release of version 1.10.0. However, the discussion they linked to is talking about CPU usage problems with multipathing enabled and not directly memory leaks so it remains to be seen what the future fix will be. [5.]

On February 15, the version 1.10.0 was released and as was to be expected it did not contain a mention of the memory leak having been fixed. However, it did have a note saying that Ubiquiti had added a watchdog, which monitors the systems internal processes and tries to correct any faults detected, for the routing daemons. “[Routing] - Add watchdog for critical routing daemons (nsm, ribd, ospfd, bgpd...) which will restore crashed daemon.” [6.] In the current software the BGP daemon did not recover after its crash and could not be manually brought back up but the whole device had to be rebooted. A watchdog that would recover the crashed daemon automatically would minimize network downtime instead of having a network engineer troubleshoot the problem, locate the faulty device, and finally reboot the device. Rebooting it would require the engineer to be in the same physical location if there is no way to control the power supply remotely, like a UPS with networking capabilities that is behind a different router. This can add up to a lot of time spent all the while network could be completely down if there are no redundant links.

On March 21, a new patch 1.10.1 was released, but again there was no fix mentioned for BGP multipathing issues. Instead the release notes listed multiple other fixes, as is often the case when a new major or minor version has been released and it is exposed to different networks that weren’t tested in the internal QA process. [7.] Due to the latest minor version release being still quite new and not as well tested, and because it doesn’t contain a fix for the multipath memory leaks, it was decided to wait for a later release before upgrading again despite the addition of useful things like a watchdog for the routing daemons.

2.3.2 Upgrading a Ubiquiti EdgeRouter PRO

While there are many mirrors available, the safest option is to download the software from the downloads section of Ubiquiti’s website. As the Metropolia lab network’s routers are already connected to the internet, the simplest way is to download the software image directly to the router.

```
add system image https://dl.ubnt.com/firmwares/edgemax/v1.9.7/ER-
e200.v1.9.7+hotfix.4.5024021.tar
```

Listing 4. Command used to add the image of version 1.9.7+hf4 to the router.

On Ubiquiti’s website pressing download on an image offers you two options: downloading the file to the computer’s local storage accessing the website, or copying the direct

URL to the file. By replacing the URL on listing 4 with the desired version's direct link to the file, the router can download the file from the internet and into its storage and it can then be used to install it on to the router.

Another option of transferring an image to the device is downloading it locally to a computer that is connected to the internet and then transferring it to the router. To transfer it, it's possible to set up a TFTP or and SFTP server on the computer with the image and have it in the same LAN as the routers or another network the routers have access to, then downloading it to the router from the server. If this is not a possibility due to how the network has been setup, the routers do have a USB port and transferring files via a USB memory stick is possible if physical access to the routers is available.

The Ubiquiti EdgeRouter PROs can hold 2 images. One that is currently being used to run the router and a second one that it is being upgraded or downgraded to from the current version.

```
admin@bule1:~$ show system image
The system currently has the following image(s) installed:

v1.9.7+hotfix.4.5024021.171005.0533 (running image) (default boot)
v1.6.0.4716006.141031.1738

admin@bule1:~$ set system image default-boot
The system currently has the following image(s) installed:

v1.9.7+hotfix.4.5024021.171005.0533 (running image) (default boot)
v1.6.0.4716006.141031.1738

Are you sure you want to switch images? (Yes/No) [Yes]: N
Canceling switch
```

Listing 5. Commands used to verify and change default boot image.

Once the image has been transferred to the router, all that is needed is to set the new image as the default image to boot with and reload the router. By default, when a new image has been downloaded the system sets it as the primary boot image for the next reload. This can be verified using the command in listing 5: show system image. If the desired image is not set as the default boot image, it can be changed by issuing the other command in listing 5: set system image default-boot.

```
admin@bule1:~$ show version
Version:      v1.9.7+hotfix.4
Build ID:     5024021
Build on:     10/05/17 05:33
Copyright:    2012-2017 Ubiquiti Networks, Inc.
HW model:     EdgeRouter Pro 8-Port
HW S/N:       24A43C3C45F8
```

```
Uptime:       17:50:32 up 5 days, 20:36,  1 user,  load average: 0.07, 0.12,
0.13
```

Listing 6. Command used to verify the current running software version and other details.

After the default boot image has been selected and the router reloaded, the running version can be checked by issuing the command in listing 6.

2.3.3 Rollback

In the context of network devices, a rollback is an operation where an implemented change is reverted and the original configuration or state is brought back. This is often done because the implemented change did not work or it had unforeseen side effects causing other problems. Having a rollback plan is an important part of any larger change but should be considered for smaller, more routine changes as well. Time spent on imagining everything that could go wrong and how to recover from those situations should be proportionate to critical the uptime of the network is.

When a change is being planned it is a good idea to create checklist of things that need to be done as a part of the change and things that need to be checked after the change has been completed to make sure they're still working normally. Most of the time when a specific change is being performed for the first time there will be things that were missed when planning for the change. Those lessons learned should be used to improve the checklist so that the next time goes smoother. The plan should also include a plan on how to recover from the potential problems.

```
admin@bule1# set system config-management commit-archive location
Possible completions:
  <uri>         Uniform Resource Identifier

Detailed information:

    "scp://<user>:<passwd>@<host>/<dir>"
    "ftp://<user>:<passwd>@<host>/<dir>"
    "tftp://<host>/<dir>"

[edit]
admin@bule1# commit-confirm
Possible completions:
<enter> Commit, rollback/reboot in 10 minutes if no confirm
<N>     Commit, rollback/reboot in N minutes if no confirm
```

Listing 7. The command to automatically export configuration after a commit to an external server and the command to require a manual confirmation after a commit.

Performing a rollback on most modern network hardware is relatively simple as there are built-in tools to help with the task. For example, devices may have a configuration option to automatically export the running configuration to an external server whenever a change is applied as shown in listing 7. The configuration can also be rollbacked automatically on a timer if no confirmation is entered within a specified time after a commit. This command is especially useful for situations when configuration is being modified remotely and the changes planned could potentially sever the remote connection permanently. Once the connection is severed and the commit can't be confirmed within the specified time the change is reverted and the remote connection should return to its original state.

```
admin@bule1# set system config-management commit-revisions
20
[edit]
admin@bule1# rollback
Possible completions:
<N>     Rollback to revision N (currently requires reboot)

Revisions:
    0 2018-04-02 21:07:10 admin by cli
```

Listing 8. The commands related to rollbacking to a previous version of the configuration.

Ubiquiti's routers also keep a history of the most recent commits to the configuration that you can use to rollback to an older version with one command. As shown in listing 8, the number of configurations kept in storage can be adjusted. If the software needs rollbacking, the default boot image can be switched as shown in listing 5 and then the router is rebooted by issuing the reboot-command.

3 Documentation

3.1 What should documentation include?

Good documentation should include everything that is relevant to managing and maintaining a network. However, there is no single clear answer to what is considered relevant, as it varies based on many variables. A big variable is the size of the network being documented – a small network of only a few devices doesn't require as thorough documentation as larger networks.

There are a few key things that even a minimalistic documentation should include. A network diagram is one of those. For smaller networks a single diagram can be used with all the info put into it, whereas larger networks may want to separate different OSI layers to separate diagrams. Most diagrams include a layer 1 (physical) and layer 2 (data link) diagram, which often may be combined into one diagram showing all the physical connections between devices, and a layer 3 diagram, which shows routing and any info related to it such as VLANs and logical interfaces. If necessary, additional diagrams going even deeper in the OSI model may be used to show, for example, how applications interact with servers.

Another key information is how to access the network devices, both logging in remotely as well as physically. Remotely logging in to a network device requires that the user knows the username and password as well as the management IP address. Additionally, the network being accessed may have ACLs or a firewall blocking attempts from any other networks than the ones specified. Physical location information of the devices may be required if power supplies need to be checked or a console connection to the device is needed. Accurate rack location information is especially crucial if the devices are located in a third party's premises, such as a colocation center abroad.

Other useful information may include, but is not limited to, who to contact in case of warranty or technical support cases, an inventory of network devices and their serial numbers, how long licensing is valid for and who to contact for renewal, IP allocation spreadsheets if they're not clear from the network diagrams, and the WAN circuit ID as well as ISP contact information in case of issues with the circuit.

3.2 Drawing a network diagram

There are no set rules for how to create a network diagram. Everyone has their own style, usually first learned by using someone else's complete diagram as a base and then creating new diagrams in the future using similar ideas.

However, there are several unwritten rules that most network diagrams follow. Routers are marked with a cylinder that has four arrows on top. Switches are marked with a cube that has arrows on it, but the size of the cube and the placement of the arrows varies

depending on the role of the switch – L3 switches, such as core switches, are very different from regular L2 switches and thus may be marked differently. VLANs and subnets are often marked with a tubular icon or a line, with the network and subnet mask written in the tube or along the line. Unknown big networks, for example the ISP network between two offices of a company, are marked with a cloud icon with the known details written inside the cloud.

3.3 Metropolia lab-network's documentation

3.3.1 Network diagrams

First step for the thesis was to create a diagram of the network as it was with only IPv4 configured. This is to assist in designing the IPv6 network.

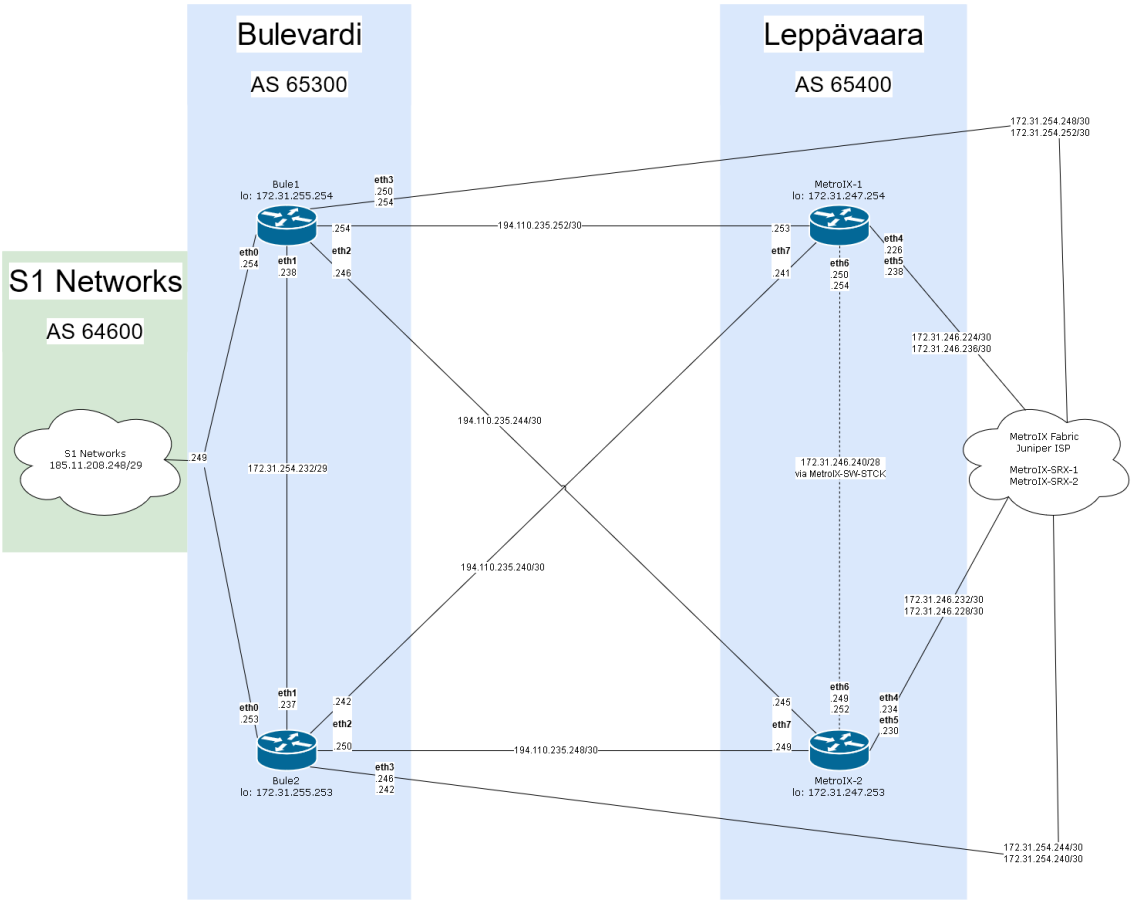


Figure 4. Network diagram of Metropolia lab IPv4 network.

As can be seen in figure 4, the network in question is relatively small with just four Ubiquiti EdgeRouter PROs and a few other devices that were not a part of the thesis, which are marked with a cloud to the right of the diagram. As such, it was decided that there was no need for a separate network diagram to show layers 1 & 2. Instead a single network diagram was created, a hybrid of sort showing all L1-L3 layers in one.

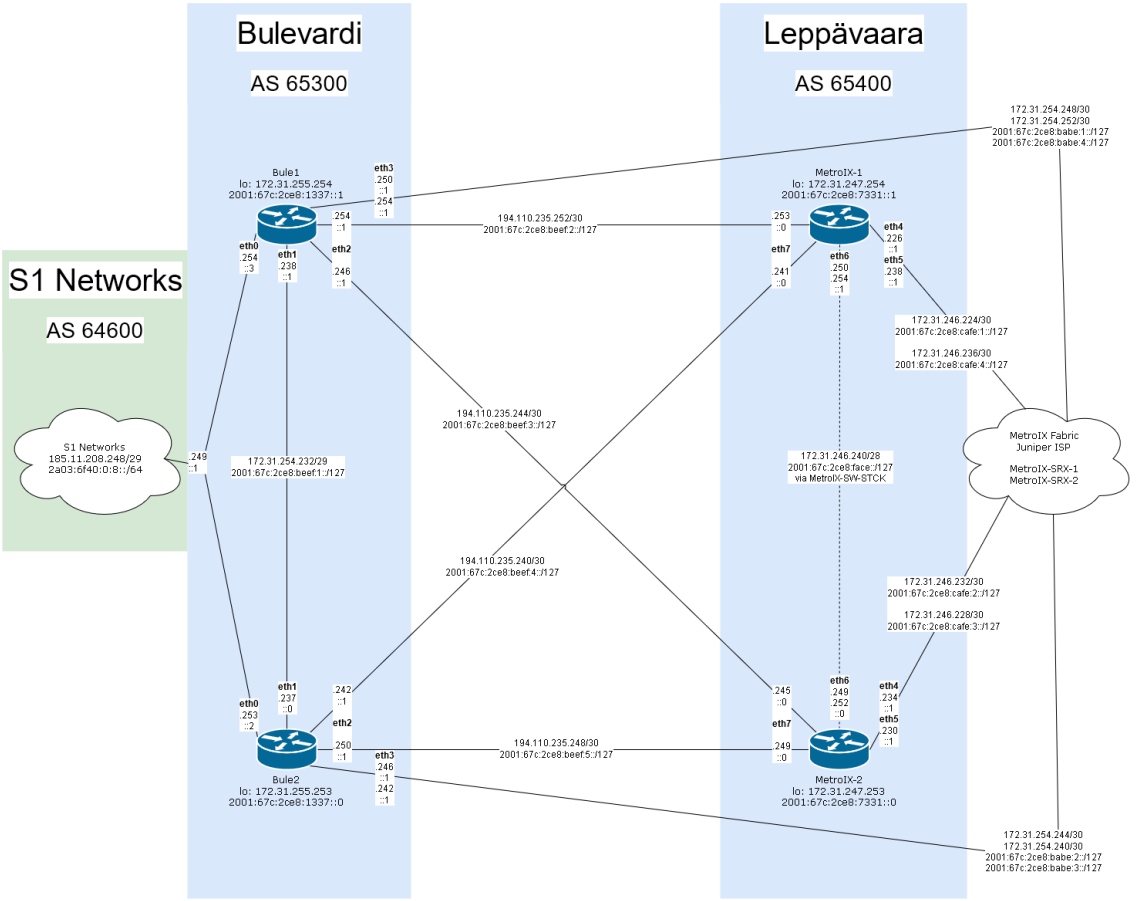


Figure 5. Network diagram of Metropolia lab network with IPv6 added.

Figure 5 illustrates the addition of IPv6 addresses to the network diagram. Each line, or a link, has its own IPv4 and IPv6 subnets and all router interfaces show the host address assigned to them. All four routers have multiple links to each other to provide redundancy. In case one router from each site crashes or is otherwise unusable, the network will still operate normally.

The IPv6 address block assigned to Metropolia is so big, a /48-block, that subnetting IPv6 to conserve addresses is not sensible. As such, the subnets were decided with the relative ease of reading in mind. IPv6 addresses are in hexadecimal, which includes

numbers from 0 to 9 and letters from A to F. Remembering words is easier than remembering numbers, and with hexadecimals it is possible to write words that only use the first 6 letters of the alphabet. Words such as cafe, face, and beef were used in this thesis. IPv6 addressing is talked about in more detail during chapter 4.

3.3.2 Access to routers

Most of the time accessing the routers remotely is enough. There are, however, access lists and a firewall blocking access from unknown sources and only Metropolia's address spaces are allowed. Metropolia provides a shell service to students which can be used to proxy a connection from a remote location. Then connecting to the routers is as simple as connecting to the routers' addresses from the shell using the credentials configured on the devices. For security reasons, the credentials will not be listed in this thesis, but the network department lecturers should have the necessary information.

Should the login credentials be forgotten, the router will need to be reset to factory-defaults and then configured from a backup. If no external backups are available, the device itself should still have its previous configuration saved in a file after the factory reset has been performed.

```
cat /root.dev/w.<random string>/config/config.boot
```

Listing 9. The command to access the previous boot-up configuration after a factory reset.

The command shown in listing 9 can be used to recover most of the previous configuration. [8.] Obviously, any encrypted data, such as passwords, can't be recovered this way and needs to be replaced in the new configuration.

The routers are physically located in both Bulevardi and Leppävaara's campuses, two routers each. In Bulevardi they're located in classrooms on the 2nd floor of Uusi Kemia building. In Leppävaara, the routers are in a separate server room in the networking wing. In both cases a key separate from the electronical access keys is required to access the routers. This is to prevent unauthorized access to the devices physically, as one can wipe the configuration simply by having access to the power and reset buttons. After resetting configuration, it is possible, as noted above, to recover most of the configuration and that information could be used for malicious purposes.

If console access is required to the routers, it should be noted that the serial connection's speed should be increased from the usual default value of 9 600, on clients such as PuTTy, to 115 200 – otherwise the console connection will not open.

For security reasons it is also important to always remove any default usernames, or at the very least change the usernames' passwords to something else. Removing all default information should also be considered and handled on a case by case basis. Default IP addresses are usually changed by necessity as the default IPs do not match the design plan.

3.3.3 Troubleshooting and support

In Ubiquiti's case there is no separate TAC that could be contacted in case of any issues with the configuration. The only official option is to post on their forums, where a Ubiquiti employee may comment on the issue at hand or other users may share their solutions, if they've run into similar problems and resolved them. There is no guarantee of a solution, however, and it is closer to the best effort approach.

For that reason, it is a good idea to take advantage of any configuration options assisting in troubleshooting potential problems.

```
admin@bule2# set system config-management commit-archive location
Possible completions:
  <uri>         Uniform Resource Identifier

Detailed information:

    "scp://<user>:<passwd>@<host>/<dir>"
    "ftp://<user>:<passwd>@<host>/<dir>"
    "tftp://<host>/<dir>"

[edit]
admin@bule2# set system syslog host 1.1.1.1 facility all level
alert    crit     debug    emerg    err      info     notice   warning
```

Listing 10. Commands for archiving commits and pushing logs to an external log server.

The command for archiving a backup of the configuration after every commit, and the command for pushing syslog entries to an external server are shown in listing 10. Currently in the MetroIX network there is no server taking scheduled backups nor are the syslog or commits pushed from the routers to a server. The advantage of exporting syslog to another server is that in case the router is inaccessible for one reason or another, the syslog may provide a clue as to why the issue has happened, and as the router is

rebooted the syslog within the router is wiped out. The external server on the other hand will keep its copy of the syslog for as long as it doesn't run out of disk space, or if the older logs are automatically deleted after a certain amount of time and that time has not expired. The reason why having a recent backup of the configuration is important is that the router may have a hardware or a configuration issue that prevents it from booting up properly, and the only option may be to replace the whole router. This possibility is easy to overlook, especially when the router keeps an automatic archive of the most recent commits if configured to do so.

4 Internet Protocols

4.1 IPv4

4.1.1 About IPv4

Internet Protocol version 4 was developed in the late 1970s and early 1980s to continue improving the earlier versions, defined in RFC791. It is used in packet-switching computer networks and provides the data blocks called datagrams that tell the source and destination, which are hosts identified by fixed length addresses, of the packets. [9.] IPv4 is still extremely widely used and the core of how present-day internet works.

RFCs were originally, as the name 'request for comments' implies, suggestions rather than decisions meant to encourage discussion among researchers. Since then they have evolved into more official documentation, including but not limited to defining standards on how internet-connected systems and applications should work. RFCs are never edited after publication and have thus become a sort of an archive of how the internet has evolved. [10.]

4.1.2 What is an IPv4 address and what are they used for?

Every host connected to a TCP/IP network, whether that network is a home LAN or the internet, needs to have an IP address to be able to communicate with other hosts, such as web servers. IPv4 addresses are 32 bits long and divided into four equal length groups, called octets due to the 8 bits per octet. The representation is called a dotted-decimal notation. The name stems from the fact that the addresses are usually shown in

their decimal format, rather than binary as decimals much easier for humans to understand and remember, and the octets are separated by a dot. [11, p. 86–87.] All hosts belonging to the same network should not be separated by a router.

Each octet is 8 bits there are 2^8 or 256 numbers, and since counting starts from 0 meaning each octet can have a number from 0 to 255, for example 1.2.3.4. With all four octets having 2^8 possibilities, the whole IPv4 address space is 2^{32} or 4 294 967 296 addresses. Many of those addresses are reserved for special purposes so in reality the available address space is much smaller.

The address space is subdivided into classes. The first octet tells which class an IP address belongs to. Class A networks, which cover almost half of the IPv4 address space, begin with a number between 1–126. Class B networks cover a fourth of the address space and their addresses begin with a number between 128–191. Lastly, class C networks begin with 192–223 and cover an eighth of the address space. [11, p. 88.] Each class is a different size to better fit different needs. Class A is by far the largest with 2^{24}-2 addresses but there are only 126 of class A networks. Addresses belonging to the same class A network all begin with the same first octet. Class B networks have 16 384 networks of 2^{16}-2 addresses each, and all networks belonging to the same class B network have the same first and second octet. Class C networks are the smallest with 2^8-2 addresses each, but on the flipside, there are most of them with 2 097 152 networks. Addresses in the class C network share the first three octets. The reason why there are always two addresses deducted from each classful network is because the first and the last address is reserved and can’t be used for anything else.

While the classful networks are of varying sizes to better fit how big a network is needed, there are a lot of addresses wasted if a network needs just slightly above a certain class’ addresses or if a network only requires 2 addresses for a point-to-point link. For example, a class C network wastes 252 addresses when configured for a link between two routers. To waste fewer addresses, subnetting is used. Subnetting, short for subdivided network, basically means dividing a network into two or more smaller networks.

Subnets are marked with their own 32-bit address divided into four octets, same as IPv4 addresses, that is called a subnet mask or netmask. A class B network 172.16.0.0 would have a subnet mask of 255.255.0.0, which can be denoted as 172.16.0.0/16 due

to the first 16 bits being 1 when the subnet mask is translated into binary, and it's often called a prefix. The network can then be divided into smaller subnets such as 172.16.1.0/24 and 172.16.2.0/24 resulting in two networks of 254 addresses each. Subnetting has the benefit of being able to assign a fitting network size to any situation for more efficient usage.

4.1.3 Running out of addresses

Efficient usage is even more important now as the internet has grown much larger than originally anticipated and the world is running out of IPv4 addresses. Originally when IPv4 was developed and certain address spaces were assigned for organizations and companies to use, the number of devices connected was much smaller and it was thought that 4,3 billion addresses were more than enough. Thus, IP address blocks were assigned quite generously. Already in the late 1980s, however, it was realized that the internet was growing at a much faster rate and the addresses wouldn't last very long after all.

There are a few ways to combat the exhaustion of IPv4 addresses, and one of the more important ways are private address ranges. In RFC1918, three blocks totaling roughly 18 million addresses were reserved for private address space. One class A network (10/8), 16 class B networks (172.16/12), and 256 class C networks (192.168/16). These addresses can freely be used within private networks without an approval from the internet registries. [12.] In contrast, most of the rest of the class A, B, and C address spaces are public IP addresses, with several exceptions, assigned to organizations and companies by an internet registry. Private addresses need to be translated using NAT to a public address if they want to communicate in the public internet, and this way a very large network can be hidden in theory behind a single address. Practically there are much less private addresses masked behind any single public address as many resources do not need to communicate with the public internet resources at all.

Public IP addresses are typically assigned to ISPs and other organizations by a regional internet registry, or RIR for short. The RIR operating in Europe is called Réseaux IP Européens Network Coordination Centre (RIPE NCC), and they're in charge of Europe, Middle East and parts of Central Asia. [13.] RIPE and other RIRs get their IP address blocks assigned by IANA.

IANA has reserved another IP address space, specifically 100.64/10, to be used for carrier-grade NAT, which is another way of prolonging IPv4 usage. It is used by carriers similarly to organizations and individuals use the private address space defined in RFC1918. In theory ISPs could use the same private address space but it might cause problems as the customers may be using the same addresses in their internal network. [14.] Using carrier-grade NAT service providers would assign an address from the reserved pool to the customers to act as gateways, and would later NAT the traffic in their own network to a public address.

The most significant tool to fight IPv4 exhaustion is IPv6, version 6 of Internet Protocol that uses a much, much bigger address space. So big, in fact, that “we could assign an IPV6 address to EVERY ATOM ON THE SURFACE OF THE EARTH, and still have enough addresses left to do another 100+ earths.” [15.]

4.2 IPv6

4.2.1 Brief history of IPv6

Internet Protocol version 6 was developed in the 1990s. It was defined in RFC2460 in 1998 as a draft standard. It wasn’t officially standardized until RFC8200 in 2017, despite having been widely used before the standardization. However, the new RFC8200 is simply a combination of the original draft as well as several other RFCs developed over the years that improved upon the original, and as such doesn’t change much for the already configured IPv6 setups. [16.]

Version 6 was developed due to the realization of IPv4 address space not being able to support the extremely rapid growth of the internet and devices connected to it. The growth was accelerated even further by the invention of the World Wide Web and its release to the public in 1991 allowing the commercialization of the internet. While the main motivation was increasing the address space there were also other improvements and new features developed as a result of what was learned from using IPv4.

4.2.2 What is an IPv6 address and how does it differ from IPv4?

The most noticeable difference to IPv4 is the format and length of the IP address. IPv6 addresses are 128 bits in length and they're written as eight groups of 16 bits each, as opposed to the 32 bits of IPv4 addresses divided into four groups of 8 bits each. Where IPv4 address space is roughly 4,3 billion, or more accurately 2^{32}, addresses, the IPv6 address space is 2^{128} or roughly $3{,}8 \times 10^{38}$ addresses. Due to how long the address would be if it was written with dotted-decimal notation like IPv4, IPv6 addresses are instead written in hexadecimal. Each group is separated by a colon (:).

Additionally, for more convenience any leading zeroes can be removed from the address and if there are consecutive sections of nothing but zeroes they can be replaced with a double colon instead. The double colon may only be used once per address as otherwise, depending on the address, it may be impossible to tell how many sections each double colon was originally. Thus, an example address 2a03:6f40:0000:0008:0000:0000:0000:0001/64 can be shortened to 2a03:6f40:0:8::1/64 (MetroIX gateway-address provided by S1 Networks) instead. [11, p. 680–681.]

Prefixes in IPv6 works somewhat similarly to IPv4. Using the above example host address, it belongs to the 2a03:6f40:0:8::/64 prefix. [11, p. 683-684.] This means all addresses in that subnet have the same leading four sections leaving the remaining four sections for use on hosts. This totals 2^{64} available addresses to be used on hosts, such as servers, computers, or interfaces of routers and other network devices.

Just like IPv4, there are many reserved address spaces for special uses. In IPv6 a public and a private address are referred to as global unicast and unique local addresses, respectively. Public address blocks, or global routing prefixes as they are called in IPv6, are assigned to companies and ISPs by the RIRs managing that geographical area. The global routing prefixes can then be broken down into smaller subnets, just like in IPv4, by the company or ISP who owns that block. If a network does not have any need to connect to the internet, an option is to use the reserved space for private addresses that are freely usable without a separate permission from a registry, similarly to IPv4. [11, p. 693.]

Another difference between version 4 and version 6 includes the protocols that use the IP addresses, such as routing protocols. As the IPv6 packets are vastly different to IPv4

packets, routing protocols needed to be updated to understand the new packets and how to forward them correctly. [11, p. 679.]

4.2.3 Importance of IPv6 implementation

The number of devices connected to the internet is increasing at ever faster speeds, with mobile phones, computers, IoT devices and other equipment becoming more and more common. IANA allocated its last /8-block of IPv4 addresses back in 2011 to the RIRs, and some of the RIRs have already run out of addresses as well, such as RIPE in 2012 and North America's ARIN in 2015. [11, p. 675; 17.]

New organizations and ISPs operating in areas managed by a RIR that has run out of addresses may not be able to get IPv4 addresses. However, there may be a few options depending on the RIR. The applicants can, for example, be put on a waitlist pending recalled IP address blocks due to non-payment, try to make do with IPv6, or try to buy an IP address from another organization. Thus, more and more new services might heavily rely on and only be available via IPv6 while they're looking into their options of obtaining an IPv4 address.

As such, implementing IPv6 is becoming more and more important. To encourage major ISPs and companies in enabling IPv6 connectivity and to promote discussion and awareness, the Internet Society organized a World IPv6 Launch day in 2012. [18.] According to statistics collected by Google, approximately only 20 % of the world has IPv6 connectivity currently. [19.] Most ISPs have started working on upgrading their network some years ago with varying degrees of completeness, and likely prioritizing their own backbone and internal networks as well as companies' circuits over individual homes.

In the beginning, most companies and organizations are going to implement IPv6 side-by-side IPv4 so that they're reachable by both versions. This is called dual-stacking, where the network is configured for both IPv4 and IPv6. Full IPv6 connectivity is unlikely to happen for many years to come despite the urgency. For any organization with a large network, implementing full IPv6 connectivity is going to take a lot of resources. There will be issues with implementation, sometimes due to the hardware being used, and troubleshooting all the various issues is very time consuming. Starting the process as early as possible is heavily recommended even if the rest of the world isn't ready. It is a question

of when, rather than if, the day when IPv6 will dethrone IPv4 as the primary Internet Protocol version.

4.3 Dynamic routing protocols

The primary job of L3 network devices, such as routers, is to act as gateways between different networks. To do that, they need to know the route to the destination network. While it is sort of possible to configure static routes everywhere it's extremely inefficient and complicated once the network grows beyond several devices. With the size of a network as large as the whole internet, this simply isn't feasible. Dynamic routing protocols were developed to solve routing. In dynamic routing protocols, routers advertise routes they know to a router configured as their neighbor and vice versa, so that the neighboring router knows what networks are behind the other router. If there are multiple routes to the destination, the router chooses the best one depending on factors that vary from one routing protocol to another.

Routing protocols can be divided into two categories. Interior gateway protocols (IGP) and exterior gateway protocols (EGP). As the names suggest, interior was designed to be used within a single autonomous system whereas exterior was designed to share routing information between different autonomous systems. Currently, BGP is the only EGP in use. [20, p. 173–174.]

Cisco's proprietary protocol RIP was the first popularly used dynamic routing protocol. Soon the demands for routing protocols grew, and new protocols were developed by early 1990s. [11, p. 437.] One of the protocols developed was OSPFv2, which is still very widely used. It is possibly the most common protocol for internal network routing due to it not being a proprietary and it is relatively easy to set up.

Both BGP and OSPF were used in this thesis and have IPv6 support. OSPF added IPv6 support in version 3, or OSPFv3 for short, whereas BGP is still using the same version 4 from the 1990s, but it has evolved to include IPv6 support as well as many other features. The current standard is defined in RFC4271 from 2006. [21.]

5 Implementing IPv6 in MetroIX

5.1 Configuring the routers

S1 Networks assigned a /48 global routing prefix to Metropolia, which was used to distribute subnets for all the links between the routers as well as reserve subnets for the links to the MetroIX Fabric. The first step was to look into the existing configuration, and try to imitate it as much as possible so as to not confuse anyone who would work on the setup in the future.

The configuration uses a lot of objects for variables like IP addresses whenever possible. The reason for that is the simplicity in configuring firewall rules, route-maps, and others. Should there be any changes to the networks or addresses, it's much easier to change the object rather than modify the address or network to every configuration line that uses it. The downside is, however, that it takes quite a while to understand all the objects, where are they being used and what do they include, as well as a lot of jumping from one place to another when mapping the configuration.

First step was to subnet the address block for all the point-to-point links between the routers. Up until several years ago, the best practice was to use /64 subnets even for point-to-point links. This recommendation was obsoleted in RFC6164, which argues that /127 subnets, which only have 2 addresses, are more secure as they do not leave unused host addresses for a possible attacker to use. Originally there was some concern that not all manufacturers' devices would support using /127 subnets for point-to-point links, but at this point they have had enough time to implement required changes in their software. [22.] In addition, while IPv6 address conservation is not strictly necessary, there is no reason not to practice it either. As a result of this RFC, the links are using /127 subnets for their point-to-point links as seen in the network diagram in figure 5.

Next step was to set up BGP peering. To understand how Ubiquiti's syntax for IPv6 peering works, it was decided to set up internal BGP session between Bulevardi's routers first. Once it was up and running at a satisfactory level the configuration could be replicated to the other routers. There were multiple issues along the way, outlined in more detail in the next chapter, but eventually the peering was up for all 4 routers and their peering sessions with each other.

Following the internal BGP configurations, it was time to ask S1 Networks for the peering info to set up IPv6 connectivity to the internet. They provided the gateway address as well as two hosts to be used on the Bulevardi routers. On the ISP's end, only the Metropolia's /48 block would be accepted – no subnets below that or any other networks altogether. This was achieved by configuring an IPv6 prefix-list containing only the Metropolia block, and attaching it to a route-map that was used to export routes towards the ISP. The same prefix-list was used to disallow advertising the /48 block back to the internal routers. Because the routers used Google's public free DNS servers as their IPv4 name-servers, Google's IPv6 DNS servers were used for IPv6 as well. [23.]

Once the internet's IPv6 routing table was also advertised by S1 Networks, the resource monitoring was checked. CPU spiked whenever the session was cleared or otherwise dropped, as often happens with BGP when it needs to recalculate routes, but outside of the initial spike both CPU usage and memory are holding well under any unsafe thresholds. Once more and more IPv6 networks are being implemented worldwide it remains to be seen how the Ubiquiti EdgeRouter PROs will handle the number of routes being advertised to them and re-advertised forward to MetroIX-Fabric and Leppävaara routers.

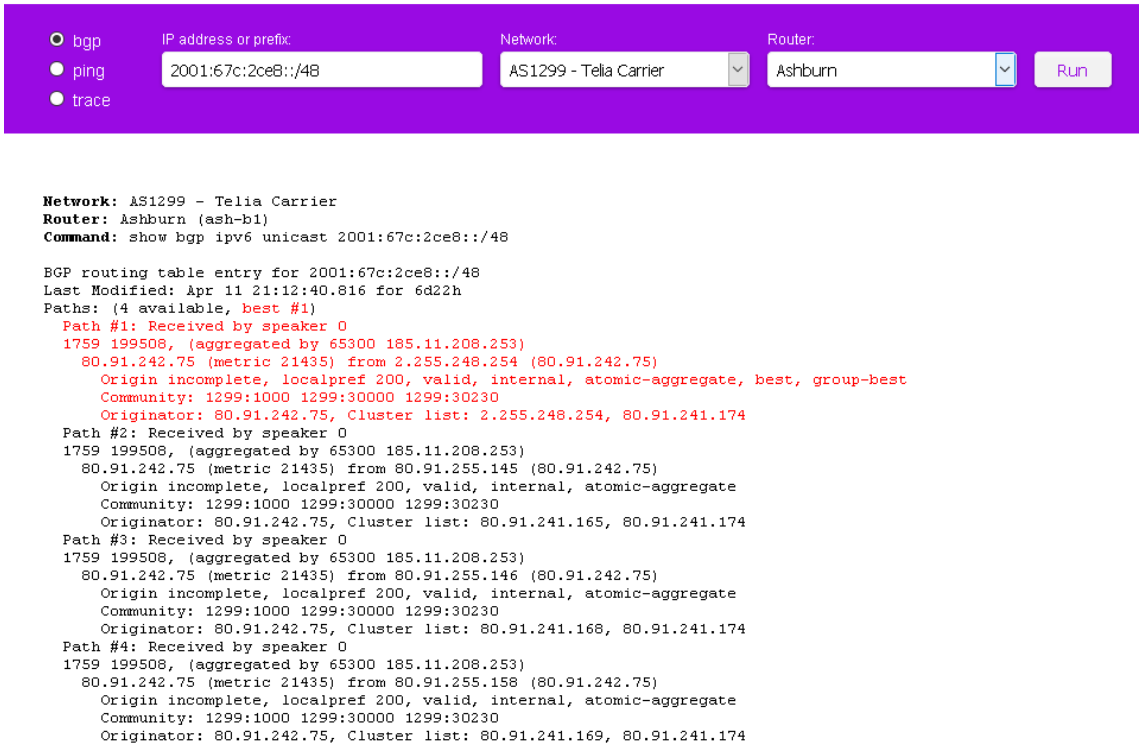


Figure 6. A screenshot of Telia's looking glass output showing Metropolia's global routing prefix being advertised.

The last step was to make sure all the configurations were consistent and that they were working correctly; all the peering sessions were up and running with routes being forwarded, testing that IPv6 domain names are resolved, and the addresses were reachable from the internet. These could be tested by, for example, pinging google.com using IPv6-ping from Leppävaara's routers. The Metropolia /48 global routing prefix being advertised was also checked by using a publicly accessible looking glass, in this case Telia's which is available at http://lg.telia.net/, as shown in figure 6.

5.2 Problems encountered

One major issue seemed to be the inconsistency on how the router handles certain objects, mainly IPv6 related. Especially BGP peer-groups caused a lot of issues with not applying the configuration when a peer-group was being used, but after using the same commands to configure the neighbor directly everything would work just as was to be

expected. This was especially troublesome to troubleshoot as randomly the peer-groups would work for some of the peers, but identical configuration apart from the peer address did not. Peering might not even come up at all despite the neighbor being reachable by ping and it being a point-to-point link.

Route-maps were also not always being applied and would sometimes require several restarts of the peering session before they worked as configured, regardless of whether or not they were applied via a peer-group. This was happening with the IPv6 neighbor sessions on both IPv4 as well as IPv6 route-maps.

To keep the configuration consistent, all IPv6 peering related configurations were applied without a peer-group. From management point of view this means slightly more work should any of the configurations change, but it is also much easier to read and understand the configuration if all four routers are configured in an identical way.

All in all, the IPv6 features do not seem to be as polished with the currently running software release v1.9.7+hf4 as their counterparts on IPv4 side. This may be due to the limited number of users having implemented IPv6 in their networks, and as such fewer bug reports have been reported to Ubiquiti. The problems encountered during this thesis have not been reported to Ubiquiti as the circumstances, when and how the problems occurred, are unclear. Simply stating something doesn't work is unhelpful to the manufacturer as they could be linked to other configurations on the router, and much more testing would be required to troubleshoot and limit the problem.

5.3 What is still missing?

Finishing the MetroIX IPv6 configuration on the MetroIX-Fabric devices is the biggest task. In addition to that, firewall rules and other configurations on the routers need to be finalized and refined. Another option is to rework the firewall concept completely by removing the configuration from the routers and instead setting up a dedicated firewall cluster in between the S1 Networks router and MetroIX network.

The original software upgrade failed due to a memory leak in BGP multipathing on the newer software. The issue has not been resolved yet, as Ubiquiti hasn't released a software with a fix. Once a software with the fix comes available, the routers need to be

upgraded once again, and BGP multipathing reverted to how it was configured, shown in listing 3, and monitored if it was indeed fixed.

Additionally, as the new software is installed on the routers, the issues with peer-groups and IPv6 should be looked into once again to see if there have been any improvements made to how the objects are handled. There may also be new beneficial features available, such as the watchdog additions in version 1.10.0 discussed in chapter 2.3.1.

References

1 Meltdown and Spectre [online]. URL: https://meltdownattack.com. Retrieved on April 4, 2018.

2 Semantic Versioning 2.0.0 [online]. URL: https://semver.org. Retrieved on April 5, 2018.

3 EdgeMAX EdgeRouter software release v1.8.5 [online]. June 14, 2016. URL: https://community.ubnt.com/t5/EdgeMAX-Updates-Blog/EdgeMAX-EdgeRouter-software-release-v1-8-5/ba-p/1591710. Retrieved on November 8, 2017.

4 eBGP multipath memory leak / EdgeRouter Pro v1.9.7+hotfix.1 [online]. August 22, 2017. URL: https://community.ubnt.com/t5/EdgeRouter/eBGP-multipath-memory-leak-EdgeRouter-Pro-v1-9-7-hotfix-1/m-p/2037492. Retrieved on November 8, 2017.

5 eBGP multipath memory leak / EdgeRouter Pro v1.9.7+hotfix.1 [online]. December 13, 2017. URL: https://community.ubnt.com/t5/EdgeRouter/eBGP-multipath-memory-leak-EdgeRouter-Pro-v1-9-7-hotfix-1/m-p/2172119/highlight/true#M189414. Retrieved on January 18, 2018.

6 EdgeMAX EdgeRouter software release v1.10.0 [online]. February 15, 2018. URL: https://community.ubnt.com/t5/EdgeMAX-Updates-Blog/EdgeMAX-EdgeRouter-software-release-v1-10-0/ba-p/2233263. Retrieved on April 6, 2018.

7 EdgeOS Firmware Changelog [online]. March 21, 2018. URL: https://dl.ubnt.com/firmwares/edgemax/v1.10.x/changelog.v1.10.1.txt. Retrieved on April 6, 2018.

8 Lost password? [online]. June 14, 2013. URL: https://community.ubnt.com/t5/EdgeRouter/Lost-password/m-p/490617/highlight/true#M10246. Retrieved on April 10, 2018.

9 Internet Protocol [online]. September 1981. URL: https://tools.ietf.org/html/rfc791. Retrieved on April 13, 2018.

10 Frequently Asked Questions [online]. URL: https://www.rfc-editor.org/faq/. Retrieved on April 13, 2018.

11 Odom, Wendell. 2016. CCENT/CCNA ICND1 100-105 Official Cert Guide. Indianapolis, IN: Cisco Press.

12 de Groot, G. J.; Karrenberg, D.; Lear, E.; Moskowitz, B. & Rekhter, Y. Address Allocation for Private Internets [online]. February 1996. URL: https://tools.ietf.org/html/rfc1918. Retrieved on April 13, 2018.

13 What We Do [online]. February 7, 2017. URL: https://www.ripe.net/about-us/what-we-do. Retrieved on April 14, 2018.

14 Azinger, M.; Donley, C.; Kuarsingh, V.; Liljenstolpe, C. & Weil, J. IANA-Reserved IPv4 Prefix for Shared Address Space [online]. April 2012. URL: https://tools.ietf.org/html/rfc6598. Retrieved on April 14, 2018.

15 Leibson, Steve. IPV6: How Many IP Addresses Can Dance on the Head of a Pin? [online]. March 28, 2008. URL: https://www.edn.com/electronics-blogs/other/4306822/IPV6-How-Many-IP-Addresses-Can-Dance-on-the-Head-of-a-Pin-. Retrieved on April 15, 2018.

16 Siddiqui, Aftab. RFC 8200 – IPv6 has been standardized [online]. July 17, 2017. URL: https://www.internetsociety.org/blog/2017/07/rfc-8200-ipv6-has-been-standardized/. Retrieved on April 15, 2018.

17 van Beijnum, Iljitsch. Europe officially runs out of IPv4 addresses [online]. September 14, 2012. URL: https://arstechnica.com/information-technology/2012/09/europe-officially-runs-out-of-ipv4-addresses/. Retrieved on April 15, 2018.

18 World IPv6 Launch [online]. URL: http://www.worldipv6launch.org/. Retrieved on April 10, 2018.

19 Statistics [online]. April 13, 2018. URL: https://www.google.com/intl/en/ipv6/statistics.html#tab=ipv6-adoption. Retrieved on April 15, 2018.

20 Odom, Wendell. 2017. CCNA Routing and Switching ICDN2 200-105 Official Cert Guide. Indianapolis, IN: Cisco Press.

21 Hares, S.; Li, T. & Rekhter, Y. A Border Gateway Protocol 4 (BGP-4) [online]. January 2006. URL: https://tools.ietf.org/html/rfc4271. Retrieved on April 12, 2018.

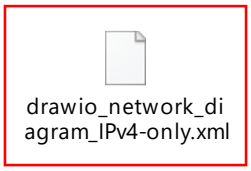
22 Bush, R.; Colitti, L.; Kohno, M.; Matsuzaki, Y.; Narten, T. & Nitzan, B. Using 127-Bit IPv6 Prefixes on Inter-Router Links [online]. April 2011. URL: https://tools.ietf.org/html/rfc6164. Retrieved on April 18, 2018.

23 Configure your network settings to use Google Public DNS [online]. June 3, 2016. URL: https://developers.google.com/speed/public-dns/docs/using. Retrieved on April 3, 2018.

Appendix 1. Draw.io network diagrams

To import the diagrams into the editor, all that's needed is to navigate to https://www.draw.io, choose an existing diagram and open the XML files from below.

IPv4 network diagram:



IPv6 network diagram:

Files can be exported from draw.io in multiple formats, including an MS visio document (beta version at the time of writing this thesis), by going to *File -> Export As* and choosing the most fitting option.