



Open Networks

Author:
Diarmuid Ó Briain



FREE
TECHNOLOGY
ACADEMY



Diarmuid Ó Briain

Diarmuid is a Chartered Engineer (CEng) with experience in Telecommunications, Information Networking and Security. He has designed and implemented next-generation networks and information security solutions for major multinational communications companies as well as an Irish Internet Service Provider. He has also lectured on Telecommunications and Computing programmes at the Lifelong Learning Department of the Institute of Technology, Carlow (ITC) in Ireland.

Institiúid Teicneolaíochta Cheatharlach



Second edition: April 2015

© Diarmuid Ó Briain

All rights are reserved

© of this edition, Diarmuid Ó Briain

Copyright © 2015 Diarmuid Ó Briain.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

ISBN-13: 978-1512135145

ISBN-10: 1512135143

v2.0.3

Preface

Software has become a strategic societal resource in the last few decades. The emergence of Free Software, which has entered in major sectors of the Information ICT market, is drastically changing the economics of software development and usage. Free Software – sometimes also referred to as “Open Source” or “Libre Software” – can be used, studied, copied, modified and distributed freely. It offers the freedom to learn and to teach without engaging in dependencies on any single technology provider. These freedoms are considered a fundamental precondition for sustainable development and an inclusive information society.

Although there is a growing interest in free technologies (including Free Software and Open Standards), still a limited number of people have sufficient knowledge and expertise in these fields. The FTA attempts to respond to this demand.

Introduction to the FTA

The Free Technology Academy (FTA) is a joint initiative from several educational institutes in various countries. It aims to contribute to a society that permits all users to study, participate and build upon existing knowledge without restrictions.

What does the FTA offer?

The Academy offers an online master level programme with course modules about Free Technologies. Learners can choose to enrol in an individual course or register for the whole programme. Tuition takes place online in the FTA virtual campus and is performed by teaching staff from partner universities. Credits obtained in the FTA programme are recognised by these universities. See <http://ftacademy.org/courses/recognition>

Who is behind the FTA?

The FTA was initiated in 2008 supported by the Life Long Learning Programme (LLP) of the European Commission, under the coordination of the Free Knowledge Institute (FKI) and in partnership with three European universities: Open Universiteit Nederland (The Netherlands), Universitat Oberta de Catalunya (Spain) and University of Agder (Norway).

Since 2012 the FTA has continued hosted by the Free Knowledge Institute in collaboration with its Partner Network and community of volunteers. <http://ftacademy.org/about/partners>

For who is the FTA?

The Free Technology Academy is specially oriented to IT professionals, educators, students and decision makers.

What about the licensing?

All learning materials used in and developed by the FTA are Open Educational Resources, published under copyleft free licenses that allow them to be freely used, modified and redistributed. Similarly, the software used in the FTA virtual campus and in its courses is Free Software and is built upon an Open Standards framework.

Evolution of this book

The original course was developed by Enric Peig Olivé of the Universitat Oberta de Catalunya (UOC) together with LibreSoft staff from the Universidad Rey Juan Carlos. Since its publication in 2010 there have been significant changes in networking and this Second version of the course aims to address this change.

Participation

Users of FTA learning materials are encouraged to provide feedback and make suggestions for improvement. A specific space for this feedback is set up on the FTA website. These inputs will be taken into account for next versions. Moreover, the FTA welcomes anyone to use and distribute this material as well as to make new versions and translations.

See for specific and updated information about the book, <http://ftacademy.org/materials/fsm/3>. For more information and enrolment in the FTA online course programme, please visit the Academy's website: <http://ftacademy.org/>. I sincerely hope this course book helps you in your personal learning process and helps you to help others in theirs. I look forward to see you in the free knowledge and free technology movements!

Happy learning!

Wouter Tebbens

President of the Free Knowledge Institute

Table of Contents

1. Networking Device.....	11
1.1 Introduction.....	11
1.1.1 Minibian.....	12
1.1.2 Add SWAP partition.....	15
1.1.3 Update and upgrade.....	15
1.1.4 Set-up users.....	15
1.1.5 Set-up locale.....	16
1.1.6 Set-up hostname.....	16
1.2 mactelnet.....	17
1.2.1 Install mactelnet on GNU/Linux.....	17
1.2.2 Create user.....	17
1.2.3 Set default IP address.....	18
1.2.4 Running a server on your GNU/Linux Server.....	18
1.2.5 Accessing the server.....	18
1.2.6 mactelnet frames.....	19
1.3 Additional Interface.....	21
1.3.1 DHCP request on new interface.....	21
1.3.2 Default gateway to eth2.....	22
1.3.3 Test routing.....	22
2. Network Simulation.....	23
2.1 Getting started.....	23
2.2 Operating the CORE environment.....	24
3. Networking.....	27
3.1 Introduction to Network Administration.....	27
3.2 Introduction to TCP/IP (TCP/IP suite).....	28
3.2.1 Services on TCP/IP.....	29
3.2.2 What is TCP/IP?.....	30
3.3 Physical network devices (hardware).....	31
3.3.1 Copper.....	31
3.3.2 Fibre Optic.....	32
3.3.3 Wireless.....	33
3.4 GNU/Linux interface.....	34
3.5 TCP/IP Concepts.....	36
4. Switching.....	41
4.1 Bridging and Switching.....	41
4.1.1 Why use Bridges.....	42
4.1.2 Switches.....	42
4.1.3 Transparent Bridging.....	43
4.2 Spanning Tree Protocol.....	45
4.2.1 Configuration of a Bridge interface on GNU/Linux.....	48
4.2.2 Create a bridge and add interfaces.....	49
4.3 Virtual LANs (VLANs).....	49
4.3.1 Removing the Physical Boundaries.....	50
4.3.2 IEEE 802.1P/Q.....	51
4.4 Provider tagging.....	55

4.5 VLANs on GNU/Linux.....	57
4.5.1 IEEE 802.1ad support on GNU/Linux.....	59
4.5.2 IEEE 802.1ad support on GNU/Linux as a switch.....	60
4.6 GNU/Linux as a Service Provider bridge.....	61
5. Internet Protocol.....	63
5.1 GNU/Linux IP networking (iproute2).....	67
5.1.1 iproute2 ip command.....	68
5.1.2 Network Manager (network-manager).....	69
5.1.3 Check there is no configuration in /etc/network/interfaces.....	70
5.2 Network interfaces.....	71
5.2.1 Bring up the eth0 interface.....	71
5.2.2 Add IP Address to the eth0 interface.....	71
5.2.3 Confirm IP Address is configured.....	71
5.2.4 Add an IPv4 Default gateway.....	71
5.2.5 Add a static route.....	72
5.2.6 Confirm that the route has taken.....	72
5.3 Monitoring.....	72
5.4 Internet Protocol v6.....	72
5.4.1 Features of IPv6.....	73
5.4.2 IPv6 Address Architecture.....	74
5.4.3 IPv6 Address Scope.....	77
5.4.4 IPv6 Addressing Model.....	78
5.4.5 Loopback Address.....	78
5.4.6 IPv6 Packet Structure.....	79
5.4.7 Applications for IPv6.....	80
5.4.8 IPv6 EUI-64 host.....	82
5.4.9 IPv6 link-local.....	82
5.4.10 IPv6 Stateless Address Auto-configuration (SLAAC).....	82
5.4.11 IPv6 transition mechanisms.....	85
5.4.12 IPv6 Interior Gateway Routing.....	87
5.4.13 IPv6 Exterior Gateway Routing.....	87
5.4.14 IPv6 Configuration.....	87
6. Routing.....	91
6.1 Introduction to Routing.....	91
6.1.1 Standard Routing Model.....	92
6.1.2 Routing Tables.....	93
6.2 Open Shortest Path First (OSPF).....	93
6.2.1 OSPF Overview.....	94
6.2.2 Benefits of Using OSPF versus Distance Vector protocols.....	94
6.2.3 OSPF Concepts.....	94
6.2.4 SPF Algorithm.....	96
6.3 Quagga Introduction.....	102
6.4 Install Quagga.....	103
6.5 Configure the Quagga configuration files.....	103
6.5.1 debian.conf.....	103
6.5.2 vtysh.conf - the VTY terminal conf file.....	104
6.6 zebra.conf - the routing daemon conf file.....	104
6.6.1 The OSPFv2 (IPv4) daemon conf file.....	105

6.6.2 The OSPFv3 (IPv6) daemon conf file.....	105
6.7 Restart the Quagga service.....	105
6.8 Accessing the Quagga router for configuration.....	106
6.8.1 Access TCP Ports.....	106
6.8.2 Accessing the zebra daemon.....	106
6.9 Configuring zebra daemon - the routing daemon.....	107
6.10 Configuring the OSPFv2 daemon.....	109
6.11 Configure the OSPFv3 (for IPv6) daemon.....	110
6.12 Quagga Summary.....	112
7. Wireless LANs.....	113
7.1 Introduction to WiFi.....	113
7.1.1 Spectrum.....	114
7.1.2 IEEE 802.11 WLAN Summary.....	117
7.1.3 IEEE 802.11 MAC (Media Access Control).....	117
7.1.4 WiFi Elements.....	118
7.1.5 WiFi Security.....	119
7.2 Configuration of a WiFi network on GNU/Linux.....	124
7.2.1 Install the wireless-tools package.....	124
7.2.2 Using WPA2.....	126
7.2.3 WPA Supplicant.....	126
8. Virtual Private Networks (VPN).....	131
8.1 IPv4 OpenVPN tunnel.....	132
8.1.1 Server set-up.....	132
8.1.2 Client set-up.....	133
8.1.3 Run the OpenVPN Server.....	134
8.1.4 Connect with the OpenVPN client.....	134
8.2 IPv6 OpenVPN tunnel.....	136
8.2.1 OpenVPN Server - tap.....	136
8.2.2 OpenVPN Client - tap.....	137
8.3 SSH VPN.....	139
8.3.1 Set-up VNC Server ran as localhost only.....	139
8.4 VNC on the client side.....	140
8.5 SSH connection and VNC connection.....	141
9. IP Telephony.....	143
9.1 Audio Streams.....	144
9.2 Real-Time Transport Protocol.....	144
9.3 Delay.....	146
9.3.1 Network Delay.....	147
9.3.2 CODEC Latency.....	147
9.3.3 Jitter.....	148
9.3.4 Packet Loss.....	150
9.3.5 Voice Compression.....	150
9.4 CODEC.....	151
9.4.1 RTP Audio & Video Payloads.....	152
9.5 Other Voice Quality Factors.....	153
9.5.1 Silence Suppression.....	153
9.5.2 Echo.....	153
9.6 Voice Quality Measurements.....	153

9.6.1 P.800 MOS.....	153
9.6.2 P.861 PSQM.....	154
9.7 The SIP Protocol and Server Functions.....	154
9.7.1 Session Description Protocol.....	155
9.7.2 SIP Redirect (Proxy) Server.....	156
9.7.3 SIP Registrar.....	156
9.7.4 Location Server.....	156
9.7.5 User Agent Client (UAC).....	156
9.7.6 User Agent Server.....	157
9.7.7 SIP UA and Server Roles.....	157
9.7.8 SIP Multimedia Protocol Stack.....	157
9.7.9 SIP Commands and Responses.....	158
9.7.10 SIP Registration.....	160
9.7.11 SIP Call Setup.....	162
9.7.12 SIP Call Terminate.....	165
9.8 IPT and the PSTN.....	166
9.8.1 Softswitch.....	166
9.8.2 Call Agent.....	167
9.8.3 Media Gateway.....	167
9.9 MG Controllers.....	168
9.9.1 Signalling Gateway.....	168
9.10 Services.....	169
9.11 FOSS Implementations.....	170
9.12 Test network.....	171
9.12.1 Asterisk Server.....	171
9.12.2 SIP Softphone Client.....	176
9.12.3 SIP Phone.....	178
9.12.4 Configuring voice-mail.....	178
9.13 Testing the configuration.....	180
9.13.1 Registration test - IP Phone.....	181
9.13.2 Registration test - Softphone.....	183
9.13.3 Voice call test.....	186
9.13.4 Hangup test.....	194
9.14 Asterisk GUI.....	196
9.14.1 Install Asterisk-gui.....	196
9.14.2 Asterisk configuration files.....	198
9.14.3 Connect to the Asterisk Server GUI.....	199
9.15 Conclusion.....	200
10. IP Services.....	201
10.1 Configuration of inetd or xinetd.....	201
10.2 Other network services.....	204
10.2.1 Additional configuration: protocols and networks.....	204
10.2.2 Security aspects.....	205
10.2.3 IP Options.....	206
10.2.4 Commands for solving problems with the network.....	207
10.3 DHCP Configuration.....	207
10.3.1 DHCP Server.....	208
10.4 IP Masquerade.....	209

11. Software Defined Networking (SDN).....	211
11.1 Introduction.....	211
11.2 Software Defined Networking.....	211
11.3 SDN operation.....	212
11.3.1 Flow Tables.....	215
11.3.2 Group Tables.....	215
11.3.3 Meter Tables.....	215
11.4 SDN Controllers.....	216
11.5 SDN Applications.....	216
11.5.1 SDN Routing Service.....	216
11.6 Link Discovery Module.....	217
11.7 Topology Manager.....	217
11.8 Virtual Routing Engine (VRE).....	218
11.9 Using Mininet to experment with SDN.....	218
11.10 Set-up a guest VM with the mininet image.....	218
11.10.1 Add rights to wireshark for user.....	219
11.11 Confirm Wireshark works over SSH.....	220
11.12 Build a mininet test network.....	220
11.12.1 Exiting mininet.....	225
11.13 Configuring hosts.....	226
11.14 Configuring links.....	226
11.15 Reviewing OpenFlow traffic.....	227
11.16 Webserver test.....	233
11.17 Custom Topologies.....	236
11.17.1 Create custom topology.....	238
11.18 OpenDaylight.....	242
11.18.1 Install OpenDaylight.....	242
11.18.2 Running OpenDaylight.....	243
11.18.3 OpenDaylight User Experience (DLUX).....	243
11.18.4 Start mininet network.....	245
12. Networks Function Virtualisation (NFV).....	249
12.1 Providing NFV to the customer.....	251
12.2 NFV Standards.....	252
12.3 OPNFV.....	253
12.4 Conclusion.....	254
13. Abbreviations.....	255
14. Bibliography.....	263
15. GNU Free Documentation License.....	269

This page is intentionally blank

1. Networking Device

1.1 Introduction

In order to participate on this course it will be necessary to have a number of networking devices for labs. To keep costs to a minimum I recommend the use of the versatile Raspberry Pi. Here is a tool-kit I put together for a single networking device. I used a Raspberry Pi 2 Starter Kit which includes:

- Raspberry Pi 2 Model B (Quad Core, 1GB RAM)
- Sandisk Ultra Class 10 MicroSD
- The PiHut UK 5V 2A Power Supply
- Black Case
- High-Definition Multimedia Interface (HDMI) cable
- Ethernet Cable
- Ethernet USB adaptors



For the initial set-up a screen with a HDMI interface or with a HDMI - Video Graphics Array (VGA) adaptor.

1.1.1 Minibian

For the networking device required for this course it is not necessary to have the full *Rasbian* Operating System (OS) with graphical interface. Download the latest version of *Minibian*. *Minibian* is a Debian based OS without a graphical interface. Install it on the MicroSD card as follows:

```
$ ls
2015-02-18-wheezy-minibian.tar.gz

$ tar -xzvf 2015-02-18-wheezy-minibian.tar.gz
2015-02-18-wheezy-minibian.img

$ ls
2015-02-18-wheezy-minibian.img  2015-02-18-wheezy-minibian.tar.gz
```

Put the microSD card in a slot on your computer (may have to use an SD/microSD adaptor). The *lsblk* command will show the devices on the system. The microSD card on the system is */dev/mmcblk0*.

```
$ lsblk
NAME                                MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
sda                                  8:0    0 465.8G  0 disk
├─sda1                               8:1    0   243M  0 part  /boot
├─sda2                               8:2    0     1K  0 part
├─sda5                               8:5    0 465.5G  0 part
│   └─sda5_crypt (dm-0)             252:0    0 465.5G  0 crypt
│       ├─mint--vg-root (dm-1)     252:1    0 457.9G  0 lvm    /
│       └─mint--vg-swap_1 (dm-2)  252:2    0   7.6G  0 lvm    [SWAP]
sr0                                  11:0    1  1024M  0 rom
mmcblk0                             179:0    0   59.5G  0 disk
└─mmcblk0p1                         179:1    0   59.5G  0 part  /media/alovelace/3866-363
```

Unmount the device, check it is unmounted and all is ready to copy the *minibian* image to it.

```
$ sudo umount /dev/mmcblk0p1

$ mount | grep mmcblk0p1
```

It is a good idea to remove the default partition that is on the microSD first.

```
$ sudo gdisk /dev/mmcblk0
```

```
Command (? for help): d
Using 1
```

```
Command (? for help): w
```

```
Final checks complete. About to write GPT data. THIS WILL OVERWRITE
EXISTING PARTITIONS!!
```

```
Do you want to proceed? (Y/N): Y
```

```
OK; writing new GUID partition table (GPT) to /dev/mmcblk0.
The operation has completed successfully.
```

```
Install the Minibian image to the microSD drive. Be careful to use the
device (mmcblk0) and not the partition name (mmcblk0p1).
```

```
$ sudo -s
```

```
# dd bs=4M if='2015-02-18-wheezy-minibian.img' | pv | dd of=/dev/mmcblk0
```

```
250MB 0:00:38 [9.67MB/s] [          <=>          ]
999424+0 records in
999424+0 records out
511705088 bytes (512 MB) copied, 63.9125 s, 8.0 MB/s
```

where:

- `/dev/mmcblk0` is the microSD Drive
- `pv`- allows for the monitoring of data through the pipe.

You can use a tool like *gparted* to realign the partitions and add SWAP space.

Partition	File System	Size	Used	Unused	Flags
/dev/mmcblk0p1	fat16	47.71 MiB	14.31 MiB	33.40 MiB	
/dev/mmcblk0p2	ext4	57.42 GiB	2.12 GiB	55.30 GiB	
/dev/mmcblk0p3	linux-swap	2.01 GiB	0.00 B	2.01 GiB	

```
$ sudo gdisk -l /dev/mmcb1k0
```

```
GPT fdisk (gdisk) version 0.8.8
```

```
Partition table scan:
```

```
  MBR: MBR only
  BSD: not present
  APM: not present
  GPT: not present
```

```
*****
Found invalid GPT and valid MBR; converting MBR to GPT format
in memory.
*****
```

```
Warning! Main partition table overlaps the first partition by 18 blocks!
Try reducing the partition table size by 72 entries.
(Use the 's' item on the experts' menu.)
```

```
Warning! Secondary partition table overlaps the last partition by
33 blocks!
You will need to delete this partition or resize it in another utility.
Disk /dev/mmcb1k0: 124735488 sectors, 59.5 GiB
Logical sector size: 512 bytes
Disk identifier (GUID): 5F7BE4E9-8717-4B7C-A899-BE3ECEED20EA
Partition table holds up to 128 entries
First usable sector is 34, last usable sector is 124735454
Partitions will be aligned on 16-sector boundaries
Total free space is 576 sectors (288.0 KiB)
```

Number	Start (sector)	End (sector)	Size	Code	Name
1	16	97727	47.7 MiB	0700	Microsoft basic data
2	98304	120520703	57.4 GiB	8300	Linux filesystem
3	120520704	124735487	2.0 GiB	8200	Linux swap

Install in the Raspberry Pi and boot. On first boot it maybe necessary to do a file system check with the *fsck* utility, The Raspberry Pi will look for a password. The root password on the image is *raspberrypi*.

```
Give the root password for maintenance
(or type Control-D to continue): raspberrypi
```

```
root@raspberrypi:~# fsck
....
```

```
Fix<y>? yes
```

Reboot the device.

```
root@raspberrypi:~# init 6
```

```
Rasbian GNU/Linux 7 raspberrypi tty1
```

```
raspberrypi login: root
Password: raspberrypi
```

1.1.2 Add SWAP partition

While the SWAP partition was added to the microSD card, it is not included in the SWAP space on the Raspberry Pi. The *mkswap* utility sets up a Linux swap area on a device or in a file.

```
$ sudo mkswap /dev/mmcblk0p3
Setting up swapspace version 1, size = 2107388 KiB
no label, UUID=f94b3d6b-da3c-4392-b252-b4bf84b2c810
```

The *swapon* utility specifies that the */dev/mmcblk0p3* device is part of the system SWAP space.

```
$ sudo swapon /dev/mmcblk0p3
```

Confirm that the new SWAP partition has indeed been added to the available SWAP space.

```
$ cat /proc/swaps
Filename                                Type              Size              Used              Priority
/dev/mmcblk0p3                          partition         2107388          0                 -1
```

Now make the addition of this SWAP space persistent.

```
$ sudo -s
# cat << FSTAB >> /etc/fstab

# Add lines to mount /dev/mmcblk0p3 as a SWAP partition on boot
/dev/mmcblk0p3          none      swap      sw        0         0

FSTAB
```

1.1.3 Update and upgrade

Upgrade and update from the repositories.

```
root@raspberrypi:~# apt-get upgrade
root@raspberrypi:~# apt-get update
```

1.1.4 Set-up users

Change the root password from *raspberry* and create a working user *Ada Lovelace*.

```
root@raspberrypi:~# passwd
Enter new UNIX password: newpass
Re-enter new UNIX password: newpass

root@raspberrypi:~# useradd -m -c "Ada Lovelace" -s /bin/bash alovelace

root@raspberrypi:~# passwd alovelace
Enter new UNIX password: countess
Re-enter new UNIX password: countess
```

Add *Ada Lovelace* to the sudoers group.

```
root@raspberrypi:~# apt-get install sudo
```

```
root@raspberrypi:~# vi /etc/group
```

Add the user *alovelace* to the group *sudo*

```
...
sudo:x:27:alovelace
...
```

1.1.5 Set-up locale

Update the keyboard if necessary.

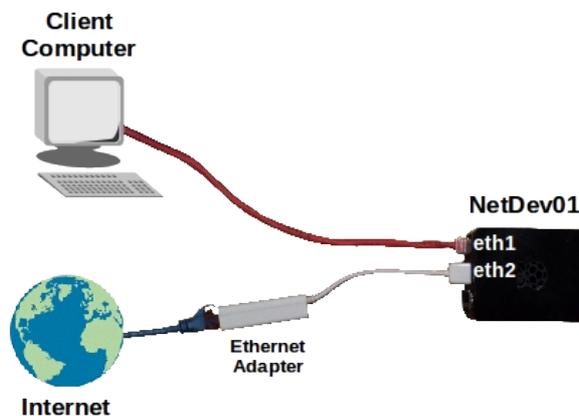
```
root@raspberrypi:~# apt-get install console-data console-common kdb
```

An *ncurses* menu will be displayed. In this example I selected a British keyboard, this is obviously different depending on the locale.

- Select *Keymap from full list* and click OK
- Select *pc / qwerty / British / Standard / Standard* and click OK

1.1.6 Set-up hostname

For the rest of this section I will describe the building of a device called *NetDev01* built to operate in the following network configuration.



Set the device hostname as *NetDev01*.

```
root@raspberrypi:~# echo NetDev01 > /etc/hostname
```

```
root@raspberrypi:~# sed -i .bak 's/raspberrypi/NetDev01/' /etc/hosts
```

1.2 mactelnet

Håkon Nessjøen developed a GNU/Linux implementation of the MikroTik MAC Telnet (*mactelnet*) tool. This tool allows for the connection to networking devices without the need for a layer 3 Internet Protocol (IP) connection. Instead communication is via UDP packets with MAC address used to identify the destination. It includes a MikroTik Neighbour Discovery Protocol (MNDP) tool called *mndp* or *mactelnet -l* which uses the *mndp* protocol for the same purpose. A *macping* tool for confirming connectivity is also included.

1.2.1 Install mactelnet on GNU/Linux

Install the *mactelnet* server and client on the device.

```
$ sudo apt-get install mactelnet-server
$ sudo apt-get install mactelnet-client
```

The following tools are installed.

- *mndp* - A tool for discovering other RouterOS or mactelnetd devices
- *mactelnet* - A tool for telneting via MAC addresses
- *macping* - A tool for pinging other RouterOS or mactelnetd devices

1.2.2 Create user

Let's add Ada Lovelace (*alovelace*) to the *mactelnet* users file.

```
$ sudo vi /etc/mactelnetd.users

# Users file for MAC-Telnetd
#
#####
# WARNING: This file has passwords written in plain-text.      #
#           Make sure this file is owned and only readable by root. #
#####
#
# Each line consists of a username and a password seperated by :.
# Usernames must be existing users from passwd.
#
# Format:
#username:password

alovelace:countess
```

1.2.3 Set default IP address

Set IP Address on first interface. For the *mactelnetd* server to run an interface must have an IP address though it is not needed to connect to the device.

```
$ sudo -s

# cat <<EOM >> /etc/network/interfaces

auto eth1
    iface eth1 inet static
        address 192.168.99.2
        netmask 255.255.255.0
        gateway 192.168.99.1

EOM

# exit
```

1.2.4 Running a server on your GNU/Linux Server

Run the server.

```
$ sudo service mactelnet-server
Usage: /etc/init.d/mactelnet-server {start|stop|status|restart|force-reload}

$ sudo service mactelnet-server start

$ service mactelnet-server status
* mactelnet-server is running
```

1.2.5 Accessing the server

On a client computer connected directly to the *NetDev01* device install the *mactelnet-client* package. This computer must be configured with a static IP address but again any will do. It is now possible to connect to the *NetDev01* device as it is on the directly connected Local Area Network (LAN).

```
$ mndp

Searching for MikroTik routers... Abort with CTRL+C.

MAC-Address      Identity (platform version hardware) uptime
b8:27:eb:95:47:39 NetDev01 (Linux 3.18.7-v7+ armv7l) up 0 days 0 hours
```

Alternatively use *mactelnet -l* option.

```
$ mactelnet -l

Searching for MikroTik routers... Abort with CTRL+C.

MAC-Address      Identity (platform version hardware) uptime
b8:27:eb:95:47:39 NetDev01 (Linux 3.18.7-v7+ armv7l) up 0 days 0 hours
```

Now *macping* the PC *myDebian-PC*.

```
$ sudo macping NetDev01
Searching for 'NetDev01'...found
b8:27:eb:95:47:39 56 byte, ping time 0.57 ms
b8:27:eb:95:47:39 56 byte, ping time 0.51 ms
b8:27:eb:95:47:39 56 byte, ping time 0.52 ms
```

Connect to the device *NetDev01* with the Ada Lovelace user. It is possible to specify the *username* and *password* on the shell or simply respond to prompts given. The identity of the server can be given as the name retrieved from the *mndp* or *mactelnet -l* commands.

```
$ mactelnet NetDev01 -u alovelace -p countess
Connecting to b8:27:eb:95:47:39...done
alovelace@NetDev01:~$
```

```
$ mactelnet b8:27:eb:95:47:39 -u alovelace -p countess
Connecting to b8:27:eb:95:47:39...done
alovelace@NetDev01:~$
```

```
$ mactelnet b8:27:eb:95:47:39 -u alovelace
Password: countess
Connecting to b8:27:eb:95:47:39...done
alovelace@NetDev01:~$
```

```
$ mactelnet b8:27:eb:95:47:39
Login: alovelace
Password: countess
Connecting to b8:27:eb:95:47:39...done
alovelace@NetDev01:~$
```

```
alovelace@NetDev01:~$ id
uid=1001(alovelace) gid=1001(alovelace) groups=1001(alovelace),0(root)
```

1.2.6 *mactelnet frames*

The see what happened a review of the frames captured on the wire below shows a connection.

No.	Time	Source	Destination	Protocol	Length
1249	14.975831000	1.1.1.1	255.255.255.255	MAC-Telnet	66

Info: 28:d2:44:19:83:95 > b8:27:eb:95:47:39 Direction: Client->Server Type: Start session

```
Frame 1249: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Linux cooked capture
Internet Protocol Version 4, Src: 192.168.25.155, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 1984, Dst Port: 20561
MikroTik MAC-Telnet Protocol
```

No.	Time	Source	Destination	Protocol	Length
1250	14.975904000	0.0.0.0	255.255.255.255	MAC-Telnet	66

Info: b8:27:eb:95:47:39 > 28:d2:44:19:83:95 Direction: Server->Client Type: Acknowledge

```

Frame 1250: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Linux cooked capture
Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 20561, Dst Port: 1984
MikroTik MAC-Telnet Protocol

```

No.	Time	Source	Destination	Protocol	Length
1255	14.976545000	1.1.1.1	255.255.255.255	MAC-Telnet	75

```

Info: 28:d2:44:19:83:95 > b8:27:eb:95:47:39 Direction: Client->Server Type: Data

```

A fully expanded *Acknowledge* packet is displayed below.

```

Frame 15: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface 0
Interface id: 0
Encapsulation type: Ethernet (1)
Arrival Time: Mar  8, 2015 18:58:41.034002000 GMT
Epoch Time: 1425841121.034002000 seconds
Frame Number: 15
Frame Length: 64 bytes (512 bits)
Capture Length: 64 bytes (512 bits)
Ethernet II, Src: Raspberr_95:47:39, Dst: LcfcHefe_19:83:95
Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
User Datagram Protocol, Src Port: 20561, Dst Port: 1532
MikroTik MAC-Telnet Protocol
  Protocol Version: 1
  Type: Acknowledge (2)
  Source MAC: b8:27:eb:95:47:39
  Destination MAC: 28:d2:44:19:83:95
  Session ID: 0x24f6
  Client Type: MAC Telnet (0x0015)
  Session Data Bytes: 0

0000  28 d2 44 19 83 95 b8 27 eb 95 47 39 08 00 45 10  (.D....'..G9..E.
0010  00 32 00 33 40 00 40 11 3a 79 00 00 00 00 ff ff  .2.3@.@.:y.....
0020  ff ff 50 51 05 fc 00 1e a7 e0 01 02 b8 27 eb 95  ..PQ.....'..
0030  47 39 28 d2 44 19 83 95 00 15 24 f6 00 00 00 00  G9(.D.....$.

```

1.3 Additional Interface

Add a USB Ethernet interface to *NetDev01* and connect it to an Ethernet switch upon which an IP address is given out via the Dynamic Host Configuration Protocol (DHCP). Using *dmesg* and *ip link show* the new interface can quickly be determined as */dev/eth2*.

```
$ dmesg
```

```
[ 1229.349092] usb 1-1.2: new high-speed USB device number 5 using dwc_otg
[ 1229.473567] usb 1-1.2: New USB device found, idVendor=9710, idProduct=7830
[ 1229.480642] usb 1-1.2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 1229.488121] usb 1-1.2: Product: USB-MAC Controller
[ 1229.493928] usb 1-1.2: Manufacturer: Moschip Semiconductor
[ 1229.499595] usb 1-1.2: SerialNumber: 6e0002c5
[ 1229.577159] usb 1-1.2: applying rev.C fixup
[ 1229.599138] usb 1-1.2: applying rev.C fixup
[ 1229.620845] MOSCHIP usb-Ethernet driver 1-1.2:1.0 eth0: register 'MOSCHIP usb-Ethernet driver' at usb-bcm2708_usb-1.2, MOSCHIP 7830/7832/7730 usb-NET adapter, 00:60:6e:00:66:13
[ 1229.643830] usbcore: registered new interface driver MOSCHIP usb-Ethernet driver
[ 1229.748355] MOSCHIP usb-Ethernet driver 1-1.2:1.0 eth2: renamed from eth0
```

```
$ ip link show
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode
DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP mode DEFAULT qlen 1000
    link/ether b8:27:eb:95:47:39 brd ff:ff:ff:ff:ff:ff
3: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode
DEFAULT qlen 1000
    link/ether 00:60:6e:00:66:13 brd ff:ff:ff:ff:ff:ff
```

1.3.1 DHCP request on new interface

Perform a DHCP request on this interface to assign an IP address to the interface.

```
$ sudo dhclient eth2
```

```
$ ip addr show dev eth2
```

```
3: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP qlen 1000
    link/ether 00:60:6e:00:66:13 brd ff:ff:ff:ff:ff:ff
    inet 192.168.22.86/24 brd 192.168.22.255 scope global eth2
        valid_lft forever preferred_lft forever
    inet6 fe80::260:6eff:fe00:6613/64 scope link
        valid_lft forever preferred_lft forever
```

1.3.2 Default gateway to eth2

The default gateway is currently set to the management interface being used by *mactelnet*. Change the default route to the */dev/eth2* interface.

```
$ sudo ip route change default dev eth2
```

```
$ ip route show
default dev eth2 scope link
192.168.22.0/24 dev eth2 proto kernel scope link src 192.168.22.86
192.168.99.0/24 dev eth1 proto kernel scope link src 192.168.99.2
```

1.3.3 Test routing

Perform a test ping to a global IP address to confirm connectivity.

```
$ ping -c1 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_req=1 ttl=56 time=26.0 ms
```

2. Network Simulation

It is very useful to have a mechanism to simulate the networks discussed in this document. The Common Open Research Emulator (CORE) offers an Open Source tool ideal for the purpose.

CORE is a tool for emulating networks on one or more machines. The emulated networks can even be connected to live networks. CORE consists of a Graphical User Interface (GUI) for drawing topologies of lightweight virtual machines, and Python modules for scripting network emulation.

CORE was developed by a Network Technology research group that is part of the Boeing Research and Technology division. The Naval Research Laboratory is supporting further development of the project.

Access to the software is at: <http://www.nrl.navy.mil/itd/ncs/products/core>

2.1 Getting started

The easiest way to get started is to use the pre-built Virtual Machine (VM). This is built on Lubuntu, a fast and lightweight variant of Ubuntu.

- Install Oracle VirtualBox or similar hypervisor.
- Download the VM from: <http://downloads.pf.itd.nrl.navy.mil/core/vmware-image/>

```
$ unzip vcore-4.7.zip
```

- Run Oracle VirtualBox.
 - Select *New*
 - Name: **CORE Network Emulator**
 - Type: **Linux**
 - Version: **Ubuntu (64 bit)**
 - Select *Next*>
 - Set Memory Size: 2048 MB, Select *Next*>

- Select the vCORE VM.
 - Select *Use an existing virtual hard drive file*
 - Browse to *vcore-4.7.vmdk*, Select *Open*
 - Select **Create>**
- Configure and run VM
 - Right click on the newly created VM
 - Select *Network*
 - Attached to: *Bridged Adapter*
 - Select **OK**
- Start the newly created VM
 - Select **Start**

2.2 Operating the CORE environment

While it is possible to use the graphical interface within the VM window it is not recommended. Within the graphical environment select the LXTerminal icon to obtain a shell and get the IP address of the VM.

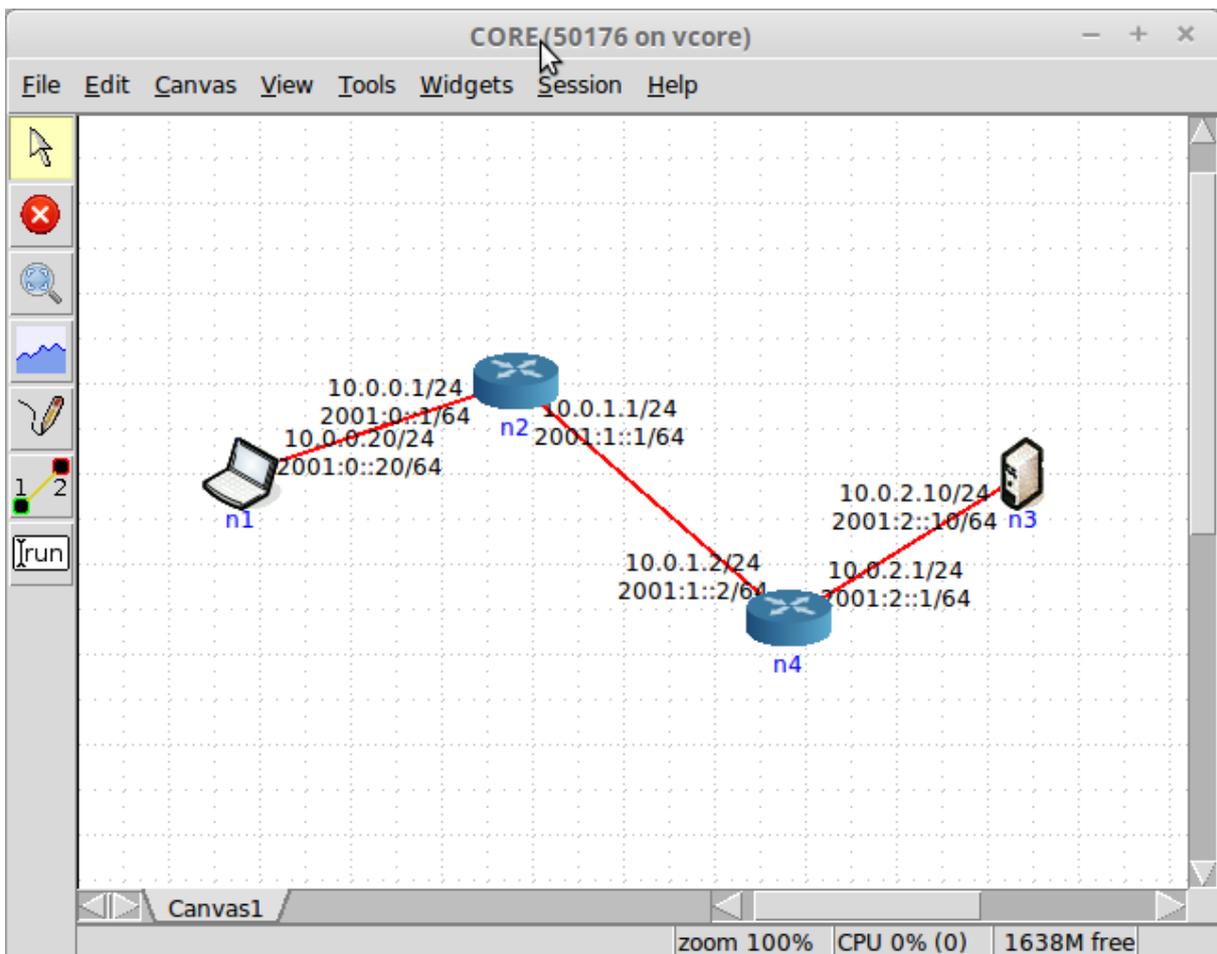
```
$ ip addr
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP qlen 1000
    link/ether 08:00:27:06:a9:03 brd ff:ff:ff:ff:ff:ff
    inet 192.168.178.31/24 brd 192.168.178.255 scope global eth1
    inet6 fe80::a00:27ff:fe06:a903/64 scope link
        valid_lft forever preferred_lft forever
```

Now connect to the CORE GUI by forwarding it via X11 forwarding to the workstation. Run the *core-gui*.

```
$ ssh -X core@192.168.178.31  
Password: core
```

```
$ core-gui
```



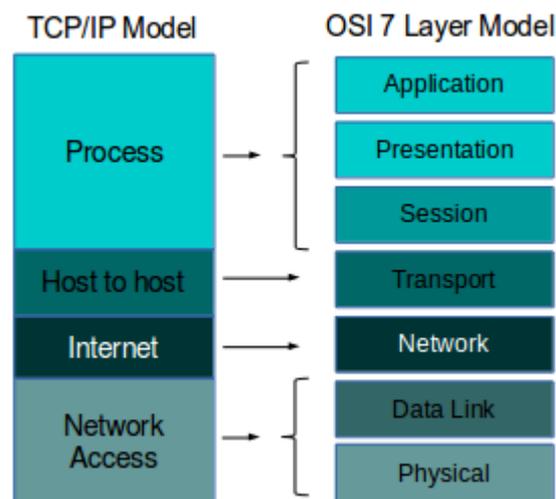
Build networks and test as necessary. Full documentation on the CORE Network Simulator can be found at: <http://downloads.pf.itd.nrl.navy.mil/docs/core/core-html/>

This page is intentionally blank

3. Networking

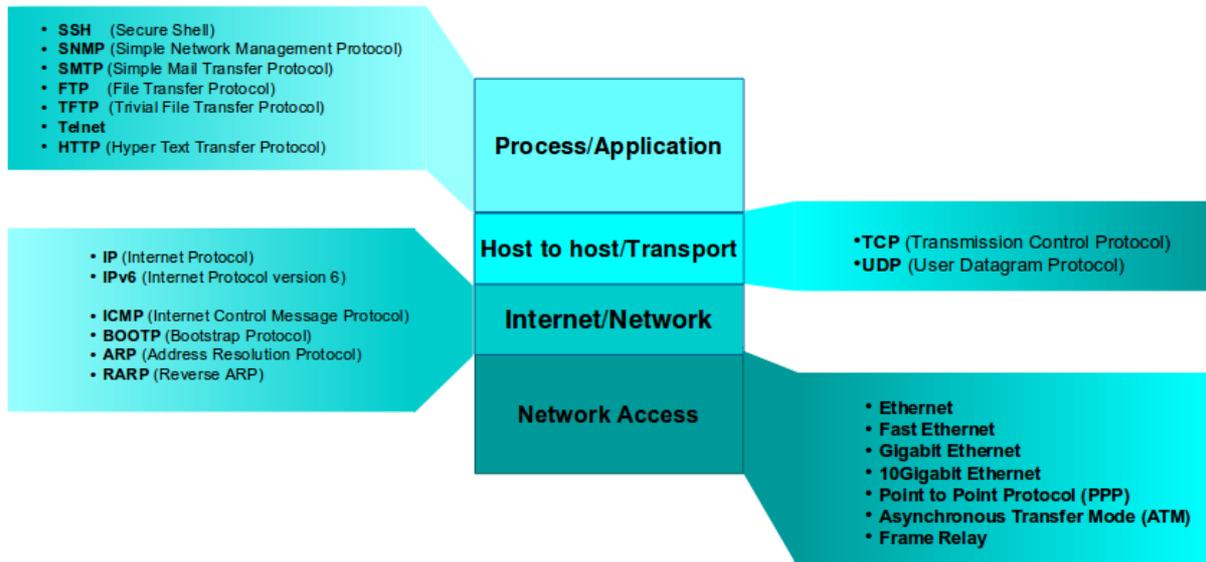
3.1 Introduction to Network Administration

The UNIX and GNU/Linux operating system has always proven its versatility in aspects related to communication and information exchange. Wide Area Networks (WAN) networks based on serial modems, Frame Relay, Plesiochronous Digital Hierarchy (PDH) E1 and T1 circuits in 2.048 and 1.544 Megabit/second (Mb/s) blocks as well as 155 Mb/s Synchronous Optical Networking (SONET) and Synchronous Digital Hierarchy (SDH) have been replaced by Gigabit/second (Gb/s) speed fibre and copper Metro Ethernet Metropolitan Area Networks (MAN) technologies. Home users are receiving broadband with a mix of Passive Optical Network (PON) technologies, various flavours of Digital Subscriber Line (DSL), Data Over Cable Service Interface Specification (DOCSIS), wireless technologies from Institute of Electrical and Electronics Engineers (IEEE) 802.16 WiMAX, European Telecommunications Standards Institute (ETSI) Long Term Evolution (LTE) and IEEE 802.11 based Wireless technologies offering speeds from 3 Megabit/second (Mb/s) to 200 Mb/s. Home users and companies have Local Area Networks (LAN) within their premises to interconnect their computing devices.



All of these technologies offer a Data Link framing at layer 2 of the Open Standards Interconnect (OSI) 7 layer communications model from the International Standards Organisation (ISO). The handling of the upper layers is mapped to the Department of Defence (DoD) 4 Layer Model, particularly layer 3, Internet layer (OSI Network layer) and layer 4, Host to Host layer (OSI Transport layer) are carried out by the Internet Protocol Suite (IPS) commonly called the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol suite. While TCP/IP is mapped to the DOD 4 layer model it is common that layers 3 and 4 are referred to by their OSI names, Network and Transport layers.

3.2 Introduction to TCP/IP (TCP/IP suite)



TCP/IP is a set of basic protocols that meet the different needs in computer-to-computer communication, such as Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Internet Protocol (IP), Internet Control Message Protocol (ICMP), Address Resolution Protocol (ARP), Dynamic Host Configuration Protocol (DHCP), Domain Name Service (DNS) and in Internet Protocol version 6 (IPv6), ICMPv6, DHCPv6,

TCP/IP is most frequently used by most current users. To remotely connect to other computers Secure Shell (SSH) offers a popular solution, to use remote files with Network File System (NFS) or to transfer them with File Transfer Protocol (FTP) or Secure FTP (SFTP). To access webpages the universal HyperText Transfer Protocol (HTTP) is the standard markup protocol.

Here is an example login to a remote server `ftacademy.org` by user `alovelace`.

```
$ ssh avelace@ftacademy.org
```

```
alovelace@ftacademy.org's password: password
Linux fta.obriain.com 3.2.0-4-amd64 #1 SMP Debian 3.2.54-2 x86_64
```

```
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Mar 26 16:21:21 2014
alovelace@ftacademy:~$
```

3.2.1 Services on TCP/IP

The most important traditional TCP/IP services are:

- *File transfer*: FTP and SFTP allow a user to obtain files or send files from one computer to another. The user must have an account in the remote computer and identify themselves through their login name and password or the user must connect to computers containing an information repository (software, documentation etc.) under an anonymous account to read those computers on their computer.
- *Remote connection (login)*: SSH or the older insecure TERminal NETwork protocol (telnet) allows a user to remotely connect to a computer. The local computer is used as a terminal to the remote computer and everything is executed over it, whilst the local computer remains invisible from the perspective of the user that started the session.
- *Email*: The e-mail service makes it possible to send messages to users of other computers.

The progress in the technology and the increasingly lower cost of computers has meant that services have specialised and are now configured on computers working in a client-server model. A server is a system that performs specific services for the rest of the network or connected clients. A client is another computer that uses this service. All of these services are generally offered within TCP/IP:

- *File systems in NFS*: Allow a system to access the files through a remote system in a manner that is more integrated than FTP. The storage devices are exported to the system that wishes to access the files and this system can access them as if they were local devices. This protocol permits in the server side to establish the rules and ways of accessing the files, which makes the place where the information physically resides independent from the place where the information is *accessed*.
- *Remote printing*: Permits users to access printers connected to other computers.
- *Remote execution*: Permits a user to execute a program on another computer. There are various ways of executing a program in this way: either through a command (rsh, ssh, rexec) or through systems with Remote Procedure Call (RPC), which allows a program on a local computer to execute a function in a program on another computer.
- *Name servers*: The Network Information Service (NIS) (Yellow Pages (YP)) is the original UNIX client server directory service protocol for distribution of configuration data like usernames and hostnames. It was more or less replaced by Lightweight Directory Access Protocol (LDAP) within organisations and for large scale directory services the DNS keeps a direct relationship between the hostname and the logical identification name of this machine (IP address).

- *Terminal Servers*: Connect terminals to a server that itself offers a telnet or SSH daemon. Different terminals are accessed via the telnet/SSH daemon with different port numbers matching the different terminals. These types of set-up are basically useful for reducing costs and improving the connections to the central computer (in some cases).
- *Graphical terminal servers* (network-oriented window systems): these permit a computer to visualise graphic information on a display that is connected to another computer. The most common of these systems is X Window (X.Org).

3.2.2 What is TCP/IP?

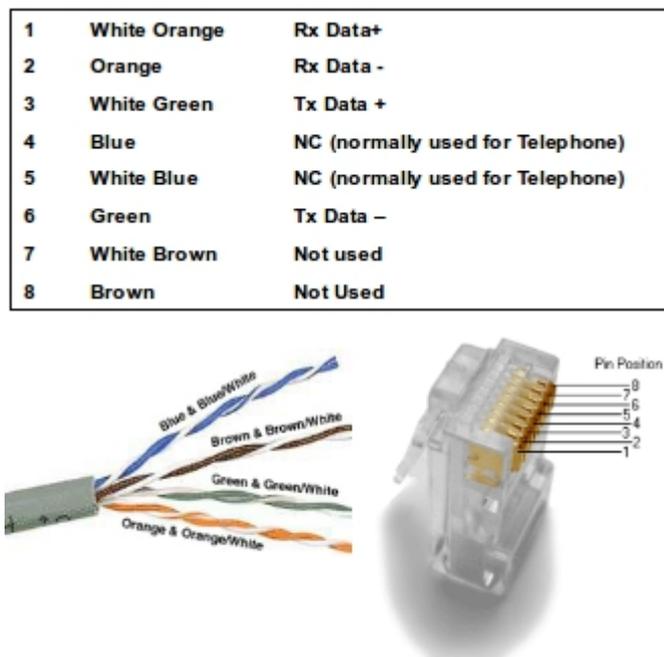
TCP/IP is in fact two communication protocols between hosts (computers) that are independent to each other. IP defines the protocol at the Internet/Network layer to identify the networks and establish the pathways between different computers. TCP defines the communication rules so that a host computer can talk to another computer (Host to Host/Transport layer). TCP is a connection-oriented protocol and the communication is considered as a data stream. The receiving host verifies receipt of data in blocks to the sending host before it sends another block, in this way the data sent over the link is verified as received.

Another Host to Host/Transport layer protocol is the UDP, which treats the data as a message (datagram) and sends packets. It is considered a connectionless protocol as the sending host receives no confirmation of receipt of the data that is sent (much like a standard letter in the mail system). The advantage of this is less overload on the network than a TCP connection, but it is obviously less reliable (the packets may not arrive or arrive duplicated). It is commonly used with protocols that are sensitive to packet delay like Voice over IP (VoIP) and video streaming.

To summarise, TCP/IP is a set of protocols including IP, TCP, UDP that provide a set of low-level functions used by most of the applications. Some of the protocols that use the above mentioned services were designed by Berkeley, Sun or other organisations. They are not included (officially) as part of the IPS. However, they are implemented using TCP/IP and they are therefore considered as a formal part of IPS. A description of the protocols are available in the Internet Engineering Task Force (IETF) Request For Comments (RFC) 1011 and 6093. There is currently a new version of the IP protocol known as IPv6, also called IP Next Generation (IPng) which replaces IP version 4 (IPv4). It is defined in RFC2460 and is updated in a series of additional RFCs. This protocol significantly improves on the previous version in elements such as having a greater number of nodes, traffic control, security or improvements in the routing.

3.3 Physical network devices (hardware)

From the physical point of view (layer 1 of the OSI model), the most commonly used hardware for LAN is that known as Ethernet (FastEthernet (FE) or GigaEthernet (GbE)). Its advantages consist of a lower cost, acceptable speeds (100 Mb/s, 1 Gb/s or 10 Gb/s) and its user-friendly installation.



Presentation is either via copper twisted pair, fibre or wireless.

3.3.1 Copper

100Base-TX (uses 2 bi-directional pairs in Category 5e (CAT5e) or above), 1000Base-T (uses 4 bi-directional pairs in CAT-5e or above) and 1000Base-TX (uses 2 bi-directional pairs in CAT-6, CAT-7 only). The standard copper pinout is given in the diagram above. All of these copper technologies are limited to a distance of 100 metres.

Color coding of Premise Fibre Cable		
Fibre Type / Class	Diameter (μm)	Jacket Colour
Multimode 1a	50/125	Orange
Multimode 1a	62.5/125	Slate
Multimode 1a	85/125	Blue
Multimode 1a	100/140	Green
Singlemode IVa	All	Yellow
Singlemode IVb	All	Red



3.3.2 Fibre Optic

Alternatively Ethernet can be delivered via optical fibre. An optical fibre is a small narrow tube plastic or glass which guides light along its length by total internal reflection. The particular wavelengths used, 850, 1300 and 1550 nano metres (nm), correspond to wavelengths where optical light sources, lasers or Light Emitting Diodes (LED) are easily manufactured. There are main types of fibre;

- Multi-Mode Fibre (MMF)
 - 850 and 1300 nm
 - Based on cheap to manufacture LED or Vertical Cavity Surface Emitting Laser (VCSEL) transmitters
 - Fibre core of 50 μm and a cladding diameter of 125 μm
 - Range limits are up to 2 km for 100BASE-FX, up to 550 m for 1000BASE-SX (850 nm) and , up to 550 m for 1000BASE-LX (1300 nm)
- Single-Mode Fibre (SMF)
 - based on laser transmitters
 - Fibre core of 8 μm and a cladding diameter of 125 μm
 - Range limits are up to 5 km for 1000BASE-LX (1300 nm), up to 10 km for 1000BASE-LX10 (1300 nm), up to 10 km for 1000BASE-BX10 (1300 nm upstream and 1550 nm downstream in a single strand), up to 40 km for 1000BASE-EX (1300 nm) and up to 70 km for 1000BASE-ZX (1550 nm)

3.3.3 Wireless

The other type of Ethernet LAN is Wireless. These are IEEE 802.11 based Wireless Fidelity (WiFi) family of specifications for wireless LAN (WLAN) technology. WLANs are organised with Access Points (AP) radio transmitters that allow hosts (computers, mobile devices, ...) to connect to a specific Service Set Identifier (SSID) which defines the wireless network. Security is an essential element of wireless networks and WiFi Protected Access version 2 (WPA2) (also called Robust Security Network (RSN)) is an implementation of the IEEE 802.11 standard used today. Older security protocols like Wireless Encryption Protocol (WEP) and WiFi Protected Access (WPA) are considered less secure.

Standard	Description
802.11	Initial WLAN standard providing 1 or 2 Mbps transmission in the 2.4 GHz band using either Frequency Hopping Spread Spectrum (FHSS) or Direct Sequence Spread Spectrum (DSSS).
802.11a	This is an extension to IEEE 802.11 that applies to WLANs and provides typically 25 Mbps to a maximum of 54 Mbps in the 5GHz band. 802.11a uses an Orthogonal Frequency-Division Multiplexing (OFDM) encoding scheme rather than FHSS or DSSS. Max range is 30 M.
802.11b	An extension to 802.11 that applies to WLANs and provides 11 Mbps transmission (with a fallback to 5.5, 2 and 1 Mbps) in the 2.4 GHz band. IEEE 802.11b uses only DSSS. IEEE 802.11b was a 1999 ratification to the original IEEE 802.11 standard, allowing wireless functionality comparable to Ethernet. Max range is 30 M.
802.11g	A further evolution to WLANs that provides typically 24 Mbps to a maximum of 54 Mbps in the 2.4 GHz band. It also uses OFDM. Max range is 30 M.
802.11n	A move towards the 5 GHz band instead of 2.4 GHz for home WiFi. A speed of 200 Mbps typical to a maximum of 540 Mbps out to 50 M in either the 2.4 or 5 GHz bands. It uses Multiple In, Multiple Out (MiMo) antennas.
802.11ac	The 1 Gb/s standard with a single link throughput of at least 500 Mb/s. It has a 160 MHz Radio Frequency bandwidth, up to 8 spatial streams for MiMo and 256 QAM modulation.

Determine the wireless interface from the kernel ring buffer. Also use the *iw dev* to get a listing of all wireless hardware devices on the system.

```
$ dmesg |grep Wireless
[ 20.615523] eth1: Broadcom BCM4359 802.11 Hybrid Wireless Controller
6.30.223.141 (r415941)

$ iw dev
phy#0
    Interface eth1
        ifindex 3
        type managed
```

Now using the wireless *iw* utility with the interface *eth1* just discovered get information on the wireless interface.

```
$ iw dev eth1 info
Interface eth1
    ifindex 3
    type managed
    wiphy 0
```

3.4 GNU/Linux interface

In GNU/Linux, Ethernet interfaces are labelled like *ethx* (where, "x" indicates an order number beginning with 0), the interface to serial lines (modems) is called up with *pppx*, Point to Point Protocol (PPP). These names are used by the commands to configure them and assign them the identification that will, subsequently permit them to communicate with other devices in the network. This may mean that the appropriate modules for the appropriate device Network Interface Card (NIC) will need to be included in the kernel, or as modules. If this proves necessary then the kernel will require compiling, after choosing, the appropriate NIC, with, for example, *make menuconfig*, indicating it as internal or as a module (in the latter case, the appropriate module must also be compiled). In reality it is typical for the GNU/Linux installation to incorporate the appropriate generic module for the hardware.

Listing the available interfaces:

```
$ ifconfig -a
```

```
eth0      Link encap:Ethernet  HWaddr 28:d2:44:19:83:95
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

eth1      Link encap:Ethernet  HWaddr 1c:3e:84:ed:99:0b
          inet addr:192.168.43.222  Bcast:192.168.43.255  Mask:255.255.255.0
          inet6 addr: fe80::1e3e:84ff:feed:990b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1068090 errors:0 dropped:0 overruns:0 frame:1016162
          TX packets:908026 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1014889719 (1.0 GB)  TX bytes:135467503 (135.4 MB)
          Interrupt:17

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:92579 errors:0 dropped:0 overruns:0 frame:0
          TX packets:92579 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:8690656 (8.6 MB)  TX bytes:8690656 (8.6 MB)
```

This command shows all of the default interfaces/parameters for each one, it is a deprecated command however and the new `ip link show` gives the same output but in a more untidy manner.

```
$ ip link show
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode
  DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast
  state DOWN mode DEFAULT qlen 1000
    link/ether 28:d2:44:19:83:95 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
  state UP mode DORMANT qlen 1000
    link/ether 1c:3e:84:ed:99:0b brd ff:ff:ff:ff:ff:ff
```

The network devices can be seen in the `/dev` directory, where there is a special file, (which may be a block file or a character file, according to the transfer) that represents each hardware device.

3.5 TCP/IP Concepts

Communication involves a series of concepts:

- *Internet/intranet*: The term *intranet* refers to the application of Internet technology (the network of networks) within an organisation, essentially to distribute an organisation's internal information and to have it available within the organisation. For example, the services offered by GNU/Linux as Internet and Intranet services include email, WWW, news, ...
- *Node*: the (host) node refers to a machine that is connected to the network (in a wider sense, a node may be a computer, a printer, a CD (rack) etc.); in other words, an active and differentiable element in the network that requires or provides some kind of service and/or shares information.
- *Ethernet Network Address (Ethernet address or MAC address)*: This is an IEEE Extended Unique Identifier (EUI). It can be either an EUI-48, a 48-bit number (i.e. *00:88:40:73:AB:FF* (octal) *0000 0000 1000 1000 0100 0000 0111 0011 1010 1011 1111 1111* (binary)) or an EUI-64, a 64-bit number (i.e. *00:88:40:FF:FE:73:AB:FF* (octal) *0000 0000 1000 1000 0100 0000 1111 1110 0111 0011 1010 1011 1111 1111* (binary)). The EUI is burnt on the physical hardware of the the Ethernet driver (NIC) and that is recorded by the manufacturer as it must be is globally unique. (this number must be the only one globally, each NIC manufacturer has a pre-allocated range).

```
$ cat /etc/hostname
```

```
myHostname
```

- *Host name*: each node must also have a unique network name. These may simply be names or they may use a scheme based on a hierarchical domain naming scheme. The names of the nodes must be unique, which is easy in small networks, more complex in large networks and impossible on the Internet unless some form of control is implemented. The names must have a maximum of 32 characters within the a-z, A-Z and 0-9 ranges and they may not contain spaces or # beginning with an alphabetic character.

```
$ cat /etc/hosts
```

```
127.0.0.1    localhost
127.0.1.1    myHostname
```

```
# The following lines are desirable for IPv6 capable hosts
```

```
::1        ip6-localhost ip6-loopback
fe00::0    ip6-localnet
ff00::0    ip6-mcastprefix
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters
```

- *Internet Address (IP Address)*

- *IPv4:* IPv4 addresses consists of four numbers within the range of 0-255 separated by dots (for example, 192.168.0.1) and it is used universally to identify the computers on a network or on the Internet. The names are translated into IP addresses by a DNS server, that transforms the node names (legible to humans) into IP addresses (this service is performed by an application called *named*).
- *IPv6:* IPv6 addresses consists of eight groups of four hexadecimal digits. (for example, 2a02:2158:435a:0000:83:314:ea21:b33f) and it is also used universally to identify the computers on a network or on the Internet. The names are translated into IP addresses by a DNS server using AAAA (Quad 'A') records, that transforms the node names in IP addresses (this service is performed by an application called *named*).

```
$ less /etc/services
```

```
tcpmux      1/tcp          # TCP port service multiplexer
echo        7/tcp
echo        7/udp
discard     9/tcp          sink null
discard     9/udp          sink null
systat      11/tcp         users
daytime     13/tcp
daytime     13/udp

~~~~~          ~~~~/~~~~          ~~~~

vboxd      20012/udp
binkp      24554/tcp      # binkp fidonet protocol
asp        27374/tcp      # Address Search Protocol
asp        27374/udp
csync2     30865/tcp      # cluster synchronization tool
dircproxy  57000/tcp      # Detachable IRC Proxy
tfido      60177/tcp      # fidonet EMSI over telnet
fido       60179/tcp      # fidonet EMSI over TCP
...
```

- *Port*: is a numerical identifier that uniquely identify applications and processes running on the computer. For example: well known ports, FTP 21, SSH 22, telnet 23, Simple Mail Transfer Protocol (SMTP) 25, Post Office Protocol version 3 (POP3) 110, Interim Mail Access Protocol (IMAP) 143, Simple Network Management Protocol (SNMP) 161/162, Remote Access Dialin User Service (RADIUS), 1812/1813 are just a few of the more popular ones.

\$ ip -4 route list

```
default via 192.168.22.1 dev eth1 proto static
192.168.22.0/24 dev eth1 proto kernel scope link src 192.168.22.159 metric 9
```

- *Router node (gateway)*: it is a node that performs the routing (data transfer) function. A router, depending on its characteristics, may transfer information between two similar or different network protocols and may also be selective.

\$ cat /etc/resolv.conf

```
nameserver 8.8.8.8
```

- *DNS*: Provides databases that perform the translation between the name and Internet address and that are structured in the form of a tree. In order to do this, domains separated by points are defined, of which the highest (from right to left) describes a category, institution or country (COM stands for Commercial, EDU for Education, GOV for Governmental, MIL for Military (government), ORG, non-profit Organisation, XX which could be any two letters to indicate the country. The second level represents the organisation and the third and remaining sections indicate the departments, sections or divisions within an organisation (for example, www.ftacademy.org or john@freeknowledge.eu). The first two names (from right to left), ftacademy.org in the first case, freeknowledge.eu (in the second) must be assigned (approved) by the Internet Corporation for Assigned Names and Numbers (ICANN) and the rest may be configured/assigned by the organisation.
- *DHCP, Bootstrap Protocol (bootp)*: *DHCP* and *bootp* are protocols that permit a client node to obtain information on the network (such as the node's IP address). Many organisations with many machines use this mechanism to facilitate the administration of large networks or networks in which there are roaming users. On IPv6 this function is carried out by either an internal IPv6 Stateless Address Auto Configuration (SLAAC) or a DHCPv6 Server.

\$ ip -4 neigh list

```
192.168.43.1 dev eth1 lladdr 5c:f8:a1:b3:79:c2 REACHABLE
```

- *ARP, Reverse ARP (RARP)*: in some networks (like Ethernet), the IP addresses are dynamically discovered through the use of two other members of the Internet protocol suite: ARP and RARP. ARP uses broadcast messages to determine the Ethernet address (MAC specification for layer 3 of the OSI model), corresponding to a particular network-layer address (IP). RARP uses broadcast messages (messages that reach all of the nodes) to determine the network-layer address associated with a particular hardware address. RARP is especially important to diskless nodes, for which network-layer addresses are usually unknown at boot time.

```
$ ip -6 neigh list
```

```
2501:f2b30:2a0a::1 lladdr 5c:f8:a1:b3:79:c2 router REACHABLE
```

- *Neighbour Discovery Protocol (NDP)*: This is the IPv6 equivalent to the ARP/RARP process in IPv4.
- *Socket Library*: in UNIX, all TCP/IP implementation is part of the kernel of the operating system (either within the same or as a module that loads at boot time, as is the case with the device drivers in GNU/Linux). A programmer can use sockets through an Application Programming Interface (API). These API implement a source-code interface to the system. For TCP/IP, the most common API is the Berkeley Socket Library (Windows uses an equivalent library that is called *Winsocks*). This library makes it possible to create a communication end-point (socket), associate it to a remote node and port (*bind*) and offer the communication service (through *connect*, *listen*, *accept*, *send*, *sendto*, *recv*, *recvfrom*, for example). The library also provides a more general communication mode (AF_INET family) and more optimised communications for cases in which the process are communicating within the same machine (AF_UNIX family). In GNU/Linux, the socket library is part of the C standard library, *Libc*, (*Libc6* in current versions), and it supports AF_INET, AF_UNIX, AF_IPX (for Novell protocols), AF_X25 (for the X.25 protocol), AF_ATMPVC-AF_ATMSVC (for the ATM protocol) and AF_AX25,F_NETROM, AF_ROSE (for amateur radio protocol).

This page is intentionally blank

4. Switching

4.1 Bridging and Switching

A bridge is a device used to connect two or more Local Area Networks (LAN) that use identical LAN (Medium Access Control (MAC) Layer) protocols. The bridge acts as an address filter, picking up frames from one LAN segment (collision domain) that are intended for a destination on another LAN segment, and passing those frames on. The bridge does not modify the contents of the frames and does not add anything to the frame. The bridge operates at layer 2 of the OSI model.

The original concept of a bridge was a device that would interface similar LAN segments and would *filter* and *forward* transmissions (pass those which were for address not on the source segment, and not pass those whose destination address is on the source segment). So bridges maintain tables of addresses associated with each port on the bridge. The IEEE 802.1 standard defines bridges, often called "Ethernet bridges" or transparent bridges, and some vendors call them *Ethernet switches*.

In the simplest terms, a bridge *forwards* (sends) frames between LAN segments that are attached to its ports using information it finds in the OSI Model Layer 2, the Data Link Layer (actually the IEEE 802.3 MAC layer addressing), of a frame. It ignores the other layers of the OSI Model. In other words, it looks at the destination address field, compares the address to its address tables for all its ports. If it finds the address associated with a port, it sends the frame out on that port. If it does not find an address, it sends the frame out on all ports.

In a multi-bridge IEEE LAN environment, the bridges usually communicate with each other using the IEEE 802.1d *Spanning Tree Algorithm* (STA) protocol or other protocol. Bridges have a problem when an address is unknown the frame is forwarded to all ports off the bridge. This could cause address table problems and frame propagation in multi-LAN environments without the spanning tree algorithm capability.

In pure bridges, there are 2 types of transparent (with or without Spanning Tree) bridging and Source Routing bridging. Bridges are most effective when there are few links in a network. Larger networks usually use Routers where links are numerous. Transparent bridges are normally *connectionless* switching devices, which means they themselves do not help maintain connections in the network. Transparent bridges just send frames or frames out a port, they do not route them to another device.

4.1.1 Why use Bridges

- Limit number of stations (contention) transmitting on specific segments.
- Limit Size of LANs.
- Limit volume of traffic (bandwidth).
- Reduces traffic across segments of a single LAN.
- Connect multiple local LANs into a single network at a local level.

4.1.2 Switches

A switch is a device designed to segment LANs with one idea in mind, increase the bandwidth. This differs from a bridge or router whose purpose is to limit the amount of traffic flowing between LANs (a LAN will be sometimes referred to as a collision domain).

The Layer 2 (L2) Switch interconnects LAN segments. Traffic between the LAN segments will be switched at near *wire speed*.

A bridge normally will have 2 or 3 ports, where a switch will have 4, 6, or more ports for attaching separate LANs or collision domains. 10/100/1000 Mb/s switches have two or more 1000 Mb/s and 4 or more 10/100 Mb/s ports. Consequently, collision domains can have more segmentation than with a bridge.

Why Switching

- Switches operate at Layer 2 of the OSI Model.
- Switching is an advance in bridging technology.
- Switches forward frames based on the MAC layer address (the actual Network Interface Card (NIC) address).
- Switches forward frames with very low delay time (wire speed).
- Switches, in most cases, use the IEEE 802.1d Spanning Tree Protocol (STP) or IEEE 802.1w Rapid Spanning Tree Protocol (RSTP) allowing for redundant switches in the network.
- Switches will forward *broadcast* traffic to all LANs attached to them.

When is switching used

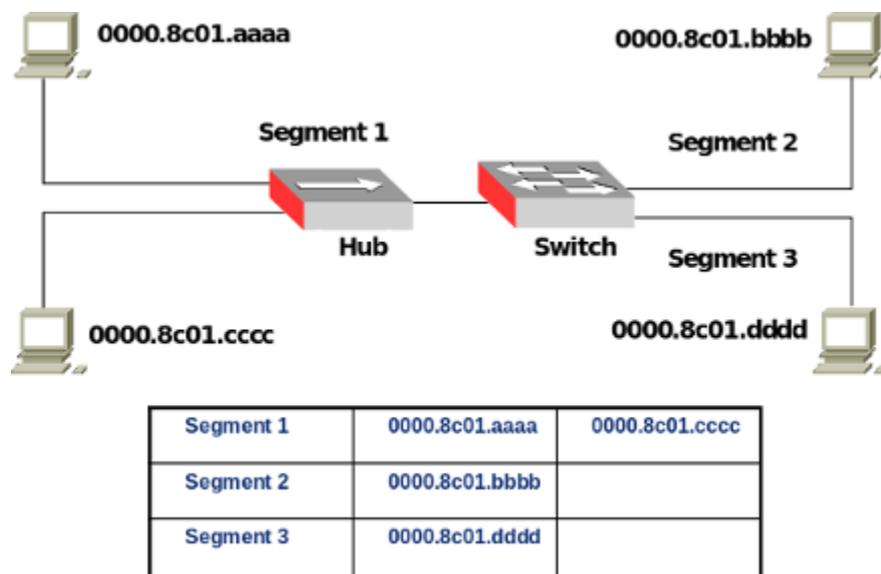
Switching is used when segmentation/connection of several LAN segments is required with increased bandwidth. If security among the LANs is not a significant issue then a switch can be used rather than a router if the following services are not required;

- Support redundant paths.
- Have intelligent frame forwarding.
- Connect to a WAN.

4.1.3 Transparent Bridging

L2 Switches and Transparent Bridges use transparent bridging to create their address lookup tables. Transparent bridging allows a switch to learn everything it needs to know about the location of nodes on the network without the network administrator having to statically add entries. Transparent bridging consists of five parts or steps:

- Learning
- Flooding
- Filtering
- Forwarding
- Ageing



The switch is added to the network, and the various segments are plugged into the switch's ports. A host with the MAC *0000.8c01.aaaa* (aaaa) on the first segment sends data to a host *0000.8c01.bbbb* (bbbb) on another segment 2.

The switch gets the first frame of data from *aaaa*. It reads the MAC address and saves it to the lookup table for Segment 1. The switch now knows where to find *aaaa* any time a frame is addressed to it. This process is called *learning*.

Since the switch does not know where *0000.8c01.bbbb* (bbbb) is, it sends the frame to all of the segments except the one that it arrived on (Segment 1). When a switch sends a frame out to all segments to find a specific node, it is called *flooding*.

The host *bbbb* gets the frame and sends a frame back to *aaaa* in acknowledgement. The frame from *bbbb* arrives at the switch. Now the switch can add the MAC address of *0000.8c01.bbbb* to the lookup table for Segment 2. Since the switch already knows the address of *aaaa*, it sends the frame directly to it. Because *aaaa* is on a different segment than *bbbb*, the switch must connect the two segments to send the frame. This is known as *forwarding*.

The next frame from *aaaa* to *bbbb* arrives at the switch. The switch now has the address of *bbbb* in its tables, so it *forwards* the frame directly to *bbbb*. *0000.8c01.cccc* (*cccc*) sends information to the switch for *aaaa*. The switch looks at the MAC address for *cccc* and adds it to the lookup table for Segment 1. The switch already has the address for *aaaa* and determines that both nodes are on the same segment, so it does not need to connect Segment 1 to another segment for the data to travel from *cccc* to *aaaa*. Therefore, the switch will ignore frames travelling between nodes on the same segment. This is *filtering*.

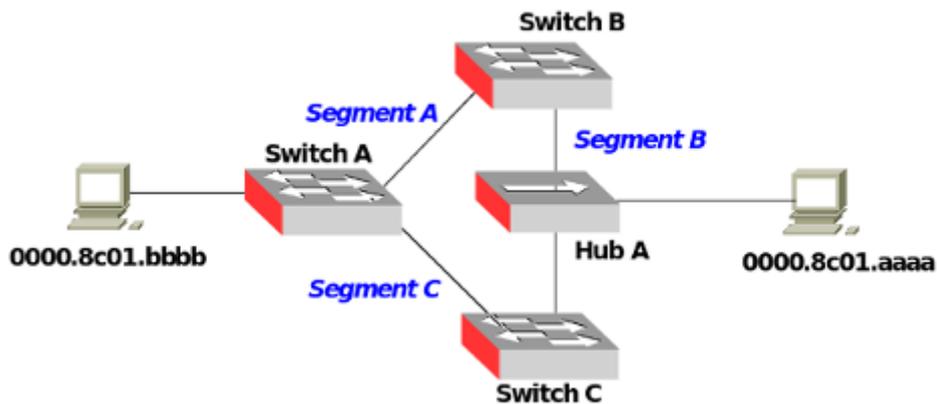
Learning and flooding continue as the switch adds nodes to the lookup tables. Most switches have plenty of memory in a switch for maintaining the lookup tables; but to optimise the use of this memory, they still remove older information so that the switch doesn't waste time searching through stale addresses. To do this, switches use a technique called *ageing*. Basically, when an entry is added to the lookup table for a node, it is given a time-stamp. Each time a frame is received from a node, the time-stamp is updated. The switch has a user-configurable timer that erases the entry after a certain amount of time with no activity from that node. This frees up valuable memory resources for other entries. As can be seen, transparent bridging is a great and essentially maintenance-free way to add and manage all the information a switch needs.

In the example, two nodes share segment 1, while the switch creates independent segments for *bbbb* and *dddd*. In an ideal LAN-switched network, every node would have its own segment. This would eliminate the possibility of collisions and also the need for filtering.

Address Resolution

To allow forwarding and filtering of frames at wire speed, LAN switches should be able to decode MAC addresses very quickly. Since Central Processing Unit (CPU) based lookups are expensive, hardware solutions may be used. Switches maintain address tables just like transparent bridges. They learn the addresses of their neighbours, and when a frame is to be forwarded, they first look up the address table and broadcast only if no entry corresponding to that destination is found. Stations that have not transmitted recently are aged out. This way a small address table can be maintained and the switch can relearn if a station starts transmitting again.

Broadcast Storm



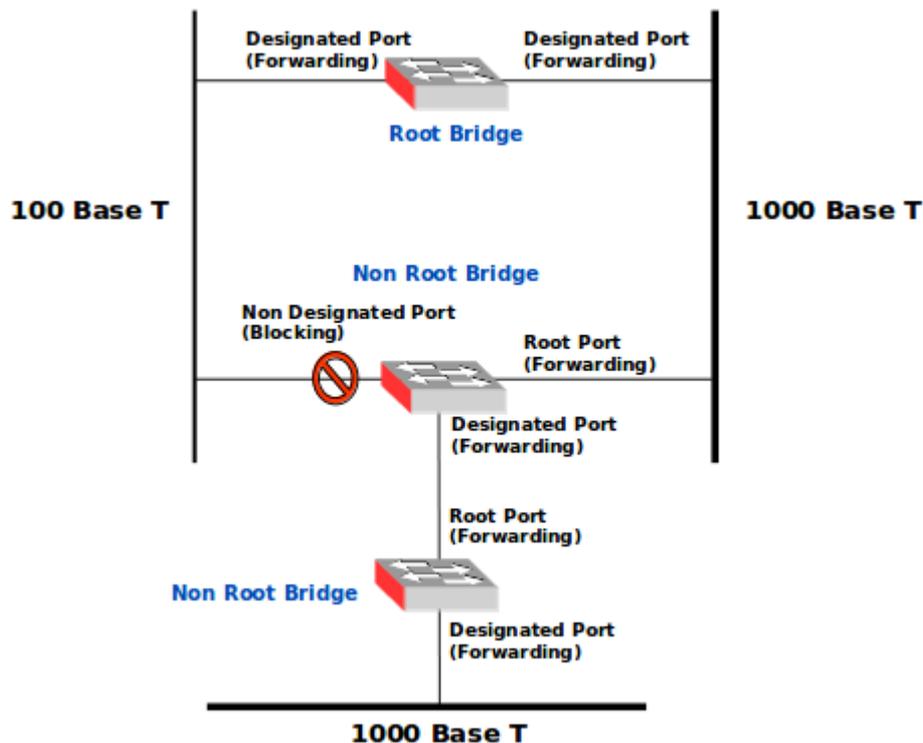
In the example shown in the diagram, even if one of the switches fails, the network will continue to function. The loop provides redundancy, effectively eliminating a single point of failure. However it introduces a new problem. With all of the switches now connected in a loop, a frame from a node could quite possibly come to a switch from two different segments. For example, imagine that *0000.8c01.bbbb* (bbbb) is connected to Switch A, and needs to communicate with *0000.8c01.aaaa* (aaaa) on Segment B. Switch A does not know who *aaaa* is, so it floods the frame.

The frame travels via Segment A or Segment C to the other two switches (B and C). Switch B will add *bbbb* to the lookup table it maintains for Segment A, while Switch C will add it to the lookup table for Segment C. If neither switch has learned the address for *aaaa* yet, they will flood Segment B looking for *aaaa*.

Each switch will take the frame sent by the other switch and flood it back out again immediately, since they still don't know who *aaaa* is. Switch A will receive the frame from each segment and flood it back out on the other segment. This causes a broadcast storm as the frames are broadcast, received and rebroadcast by each switch, resulting in potentially severe network congestion.

4.2 Spanning Tree Protocol

To prevent broadcast storms and other unwanted side effects of looping, Digital Equipment Corporation (DEC) created the Spanning Tree Protocol (STP), which has been standardised as the IEEE 802.1d specification by the IEEE. Essentially, a spanning tree uses STA, which senses that the switch has more than one way to communicate with a node, determines which way is best and blocks out the other path(s). It also keeps track of the other path(s), just in case the primary path is unavailable.



Each switch is assigned a group of IDs, one for the switch itself and one for each port on the switch. The switch's identifier, called the Bridge ID (BID), is 8 bytes long and contains a bridge priority (2 bytes) along with one of the switch's MAC addresses (6 bytes). Each port ID is 16 bits long with two parts: a 6-bit priority setting and a 10-bit port number.

A path cost value is given to each port. The cost is typically based on a guideline established as part of IEEE 802.1d and further enhanced with IEEE 802.1w Rapid STP (RSTP). According to the original specification, cost is 1,000 Mb/s (1 gigabit per second) divided by the bandwidth of the segment connected to the port. Therefore, a 10 Mb/s connection would have a cost of $(1,000/10) 100$. To compensate for the speed of networks increasing beyond the Gb/s range, the standard cost has been modified over time. The new values are:

Data rate	STP Cost - 802.1d-1998	RSTP Cost - 802.1w-2004
4 Mb/s	250	5000000
10 Mb/s	100	2000000
16 Mb/s	62	1250000
100 Mb/s	19	200000
1 Gb/s	4	20000
2 Gb/s	3	10000
10 Gb/s	2	2000

It should also be noted that the path cost can be an arbitrary value assigned by a network administrator in most switches, instead of one of the standard cost values.

Each switch begins a discovery process to choose which network paths it should use for each segment. This information is shared between all the switches by way of special network frames called Bridge Protocol Data Units (BPDU). The BPDU consists of:

- Root BID
 - This is the BID of the current root bridge.
- Path cost to root bridge
 - This determines how far away the root bridge is. For example, if the data has to travel over three 100 Mb/s segments to reach the root bridge, then the cost is $(19 + 19 + 0) = 38$. The segment attached to the root bridge will normally have a path cost of zero.
- Sender BID
 - This is the BID of the switch that sends the BPDU.
- Port ID
 - This is the actual port on the switch that the BPDU was sent from.

A root bridge is chosen based on the results of the BPDU process between the switches. Initially, every switch considers itself the root bridge. When a switch first powers up on the network, it sends out a BPDU with its own BID as the root BID. When the other switches receive the BPDU, they compare the BID to the one they already have stored as the root BID. If the new root BID has a lower value, they replace the saved one. But if the saved root BID is lower, a BPDU is sent to the new switch with this BID as the root BID. When the new switch receives the BPDU, it realises that it is not the root bridge and replaces the root BID in its table with the one it just received. In this way the switch that has the lowest BID is elected by the other switches as the root bridge.

Based on the location of the root bridge, the other switches determine which of their ports has the lowest path cost to the root bridge. These ports are called root ports, and each switch (other than the current root bridge) must have one.

The switches determine who will have designated ports. A designated port is the connection used to send and receive frames on a specific segment. By having only one designated port per segment, all looping issues are resolved.

Designated ports are selected based on the lowest path cost to the root bridge for a segment. Since the root bridge will have a path cost of 0, any ports on it that are connected to segments will become designated ports. For the other switches, the path cost is compared for a given segment. If one port is determined to have a lower path cost, it becomes the designated port for that segment. If two or more ports have the same path cost, then the switch with the lowest BID is chosen.

Once the designated port for a network segment has been chosen, any other ports that connect to that segment become non-designated ports. They block network traffic from taking that path so it can only access that segment through the designated port.

Each switch has a table of BPDUs that it continually updates. The network is now configured as a single spanning tree, with the root bridge as the trunk and all the other switches as branches. Each switch communicates with the root bridge through the root ports, and with each segment through the designated ports, thereby maintaining a loop-free network. In the event that the root bridge begins to fail or have network problems, STP allows the other switches to immediately reconfigure the network with another switch acting as Root Bridge. This process gives a company the ability to have a complex network that is fault-tolerant and yet fairly easy to maintain.

4.2.1 Configuration of a Bridge interface on GNU/Linux

GNU/Linux through the *bridge-utils* offers the functionality to create an internal Ethernet switch and put selected interfaces into it. Control of the bridge is via the *brctl* command. This command gives the configuration options expected of a typical Ethernet switch. It supports functionality like Spanning Tree Protocol (*STP*).

Install bridge-utils

```
$ sudo apt-get install bridge-utils
```

```
$ sudo brctl --help
```

```
Usage: brctl [commands]
```

```
commands:
```

addbr	<bridge>	add bridge
delbr	<bridge>	delete bridge
addif	<bridge> <device>	add interface to bridge
delif	<bridge> <device>	delete interface from bridge
hairpin	<bridge> <port> {on off}	turn hairpin on/off
setageing	<bridge> <time>	set ageing time
setbridgeprio	<bridge> <prio>	set bridge priority
setfd	<bridge> <time>	set bridge forward delay
sethello	<bridge> <time>	set hello time
setmaxage	<bridge> <time>	set max message age
setpathcost	<bridge> <port> <cost>	set path cost
setportprio	<bridge> <port> <prio>	set port priority
show	[<bridge>]	show a list of bridges
showmacs	<bridge>	show a list of mac addrs
showstp	<bridge>	show bridge stp info
stp	<bridge> {on off}	turn stp on/off

4.2.2 Create a bridge and add interfaces

To use the bridge utilities, a bridge must be created and then interfaces added to it. It is important to note that once interfaces are added to the bridge, addressing at the Internet Layer (Open Systems Interconnection (OSI) Network Layer) is applied to the bridge interface and not the physical interface (eth0, eth1, ..). In the example below the interfaces *eth0* and *eth1* are now within the bridge *br0*. IP addressing at the next stage is applied to *br0* (as shown in the right side of the diagram above).

```
$ sudo brctl addbr br0
$ sudo brctl addif br0 eth0 eth1
```

Enable the bridge interface.

```
$ sudo ip link set dev br0 up
```

Review bridge

Now that the bridge is created review it.

```
$ sudo brctl show
bridge name      bridge id          STP enabled      interfaces
br0              8000.2a7c0a401a31 no                eth0
                                                         eth1
```

```
$ sudo brctl showmacs br0
port no         mac addr          is local?      ageing timer
1      00:04:23:b1:8f:e2 no                0.02
1      00:14:22:09:57:0a no                23.24
1      00:60:e0:50:a0:29 no                9.50
1      2a:7c:0a:40:1a:31 yes               0.00
3      32:e5:4e:7d:ad:de no                43.32
4      36:96:9e:e5:38:fc yes               0.00
4      3a:7f:77:2a:1e:46 no                69.38
2      5e:cd:68:17:2e:d1 no                37.73
1      b6:7a:b1:d5:03:8c no                31.97
3      c6:33:b1:f0:f1:a4 yes               0.00
```

4.3 Virtual LANs (VLANs)

A virtual LAN, commonly known as a VLAN, is a logically segmented network mapped over physical hardware. The IEEE 802.1q standard is the predominant protocol.

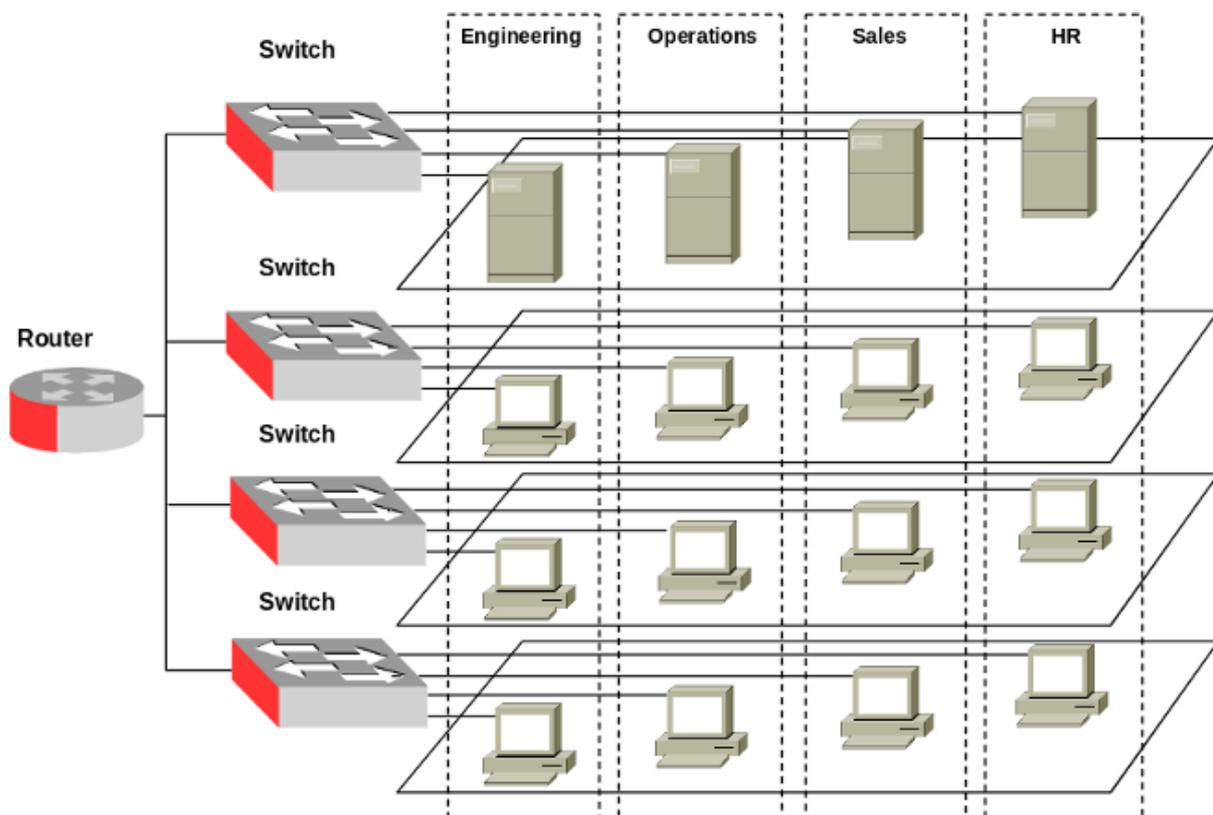
Early VLANs were often configured to reduce the size of the collision domain in a large single Ethernet segment to improve performance. When Ethernet switches made this a non-issue (because they have no collision domain), attention turned to reducing the size of the broadcast domain at the MAC layer. Another purpose of a virtual network is to restrict access to network resources without regard to physical topology of the network, although the strength of this method is debatable.

VLANs operate at layer 2 of the OSI model. Although often a VLAN is configured to map directly to an IP network, or subnet, which gives the appearance it is involved in layer 3.

Switch to switch links and switch to router links are called trunks. A router serves as the backbone for traffic going across different VLANs.

4.3.1 Removing the Physical Boundaries

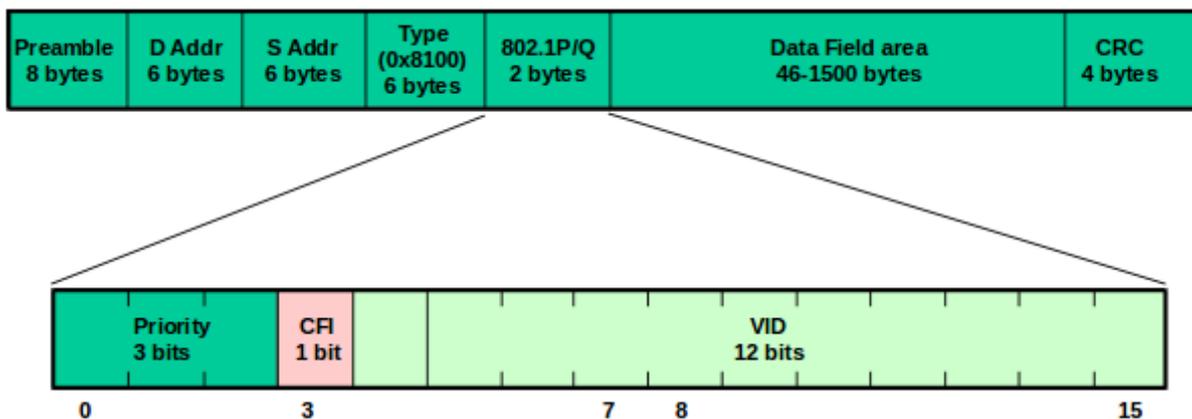
Conceptually, VLANs provide greater segmentation and organisational flexibility. VLAN technology allows network managers to group switch ports and users connected to them into logically defined communities of interest. These groupings can be co-workers within the same department, a cross-functional product team, or diverse users sharing the same network application or software (such as LibreOffice users). Grouping these ports and users into communities of interest, referred to as VLAN organisations, can be accomplished within a single switch, or more powerfully, between connected switches within the enterprise. By grouping ports and users together across multiple switches, VLANs can span single building infrastructures, interconnected buildings, or even Wide Area Networks (WAN). VLANs completely remove the physical constraints of workgroup communications across the enterprise.



VLANs provide the ability for any organisation to be physically dispersed throughout the company while maintaining its group identity. For example, engineering personnel can be

located on the manufacturing floor, in the research and development centre, in the Professional Services demonstration centre, and in the corporate offices, while at the same time all members reside on the same virtual network, sharing traffic only with each other. The graphic above illustrates a typical VLAN architecture that places these employees closer to their assigned areas of management and the people with whom they interact, while maintaining communication integrity within their respective organisation. Today's VLANs better match the way that companies are organised, and allow network managers to more closely align the network to the way that employees work and communicate.

4.3.2 IEEE 802.1P/Q



CRC - Cyclical Redundancy Check

CFI - Canonical Format Indicator is a single-bit flag, always set to zero for Ethernet switches.

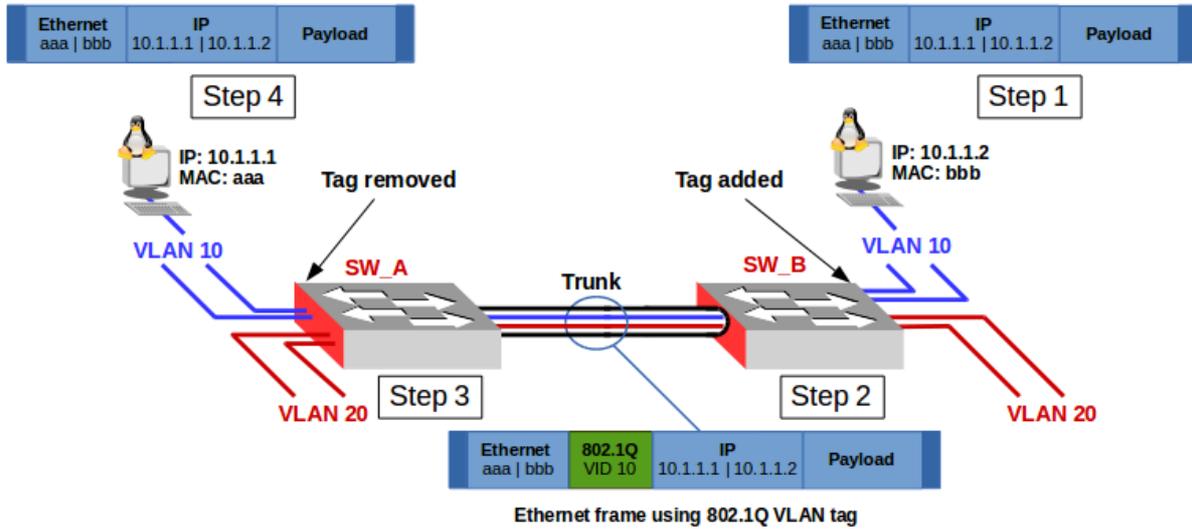
VID - VLAN Identifier $2^{12} = 4096$ VLANs. VID 0 is used to identify priority frames and value 4095 (FFF) is reserved, maximum VLAN configurations are 4,094.

The IEEE 802.1Q specification is the standard method for inserting VLAN membership information into Ethernet frames. A tag field containing VLAN information can be inserted into an Ethernet frame. If a port has an IEEE 802.1Q compliant device attached (such as another switch), these tagged frames can carry VLAN membership information between switches, thus letting a VLAN span multiple switches.

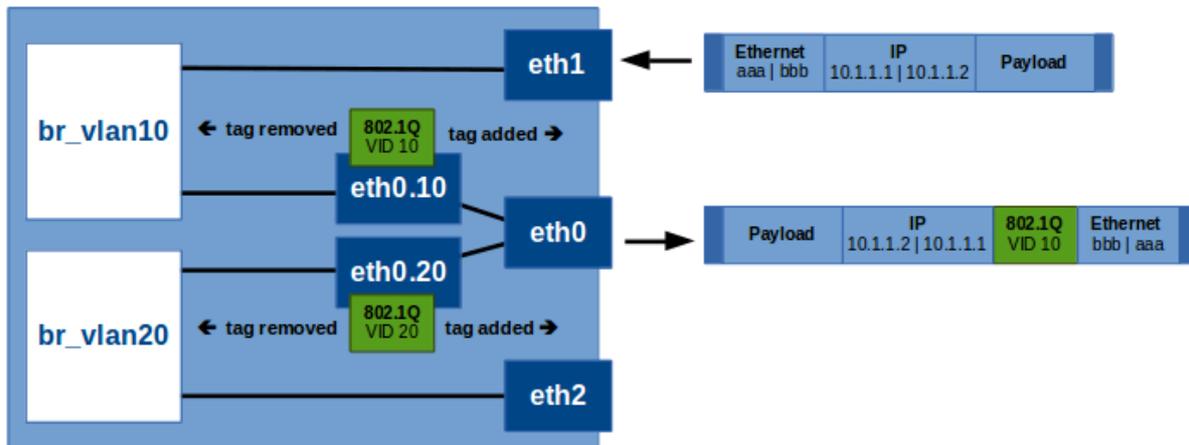
Note that VLAN functionality is shared in the IEEE 802.1 two bytes 3 priority bits. These 3 bits define 8 classes, the highest priority is 7 for say network-critical traffic such as routing, values 5 and 6 for say delay-sensitive applications such as video and VoIP. The 0 value is used as a best-effort default, invoked automatically when no other value has been set.

The priority function of IEEE 802.1 is known as IEEE 802.1P and the VLAN function as IEEE 802.1Q while combined they are referred to as IEEE 802.1P/Q.

IEEE 802.1Q



The diagram above shows a frame traversing the VLAN. Step 1 the host with MAC: *bbb* puts a frame on the wire for MAC: *aaa*. The Switch *SW_B* determines the frame belongs to *VLAN 10*, either by the protocol within the frame or in this example the port it is received on. In Step 2 the *SW_B* switch then encapsulates the frame with an IEEE 802.1Q tag and a Frame Check Sequence (FCS), this tag is then used to identify the VLAN the frame is from on all IEEE 802.1Q enabled switches. The tagged frame is then passed on the trunk to the *SW_A* switch. In Step 3 the *SW_A* determines the frame is for *VLAN 10*, removes the tag and puts the frame out the ports associated with *VLAN 10*. Step 4 The workstation with the MAC: *aaa* receives an untagged frame.



Now dissecting further what happens within the switch. Each VLAN has a bridge configured for it. The interface considered to be the *trunk* between the switches has sub-interfaces configured, one for each VLAN. These sub-interfaces perform the tagging and untagging. The sub-interfaces are added to their respective bridges. Each other port considered to be an *access* port is added to the bridge associated with the VLAN for that port. In the example therefore eth1 is added to the bridge *br_vlan10* and eth2 to *br_vlan20*.

A frame arrives as *eth1* and is passed to the bridge *br_vlan10* and as a result is forwarded to the sub-interface *eth0.10* where an IEEE 802.1Q tag is added to the frame and it is forwarded to the physical interface eth0. The frame leaving eth0 is therefore tagged.

Here is an example of a frame captured on the wire on a *trunk* between two switches using IEEE 802.1Q. Note the Ethernet type field has a value of *0x8100* indicating that the next field is IEEE 802.1Q VLAN. This field contains the value *000000001010* (10) which is the VLAN tag and it follows with a type field of *0x0800* indicating that the next field is the IP header.

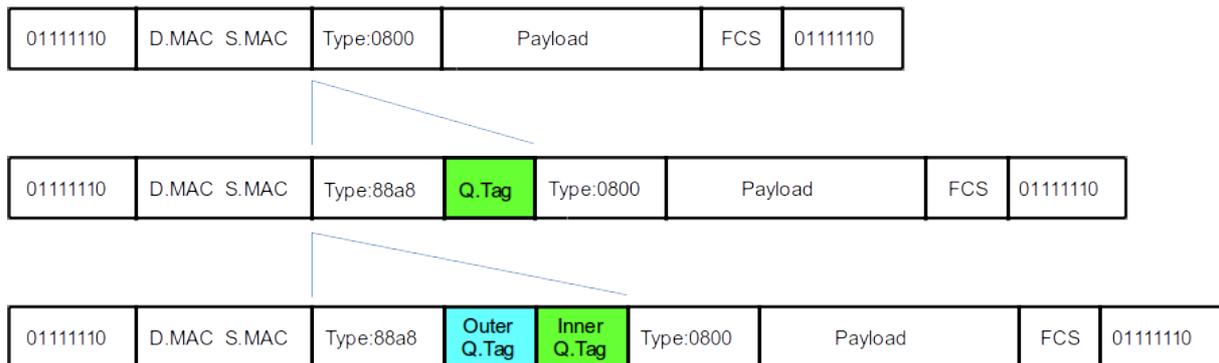
```

Frame: 102 bytes on wire (816 bits)
Ethernet II, Src: d4:ca:6d:61:dd:89, Dst: 00:12:3f:dc:ab:47
  Destination: 00:12:3f:dc:ab:47
  Source: d4:ca:6d:61:dd:89
  Type: 802.1Q Virtual LAN (0x8100)
802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 10
  000. .... .... = Priority: Best Effort (default) (0)
  ...0 .... .... = CFI: Canonical (0)
  .... 0000 0000 1010 = ID: 10
  Type: IP (0x0800)
Internet Protocol Version 4, Src: 10.10.10.40, Dst: 10.10.10.30
Internet Control Message Protocol

```

IEEE 802.1ad

One of the difficulties presented by IEEE 802.1Q is the fact that tags cannot be *stacked*. Imagine a service provider wants to use VLANs to separate services to different customers. As the provider used VLAN tags for that purpose it prevents the customers using VLANs themselves as IEEE 802.1Q does not support VLANs within VLANs. IEEE 802.1ad is an amendment to the IEEE 802.1Q VLAN standard. It provides for the stacking of VLANs within VLANs which is called names such as provider stacking, stacked VLANs, Q-in-Q or simply QinQ. IEEE 802.ad allows for multiple VLAN headers to be inserted into a single frame an essential capability for implementing Metro Ethernet. QinQ allows multiple VLAN tags in an Ethernet frame; together these tags constitute a tag stack.



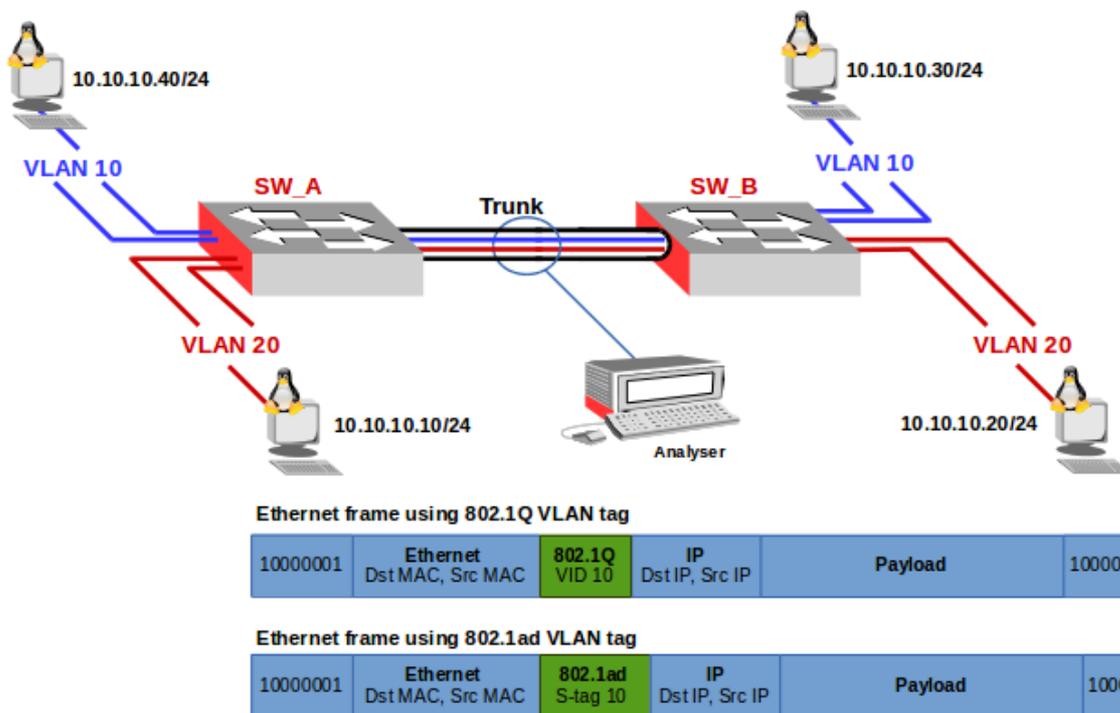
This first example demonstrates the use of IEEE 802.1ad instead of IEEE 802.1Q where there is no stacking of VLANs. Here the Ethernet type field contains *0x88a8* such that the next field is treated as IEEE 802.1ad. Like the earlier example for IEEE 802.1Q this field contains a VLAN ID of *000000001010* (10) and a type field of *0x0800* to indicate that the next field is the IP header.

```

Frame: 102 bytes on wire (816 bits)
Ethernet II, Src: d4:ca:6d:61:dd:89, Dst: 00:12:3f:dc:ab:47
  Destination: 00:12:3f:dc:ab:47
  Source: d4:ca:6d:61:dd:89
  Type: 802.1ad Provider Bridge (Q-in-Q) (0x88a8)
IEEE 802.1ad, ID: 10
  000. .... .. = Priority: 0
  ...0 .... .. = DEI: 0
  ... 0000 0000 1010 = ID: 10
  Type: IP (0x0800)
Internet Protocol Version 4, Src: 10.10.10.40, Dst: 10.10.10.30
Internet Control Message Protocol

```

4.4 Provider tagging



Considering the graphic the traffic in the trunks will be treated by *ISP_1* and *ISP_2* as *access* ports despite they containing VLAN tags already. In fact *ISP_1* and *ISP_2* ignore these tags as they are customer tags (C-tags).

Before forwarding to the other ISP switch each switch adds a provider tag (S-tag) of 1001. Thus the C-tag is stacked inside the S-tag from the provider. This can be seen by considering the frame capture from the wire between *ISP_1* and *ISP_2* below. In this packet a customer IEEE 802.1Q VLAN is outer labelled with an ISP IEEE 802.1ad (Q-in-Q) S-tag of *001111101001* (1001). The Ethernet type field indicates *0x88a8* the next header containing an IEEE 802.1ad tag. This headers type field indicates that the next header is *0x8100* IEEE 802.1Q. This headers type field in turn contains a type field of *0x0800* indicating the next header is the IP header. So in this example a customer IEEE 802.1Q tag is stacked by an IEEE 802.1ad S-tag.

```

Frame: 106 bytes on wire (848 bits)
Ethernet II, Src: d4:ca:6d:61:dd:89, Dst: 00:12:3f:dc:ab:47
  Destination: 00:12:3f:dc:ab:47
  Source: d4:ca:6d:61:dd:89
  Type: 802.1ad Provider Bridge (Q-in-Q) (0x88a8)
IEEE 802.1ad, ID: 1001
  000. .... .... = Priority: 0
  ...0 .... .... = DEI: 0
  .... 0011 1110 1001 = ID: 1001
  Type: 802.1Q Virtual LAN (0x8100)
802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 10
  000. .... .... = Priority: Best Effort (default) (0)
  ...0 .... .... = CFI: Canonical (0)
  .... 0000 0000 1010 = ID: 10
  Type: IP (0x0800)
Internet Protocol Version 4, Src: 10.10.10.40, Dst: 10.10.10.30
Internet Control Message Protocol

```

Now consider the frame below where the customer tag is also IEEE 802.1ad. The Ethernet type field is *0x88a8* indicating the next header is IEEE 802.1ad Q-in-Q. In this header two tags can be seen, a provider S-tag of *001111101001 (1001)* with a customer C-tag of *000000001010 (10)*. This headers type field is *0x0800* indicating the next header is the IP header.

```

Frame: 106 bytes on wire (848 bits)
Ethernet II, Src: d4:ca:6d:61:dd:89, Dst: 00:12:3f:dc:ab:47
  Destination: 00:12:3f:dc:ab:47
  Source: d4:ca:6d:61:dd:89
  Type: 802.1ad Provider Bridge (Q-in-Q) (0x88a8)
IEEE 802.1ad, S-VID: 1001, C-VID: 10
  000. .... .... = Priority: 0
  ...0 .... .... = DEI: 0
  .... 0011 1110 1001 = ID: 1001
  000. .... .... = Priority: 0
  ...0 .... .... = DEI: 0
  .... 0000 0000 1010 = ID: 10
  Type: IP (0x0800)
Internet Protocol Version 4, Src: 10.10.10.40, Dst: 10.10.10.30
Internet Control Message Protocol

```

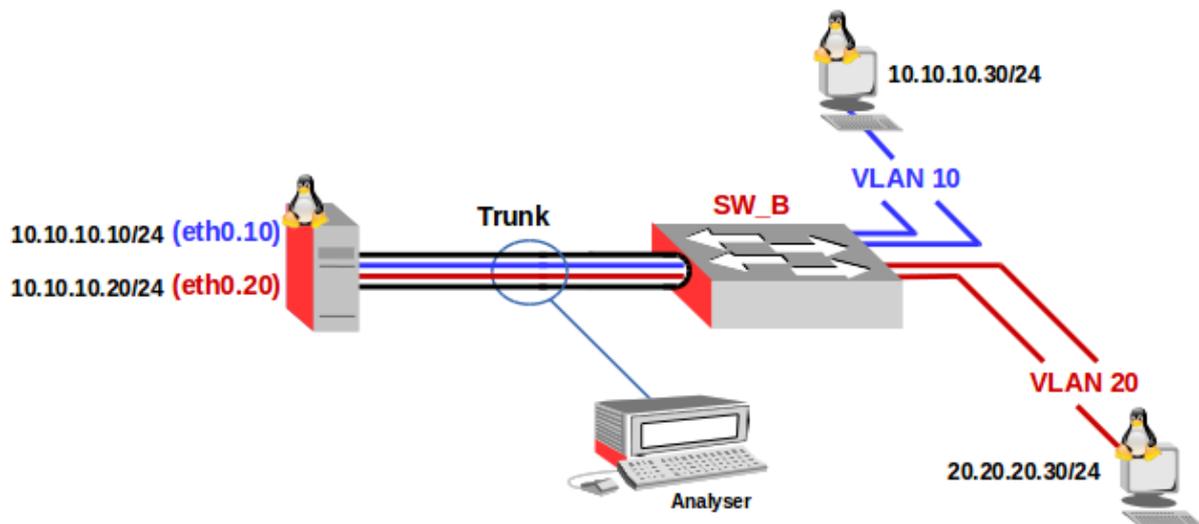
This packet shows a customer IEEE 802.1ad (Q-in-Q) label as an inner C-tag *0000000010100 (20)* which is also outer labelled with an ISP IEEE 802.1ad (Q-in-Q) S-tag *001111101001 (1001)*.

```

Frame: 106 bytes on wire (848 bits)
Ethernet II, Src: d4:ca:6d:61:dd:89, Dst: 00:12:3f:dc:ab:47
  Destination: 00:12:3f:dc:ab:47
  Source: d4:ca:6d:61:dd:89
  Type: 802.1ad Provider Bridge (Q-in-Q) (0x88a8)
IEEE 802.1ad, S-VID: 1001, C-VID: 20
000. .... .. = Priority: 0
...0 .... .. = DEI: 0
... 0011 1110 1001 = ID: 1001
000. .... .. = Priority: 0
...0 .... .. = DEI: 0
... 0000 0001 0100 = ID: 20
Type: IP (0x0800)
Internet Protocol Version 4, Src: 10.10.10.40, Dst: 10.10.10.30
Internet Control Message Protocol

```

4.5 VLANs on GNU/Linux



iproute2 supports IEEE 802.1Q VLAN and IEEE 802.1ad VLAN Stacking. IEEE 802.1Q or IEEE 802.1ad traffic received on the *eth0* interface will have the VLAN tag removed and the frame passed to the VLAN interface. Traffic passing out the sub-interface will have the IEEE 802.1Q or IEEE 802.1ad tag added. Create the sub-interfaces with the following commands. These create sub-interfaces for VLAN ID 10 and VLAN ID 20 on the *eth0* interface and gives them labels of *eth0.10* and *eth0.20*.

Confirm that the 8021q kernel module is loaded and if not then load it.

```

$ lsmod |grep 8021q
$ sudo modprobe 8021q
$ lsmod |grep 8021q
8021q          18824  0
garp           13025  1 8021q

```

Add sub-interfaces for each VLAN expected on the physical eth0 interface.

```
$ sudo ip link add link eth0 name eth0.10 type vlan id 10
$ sudo ip link add link eth0 name eth0.20 type vlan id 20
```

Add IP addresses to the sub-interfaces.

```
$ sudo ip addr add 10.10.10.10/24 dev eth0.10
$ sudo ip addr add 20.20.20.20/24 dev eth0.20
```

Bring up the interface and its new sub-interfaces.

```
$ sudo ip link set dev eth0 up
$ sudo ip link set dev eth0.10 up
$ sudo ip link set dev eth0.20 up
```

Making VLAN changes persistent on Debian GNU/Linux.

Add the following lines to the `/etc/network/interfaces` file.

```
## VLAN 10 on eth0 ##

auto eth0.10
iface eth0.10 inet static
    address 10.10.10.10
    netmask 255.255.255.0
    vlan-raw-device eth0

## VLAN 20 on eth0 ##

auto eth0.20
iface eth0.20 inet static
    address 20.20.20.20
    netmask 255.255.255.0
    vlan-raw-device eth0
```

Review by monitoring packets on the trunk interface between switches.

```
Frame: 102 bytes on wire (816 bits)
Ethernet II, Src: 00:12:3f:dc:ab:47 (GNU/Linux PC), Dst: d4:ca:6d:61:dd:89
    Destination: d4:ca:6d:61:dd:89
    Source: 00:12:3f:dc:ab:47 (GNU/Linux PC)
    Type: 802.1Q Virtual LAN (0x8100)
802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 10
    000. .... .. = Priority: Best Effort (default) (0)
    ...0 .... .. = CFI: Canonical (0)
    .... 0000 0000 1010 = ID: 10
    Type: IP (0x0800)
Internet Protocol Version 4, Src: 10.10.10.10, Dst: 10.10.10.30
Internet Control Message Protocol
```

4.5.1 IEEE 802.1ad support on GNU/Linux

Support for IEEE 802.1ad was incorporated in the GNU/Linux kernel from Kernel version 3.10. Check the kernel version of your system and if less than 3.10, download the latest stable kernel, compile and use it. Also check the version of *iproute** installed, it needs to be a version 3.10 or higher.

```
$ uname -r
3.14.0-4-686-pae

$ dpkg -l iproute*

Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Inst/Conf-files/Unpacked/half-f-inst/Trig-await/Trig-pend
|/ Err?=(none)/Reinst-required (Status,Err: uppercase=bad)
||/ Name                Version                Architecture           Description
+++-+-----+-----+-----+-----+-----+-----+-----+-----+
=====
rc  iproute                20120521-3+b3         i386                   networking and traffic control tools
un  iproute-doc            <none>                 i386                   (no description available)
ii  iproute2               3.12.0-2~bpo70+1     i386                   networking and traffic control tools
un  iproute2-doc           <none>                 i386                   (no description available)
```

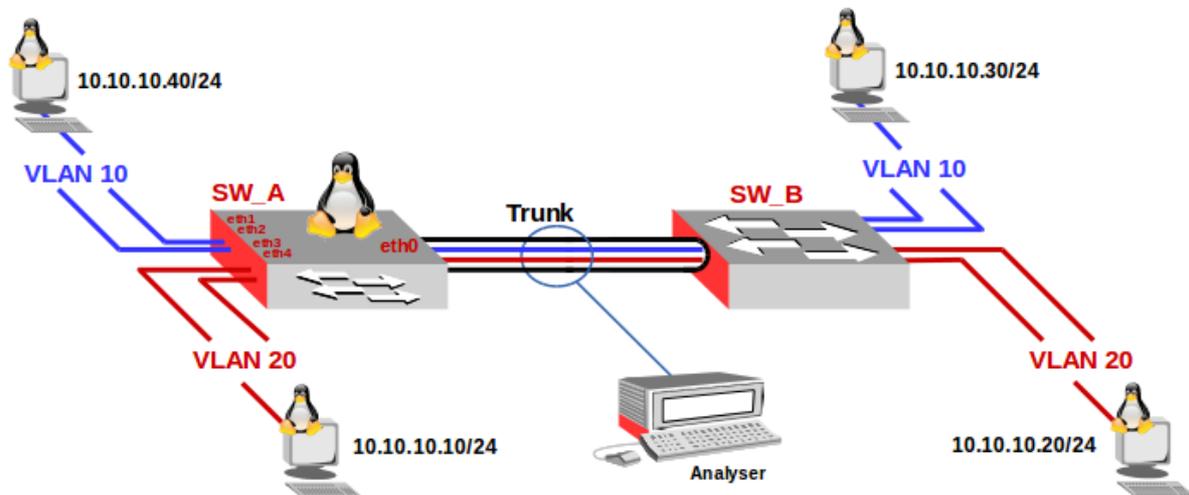
Configure interfaces as shown already with the addition of *proto 802.1ad* with creating the sub-interfaces.

```
$ sudo ip link add link eth0 name eth0.10 type vlan proto 802.1ad id 10
$ sudo ip link add link eth0 name eth0.20 type vlan proto 802.1ad id 20
$ sudo ip addr add 10.10.10.10/24 dev eth0.10
$ sudo ip addr add 20.20.20.20/24 dev eth0.20
$ sudo ip link set dev eth0 up
$ sudo ip link set dev eth0.10 up
$ sudo ip link set dev eth0.20 up
```

Monitor the trunk link and notice that the GNU/Linux workstation is now terminating directly in the IEEE 802.1ad C-tag interfaces on the *eth0* sub-interfaces.

```
Frame: 102 bytes on wire (816 bits)
Ethernet II, Src: 00:12:3f:dc:ab:47, Dst: d4:ca:6d:61:dd:89
  Destination: d4:ca:6d:61:dd:89
  Source: 00:12:3f:dc:ab:47
  Type: 802.1ad Provider Bridge (Q-in-Q) (0x88a8)
IEEE 802.1ad, ID: 10
  000. .... .... .... = Priority: 0
  ...0 .... .... .... = DEI: 0
  .... 0000 0000 1010 = ID: 10
  Type: IP (0x0800)
Internet Protocol Version 4, Src: 10.10.10.10, Dst: 10.10.10.30
Internet Control Message Protocol
```

4.5.2 IEEE 802.1ad support on GNU/Linux as a switch



In this case GNU/Linux workstation will operate as a bridge with a VLAN interface.

Create the VLAN subinterfaces to deal with the incoming trunk interface containing the VLANs on physical interface *eth0*.

```
$ sudo ip link add link eth0 name eth0.10 type vlan proto 802.1ad id 10
$ sudo ip link add link eth0 name eth0.20 type vlan proto 802.1ad id 20
$ sudo ip link set dev eth0 up
$ sudo ip link set dev eth0.10 up
$ sudo ip link set dev eth0.20 up
```

Bring up the interfaces that connect to the LANs.

```
$ sudo ip link set dev eth1 up
$ sudo ip link set dev eth2 up
$ sudo ip link set dev eth3 up
$ sudo ip link set dev eth4 up
```

Create bridges to link the VLANs to their appropriate interfaces.

```
$ sudo brctl addbr br_vlan_10
$ sudo brctl addbr br_vlan_20
```

Assign interfaces to the various bridges.

```
$ sudo brctl addif br_vlan_10 eth0.10 eth1 eth2
$ sudo brctl addif br_vlan_20 eth0.20 eth3 eth4
```

Bring up the bridges.

```
$ sudo ip link set dev br_vlan_10 up
$ sudo ip link set dev br_vlan_20 up
```

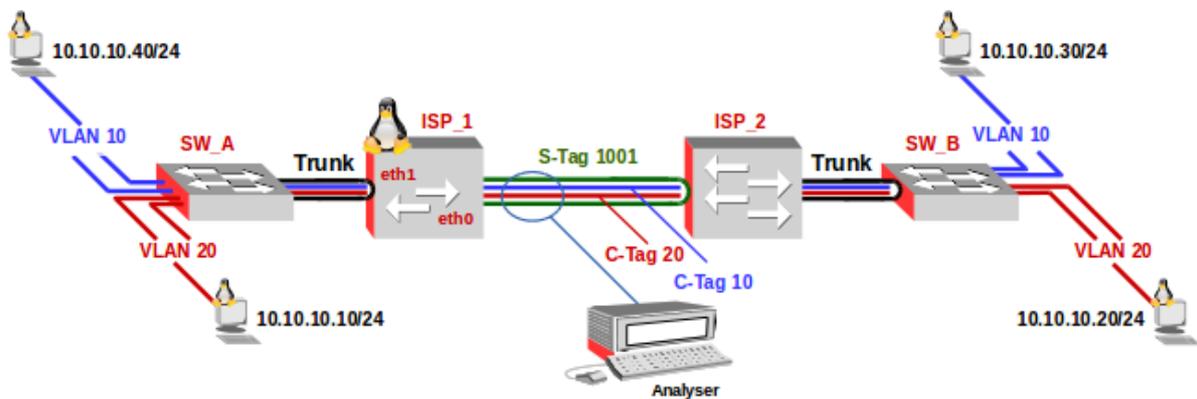
Review the packets on the wire.

```

Frame: 74 bytes on wire (592 bits)
Ethernet II, Src: d4:ca:6d:61:dd:89, Dst: 00:0c:42:8b:73:e4
  Destination: 00:0c:42:8b:73:e4
  Source: d4:ca:6d:61:dd:89
  Type: 802.1Q Virtual LAN (0x8100)
802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 10
  000. .... .. = Priority: Best Effort (default) (0)
  ...0 .... .. = CFI: Canonical (0)
  .... 0000 0000 1010 = ID: 10
  Type: IP (0x0800)
Internet Protocol Version 4, Src: 10.10.10.30, Dst: 10.10.10.40
Internet Control Message Protocol

```

4.6 GNU/Linux as a Service Provider bridge



Create a sub-interface on `eth0` to handle the VLAN S-tag `1001` and bring the physical and sub-interface up.

```

$ sudo ip link add link eth0 name eth0.1001 type vlan proto 802.1ad id 1001

$ sudo ip link set dev eth0 up
$ sudo ip link set dev eth0.1001 up

```

Bring up the `eth1` interface which will be connected to the trunk from `SW_A`.

```

$ sudo ip link set dev eth1 up

```

Create a bridge `br_vlan_1001` and put the `eth0.1001` sub-interface and `eth1` into it. Then bring the bridge up.

```

$ sudo brctl addbr br_vlan_1001
$ sudo brctl addif br_vlan_1001 eth0.1001 eth1
$ sudo ip link set dev br_vlan_1001 up

```

Now monitor the traffic on the wire between the provider switches. Note the double tag with an S-tag of `1001` and a C-tag of `10`.

```
Frame: 78 bytes on wire (624 bits)
Ethernet II, Src: d4:ca:6d:61:dd:89, Dst: 00:0c:42:8b:73:e4
  Destination: 00:0c:42:8b:73:e4
  Source: d4:ca:6d:61:dd:89
  Type: 802.1Q Virtual LAN (0x8100)
802.1Q Virtual LAN, PRI: 0, CFI: 0, ID: 1001
  000. .... = Priority: Best Effort (default) (0)
  ...0 .... = CFI: Canonical (0)
  .... 0011 1110 1001 = ID: 1001
  Type: 802.1ad Provider Bridge (Q-in-Q) (0x88a8)
IEEE 802.1ad, ID: 10
  000. .... = Priority: 0
  ...0 .... = DEI: 0
  .... 0000 0000 1010 = ID: 10
  Type: IP (0x0800)
Internet Protocol Version 4, Src: 10.10.10.30, Dst: 10.10.10.40
Internet Control Message Protocol
```

5. Internet Protocol

IP version 4 (IPv4 or IP) was defined initially in 1980 and finalised in RFC 791 in 1981. It has been the mainstay of the Internet ever since though the pressure on its limited address space of 4.3 billion addresses (2^{32}) is now telling which is forcing change to IPv6 with its 3.4×10^{38} addresses (2^{128}).

The IPv4 address defines the host at the Network/Internet layer and it has two section or parts. The left part represents network identification and the right part represents the node identification. In consideration of the point mentioned above (four numbers between 0-255, or 32 bits or four bytes), each byte represents either the network or the node.

There are some restrictions: 0 (for example, 0.0.0.0) in the network space is reserved default routing and 127 (for example, 127.0.0.1) is reserved for the local loopback or local host. 0 in the node part refers to this network (for example, 192.168.0.0) and 255 is reserved for sending packets to all devices, this is known as broadcast (for example, 198.162.255.255). There may be different types of networks or addresses in the different assignments:

- *Class A* (network.host.host.host): 1.0.0.1 to 126.254.254.254 (126 networks, 16 million nodes) define the large networks. The binary standard is: 0 + 7 network bits + 24 node bits.
- *Class B* (network.network.host.host): 128.1.0.1 to 191.255.254.254 (16K networks, 65K nodes); (usually, the first node byte is used to identify subnets within an institution). The binary standard is 10 + 14 network bits + 16 node bits.
- *Class C* (net.net.net.host): 192.1.1.1 to 223.255.255.254 (2 million of networks, 254 nodes). The binary standard is 110 + 21 network bits + 8 node bits.
- *Classes D and E* (network.network.network.host): 224.1.1.1 to 255.255.255.254 reserved for multicast (from one node to a set of nodes that form part of the group) and experimental purposes.

Some address ranges have been reserved so that they do not correspond to public networks, and are considered to be private networks. These are nterconnected computers without external connection; messages between them will not be sent through Internet, but through an intranet. These address ranges are class A 10.0.0.0 to 10.255.255.255, class B 172.16.0.0 to 172.31.0.0 and class C 192.168.0.0 to 192.168.255.0.

The broadcast address is special, because each node in a network listens to all the messages received with the broadcast destination address (as well as its own address of course). This address makes it possible to send datagrams (generally routing information and warning messages) to a network and all nodes on the network will be able to read them. For example, when ARP tries to find the Ethernet address corresponding to an IP, it uses a broadcast message, which is sent to all the machines on the network at the same time. Each node in the network reads this message and compares the IP that is being searched and sends back a message to the sender node if they match.

Two concepts that are related to the point described above are subnets and routing between these subnets. Subnets subdivide the node part into smaller networks within the same network, so as to, for example, improve the traffic. A subnet is in charge of sending traffic to certain IP address ranges, extending to the same concept of Class A, B and C networks, but only applying this rerouting in the IP node part. The number of bits interpreted as a subnet identifier is provided by a netmask, which is a 32-bit number (as is an IP). In order to obtain the subnet identifier, a logical AND operation is performed between the mask and the IP, which will provide the subnet IP. For example, an institution with a B class network, with the network 172.17.0.0, would therefore have a netmask with number 255.255.0.0 i.e. 172.17.0.0, 255.255.0.0 or written another way 172.17.0.0/16. Internally, this network is formed by small networks (one per floor in the building, for example). In this way, the range of addresses is reassigned in 20 subnets (floors in the example), 172.17.10.0 to 172.17.20.0. The point that connects all these floors, called the backbone, has its own address, for example 172.17.1.0.

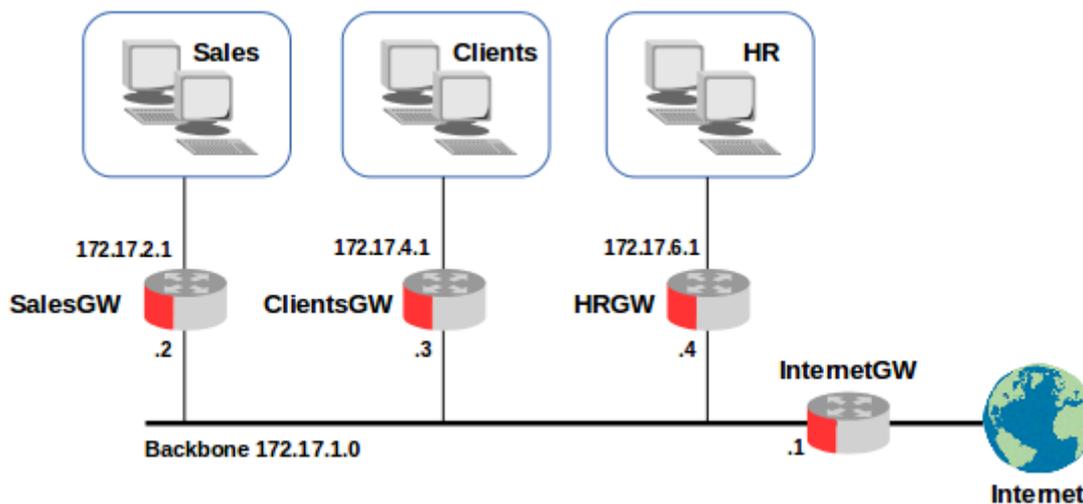
These subnets share the same network IP 172.17.0.0, whereas the third is used to identify each of the subnets within it (which is why it will use the netmask 255.255.255.0 or using netmask length format /24 ($8 \times 1 + 8 \times 1 + 8 \times 1 = 24 \times 1$). Subnets are 172.17.1.0/24, 172.17.2.0/24, 172.17.3.0/24, ...172.17.254.0/24. The subnet 172.17.255.0/24 is a reserved subnet by the RFC however most routing implementations will allow it to be routed also.

The second concept, routing, represents the mode in which the messages are sent through the subnets. For example, there are three departments with Ethernet subnets:

1. Sales (subnet 172.17.2.0),
2. Clients (subnet 172.17.4.0),
3. Human Resources (HR), (subnet 172.17.6.0)
4. Backbone with GbE (subnet 172.17.1.0).

In order to route the packets between the computers on the three networks, three gateways or routers are required that will each have two network interfaces to switch between Ethernet and GbE plus a gateway to connect to the Internet. These would be:

1. SalesGW IPs: 172.17.2.1 and 172.17.1.2,
2. ClientsGW IPs: 172.17.4.1 and 172.17.1.3
3. HRGW IPs: 172.17.6.1 and 172.17.1.4, in other words, one IP on the subnet side and another on the backbone side.
4. InternetGW IP: 172.17.1.1 and the Internet side is determined by the companies Internet Service Provider (ISP).



When messages are sent between devices in the Sales area, it is not necessary to leave the gateway, as the TCP/IP will find the destination device directly. The problem arises when the Sales device wishes to send a packet to a device on the HR subnet. The message must pass through the two respective routers. When Sales *identifies* that HR is on another network, it sends the packet through the SalesGW router, which in turn sends it to HRGW, which, in turn, sends it to the respective HR device. The advantage of having subnets is obvious, given that the traffic between all the Sales devices, for example, will not affect the Clients or HR devices (although this is more complex and expensive in terms of designing and building the network).

TCP/IP routers use a table to route the packets between the different networks. A special route exists in the table to the network 0.0.0.0 which is a route of last resort. All the IP addresses on the Internet match with this route, as none of the 32 bits are necessary; they are sent through the default gateway router to the indicated network (assuming it can route to the required network). In the SalesGW router, for example, the table would be:

Address	Mask	Gateway	Interface
172.17.1.0	255.255.255.0 /24	-	eth1
172.17.4.0	255.255.255.0 /24	172.17.1.2	eth1
172.17.6.0	255.255.255.0 /24	172.17.1.3	eth1
0.0.0.0	0.0.0.0	172.17.2.1	eth1
172.17.2.0	255.255.255.0 /24	-	eth0

The '-' means that the machine is directly connected and does not need routing. The procedure for identifying whether routing is required or not consists of performing a very simple operation with the two logic ANDs (subnet AND mask and origin AND mask) and comparing the two results.

If they match, there is no routing required, if they are not a match then the packet is sent to the respective gateway router for onward forwarding. Each device must have its default gateway router pre-configured so it knows where to send such packets.

For example, a message from 172.17.2.4 to 172.17.2.6 would mean:

- $172.17.2.4 \text{ AND } 255.255.255.0 = 172.17.2.0$
- $172.17.2.6 \text{ AND } 255.255.255.0 = 172.17.2.0$

As the results are the same, there would be no routing required, simply a local ARP request to find the MAC address of the destination device. On the other hand, for a packet from 172.17.2.4 to 172.17.6.6 routing will occur via the gateway router 172.17.2.1 with an interface change (eth0 to eth1) to 172.17.1.1 and from here to 172.17.1.2 with another interface change (eth1 to eth0) and then the packet will be forwarded to 172.17.6.6. Routes are matched in the routing table with the shortest mask first and the 0.0.0.0/0 default route is only used as a route of last resort.

In order to build the routing tables, the *route* command can be used to specify routes in the routes table, these are called *static* routes. However, for more complex networks such manual programming is unrealistic and dynamic building of the routing tables is necessary. For dynamic routing an Internal Gateway Protocol (IGP) like the Open Shortest Path First (OSPF) protocol or, between independent systems, an External Gateway Protocol (EGP) like Border Gateway Protocol (BGP) is used.

The *quagga* package is the GNU/Linux routing daemon and it supports Border Gateway Protocol version 4 (BGP4), BGP4 plus (BGP4+), OSPF version 2 (OSPFv2), OSPF version 3 (OSPFv3), Intermediate System to Intermediate System (IS-IS), Routing Internet Protocol (RIP), RIP version 2 (RIPv2), and RIP Next Generation (RIPng).

To install a host on an existing network, it is necessary to have the following information, obtained from the network provider or the administrator:

- node IP address
- network IP address
- broadcast address
- netmask address / netmask length
- gateway router address
- DNS server address

If a network is being established that will never have an Internet connection, any IP addresses scheme can be used, but it is advisable to maintain an appropriate order corresponding to the size of the network that will be needed. This avoids administrative problems within the network. This is how a network and nodes are defined for a private network (consideration must be taken, as, if the device is connected to the network, it can inconvenience another user to whom this address has been assigned already): node address 192.168.110.23, netmask 255.255.255.0 (/24), net part 192.168.110., node part .23, net address 192.168.110.0, broadcast address 192.168.110.255.

5.1 GNU/Linux IP networking (*iproute2*)

GNU/Linux depended on the *net-tools* package for network for many years, you may indeed be familiar with some of them and even use them still today. Tools like *arp*, *hostname*, *ifconfig*, *netstat* and *route* are well understood and used tools. However they were seen to be a loose collection of tools without common structure and the *iproute2* package of tools for controlling TCP and UDP IP networking and a new functionality of *network traffic control* for both IPv4 and IPv6 networks was released in 1999 (Note:: *network traffic control* did not exist in *net-tools* previously. *iproute2* has additional functionality and a common command framework much like the command line of a router. These tools are all under the *ip* command in the GNU/Linux shell. As a result of the change many distributions continue to have the *net-tools* co-existing alongside *iproute2*, for compatibility, supporting older scripts.

Here is a comparative list of the *net-tools* alongside their replacement *iproute2* tool.

Purpose	net-tools	iproute2
Address and link configuration	ifconfig	ip addr, ip link
Routing tables	route	ip route
Neighbours	arp	ip neigh
Virtual LAN (VLAN)	vconfig	ip link
Tunnels	iptunnel	ip tunnel
Multicast	ipmaddr	ip maddr
Network Traffic Control		ip rule
Statistics	netstat	ss

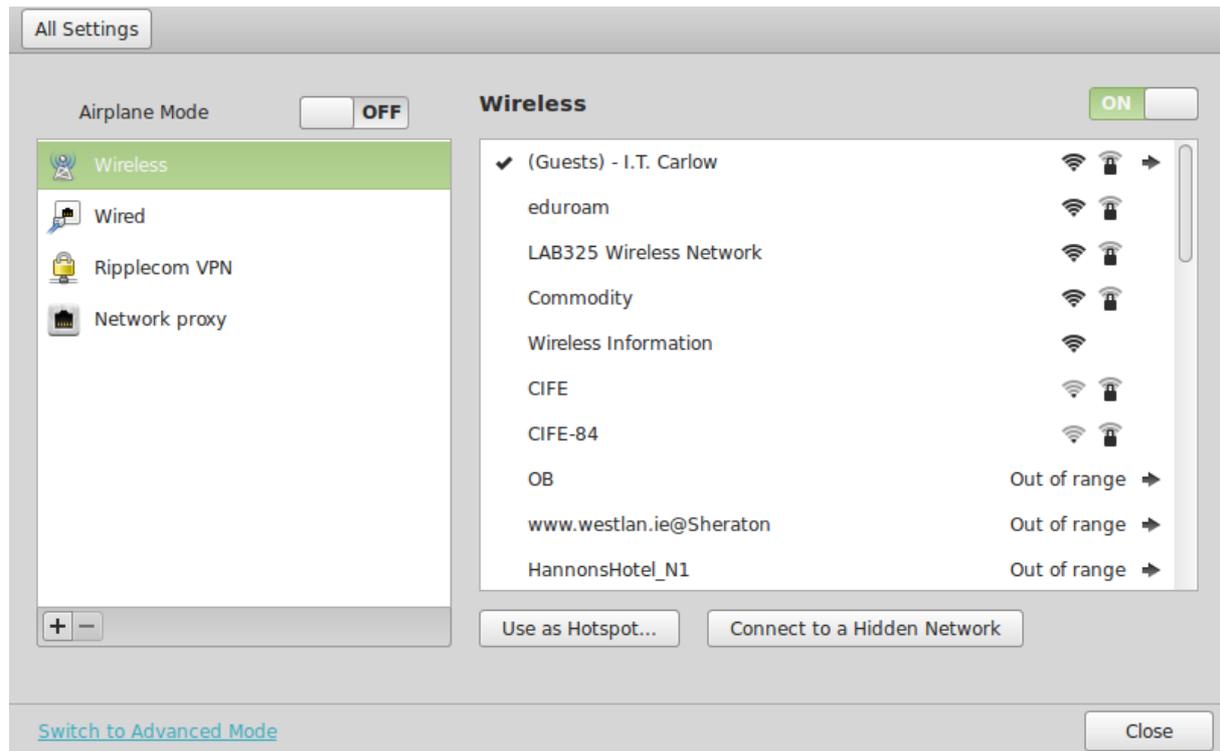
5.1.1 *iproute2 ip command*

The *ip* command offers a simple map with its *--help* option switch. For the rest of this section I will refer to the network diagram below, the commands will show the configuration of interfaces using the *ip* command from *iproute2*.

```
$ ip --help
Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }
       ip [ -force ] -batch filename
where OBJECT := { link | addr | addrlabel | route | rule | neigh | ntable |
                 tunnel | tuntap | maddr | mroute | mrule | monitor | xfrm |
                 netns | l2tp | tcp_metrics | token }
OPTIONS := { -V[ersion] | -s[tatistics] | -d[etails] | -r[esolve] |
             -f[amily] { inet | inet6 | ipx | dnet | bridge | link } |
             -4 | -6 | -I | -D | -B | -0 |
             -l[oops] { maximum-addr-flush-attempts } |
             -o[neline] | -t[imestamp] | -b[atch] [filename] |
             -rc[vbuf] [size]}
```

5.1.2 Network Manager (*network-manager*)

GNU/Linux distributions come today with a Network Manager (*network-manager*). This package is designed to remove the complexity of networking for users. *network-manager* will try and determine the correct network interface to use at any given circumstance and makes it very easy to establish WiFi connections with the Gnome applet (*nm-applet*). See the example in the diagram.



While this is a very useful feature for a laptop user it can be annoying when working with servers or network testing as unexpected and sometimes undesirable results can occur. The current status of *network-manager* can be achieved with the *nmcli* command.

```
$ nmcli nm
RUNNING STATE      WIFI-HARDWARE WIFI    WWAN-HARDWARE WWAN
running  connected  enabled  enabled  enabled  disabled
```

Disabling (and enabling) network manager is done with the *start/stop network-manager* commands. Here is an example stopping the manager and then restarting it.

```
$ sudo service network-manager stop
network-manager stop/waiting

$ sudo service network-manager start
network-manager start/running, process 5656
```

For a server or a device that will act as a router it is better to permanently disable the *network-manager*, create a file *network-manager.override* in */etc/init* containing the word *manual* and upon next reboot the *network-manager* will not be started.

```
$ sudo echo "manual" | tee /etc/init/network-manager.override
manual
```

Confirm the *network-manager* is off.

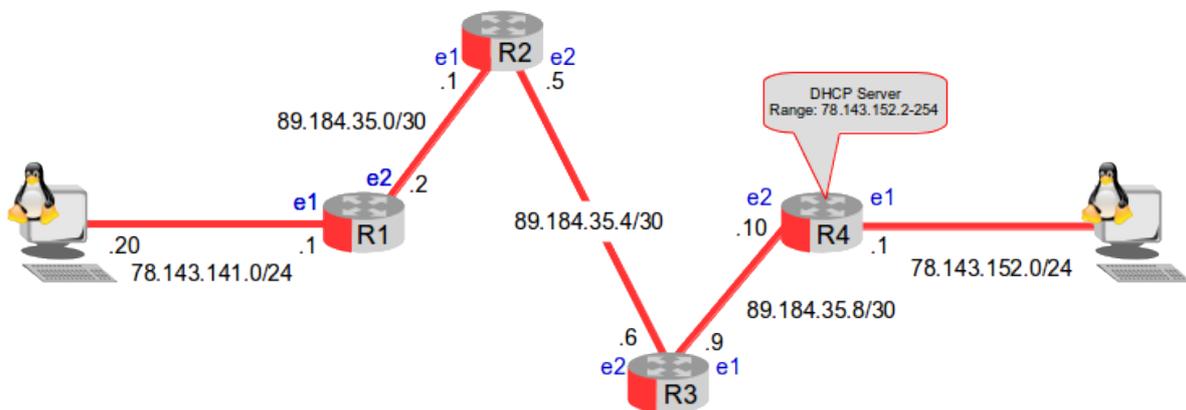
```
$ nmcli nm
RUNNING          STATE          WIFI-HARDWARE   WIFI           WWAN-HARDWARE   WWAN
not running      unknown       unknown        unknown       unknown         unknown
```

5.1.3 Check there is no configuration in */etc/network/interfaces*

To use the *ip* commands of *iproute* it is important to insure that there are no manual addresses configured in */etc/network/interfaces* file. Remove any entries except those for the loopback interface *lo*.

```
$ cat /etc/network/interfaces
# interfaces(5) file used by ifup(8) and ifdown(8)
auto lo
iface lo inet loopback
```

The configurations below are based on the following IPv4 network diagram.



5.2 Network interfaces

The *iproute2* equivalent to *ifconfig* is the *ip link list* command. Note the state of the interface *eth0* is *DOWN* as *network-manager* is disabled.

```
$ sudo ip link list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc mq state DOWN mode DEFAULT qlen 1000
    link/ether 00:12:3f:dc:ab:47 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast state DOWN mode DEFAULT qlen 1000
    link/ether 00:13:ce:01:66:92 brd ff:ff:ff:ff:ff:ff
```

5.2.1 Bring up the *eth0* interface

Change the state of the *eth0* interface from *DOWN* to *UP*.

```
$ sudo ip link set dev eth0 up

$ sudo ip link list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT qlen 1000
    link/ether 00:12:3f:dc:ab:47 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast state DOWN mode DEFAULT qlen 1000
    link/ether 00:13:ce:01:66:92 brd ff:ff:ff:ff:ff:ff
```

5.2.2 Add IP Address to the *eth0* interface

Add an IPv4 address to the *eth0* interface. The *-4* option switch is optional for IPv4 as IPv4 is assumed if *-6* is not specified.

```
$ sudo ip -4 addr add 78.143.141.20/24 dev eth0
```

5.2.3 Confirm IP Address is configured

```
$ sudo ip -4 addr list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc mq state DOWN qlen 1000
    inet 78.143.141.20/24 scope global eth0
        valid_lft forever preferred_lft forever
```

5.2.4 Add an IPv4 Default gateway

Add an IPv4 gateway router to the routes table for all routes not otherwise specified.

```
$ sudo ip route add default via 78.143.141.1
```

or

```
$ sudo ip route add default dev eth0
```

5.2.5 Add a static route

While the network diagram does not require a static route, I include one here for completeness.

```
$ sudo ip route add 78.143.152.0/24 via 78.143.141.1
```

or

```
$ sudo ip route add 78.143.152.0/24 dev eth0
```

5.2.6 Confirm that the route has taken

```
$ sudo ip route list
default via 10.10.10.1 dev eth0
10.10.10.0/24 dev eth0 proto kernel scope link src 10.10.10.10
192.168.1.0/24 via 10.10.10.1 dev eth0
```

5.3 Monitoring

iproute2 comes with a very neat set of monitoring tools. Here is an example monitoring neighbours, in IPv4 and net-tools parlance this is equivalent to the *arp* command to manipulate the system ARP cache.

```
$ ip -4 monitor neigh
78.143.141.1 dev eth0 lladdr 00:0c:42:d1:3c:38 STALE
```

5.4 Internet Protocol v6

IPv6 also called IPng is the replacement for IPv4. It has 3.4×10^{38} addresses (2^{128}) more than 7.9×10^{28} times as many as IPv4. This updated version of IP was invented by Steve Deering and Craig Mudge at Xerox PARC, it was then adopted by the Internet Engineering Task Force in 1994 as IPng.

The adoption of IPv6 has been slowed by the introduction of Network Address Translation (NAT), which partially alleviates address exhaustion. Japan and Korea implemented IPv6 in the 1990's. The European Union formed an IPv6 Task Force as a steering committee in 2001 and member states all had their own IPv6 Task Forces by 2004. The US has specified that the network backbones of all federal agencies must have deployed IPv6 by 2008. In Ireland the Irish National IPv6 Centre is situated at the Telecommunications Systems & Software Group (TSSG) at Waterford Institute of Technology.

In October 2007 Vint Cerf the founder of the Internet issued a warning to Internet Service Providers (ISP) urgently need to roll out IPv6 because the IPv4 pool is finite and has all but run out in 2012 (The European IP Research (RIPE) body started allocating its last /8 in September 2012). Each Local Internet Registry (LIR) can receive one final /22 allocation (1,024 IPv4 addresses) upon application for IPv4 resources. No new IPv4 Provider Independent (PI) space will be assigned.

It is expected that IPv4 will be supported alongside IPv6 for the foreseeable future with hosts running dual-stack software.

5.4.1 Features of IPv6

IPv6 supports many new features over IPv4, these features were developed considering the problems that were showing in IPv4.

- Much larger address space - Stateless Address Auto-configuration (SLAAC)
 - IPv6 hosts can be configured automatically when connected to a routed IPv6 network. When first connected to a network, a host sends a link-local multicast request for its configuration parameters; if configured suitably, routers respond to such a request with a router advertisement packet that contains network-layer configuration parameters.
- Multicast
 - Multicast (both on the local link and across routers) is part of the base protocol suite in IPv6. This is different to IPv4, where multicast is optional.
 - IPv6 does not have a link-local broadcast facility; the same effect can be achieved by multicasting to the all-hosts group with a hop count of one.
- Jumbograms
 - In IPv4, packets are limited to 64KB of payload. When used between capable communication partners, IPv6 has support for packets over this limit, referred to as jumbograms. Use of jumbograms improves performance over high throughput networks.
- Faster routing
 - By using a simpler and more systematic header structure, IPv6 improves the performance of routing. Recent advances in router technology, however, may have made this improvement irrelevant.
- Network-layer security
 - IPsec, the protocol for IP network-layer encryption and authentication, is an integral part of the base protocol suite in IPv6. It is, however, not yet deployed widely except for securing BGP traffic between IPv6 routers.

- Mobility
 - IPv6 was designed to support mobility. IPv6 Neighbour Discovery (ND) and SLAAC allow hosts to operate in any locations without any special support. This makes it more scalable and the performance is better because less traffic passes through the home link and less redirection and less rerouting. It also means no single point of failure.

5.4.2 IPv6 Address Architecture

IPv6 addresses are normally written as eight groups of four hexadecimal digits. For example, *2a02:2158:435a:0000:83:314:ea21:b33f* is a valid IPv6 address.

If a four-digit group is 0000, the zeros may be omitted.

2a02:2158:435a:0000:83:314:ea21:b33f --> *2a02:2158:435a::83:314:ea21:b33f*

Following this rule, any group of consecutive 0000 groups may be reduced to two colons, as long as there is only one double colon used in an address. Thus, the addresses below are all valid and equivalent:

2a02:2158:0000:0000:0000:0000:00a1:b33f
2a02:2158:0000:0000:0000::00a1:b33f
2a02:2158:0:0:0:0:0a1:b33f
2a02:2158:0::0:0a1:b33f
2a02:2158::0a1:b33f

Having more than one double-colon abbreviation in an address is invalid as it would make the notation ambiguous.

Leading zeros in a group can be omitted. Thus *2a02:0201:0000:0000:0000:0000:00a1:b33f* may be shortened to *2a02:201::a1:b33f*.

A sequence of 4 bytes at the end of an IPv6 address can also be written in decimal, using dots as separators. This notation is often used with compatibility addresses (see below). Thus, *::ffff:1.2.3.4* is the same address as *::ffff:102:304*.

Additional information can be found in IPv6 Addressing Architecture RFC's 4291, 5952, 6052, 7136, 7346 and 7371.

IPv6 Network Notation

IPv6 networks are written using Classless Inter-Domain Routing (CIDR) notation.

An IPv6 network is a contiguous group of IPv6 addresses the size of which must be a power of two; the initial bits of addresses which are identical for all hosts in the network are called the network's prefix.

A network is denoted by the first address in the network and the size in bits of the prefix, separated with a slash. For example, *2a02:2158:435a:0000::/64* stands for the network with

- First address: *2a02:2158:435a:0000::*
- Last address: *2a02:2158:435a:0000:ffff:ffff:ffff:ffff*

Because a single host can be seen as a network with a 128-bit prefix, a host address may be shown with */128* mask.

IPv6 Prefix Terminology

IPv6 does not have a *classful* concept like IPv4 but within Global unicast IPv6 address assignments have a number of prefixes, with different prefix lengths. Here is a table outlining four of the key terms.

Prefix Term	Assigned by	Example prefix
Registry Prefix	Assigned to Regional Registry (RR)	2a02::/12
ISP Prefix	Assigned to Internet Service Provider (ISP)	2a02:2158::/32
Site Prefix	Assigned to Large Organisation	2a02:2158:1111::/48
Site Prefix	Assigned to Smaller Organisation	2a02:2158:1111:100::/56
Subnet Prefix	Internal subnet within Organisation	2a02:2158:1111:110::/64
A host address	Organisation/Residential home user	2a02:2158:1111:110::10/128

The following table give an indication of IPv6 Relative Network Sizes.

Mask	Size	Description
128	1 IPv6 Address	A network interface
64	1 IPv6 subnet	18,446,744,073,709,551,616 IPv6 addresses
56	256 LAN segments	Popular prefix size for smaller subscriber site
48	65,536 LAN segments	Popular prefix size for larger subscriber site
32	65,536 /48 subscriber sites	Minimum IPv6 allocation by RR
24	16,777,216 subscriber sites	256 times larger than the min IPv6 allocation

Special Prefix's

There are a number of specific addresses within IPv6 with special meaning:

Prefix	Meaning
::/0	The default unicast route address (similar to 0.0.0.0/0 in IPv4)
::/128	The address with all zeroes is an unspecified address, and is only to be used in software
::1/128	The loopback address is a localhost address. (like 127.0.0.1 in IPv4)
::ffff:0:0/96	This prefix is used for IPv4 mapped addresses. Transparent use of Transport Layer protocols over IPv4 through IPv6 API
64:ff9b::/96	Well known prefix for 6to4 address translation.
0400::/7	Internetwork Packet Exchange (IPX) from the IPX/SPX protocol stack routed via IPv6
fc00::/7	Unique Local IPv6 Unicast Addresses are only routable within a set of cooperating sites. They were defined in RFC 4193 as a replacement for site-local addresses (see below). The addresses include a 40-bit pseudo-random number that minimises the risk of conflicts if sites merge or packets somehow leak out
fe80::/10	Link-local prefix specifies that the address only is valid in the local physical link. (like the Auto-configuration address 169.254.x.x in IPv4)
ff00::/8	The multicast prefix for multicast addresses
ff01::0/12	Pre-defined Multicast addresses
ff01::1/12	All host addresses (interface-local)
ff02::1/12	All host addresses (link-local)
ff01::2/12	All routers (interface-local)
ff02::2/12	All routers (link-local)
ff05::2/12	All routers (site-local)

Deprecated Prefix's

The following prefixes were originally defined as part of IPv6 but have since been deprecated or obsoleted. I have added them here for information in case you come across such addresses.

Prefix	Meaning
::/96	The zero prefix was used for IPv4-compatible addresses. Deprecated in February 2006.
fec0::/10	Site-local prefix specifies that the address is only valid inside the local organisation. Its use has been deprecated in September 2004 by IPv6 Deprecating Site Local Addresses RFC and future systems must not implement any support for this special type of address any more.
0200::/7	Network Service Access Point (NSAP) addresses from ISO/IEC 8348 routed via IPv6. Deprecated in December 2004.

5.4.3 IPv6 Address Scope

IPv6 address have a *scope* to specify where the address is valid. Within unicast addressing, link-local addresses and the loopback address have *link-local* scope, which means they are to be used in the directly attached network (link) only. All other addresses, including unique local addresses, have global (or universal) scope, which means they are globally routable, and can be used to connect to addresses with global scope anywhere, or addresses with *link-local* scope on the directly attached network. The scope of an *anycast* address is defined identically to that of a *unicast* address.

For multicasting, the four least-significant bits of the second address octet of a multicast address (ff0X::) identify the address scope, the span over which the multicast address is propagated.

ff0X::/8	Meaning
0x1	Interface local
0x2	Link local
0x4	Admin local
0x5	Site local (Now Deprecated)
0x8	Organisation local
0xE	Global
0x0	Reserved
0xF	Reserved

Examples:

- ff05::1 - All nodes on the local site
- ff02::2 - All routers on the link local
- ff02::5 - All OSPF routers on the link local
- ff02::a - All EIGRP routers on the link local
- ff05::101 - All Network Time Protocol (NTP) Servers on the local site
- ff02::1:3 - All DHCPv6 servers on the link local

5.4.4 IPv6 Addressing Model

Like IPv4 the IPv6 Address is constructed of two parts the Prefix + host Identifier (ID) (Sometimes the Interface ID). The idea is to separate *who u are* from *where u are connected to*. The Prefix is dependant on the routing topology and the Interface ID identifies a node. IPv6 removes the Broadcast address and instead uses special Multicast addresses *all hosts ff0X::1* or *all routers ff0X::2* where X is replaced by the scope number. IPv6 also introduces a new *anycast* address. An *anycast* address is an IPv6 address that is assigned to one or more network interfaces, with the property that a packet sent to an *anycast* address is routed to the *nearest* interface having that address, according to the routing protocols measure of distance.

- *Unicast*: from one host to another.
- *Multicast*: from one to all belonging to a group.
- *Anycast*: from one to the nearest belonging to a group.

5.4.5 Loopback Address

Similar to IPv4, IPv6 has a special address reserved for loopback.

- ::1

5.4.6 IPv6 Packet Structure

The IPv6 packet header has many changes compared to the IPv4 header while maintaining necessary elements. The IPv6 header contains.

Header	Description
Version	Describes the version as 6
Traffic Class	One byte field
Flow Label	20 bit flow label for label tagging
Payload Length	Two byte integer giving the length of the packet less the header
Next Header	Single byte selector using the same values as the IPv4 Protocol field
Hop Limit	Single byte decremented at each router, packet discarded if zero
Source Address	128 bit address of originator
Destination Address	128 bit address of ultimate recipient

Here is an example IPv6 packet which has ICMPv6 embedded within it.

```

Ethernet II, Src: 00:16:17:ba:0e:74, Dst: 00:12:3f:dc:ab:47
  Destination: 00:12:3f:dc:ab:47
  Source: 00:16:17:ba:0e:74
  Type: IPv6 (0x86dd)
Internet Protocol Version 6
  0110 .... = Version: 6
  .... 0000 0000 .... .... .... .... = Traffic class: 0x00000000
  .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
  Payload length: 40
  Next header: ICMPv6 (0x3a)
  Hop limit: 128
  Source: 2a02:aaaa:10
  Destination: 2a02:aaaa:20
Internet Control Message Protocol v6
  Type: 128 (Echo request)
  Code: 0
  Checksum: 0x94bb [correct]
  ID: 0x0001
  Sequence: 0x000b
  Data (32 bytes)

```

IPv6 Option headers

Unlike IPv4 the IPv6 options are handled outside the IPv6 header. This is achieved by the addition of extensions headers which are only processed as necessary. For example only routers process the *Hop by Hop options header*. With this method it is easier to define new extensions and options as the protocol evolves. Here is a list of some optional headers that are used with IPv6 today.

- Hop by Hop options header
- Destination options header
- Routing header
- Fragment header
- Authentication header (AH)
- Encapsulation security payload (ESP) header

5.4.7 Applications for IPv6

DHCP for IPv6 (DHCPv6)

DHCP for IPv6 (DHCPv6). Although IPv6's StateLess Address Auto Configuration (SLAAC) removes the primary motivation for DHCP in IPv4, DHCPv6 can still be used to statefully assign addresses if the network administrator desires more control over addressing. It can also be used to distribute information which is not otherwise discoverable; the most important case of this is the DNS server.

A major difference with DHCPv4 Servers is that hosts send broadcasts to find DHCP Servers whereas with DHCPv6 Servers IPv6 hosts send IPv6 multicast. The reserved address for hosts to send packets to an unknown DHCPv6 Server is *FF02::1:2*.

DNS Extensions to Support IP Version 6 (DNSv6)

DNS is similar for IPv4 and IPv6 (DNSv6). The main difference is that the *A* record is replaced by the *AAAA* record which maps a hostname to a 128-bit IPv6 address for forward lookups. Reverse lookups take place under *ip6.arpa*, where address space is delegated on nibble boundaries. This scheme is a straightforward adaptation of the familiar *A* record and *in-addr.arpa* schemes for IPv4.

ICMPv6 for IPv6

ICMP version 6 (ICMPv6) is a new version of ICMP and is an integral part of the IPv6 architecture that must be completely supported by all IPv6 implementations and nodes. ICMPv6 combines functions previously subdivided among different protocols, such as ICMP, IGMP (Internet Group Membership Protocol version 3), and ARP (Address Resolution Protocol) and it introduces some simplifications by eliminating obsolete types of messages no longer in use.

ICMPv6 is a multi-purpose protocol and it is used for reporting errors encountered in processing packets, performing diagnostics, performing ND and reporting IPv6 multicast memberships. For this reason, ICMPv6 messages are subdivided into two classes:

Error messages

The first type of ICMPv6 message is the error message. ICMPv6 is used by IPv6 nodes to report errors encountered.

Type	Message
1	Destination Unreachable
2	Packet Too Big
3	Time Exceeded
4	Parameter Problem

Information messages

The second type of ICMP message is the informational message type which is subdivided into three groups: diagnostic, management of multicast groups, and ND messages.

Type	Message
128	Echo Request
129	Echo Reply
130	Group Membership Query
131	Group Membership Report
132	Group Membership Reduction
133	Router Solicitation
134	Router Advertisement
135	Neighbour Solicitation
136	Neighbour Advertisement
137	Redirect
138	Router Renumbering

5.4.8 IPv6 EUI-64 host

IPv6 uses 64 bits for the network and subnets while it reserves the last 64 bits to identify the host. The last 64 bits can be specified specifically as shown or IPv6 can *create* the host by using its MAC address from the interface. Such MAC addresses are properly termed EUI-48 as they have 48 bits. However the range of unique EUI-48 addresses are running out and it is decided to migrate such addresses to EUI-64 in the future. IPv6 was therefore developed with this in mind. To deal with the difference between EUI-48 and EUI-64 a conversion mechanism exists where the EUI-48 address is split and *FF:FE* is embedded to *convert* it to EUI-64. Here is an example of the conversion process of an EUI-48 to EUI-64.

$00:12:3f:dc:ab:47 \Rightarrow 0012:3fdc:-:fffe:-:ab47 \Rightarrow 0012:3fdc:fffe:ab47$

5.4.9 IPv6 link-local

When an IPv6 host boots it has no IPv6 address so using its MAC address it forms an EUI-64 address and prepends it with the special network identifier FF80::. This is called the link-local address and it has only local scope.

IPv6 prefix added to EUI-64 to form link-local scope IPv6 address.

$fe80::212:3fff:fedc:ab47/64$

IPv6 prefix added to EUI-64 to form global scope IPv6 address.

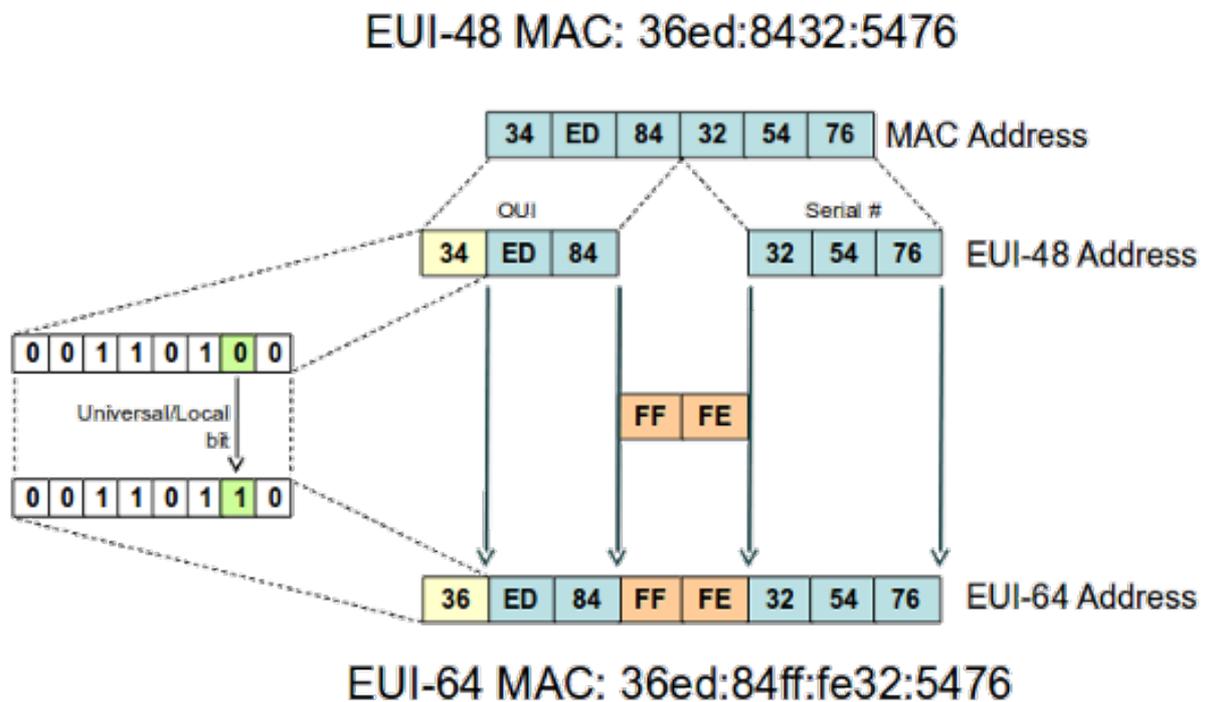
$2a02:aaaa::212:3fff:fedc:ab47/64$

5.4.10 IPv6 Stateless Address Auto-configuration (SLAAC)

SLAAC is an IPv6 process that removes the requirement for the manual configuration of hosts, minimal configuration of routers, and no additional servers. The stateless mechanism enables a host to generate its own global address. It is based on ICMPv6. The stateless mechanism uses local information as well as non-local information that is advertised by routers to generate the addresses.

Routers advertise prefixes that identify the subnet or subnets that are associated with a link. Hosts generate an interface identifier that uniquely identifies an interface on a subnet. An address is formed by combining the prefix and the interface identifier. In the absence of routers, a host can generate only *link-local* addresses. However, *link-local* addresses are only sufficient for allowing communication among nodes that are attached to the same link.

SLAAC Process



Forming a link-local address

Here are the typical steps performed by an interface during SLAAC. Auto-configuration is performed only on multicast-capable links. SLAAC begins when a multicast-capable interface is enabled, for example, during system startup. Nodes, both hosts and routers, begin the Auto-configuration process by generating a link-local address for the interface. A link-local address is formed by appending the interface's identifier to the well-known link-local prefix *fe80::* as described above.

Duplicate Address Detection (DAD)

The next step is called DAD where the host once its link-local address is created will send an *ICMPv6 Neighbour Solicitation (135)* informational message to the newly created link-local address to see if an *ICMPv6 Neighbour Advertisement (136)* informational message will be received. If one is received then it determines that the address is a duplicate and the SLAAC process stops. If none is received SLAAC proceeds to the next step.

Obtaining a Global scope prefix

After confirming that the *link-local* address is unique the host sends a *ICMPv6 Router Solicitation (133)* informational message to the well-known all-routers multicast group *ff02::2*. If an *ICMPv6 Router Advertisement (134)* informational message is received from a router the host creates a new IPv6 address with global scope by taking the advertised prefix and pre-pending it to the EUI-64 address created on bootup. *Router Advertisement (134)* informational messages contain two flags that indicate what type of stateful Auto-configuration should be performed. A Managed address configuration flag (M-Flag) indicates whether hosts should use stateful auto-configuration to obtain global scope IPv6 addresses. The other stateful configuration flag (O-Flag) if set (1) indicates that hosts should use stateful auto-configuration to obtain additional information, excluding addresses, from a stateless DHCPv6 Server.

```

Internet Protocol Version 6
  0110 .... = Version: 6
  Payload length: 64
  Next header: ICMPv6 (0x3a)
  Hop limit: 255
  Source: fe80::223:5eff:fe0e:6816
  Destination: ff02::1 (all-hosts link-local)
Internet Control Message Protocol v6
  Type: 134 (Router advertisement)
  Code: 0
  Checksum: 0x9246 [correct]
  Cur hop limit: 64
  Flags: 0x40
    0... .... = Not managed (M-Flag)
    .1.. .... = Other Configuration (O-Flag)
    ..0. .... = Not Home Agent
    ...0 0... = Router preference: Medium
  Router lifetime: 1800
  Reachable time: 5000
  Retrans timer: 0
  ICMPv6 Option (Source link-layer address)
    Type: Source link-layer address (1)
    Length: 8
    Link-layer address: 00:23:5e:0e:68:16
  ICMPv6 Option (MTU)
    Type: MTU (5)
    Length: 8
    MTU: 1500
  ICMPv6 Option (Prefix information)
    Type: Prefix information (3)
    Length: 32
    Prefix length: 64
    Flags: 0xc0
      1... .... = Onlink
      .1.. .... = Auto
      ..0. .... = Not router address
      ...0 .... = Not site prefix
    Valid lifetime: 86400
    Preferred lifetime: 86400
    Prefix: 2a02:aaaa:2::

```

5.4.11 IPv6 transition mechanisms

Until IPv6 completely replaces IPv4, a number of transition mechanisms are needed to enable IPv6-only hosts to reach IPv4 services and to allow isolated IPv6 hosts and networks to reach the IPv6 Internet over the IPv4 infrastructure. As the IPv6 Internet grows larger, the need also arises for carrying IPv4 traffic over the IPv6 infrastructure.

Dual Stack

IPv6 is a form of extension of IPv4 and therefore it is relatively easy to write a network stack that supports both IPv4 and IPv6 while sharing most of the code. Dual Stack is implemented by the various OS today. Some early experimental implementations used independent IPv4 and IPv6 stacks. There are no known implementations that implement IPv6 only. Actually when used in IPv4 communications, hybrid stacks tend to use an IPv6 API and represent IPv4 addresses in a special address format, the IPv4-mapped IPv6 address.

Proxying and translation

When an IPv6 only host needs to access an IPv4 only host, translation is necessary. The one form of translation that actually works is the use of a dual stack application-layer proxy. Techniques for application agnostic translation at the lower layers have also been proposed, but they have been found to be too unreliable in practice due to the wide range of functionality required by common application-layer protocols, and are commonly considered to be obsolete.

Stateless IP/ICMP Translation Algorithm (SIIT)

SIIT translates between the packet header formats in IPv6 and IPv4. SIIT can be used to allow IPv6 hosts, that are not Dual Stack, to communicate with IPv4-only hosts.

Network Address Translation/Protocol Translation (NAT-PT)

NAT-PT is a protocol translator between IPv6 and IPv4 that allows direct communication between hosts speaking different network protocols. However RFC 4966 outline reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to historic status.

Tunnelling

In order to reach the IPv6 Internet, an isolated host or network must be able to use the existing IPv4 infrastructure to carry IPv6 packets. This is achieved using a technique known as tunnelling which consists of the encapsulation of IPv6 packets within IPv4, in effect using IPv4 as a link layer for IPv6.

IPv6 packets can be directly encapsulated within IPv4 packets using protocol number 41. They can also be encapsulated within UDP packets e.g. in order to cross a router or NAT device that blocks protocol 41 traffic. Another options is to use generic encapsulation schemes like Generic Routing Encapsulation (GRE).

Manual tunnelling

Manual tunnelling is done by manually configuring the end points of the tunnel. This tunnelling method can be used for sites with few nodes or for a limited number of remote connections. As is the case with static routing, scalability and management overhead are major issues limiting the use of manual tunnelling.

Automatic tunnelling

Automatic tunnelling refers to a technique where the tunnel endpoints are automatically determined by the routing infrastructure.

Connection of IPv6 Domains via IPv4 Clouds (6over4)

The recommended technique for automatic tunnelling is *6to4* tunnelling, which uses *protocol 41* encapsulation. Tunnel endpoints are determined by using a well-known IPv4 anycast address on the remote side, and embedding IPv4 address information within IPv6 addresses on the local side. *6to4* is widely deployed today.

Teredo: Tunnelling IPv6 over UDP through NATs

Teredo is an automatic tunnelling technique that uses UDP encapsulation and is capable of crossing multiple NAT devices. *Teredo* gives IPv6 connectivity to IPv6 capable hosts which are on the IPv4 Internet but have have no direct native connection to an IPv6 network. *Teredo* is not widely deployed today. *Miredo* is the GNU/Linux and BSD UNIX open-source implementation of *Teredo*.

Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)

ISATAP is an IPv6 transition mechanism designed to transmit IPv6 packets between nodes with dual-stack (IPv6/IPv4) over IPv4 networks. *ISATAP* views the IPv4 network as a link layer for IPv6 and supports an automatic tunnelling abstraction similar to a Non-Broadcast Multiple Access (NBMA) model.

5.4.12 IPv6 Interior Gateway Routing

RIPng

Like its IPv4 variant *RIPng* is a Distance vector algorithm. It has a number of implementations: GateD, MRTd, Kame, route6d, Quagga as well as vendor equipment solutions from companies like Cisco, Juniper, HP, Huawei, MikroTik etc....

OSPFv3

OSPFv3 is a Link State algorithm like the IPv4 version. It is the recommended IGP of IETF. The main differences from *OSPFv2* are the removal of security as IPv6 has its own implementation embedded and the format of addresses are for IPv6. Implementations: GateD, MRTd, Kame, route6d, Quagga and vendor hardware solutions from companies like Cisco, Juniper, HP, Huawei, MikroTik etc....

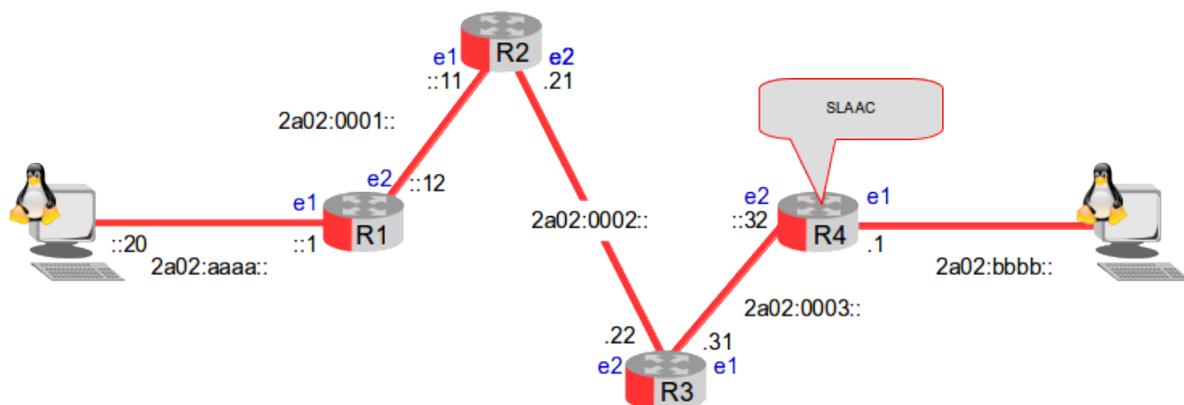
5.4.13 IPv6 Exterior Gateway Routing

BGP4+

BGP4+ is the standard Inter domain routing protocol for IPv6. It is used between ISPs and carriers and its extensions to *BGP4* are defined in RFC 2858. RFC 2545 defines how to use IPv6 extensions. It is used in 6BONE and the following are implementations today: GateD, MTRd, Kame, BGPd, Quagga and vendor hardware solutions from companies like Cisco, Juniper, HP, Huawei, MikroTik etc....

5.4.14 IPv6 Configuration

The configurations below are based on the following IPv6 network diagram.



Like IPv6 the *ip link list* command will show the interface status, the *-6* option switch makes it IPv6 specific.

```
$ sudo ip -6 link list
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT qlen 1000
    link/ether 00:12:3f:dc:ab:47 brd ff:ff:ff:ff:ff:ff
```

Review the IPv6 addresses with the *-6* option switch.

```
$ sudo ip -6 addr list
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 2a02:aaaa::1021:3ae0:b092:be7e/64 scope global temporary dynamic
        valid_lft 602791sec preferred_lft 83791sec
    inet6 2a02:aaaa::212:3fff:fedc:ab47/64 scope global dynamic
        valid_lft 259191sec preferred_lft 604711sec
    inet6 fe80::212:3fff:fedc:ab47/64 scope link
        valid_lft forever preferred_lft forever
```

Add a new IPv6 address

To add an IPv6 address to an interface follow the same format as the IPv4 equivalent with the *-6* switch.

```
$ sudo ip -6 addr add 2a02:aaaa::20/64 dev eth0
```

Testing the link to the remote router

Instead of *ping* to test connectivity to an IPv6 address, a new utility called *ping6* must be used.

```
$ ping6 2a02:aaaa::1
PING 2a02:aaaa::1(2a02:aaaa::1) 56 data bytes
64 bytes from 2a02:aaaa::1: icmp_seq=1 ttl=64 time=0.592 ms
64 bytes from 2a02:aaaa::1: icmp_seq=2 ttl=64 time=0.266 ms
```

Instead of ARP as used by IPv4, IPv6 uses the ICMPv6 ND messages. To review the IPv6 neighbours cached use the *ip -6 neigh list* command.

```
$ sudo ip -6 neigh list dev eth0
2a02:aaaa::1 lladdr 00:0c:42:d1:3c:38 router REACHABLE
fe80::20c:42ff:fed1:3c38 lladdr 00:0c:42:d1:3c:38 router REACHABLE
```

To print the list of IPv6 routes known by the host connected to the router *R1*.

\$ ip -6 route list

```
2a02:2::/64 dev eth1 proto kernel metric 256 expires 2591731sec
2a02:aaaa::/64 dev eth0 proto kernel metric 256 expires 2591689sec
fe80::/64 dev eth0 proto kernel metric 256
fe80::/64 dev eth1 proto kernel metric 256
default via fe80::20c:42ff:fed1:3c38 dev eth0 proto ra metric 1024 expires 1489sec
default via fe80::20c:42ff:fe8b:73e8 dev eth1 proto ra metric 1024 expires 1531sec
default via fe80::20c:42ff:fe8b:769a dev eth1 proto ra metric 1024 expires 1395sec
```

IPv6 Monitor

\$ ip -6 monitor address

```
3: eth1 inet6 2a02:2::213:ceff:fe01:6692/64 scope global dynamic
    valid_lft 2592000sec preferred_lft 604800sec
3: eth1 inet6 2a02:2::c1ad:a142:f05a:5539/64 scope global temporary dynamic
    valid_lft 600483sec preferred_lft 81483sec
2: eth0 inet6 2a02:aaaa::212:3fff:fedc:ab47/64 scope global dynamic
    valid_lft 2592000sec preferred_lft 604800sec
2: eth0 inet6 2a02:aaaa::1021:3ae0:b092:be7e/64 scope global temporary dynamic
    valid_lft 599887sec preferred_lft 80887sec
3: eth1 inet6 2a02:2::213:ceff:fe01:6692/64 scope global dynamic
    valid_lft 2592000sec preferred_lft 604800sec
3: eth1 inet6 2a02:2::c1ad:a142:f05a:5539/64 scope global temporary dynamic
    valid_lft 600364sec preferred_lft 81364sec
3: eth1 inet6 2a02:2::213:ceff:fe01:6692/64 scope global dynamic
    valid_lft 2592000sec preferred_lft 604800sec
```

This page is intentionally blank

6. Routing

6.1 Introduction to Routing

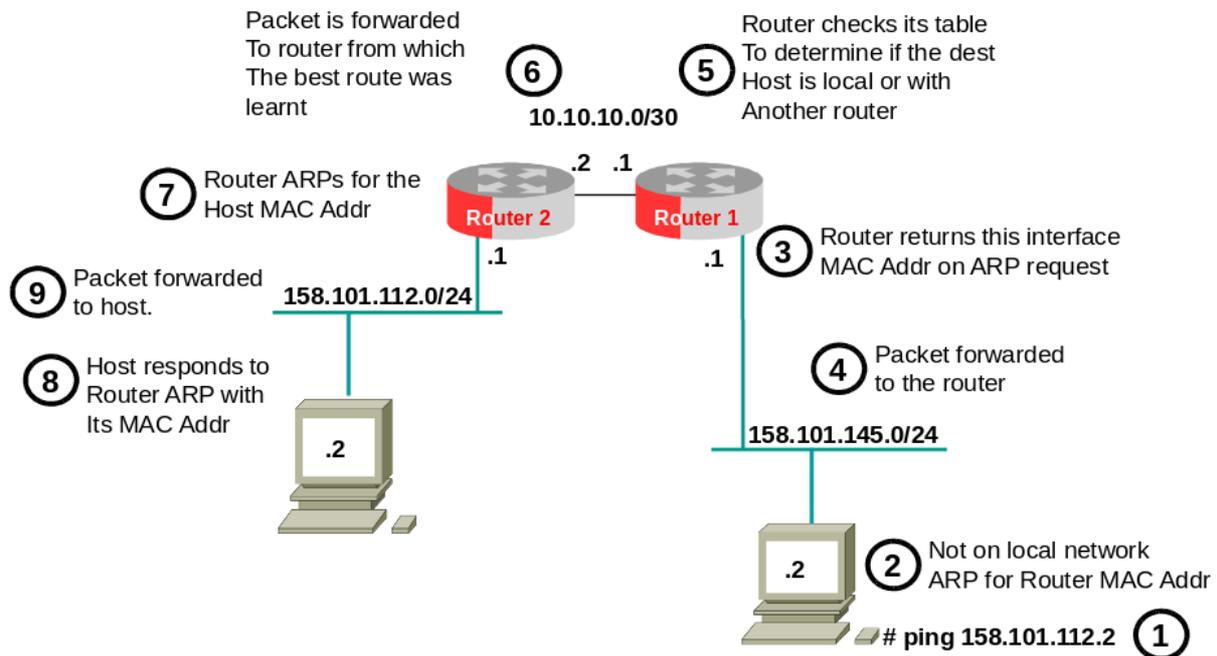
Routing refers to selection of paths in a computer network along which to send data. Routing directs forwarding, the passing of logically addressed packets from their source network, toward their ultimate destination through intermediary nodes; typically hardware devices called routers. The routing process usually directs forwarding on the basis of routing tables which maintain a record of the best routes to various network destinations. Thus constructing routing tables, which are held in the routers' memory, becomes very important for efficient routing.

Routing differs from bridging in its assumption that address-structures imply the proximity of similar addresses within the network, thus allowing a single routing-table entry to represent the route to a group of addresses. Therefore, routing outperforms bridging in large networks, and it has become the dominant form of path-discovery on the Internet.

Small networks may involve manually configured routing tables, while larger networks involve complex topologies and may change constantly, making the manual construction of routing tables very problematic. A good example is the Public Switched Telephone Network (PSTN) which uses pre-computed routing tables, with fallback routes if the most direct route becomes blocked. Dynamic routing attempts to solve this problem by constructing routing tables automatically, based on information carried by routing protocols, and allowing the network to act nearly autonomously in avoiding network failures and blockages. Dynamic routing dominates the Internet. However, the configuration of the routing protocols often requires a skilled touch, one should not suppose that networking technology has developed to the point of the complete automation of routing.

Traditional IP routing stays relatively simple because it uses next-hop routing where the router only needs to consider where it sends the packet, and does not need to consider the subsequent path of the packet on the remaining hops. However, more complex routing strategies can be, and are, often used in systems such as Multi-Protocol Label Switching (MPLS), Asynchronous Transfer Mode (ATM) or Frame Relay, which are sometimes used as underlying technologies to support IP networks.

6.1.1 Standard Routing Model



A router must be connected to at least two networks, or it will have nothing to route. A special variety of router is the one-armed router used to route packets in a VLAN environment. In the case of a one-armed router the multiple attachments to different networks are all over the same physical link.

A router creates and/or maintains a table, called a "routing table" that stores the best routes to certain network destinations and the "routing metrics" associated with those routes.

Knowing where to send packets requires knowledge of the structure of the network. In small networks, routing can be very simple, and is often configured by hand called Static Routing. In large networks the topology of the network is complex, and constantly changing, making the problem of constructing the routing tables very complex.

As the best routes can only be recalculated very slowly relative to the rate of arrival of packets, routers keep a routing table that maintains a record of only the best possible routes to certain network destinations and the routing metrics associated with those routes.

In the above diagram the host 158.101.145.2 pings 158.101.112.2. The IP Stack in the host determines that this address is not on its own network by doing a bitwise AND on the address with its own subnet mask. It then looks up its Default Gateway, or in other words the router connected to the LAN which can route to the rest of the internetwork. It sends out an Address Resolution Protocol (ARP) request to 158.101.145.1 for the MAC address of the routers interface on its own network. The router responds in kind and the host forwards the packet in a frame to the router.

The router strips off the frame and does a routing table lookup for the destination IP address in the packet header. In this case it does not have an interface in the destination network. It determines from its routing table that router2 has declared it can reach the destination network. It places the packet in a frame and forwards it to the peer router2.

Router2 receives the frame and removes the packet from it. It determines that it is connected on its other interface to the destination network so it does an ARP request on that LAN for the MAC address of the host 158.101.112.2. The host responds to the request with its MAC address and router2 encapsulates the packet in a new frame which it forwards to the destination host.

6.1.2 Routing Tables

Routers maintain Routing Tables to determine if it can reach a requested route. Routing tables can take many forms, but here is a simple model that can explain most Internet routing. Each entry in a routing table has at least two fields - IP Address Prefix and Next Hop. The Next Hop is the IP address of another host or router that is directly reachable via an Ethernet, serial link, or some other physical connection. The IP Address Prefix specifies a set of destinations for which the routing entry is valid for. In order to be in this set, the beginning of the destination IP address must match the IP Address Prefix, which can have from 0 to 32 significant bits. For example, a IP Address Prefix of 128.8.0.0/16 would match any IP Destination Address of the form 128.8.X.X. Bridged and switched networks are regarded as single connections.

If no routing table entries match a packet's Destination Address, the packet is discarded as undeliverable (possibly with an ICMP notification to the sender). If multiple routing tables entries match, the longest match is preferred. The longest match is the entry with the most 1 bits in its Routing Mask.

6.2 Open Shortest Path First (OSPF)

Traditional IP networks in the past used the RIP or the Cisco Interior Gateway Routing Protocols (IGRP). These are Distance Vector based or hop by hop based on a distance metric of hop counts. Each router regularly passes its routing table to its neighbouring routers. Cisco developed an Enhanced version of IGRP (EIGRP) to take account of more parameters and to reduce the traffic between routers to only that the neighbours did not have already. It also incorporated Classless Inter-Domain Routing (CIDR) and Message Digest 5 (MD5) functionality. However in 1991 a DRAFT Internet Engineering Task Force (IETF) Standard RFC1247 - OSPF Version 2 led to a series of RFCs that eventually produced a new Interior Gateway Protocol (IGP) RFC2328 - Open Shortest Path First Version 2. standard in 1998. This new IGP is not based on Distance Vector but is a Link State protocol that has been the mainstay of interior networks ever since. In 2008 RFC5340 - OSPF for IPv6 incorporated support for IPv6. It is generally known as OSPFv3.

6.2.1 OSPF Overview

OSPF is an IGP most suited for use in large networks. OSPF uses a link-state algorithm to exchange routing information between routers in an Autonomous System (AS). An AS is a collection of routers and networks administratively configured to belong to a single organisation. OSPF enables the routers to quickly synchronise their topological databases, topology information for the AS only floods in response to topological change.

6.2.2 Benefits of Using OSPF versus Distance Vector protocols

Compared to other distance vector protocols like RIP and IGRP, OSPF:

- Chooses the least costly path as the best path
- Can calculate equal cost multiple paths to a destination
- Distributes external information independently
- Propagates routing information quickly and stably
- Handles Variable Length Subnet Masks (VLSM)
- Supports multicasting
- Responds quickly to topological changes by utilising reliable flooding to minimise routing traffic
- Is loop free
- Supports large metrics, external route tags and authentication of protocol exchanges

6.2.3 OSPF Concepts

Overview

To better understand the OSPF implementation, it is important to define some terms and concepts.

Link State Databases

Each OSPF router originates one or more Link State Advertisements (LSA) to describe its local part of the routing domain. The advertised link state describes the router's local interface and adjacent neighbours. Collectively, the total LSAs in the routing domain generated by OSPF routers form what is referred to as the Link State Database (LSDB). The LSDB describes the routing topology - the collection of routers and networks in the routing domain and how they are interconnected. LSDBs are exchanged between neighbouring routers soon after the routers discover each other.

Reliable Flooding

The LSDB synchronises via reliable flooding to ensure each router has an identical LSDB. When a router's link state changes, a technique called reliable flooding occurs wherein an OSPF router floods its updated LSA out of all of its interfaces. The neighbouring routers receive the updated LSA, update their own LSDB, and replicate this action out of all of their interfaces (except the interface where the LSA originated).

Shortest Path First Algorithm

Using the LSDB as input, each OSPF router runs Dijkstra's Shortest Path First (SPF) algorithm to compute the shortest path from the calculating router to all destinations. The shortest path destinations discovered by the SPF algorithm are then updated into the IP routing table.

Adjacency

Adjacency is a relationship an OSPF establishes with other routers attached to its local interface. Full adjacency is achieved once these two events take place:

- Hello packets are exchanged, allowing neighbouring routers to discover each other.
- LSDBs are synchronised between neighbouring routers.

Network Types

- Point-to-Point networks
 - This type of network connects a single pair of routers. Point-to-Point interfaces include PPP, Serial Line Internet Protocol (SLIP), ATM, and Frame Relay.
- Broadcast networks
 - This type of network allows more than two routers to share a common network, and is able to address a single message to all attached devices. Broadcast interfaces include 10, 100 and 1000 Mbps Ethernet.

Designated Router

A single router, called the Designated Router (DR), is used on Broadcast networks such as Ethernet and ATM. Once the DR establishes adjacency with other routers in an area, it is responsible for generating the network link state advertisements for the broadcast network and distributing this information to other parts of the routing domain. Conversely, the DR also receives routing information from other parts of the routing domain and distributes it to other routers on its network.

By default, the first OSPF router configured on an IP subnet is the DR. When a second router is added, it becomes the Backup Designated Router (BDR). Additional routers added to the subnet defer to the existing DR and BDR. The only time this designation changes is when the

existing DR or BDR fails, in which case other routers on the subnet will participate in a designated router election.

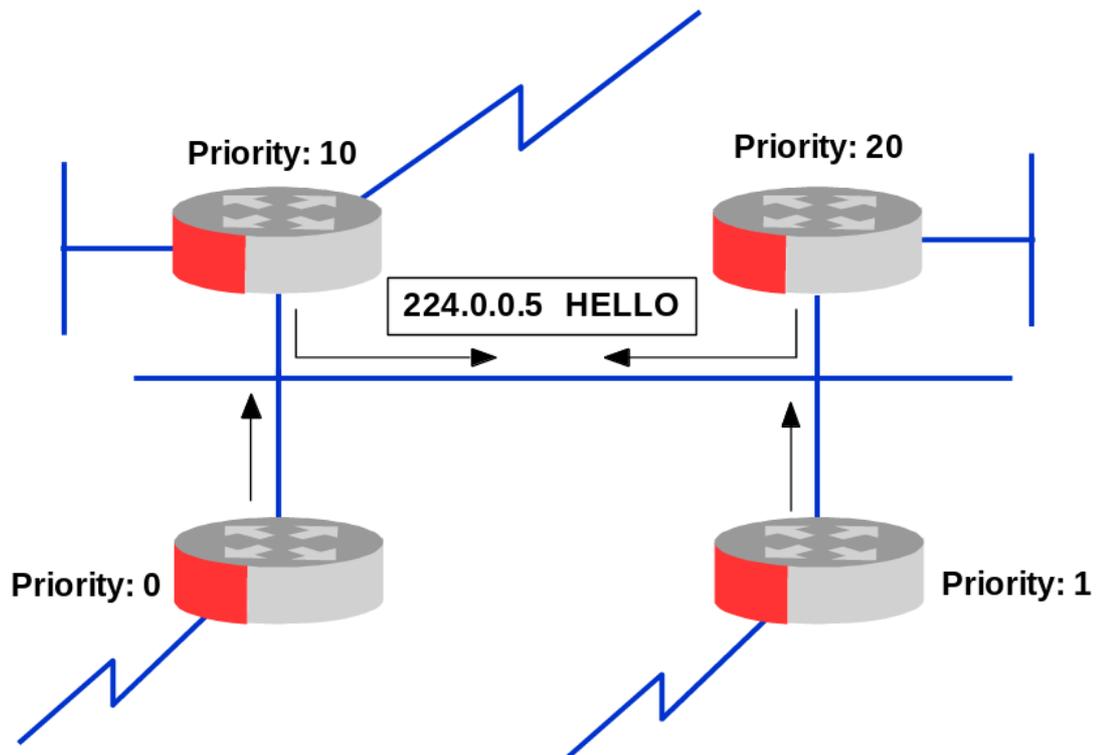
Authentication

All OSPF protocol exchanges are authenticated. OSPF authentication ensures routers exchange information only with trusted neighbours.

- Simple password
 - Configures a password included in all OSPF messages on an interface-by-interface basis.
 - When a router receives a message on an interface configured for simple password authentication, it checks the incoming OSPF message to ensure the proper password is included in the message. If the password is not included, the message is dropped.
 - The simple password is clear text, case sensitive, and is not encrypted.
- Cryptographic
 - This authentication method, also referred to as MD5 authentication uses a shared secret key that is configured in all routers attached to a common network or subnet.
 - Each key is identified by the combination of an interface and Key ID. A default key ID of 0 is automatically set when an interface is configured for cryptographic authentication.
 - An interface can have multiple active keys.
 - Each key has four time constants associated with it, governing the use of the key during specific time periods.

6.2.4 SPF Algorithm

The SFP routing algorithm is the basis for OSPF operations. When an SPF router is powered up, it initialises its routing-protocol data structures and then waits for indications from lower-layer protocols that its interfaces are functional.



Hello Protocol

After a router is assured that its interfaces are functioning, it uses the OSPF Hello protocol to acquire neighbours by multicasting to `224.0.0.5`, which is to all routers with interfaces to a common network. The router sends *hello* packets to its neighbours and receives their *hello* packets. In addition to helping acquire neighbours, hello packets also act as keep-alives to let routers know that other routers are still functional.

DR/BDR Election

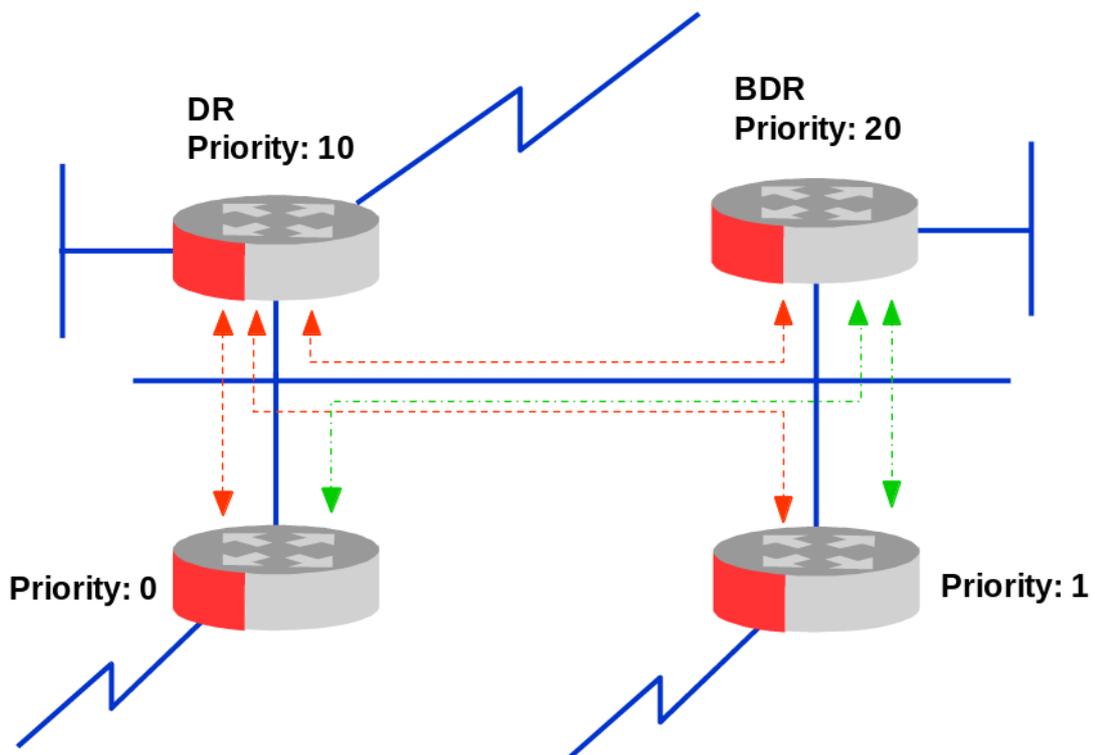
On multi-access networks (networks supporting more than two routers), the hello protocol is used to process the election of a DR and a BDR.

The DR/BDR election process is as follows:

- All routers create list of eligible routers:
 - Priority greater than 0.
 - OSPF State of 2 way.
 - DR or BDR IP Address in same network as interface.

- The BDR is chosen first which is the router with the highest priority.
- The DR is chosen from the remaining routers again the one with the highest priority.
- If there were not enough routers to have a BDR and a DR then the BDR becomes the DR.
- If the priorities are equal the Router ID is used as a tie-breaker.

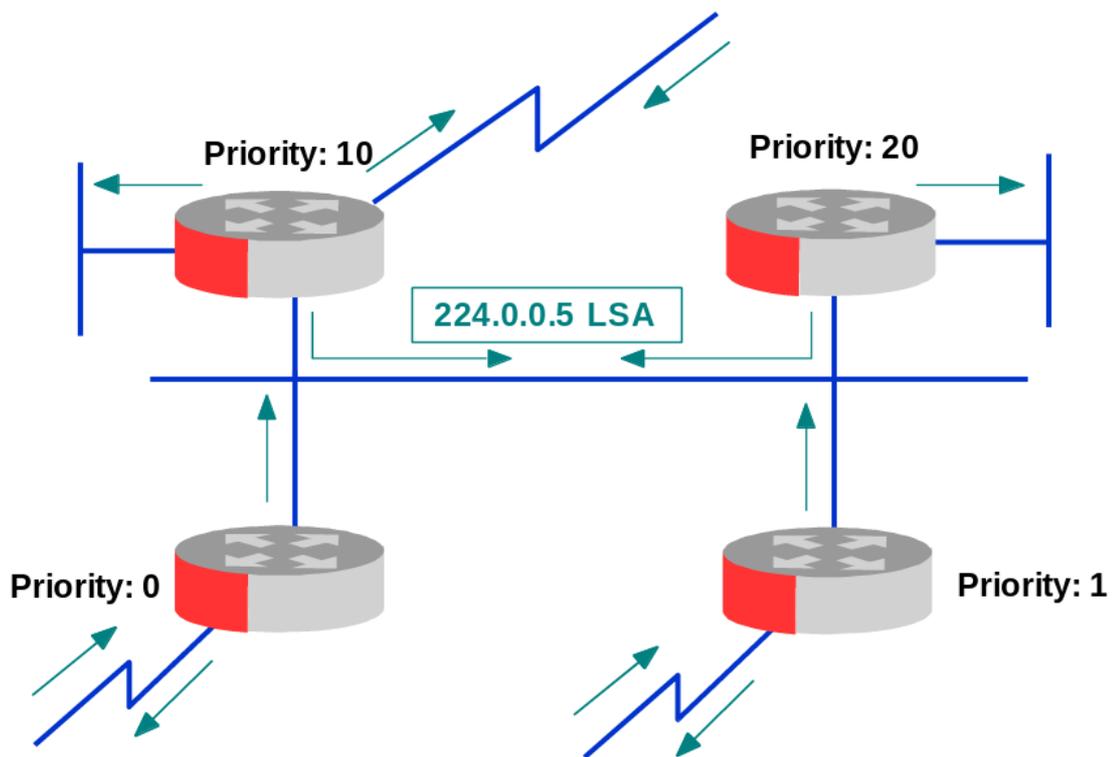
Adjacencies



Among other things, the DR is responsible for generating LSA for the entire multi-access network. Designated routers allow a reduction in network traffic and in the size of the topological database.

When the link-state databases of two neighbouring routers are synchronised, the routers are said to be adjacent.

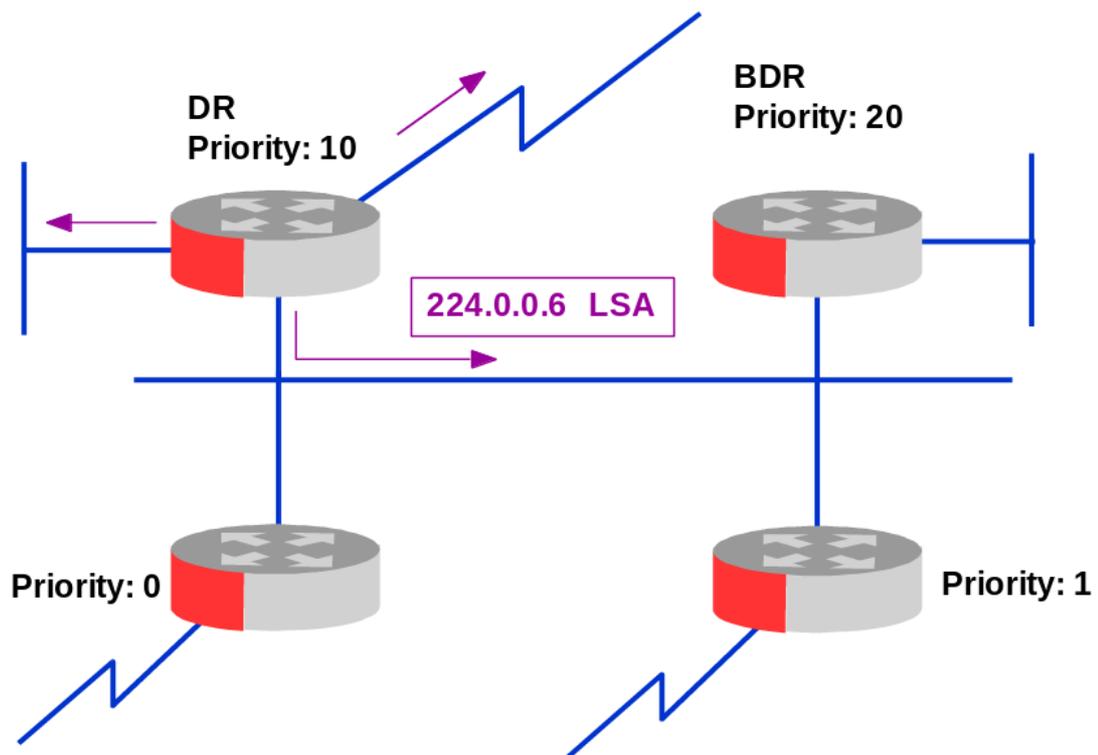
On multi-access networks, the designated router determines which routers should become adjacent. Topological databases are synchronised between pairs of adjacent routers. Adjacencies control the distribution of routing-protocol packets, which are sent and received only on adjacencies.



Router Link State Advertisements (Type 1)

Each router periodically sends an LSA to provide information on a router's adjacencies or to inform others when a router's state changes. By comparing established adjacencies to link states, failed routers can be detected quickly, and the network's topology can be altered appropriately. From the topological database generated from LSAs, each router calculates a Shortest Path Tree (SPT), with itself as root. The shortest-path tree, in turn, yields a routing table.

Once the routing table has been established from the shortest path tree only hello packets are exchanged as a form of heart beat. Every OSPF speaker sends small hello packets out each of its interfaces every ten seconds. Hello packets are not forwarded or recorded in the OSPF database, but if none are received from a particular neighbour for forty seconds, that neighbour is marked down. LSAs are then generated marking links through a down router as down. The hello timer values can be configured, though they must be consistent across all routers on a network segment.



Network Link State Advertisements (Type 2)

To reduce the effect of flooding DRs send information about the state of routers it is designated for to other DRs within the same area. These Network LSAs are sent to the multicast address `224.0.0.6`.

Other Link State Advertisements

OSPF has other LSA types associated with Area Border Routers (ABR) and Autonomous System Border Routers (ASBR) which are outside the scope of the course.

- Summary Link Advertisements (Type 3 and 4).
- External Link Advertisements (Type 5).
- Not So Stubby Area (NSSA) External Link Advertisements (Type 7).

OSPF Timers

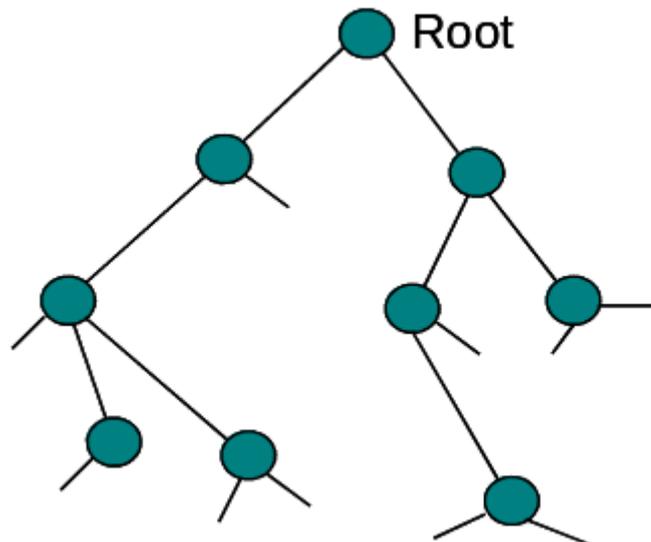
Link state advertisements also age. The originating router re-advertises an LSA after it has remained unchanged for thirty minutes. If an LSA ages to more than an hour, it is flushed from the databases. These timer values are called architectural constants by RFC 2328 and 5340.

OSPFs various timers interact as follows:

- If a link goes down for twenty seconds, then comes back up, OSPF doesn't notice.
- If a link flaps constantly, but at least one of every four Hello packets make it across, OSPF doesn't notice.
- If a link goes down for anywhere from a minute to half an hour, OSPF floods an LSA when it goes down, and another LSA when it comes back up.
- If a link stays down for more than half an hour, LSAs originated by remote routers (that have become unreachable) begin to age out. When the link comes back up, all these LSAs will be re-flooded.
- If a link is down for more than an hour, any LSAs originated by remote routers will have aged out and been flushed. When the link comes back up, it will be as if it were brand new.

OSPF Shortest Path Tree

Now that each router has an identical LSDB each router uses Dijkstra's SPF algorithm to calculate the SPT. During the computation of the SPT, the shortest path to each node is discovered. The routing table is populated from the topology tree.



6.3 Quagga Introduction

Quagga Routing Software Suite is a GNU General Public License (GPL) licensed advanced routing software package that provides a suite of TCP/IP based routing protocols.

- RIP
- RIPng
- OSPFv2
- OSPFv3
- Babel
- BGP4

The Quagga architecture consists of a core routing daemon, *zebra* and a number of routing protocol daemons. *zebra* itself is the kernel interface, handles static routes and is the zserv FTP server for the transferring of zebra files. Each routing protocol has its own daemon, with for example *opdfd* daemon handling OSPFv2 operations and *ospf6d* the OPSFv3 operations.

Having mastered the application of IP Networking on a GNU/Linux OS in the role of a single host on a network we will now look at the use of Quagga to allow the GNU/Linux host with multiple network interfaces to act as a router in a single area OSPF network for both IPv4 and IPv6.



6.4 Install Quagga

Quagga is the GNU/Linux BGP/OSPF/RIP routing daemon and *quagga-doc* is the documentation files for quagga which are generally stored in */usr/share/doc/quagga*.

```
$ sudo apt-get install quagga quagga-doc
```

Enable the *zebra*, *ospfd* and *ospf6d* daemons.

```
$ sudo vi /etc/quagga/daemons
zebra=yes
bgpd=no
ospfd=yes
ospf6d=yes
ripd=no
ripngd=no
isisd=no
```

Copy the example files for the daemons to the */etc/quagga* directory.

```
$ sudo cp /usr/share/doc/quagga/examples/zebra.conf.sample /etc/quagga/zebra.conf
$ sudo cp /usr/share/doc/quagga/examples/ospfd.conf.sample /etc/quagga/ospfd.conf
$ sudo cp /usr/share/doc/quagga/examples/ospf6d.conf.sample /etc/quagga/ospf6d.conf
$ sudo cp /usr/share/doc/quagga/examples/vtysh.conf.sample /etc/quagga/vtysh.conf
```

Set the following *users*, *groups* and file permissions on the files in */etc/quagga*.

```
$ sudo chown quagga:quagga /etc/quagga/zebra.conf
$ sudo chown quagga:quagga /etc/quagga/ospfd.conf
$ sudo chown quagga:quagga /etc/quagga/ospf6d.conf
$ sudo chown quagga:quaggavty /etc/quagga/vtysh.conf

$ sudo chmod 640 /etc/quagga/zebra.conf
$ sudo chmod 640 /etc/quagga/ospfd.conf
$ sudo chmod 640 /etc/quagga/ospf6d.conf
$ sudo chmod 660 /etc/quagga/vtysh.conf
```

6.5 Configure the Quagga configuration files

6.5.1 *debian.conf*

The *debian.conf* file has the list of devices that can telnet to the various daemons, by default this is limited to the localhost but additional IP addresses can be added as shown for *zebra*, *ospfd* and *ospf6d*.

```
$ cat /etc/quagga/debian.conf

vtysh_enable=yes
zebra_options="  --daemon -A 127.0.0.1 78.143.141.20"
bgpd_options="  --daemon -A 127.0.0.1"
ospfd_options="  --daemon -A 127.0.0.1 78.143.141.20"
ospf6d_options="  --daemon -A ::1 2a02:aaaa::20"
ripd_options="  --daemon -A 127.0.0.1"
ripngd_options="  --daemon -A ::1"
isisd_options="  --daemon -A 127.0.0.1"
babeld_options="  --daemon -A 127.0.0.1"
#
watchquagga_enable=yes
watchquagga_options=(--daemon)
```

6.5.2 vtysh.conf - the VTY terminal conf file

Edit the `/etc/quagga/vtysh.conf` file. Note that the `service integrated-vtysh-config` is disabled which is the recommended setting. Also set the `hostname`.

```
$ sudo vi /etc/quagga/vtysh.conf

!
! Configuration file for vtysh.
!
!service integrated-vtysh-config
hostname R3
username root nopassword
!
```

6.6 zebra.conf - the routing daemon conf file

Edit the `zebra.conf` file.

```
$ sudo vi /etc/quagga/zebra.conf

! *- zebra *-
!
! zebra configuration file
!
hostname R3
password obquagga
enable password quaggapass
!
!
!log file /var/log/quagga/zebra.log
log stdout
```

6.6.1 The OSPFv2 (IPv4) daemon conf file

Edit the *ospfd.conf* file.

```
$ sudo vi /etc/quagga/ospfd.conf

! -*- ospf -*-
!
! OSPFd configuration file
!
hostname R3
password obquagga
enable password quaggapass
!
!
!log file /var/log/quagga/ospfd.log
log stdout
```

6.6.2 The OSPFv3 (IPv6) daemon conf file

Edit the *ospf6d.conf* file.

```
$ sudo vi /etc/quagga/ospf6d.conf

! -*- ospf6 -*-
!
! OSPF6d configuration file
!
hostname R3
password obquagga
enable password quaggapass
!
!
!log file /var/log/quagga/ospf6d.log
log stdout
```

6.7 Restart the Quagga service

After changing the configuration files, restart the *quagga* service to reread the configuration files.

```
$ sudo service quagga restart
```

```
Stopping Quagga monitor daemon: watchquagga.
Stopping Quagga daemons (prio:0): (ospfd) (ospf6d) (zebra) (bgpd) (ripd)
(ripngd) (isisd) (babeld).
Removing all routes made by zebra.
Loading capability module if not yet done.
Starting Quagga daemons (prio:10): zebra ospfd ospf6d.
Starting Quagga monitor daemon: watchquagga.
```

6.8 Accessing the Quagga router for configuration

The quagga router can be accessed using telnet to the port for the relevant daemon, routing is managed in the *zebra* daemon, OSPFv2 in the *ospfd* daemon and OSPFv3 in the *ospf6d* daemon. The same applies if you want to use RIP, RIPng or BGP4.

6.8.1 Access TCP Ports

Daemon	TCP port	Access daemon
zebra	2601	telnet 127.0.0.1 2601
ripd	2602	telnet 127.0.0.1 2602
ripng	2603	telnet ::1 2603
ospfd	2604	telnet 127.0.0.1 2604
bgpd	2605	telnet 127.0.0.1 2605
ospf6d	2606	telnet ::1 2606

6.8.2 Accessing the zebra daemon

```
$ sudo vtysh
```

```
Hello, this is Quagga (version 0.99.22.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.
```

```
R3#
```

or via telnet

```
$ telnet 127.0.0.1 2601
```

```
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^['.
```

```
Hello, this is Quagga (version 0.99.22.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.
```

```
User Access Verification
```

```
Password: obquagga
R3> en
Password: quaggapass
R3#
```

6.9 Configuring zebra daemon - the routing daemon

If you are used of the Cisco iOS syntax then Quagga is quite similar and should not present a problem. Here IPv4 and IPv6 addresses are added to the interfaces and ND messages are not suppressed.

```
R3# conf t
R3(config)# interface eth0
R3(config-if)# description "To R4"
R3(config-if)# ip address 89.184.35.9/30 label to-R4
R3(config-if)# ipv6 address 2a02:3::31/64
R3(config-if)# no ipv6 nd suppress-ra
R3(config-if)# no shut

R3(config-if)# interface eth1
R3(config-if)# description "To R2"
R3(config-if)# ip address 89.184.35.6/30 label to-R2
R3(config-if)# ipv6 address 2a02:2::22/64
R3(config-if)# no ipv6 nd suppress-ra
R3(config-if)# no shut
R3(config-if)# <CTRL> - Z
```

Save the configuration and review.

```
R3# copy run start
Configuration saved to /etc/quagga/zebra.conf
```

```
R3# show run
```

```
hostname R3
log stdout
hostname ospfd
hostname ospf6d@plant
!
service advanced-vty
!
debug ospf6 lsa unknown
debug ospf6 neighbor state
!
password obquagga
enable password quaggapass
password zebra
!
interface eth0
  description "To R4"
  ip address 89.184.35.9/30 label to-R4
  ipv6 address 2a02:3::31/64
  no ipv6 nd suppress-ra
!
interface eth1
  description "To R2"
  ip address 89.184.35.6/30 label to-R2
  ipv6 address 2a02:2::22/64
  no ipv6 nd suppress-ra
!
```

```

interface lo
!
interface fxp0
  ipv6 ospf6 priority 0
!
interface lo0
!
router ospf6
  router-id 255.1.1.1
  redistribute static route-map static-ospf6
  interface fxp0 area 0.0.0.0
!
access-list access4 permit 127.0.0.1/32
!
ipv6 access-list access6 permit 3ffe:501::/32
ipv6 access-list access6 permit 2001:200::/48
ipv6 access-list access6 permit ::1/128
!
ipv6 prefix-list test-prefix seq 1000 deny any
!
route-map static-ospf6 permit 10
  match ipv6 address prefix-list test-prefix
  set metric 2000
  set metric-type type-2
!
line vty
  access-class access4
  exec-timeout 0 0
  ipv6 access-class access6
!

```

Reviewing the routes on the GNU/Linux host as configured by Quagga.

\$ sudo ip addr list

```

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
    link/ether 00:12:3f:dc:ab:47 brd ff:ff:ff:ff:ff:ff
    inet 89.184.35.9/30 brd 89.184.35.11 scope global to-R4
        valid_lft forever preferred_lft forever
    inet6 2a02:3::31/64 scope global
        valid_lft forever preferred_lft forever
    inet6 2a02:3::212:3fff:fedc:ab47/64 scope global dynamic
        valid_lft 2591768sec preferred_lft 604568sec
    inet6 fe80::212:3fff:fedc:ab47/64 scope link
        valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:60:6e:00:66:13 brd ff:ff:ff:ff:ff:ff
    inet 89.184.35.6/30 brd 89.184.35.7 scope global to-R2
        valid_lft forever preferred_lft forever
    inet6 2a02:2::15db:54cc:1f83:4a46/64 scope global temporary dynamic
        valid_lft 604789sec preferred_lft 85789sec
    inet6 2a02:2::260:6eff:fe00:6613/64 scope global dynamic
        valid_lft 2591989sec preferred_lft 604789sec
    inet6 2a02:2::22/64 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::260:6eff:fe00:6613/64 scope link
        valid_lft forever preferred_lft forever

```

6.10 Configuring the OSPFv2 daemon

Telnet to the OSPFv2 Daemon.

```
$ telnet 127.0.0.1 2604
```

```
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^['.
```

```
Hello, this is Quagga (version 0.99.22.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.
```

User Access Verification

```
Password:
R3> en
Password:
R3#
```

Configure OSPFv2, add a unique Router ID, i.e. *0.0.0.3* for *R3*. Add the networks that this router will advertise. In this case *89.184.35.0/24* summarises for all the *89.184.35.x/30* networks.

```
R3# conf t
R3(config)# router ospf
R3(config-router)# router-id 0.0.0.3
R3(config-router)# network 89.184.35.0/24 area 0
R3(config-if)# <CTRL> - Z
```

See what OSPFv2 neighbours R3 discovered on its interfaces.

```
R3# show ip ospf neighbor
```

Neighbor	ID	Pri	State	Dead Time	Address	Interface	RXmtL	RqstL	DBsmL
0.0.0.4		1	Full/DR	38.866s	89.184.35.10	eth0:89.184.35.9	0	0	0
0.0.0.2		1	Full/DR	31.353s	89.184.35.5	eth1:89.184.35.6	0	0	0

Review the routes learned by the OSPFv2 daemon.

```
R3# show ip ospf route
```

```
===== OSPF network routing table =====
N    78.143.141.0/24    [30] area: 0.0.0.0
      via 89.184.35.5, eth1
N    78.143.152.0/24    [20] area: 0.0.0.0
      via 89.184.35.10, eth0
N    89.184.35.0/30     [20] area: 0.0.0.0
      via 89.184.35.5, eth1
N    89.184.35.4/30     [10] area: 0.0.0.0
      directly attached to eth1
N    89.184.35.8/30     [10] area: 0.0.0.0
      directly attached to eth0
```

Save all configuration on the OSPFv2 daemon.

```
R3# copy run start
Configuration saved to /etc/quagga/ospfd.conf
R3#
```

Returning to *zebra* on TCP port 2601 review the IP routes table.

```
O>* 78.143.141.0/24 [110/30] via 89.184.35.5, eth1, 00:00:02
O>* 78.143.152.0/24 [110/20] via 89.184.35.10, eth0, 00:04:24
O>* 89.184.35.0/30 [110/20] via 89.184.35.5, eth1, 00:04:22
O 89.184.35.4/30 [110/10] is directly connected, eth1, 00:04:29
C>* 89.184.35.4/30 is directly connected, eth1
O 89.184.35.8/30 [110/10] is directly connected, eth0, 00:04:29
C>* 89.184.35.8/30 is directly connected, eth0
C>* 127.0.0.0/8 is directly connected, lo
```

or review the routes in GNU/Linux directly. One disadvantage of this however is that the information on the protocols the routes were learnt from, or the administrative distances are not displayed.

```
$ ip -4 route list
```

```
78.143.141.0/24 via 89.184.35.5 dev eth1 proto zebra metric 30
78.143.152.0/24 via 89.184.35.10 dev eth0 proto zebra metric 20
89.184.35.0/30 via 89.184.35.5 dev eth1 proto zebra metric 20
89.184.35.4/30 dev eth1 proto kernel scope link src 89.184.35.6
89.184.35.8/30 dev eth0 proto kernel scope link src 89.184.35.9
```

6.11 Configure the OSPFv3 (for IPv6) daemon

Telnet to the OSPF Daemon.

```
$ telnet ::1 2606
```

```
Trying ::1...
Connected to ::1.
Escape character is '^']'.
```

```
Hello, this is Quagga (version 0.99.22.1).
Copyright 1996-2005 Kunihiro Ishiguro, et al.
```

User Access Verification

```
Password:
R3> en
Password:
R3#
```

Configure OSPFv3 daemon is a similar way to the IPv4 version. In IPv6 case however instead of defining networks to be routed, interfaces in networks to be advertised are added to the IPSFv3 router configuration.

```
R3# conf t
R3(config)# router ospf6
R3(config-ospf6)# router-id 0.0.0.3
R3(config-ospf6)# interface eth0 area 0.0.0.0
R3(config-ospf6)# interface eth1 area 0.0.0.0
```

Review OSPFv3 neighbours and routes.

```
R3# show ipv6 ospf6 neighbor
```

Neighbor ID	Pri	DeadTime	State/IfState	Duration I/F[State]
0.0.0.4	1	00:00:40	Full/DR	00:02:50 eth0[BDR]
0.0.0.2	1	00:00:32	Full/DR	00:02:47 eth1[BDR]

```
R3# show ipv6 ospf6 route
```

*N IA 2a02:1::/64	fe80::20c:42ff:fe8b:73e4	eth1 00:03:43
*N IA 2a02:2::/64	::	eth1 00:03:53
N IA 2a02:2::/64	fe80::20c:42ff:fe8b:73e4	eth1 00:03:43
*N IA 2a02:3::/64	::	eth0 00:03:55
*N IA 2a02:aaaa::/64	fe80::20c:42ff:fe8b:73e4	eth1 00:03:43
*N IA 2a02:bbbb::/64	fe80::d6ca:6dff:fe61:dd89	eth0 00:03:45

Save the OSPFv3 daemon configuration.

```
R3# copy run start
Configuration saved to /etc/quagga/ospf6d.conf
R3#
```

Returning to *zebra* on TCP port 2601 review the IPv6 routes table.

```
R3# show ipv6 route
```

```
Codes: K - kernel route, C - connected, S - static, R - RIPng,
       O - OSPFv6, I - IS-IS, B - BGP, A - Babel,
       > - selected route, * - FIB route
```

```
K>* ::/0 via fe80::20c:42ff:fe8b:73e4, eth1
C>* ::1/128 is directly connected, lo
O>* 2a02:1::/64 [110/11] via fe80::20c:42ff:fe8b:73e4, eth1, 00:07:36
O 2a02:2::/64 [110/1] is directly connected, eth1, 00:07:46
C>* 2a02:2::/64 is directly connected, eth1
O 2a02:3::/64 [110/1] is directly connected, eth0, 00:07:48
C>* 2a02:3::/64 is directly connected, eth0
O>* 2a02:aaaa::/64 [110/21] via fe80::20c:42ff:fe8b:73e4, eth1, 00:07:36
O>* 2a02:bbbb::/64 [110/11] via fe80::d6ca:6dff:fe61:dd89, eth0, 00:07:38
C * fe80::/64 is directly connected, eth1
C>* fe80::/64 is directly connected, eth0
```

or review the routes in GNU/Linux directly. This has a similar disadvantage to that described for IPv4 above.

```
$ ip -6 route list
2a02:1::/64 via fe80::20c:42ff:fe8b:73e4 dev eth1 proto zebra metric 11
2a02:2::/64 dev eth1 proto kernel metric 256
2a02:3::/64 dev eth0 proto kernel metric 256 expires 2591777sec
2a02:aaaa::/64 via fe80::20c:42ff:fe8b:73e4 dev eth1 proto zebra metric 21
2a02:bbbb::/64 via fe80::d6ca:6dff:fe61:dd89 dev eth0 proto zebra metric 11
fe80::/64 dev eth0 proto kernel metric 256
fe80::/64 dev eth1 proto kernel metric 256
default via fe80::d6ca:6dff:fe61:dd89 dev eth0 proto static metric 1
default via fe80::d6ca:6dff:fe61:dd89 dev eth0 proto ra metric 1024 expires 1577sec
default via fe80::20c:42ff:fe8b:73e4 dev eth1 proto ra metric 1024 expires 1770sec
```

6.12 Quagga Summary

Quagga is a family of Internet Routing daemons that turn a GNU/Linux host into a full router. In this instance you have seen the use of OSPF in a single backbone (0.0.0.0) area to perform routing however this is only a sample of what can be achieved with Quagga. For example intra-area OSPF routing, IS-IS, BGP etc..

7. Wireless LANs

7.1 Introduction to WiFi

IEEE 802.11 is a set of standards for Wireless Local Area Network (WLAN) computer communication, developed by the IEEE LAN/MAN Standards Committee (IEEE 802) in the 5 GHz and 2.4 GHz public spectrum bands.

The IEEE 802.11 family includes over-the-air modulation techniques that use the same basic protocol. The most popular are those defined by the IEEE 802.11b and IEEE 802.11g protocols, and are amendments to the original standard. IEEE 802.11a was the first wireless networking standard, but IEEE 802.11b was the first widely accepted one, followed by *g*, *n* and *ac*. Security was originally purposefully weak due to export requirements of some governments, and was later enhanced via the IEEE 802.11i amendment after US governmental and legislative changes. IEEE 802.11n added Multiple In, Multiple Out (MIMO) antenna technology with multiple spacial streams to increase speeds to 200 Mb/s with a maximum throughput of 500 Mb/s. IEEE 802.11ac, the 1 Gb/s standard with a single link throughput of at least 500 Mb/s. It uses up to 160 MHz Radio Frequency bandwidth, up to 8 spacial streams for MIMO and 256 QAM modulation. Other standards in the family (*c-f*, *h*, *j*) are service amendments and extensions or corrections to previous specifications.

The segment of the radio frequency spectrum used varies between countries. In the US, IEEE 802.11a and *g* devices may be operated without a license, as explained in Part 15 of the Federal Communications Commission (FCC) Rules and Regulations. Frequencies used by channels one through six (IEEE 802.11b) fall within the 2.4 GHz amateur radio band. Licensed amateur radio operators may operate IEEE 802.11b/g devices under Part 97 of the FCC Rules and Regulations, allowing increased power output but not commercial content or encryption.

- IEEE 802.11 – This is the initial WLAN standard and provides 1 or 2 Mbps transmission in the 2.4 GHz band using either Frequency Hopping Spread Spectrum (FHSS) or Direct Sequence Spread Spectrum (DSSS).
- IEEE 802.11a – This is an extension to 802.11 that provides typically 25 Mbps to a maximum of 54 Mbps in the 5GHz band. 802.11a uses an OFDM encoding scheme rather than FHSS or DSSS. Max range is 30 M.
- IEEE 802.11b (also referred to as 802.11 High Rate or WiFi) – This is also an extension to 802.11 that provides 11 Mbps transmission (with a fallback to 5.5, 2 and 1 Mbps) in the 2.4 GHz band. 802.11b uses only DSSS. 802.11b was a 1999 ratification to the original 802.11 standard, allowing wireless functionality comparable to Ethernet. Max range is 30 M.

- IEEE 802.11g -- Provides typically 24 Mbps to a maximum of 54 Mbps in the 2.4 GHz band. It also uses OFDM. Max range is 30 M.
- IEEE 802.11n – A new standard to give typically 200 Mbps up to a maximum of 540 Mbps out to 50 M in either the 2.4 or 5 GHz bands. This standard introduced MIMO antennas to WLAN.
- IEEE 802.11ac – A further improvement to the IEEE 802.11n standard to include wider channels (80 or 160 MHz versus 20 or 40 MHz) in the 5 GHz band, up to eight spatial streams, 256 Quadrature Amplitude Modulation (QAM) and Multi-user MIMO (MU-MIMO).

7.1.1 Spectrum

2.4 GHz

The IEEE 802.11b standard defines a total of 14 frequency channels in the 2.4 GHz WiFi signal range. In Europe the first 13 channels are available for use, while in the United States, only the WiFi channels 1 - 11 can be chosen. In Japan, all 14 channels are licensed for IEEE 802.11b.

Many wireless products ship with a default WiFi channel of 6. If encountering interference from other devices within the home, consider changing the channel up or down to avoid it. Note that all WiFi devices on the network must use the same channel.

Some WiFi channel numbers overlap with each other. Channel 1 uses the lowest frequency band and each subsequent channel increases the frequency slightly. Therefore, the further apart two channel numbers are, the less the degree of overlap and likelihood of interference. If encountering interference with a neighbour's WLAN, change to a distant channel. Both channels 1 and 11 do not overlap with the default channel 6; use one of these three channels for best results.

2.4 Ghz Channels (GHz)

Channel	Frequency (GHz)	Notes
1	2.412	
2	2.417	
3	2.422	
4	2.427	
5	2.432	
6	2.437	
7	2.442	
8	2.447	
9	2.452	
10	2.457	
11	2.462	
12	2.467	No North America
13	2.472	No North America
14	2.484	Japan only

5 GHz

IEEE 802.11a defines the physical air interface for up to 200 channels in the 5 GHz unlicensed spectrum with a channel size of 20 MHz. Channel centre frequency for each channel can be calculated by the formula $5000 + 5 \times N_{ch}$ (MHz) where $N_{ch} = 0 - 200$.

It is based on an OFDM implementation using up to 64-QAM (256 QAM introduced with IEEE 802.11ac). These channels have been incorporated into the higher speed IEEE 802.11n and IEEE 802.11ac standards.

5 Ghz Channels (GHz)

Channel	Frequency (GHz)
36	5.18
40	5.2
44	5.22
48	5.24
52	5.26
56	5.28
60	5.3
64	5.32
100	5.5
104	5.52
108	5.54
112	5.56
116	5.58
120	5.6
124	5.62
128	5.64
132	5.66
136	5.68
140	5.7

In some countries the spectrum is extended for the provision of Fixed Wireless Access Networks/Metropolitan Area Networks (FWA/MAN) in the 5.8 GHz (5.725 – 5.875 GHz) band up to a maximum radiated power of 2 Watt Effective Isotropic Radiated Power (EIRP) on a licence exempt basis. This gives an additional 7 20 MHz channels, 5.745, 5.765, 5.785, 5.805, 5.825, 5.845, 5.865 GHz.

MIMO Antenna's

MIMO is the use of multiple antennas at both at the transmitter and receiver to improve communication performance. It is one of several forms of Smart Antenna (SA), and the state of the art of SA technology. MIMO technology has attracted attention in wireless communications, since it offers significant increases in data throughput and link range without additional bandwidth or transmit power. It achieves this by higher spectral efficiency (more bits per second per Hertz of bandwidth) and link reliability or diversity (reduced fading).

7.1.2 IEEE 802.11 WLAN Summary

IEEE Designation	Modulation	Max Speed	Operating Frequency	Non-overlapping channels	Antenna	Range (Indoor/Outdoor)
802.11b	DSSS	11 Mbps	2.4 GHz	3		~38 M / ~140 M
802.11a	OFDM	54 Mbps	5 GHz	12		~35 M / ~120 M
802.11g	OFDM	54 Mbps	2.4 GHz	3		~35 M / ~140 M
802.11n	OFDM	248 Mbps	2.4 (5) GHz	3 (12)	MIMO	~50 M / ~200 M
802.11ac	OFDM	867 Mbps	5 GHz	3	MU-MIMO	~50 M / ~200 M

7.1.3 IEEE 802.11 MAC (Media Access Control)

The following section describes the common Media Access Control layer used by the IEEE 802.11 family of standards. The IEEE 802.11 family uses a MAC layer known as CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) NOTE: Classic Ethernet uses CSMA/CD - Collision Detection. CSMA/CA is, like all Ethernet protocols, peer-to-peer (there is no requirement for a master station).

In CSMA/CA a Wireless node that wants to transmit performs the following sequence:

1. Listen on the desired channel.
2. If channel is idle (no active transmitters) it sends a packet.
3. If channel is busy (an active transmitter) node waits until transmission stops then a further CONTENTION period. (The Contention period is a random period after every transmit on every node and statistically allows every node equal access to the media. To allow TX to RX turn around the contention time is slotted 50 micro sec for Frequency Hopping (FH) and 20 micro sec for Direct Sequence (DS) systems).
4. If the channel is still idle at the end of the CONTENTION period the node transmits its packet otherwise it repeats the process defined in 3 above until it gets a free channel.

ACKing

At the end of every packet the receiver, if it has successfully received the packet, will return an ACK packet (if not received or received with errors the receiver will NOT respond i.e. there is no NACK). The transmit window allows for the ACK i.e. CONTENTION period starts after the ACK should have been sent.

MAC level retransmission

If no ACK is received the sender will retry the transmit (using the normal CSMA/CA procedures) until either successful or the operation is abandoned with exhausted retries.

Fragmentation

Bit error rates on wireless systems (10^{-5} , 10^{-6}) are substantially higher than wire-line systems (10^{-12}). Large blocks may approach the number of bits where the probability of an error occurring is so high that every block could fail including the re-transmission. To reduce the possibility of this happening large blocks may be fragmented by the transmitter and reassembled by the receiver node e.g. a 1500 byte block (12,000 bits) may be fragmented into 5 blocks of 300 bytes (2,400 bits). While there is some overhead in doing this - both the probability of an error occurring is reduced and, in the event of an error, the re-transmission time is also reduced.

7.1.4 WiFi Elements

IEEE 802.11 networks are organised in two ways:

- In infrastructure mode one station acts as a master with all the other stations associating to it; the network is known as a Basic Service Set (BSS) and the master station is termed an AP. In a BSS all communication passes through the AP; even when one station wants to communicate with another wireless station messages must go through the AP.
- In the second form of network there is no master and stations communicate directly. This form of network is termed an Independent Basic Service Set (IBSS) and is commonly known as an ad-hoc network.

Wireless Access Point (AP)

The AP is the hub of a wireless network. Wireless clients connect to the access point, and traffic between two clients must travel through the access point. Access Points are often abbreviated to AP, and you may also see them referred to as *wireless routers*, *wireless gateways*, and *Base Stations (BS)*.

Service Set Identifier (SSID)

An SSID is a secret key attached to all packets on a wireless network to identify each packet as part of that network. The code consists of a string of 1-32 octets. All wireless devices attempting to communicate with each other must share the same SSID. Apart from identifying each packet, an SSID also serves to uniquely identify a group of wireless network devices used in a given *Service Set*. There are two major variants of the SSID: Ad-hoc wireless networks (IBSS) that consist of client machines without an AP use the IBSS Identifier (IBSS-ID).

Infrastructure networks which include an AP (BSS or possibly an ESS) use the BSSID or ESS ID (E for Extended) instead.

The naming is for convention only as the IEEE 802.11 standard dictates that an IBSS, BSS, and ESS are each defined by an SSID, otherwise known as a *Network Name*. A network name is commonly set to the name of the network operator, such as a company name. Equipment manufacturers have liberally used all of the above SSID naming conventions to essentially describe the same thing. In some instances, the convention is wrong, as in the case of BSSID.

The SSID on wireless clients can be set either manually, by entering the SSID into the client network settings, or automatically, by leaving the SSID unspecified or blank. A network administrator often uses a public SSID that is set on the access point and broadcast to all wireless devices in range.

Most IEEE 802.11 access point vendors allow the use of an SSID of *any* to enable an IEEE 802.11 NIC to connect to any IEEE 802.11 network.

Disabling SSID Broadcasting

Many wireless AP vendors have added a configuration option which lets you disable broadcasting of the SSID. This adds little security because it is only able to prevent the SSID from being broadcast with Probe Request and Beacon frames. The SSID must be broadcast with Probe Response frames. In addition, the wireless access cards will broadcast the SSID in their association and re-association frames. Because of this, the SSID cannot be considered a valid security tool.

7.1.5 WiFi Security

The ability to enter a network while mobile has great benefits. However, wireless networking has many security issues. Hackers have found wireless networks relatively easy to break into in the past, and even use wireless technology to crack into wired networks.

Methods of counteracting security risks

There are many technologies available to counteract wireless network intrusion, but currently no method is absolutely secure. The best strategy may be to combine a number of security measures. There are three steps to take towards securing a wireless network:

1. All wireless LAN devices need to be secured.
2. All users of the wireless network need to be educated in wireless network security.
3. All wireless networks need to be actively monitored for weaknesses and breaches.

Wireless Encryption Protocol (WEP)

WEP is part of the IEEE 802.11 wireless networking standard. Because wireless networks broadcast messages using radio, they are susceptible to eavesdropping.

WEP was intended to provide confidentiality comparable to that of a traditional wired network. Several serious weaknesses were identified by cryptanalysts; a WEP connection can be cracked with readily available software within minutes. WEP was superseded by WiFi Protected Access (WPA) in 2003, followed by the full IEEE 802.11i standard in 2004. Despite its weaknesses, WEP provides a level of security that may deter casual snooping.

Standard 64-bit WEP uses a 40 bit key, which is concatenated with a 24-bit Initialisation Vector (IV) to form the 64-bit Rivest Cipher 4 (RC4) traffic key. At the time that the original WEP standard was being drafted, US Government export restrictions on cryptographic technology limited the key size. Once the restrictions were lifted, all of the major manufacturers eventually implemented an extended 128-bit WEP protocol using a 104-bit key size.

A 128-bit WEP key is almost always entered by users as a string of 26 Hexadecimal (Hex) characters (0-9 and A-F). Each character represents 4 bits of the key. $4 \times 26 = 104$ bits; adding the 24-bit initialisation vector brings us what we call a "128-bit WEP key". A 256-bit WEP system is available from some vendors, and as with the above-mentioned system, 24 bits of that is for the IV, leaving 232 actual bits for protection. This is typically entered as 58 Hexadecimal characters. $(58 \times 4 = 232 \text{ bits}) + 24 \text{ I.V. bits} = 256 \text{ bits of WEP protection}$.

Key size is not the only major security limitation in WEP. Cracking a longer key requires interception of more packets, but there are active attacks that stimulate the necessary traffic. There are other weaknesses in WEP, including the possibility of IV collisions and altered packets that are not helped at all by a longer key.

WiFi Protected Access (WPA)

WPA is a class of security standard to secure WiFi networks. It was created in response to several serious weaknesses researchers had found in the previous system, WEP. WPA implements the majority of the IEEE 802.11i standard, and was intended as an intermediate measure to take the place of WEP while IEEE 802.11i was prepared. WPA is designed to work with all wireless network interface cards, but not necessarily with first generation wireless access points. WPA2 (also called IEEE 802.11i) implements the full standard, but will not work with some older network cards. Both provide good security, with two significant issues:

- Either WPA or WPA2 must be enabled and chosen in preference to WEP. WEP is usually presented as the first security choice in most installation instructions.
- In the *Personal* mode, the most likely choice for homes and small offices, a passphrase is required that, for full security, must be longer than the typical 6 to 8 character passwords users are taught to employ.

WPA resolves the issue of weak WEP headers, which are called Initialisation Vectors (IV), and insures the integrity of the messages passed through (Message Integrity Check (MIC) using Temporal Key Integrity Protocol (TKIP) to enhance data encryption.

WPA Pre Shared Key (WPA-PSK) is a special mode of WPA for home users without an enterprise authentication server and provides the same strong encryption protection.

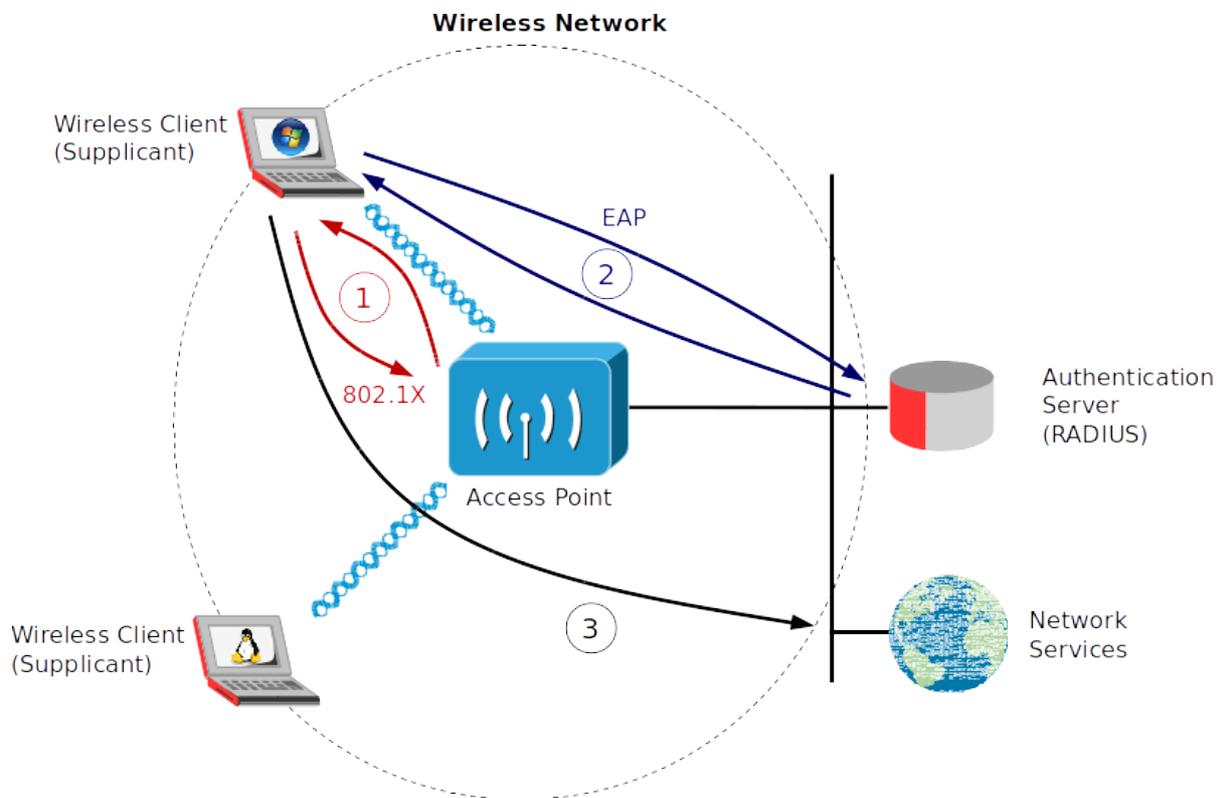
Security in pre-shared key mode

PSK mode is designed for home and small office networks that cannot afford the cost and complexity of an IEEE 802.1X authentication server. Each user must enter a passphrase to access the network. The passphrase may be from 8 to 63 printable American Standard Code for Information Interchange (ASCII) characters or 64 hexadecimal digits (256 bits). If you choose to use the ASCII characters, a hash function reduces it from 504 bits (63 characters * 8 bits/character) to 256 bits (using also the SSID). The passphrase may be stored on the user's computer at their discretion under most operating systems to avoid re-entry. The passphrase must remain stored in the WiFi access point.

Security is strengthened by employing a Password-Based Key Derivation Function version 2 (PBKDF2). However, the weak passphrases users typically employ are vulnerable to password cracking attacks. Some consumer chip manufacturers have attempted to bypass weak passphrase choice by adding a method of automatically generating and distributing strong keys through a software or hardware interface that uses an external method of adding a new WiFi adapter or appliance to a network.

Security with an Authentication Server

With WPA the use of IEEE 802.1x is supported for operation with databases of users stored in Remote Access Dialin User Service (RADIUS) and this is accessed using Extensible Authentication Protocol (EAP).



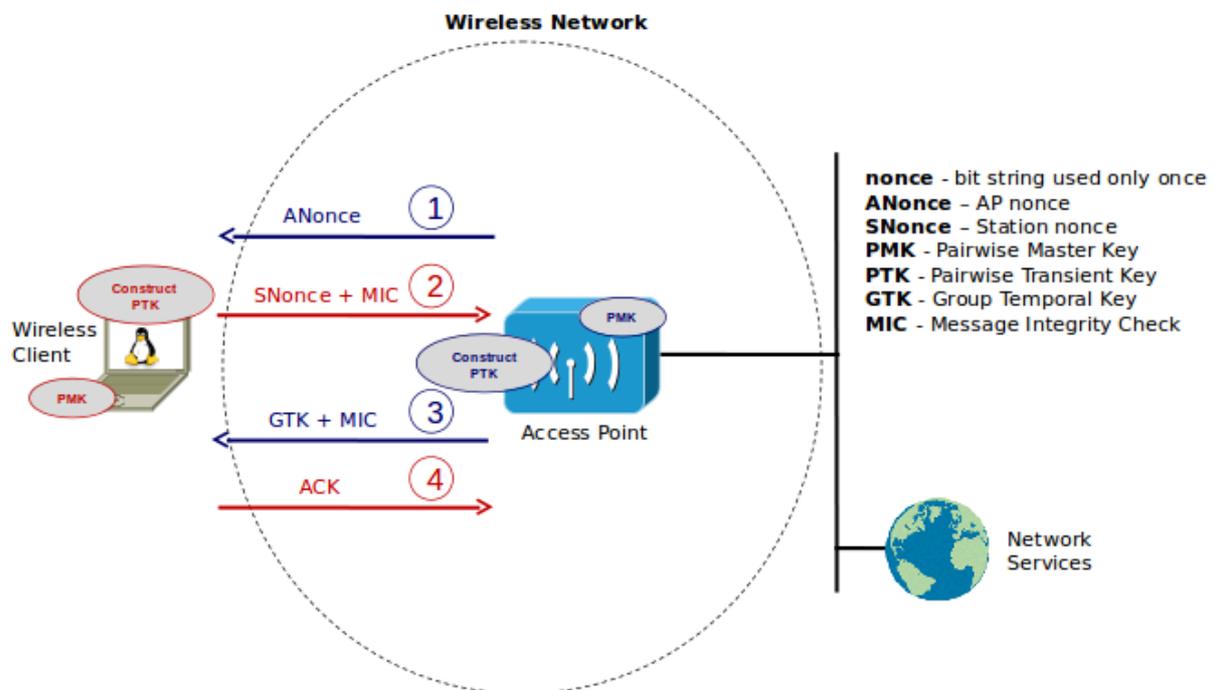
IEEE 802.1X is an IEEE standard for port-based network access control. It provides authentication to devices attached to a LAN port, establishing a point-to-point connection or preventing access from that port if authentication fails. It is used for certain closed wireless APs, and is based on the EAP.

Some vendors are implementing IEEE 802.1X for wireless APs, to be used in certain situations where an access point needs to be operated as a closed AP, addressing the security vulnerabilities of WEP. The authentication is usually done by a third-party entity, such as a RADIUS server. This provides for client-only authentication, or more appropriately, strong mutual authentication using protocols such as RFC 5216 EAP-Transport Layer Security (EAP-TLS).

Upon detection of the new client or the supplicant, the port on the switch (authenticator) will be enabled and set to the "unauthorised" state. In this state, only IEEE 802.1X traffic will be allowed; other traffic, such as Dynamic Host Configuration Protocol (DHCP) and Hypertext Transfer Protocol (HTTP), will be blocked at the data link layer. The authenticator will send out the EAP-Request identity to the supplicant, the supplicant will then send out the EAP-response packet that the authenticator will forward to the authenticating server. The authenticating server can accept or reject the EAP-Request; if it accepts the request, the authenticator will set the port to the "authorised" mode and normal traffic will be allowed. When the supplicant logs off, he will send an EAP-logoff message to the authenticator. The authenticator will then set the port to the "unauthorised" state, once again blocking all non-EAP traffic.

IEEE 802.11i WPA2

The WiFi Alliance approved full IEEE 802.11i as WPA2, also called Robust Security Network (RSN). WPA2 implements the mandatory elements of 802.11i. It introduces Advanced Encryption Standard (AES) algorithm based algorithm, Counter Mode with Cipher Block Chaining Message Authentication Code Protocol (CCMP) that is considered fully secure.



The authentication process has two considerations:

- the AP still needs to authenticate itself to the client station.
- keys to encrypt the traffic need to be derived.

An EAP exchange provides the shared secret key Pairwise Master Key (PMK). This key is however designed to last the entire session and should be exposed as little as possible. Therefore the four-way handshake is used to establish another key called the Pairwise Transient Key (PTK). The PTK is generated by concatenating the following attributes: PMK, AP Nonce (ANonce), Station Nonce (SNonce), AP MAC address and Station MAC address. In cryptography, a Nonce is a random, arbitrary number that is generated for security purposes and is used one time only. The product is then put through a cryptographic hash function.

The handshake also yields the Group Temporal Key (GTK), used to decrypt multicast and broadcast traffic. The actual messages exchanged during the handshake are depicted in the diagram.

7.2 Configuration of a WiFi network on GNU/Linux

In order to configure WiFi interfaces, the *wireless-tools* package (as well *ip*) can be used. This package uses the commands starting with *iw* command to configure a wireless interface, but this can also be carried out through the */etc/network/interfaces* file.

7.2.1 Install the wireless-tools package

Like was the case for net-tools the *wireless-tools* package has been replaced by the *iw* tool for configuring GNU/Linux wireless devices. This tool can show and manipulate wireless devices and their configurations. It replaces the *iwconfig* tool.

```
$ apt-get install iw
```

Before getting into the *iw* tool it is important to understand the WiFi card chipset information.

```
$ lspci | grep Network
04:00.0 Network controller: Broadcom Corporation BCM43228 802.11a/b/g/n
```

Here is a demonstration of some of these in action. Make sure that the *network-manager* is disabled.

```
$ sudo stop network-manager
```

Search for the interface on the host that is wireless capable.

```
$ iw dev eth1 info
Interface eth1
  ifindex 3
  type managed
  wiphy 0
```

Now bring up the discovered wireless interface.

```
$ sudo ip link set dev eth1 up
```

Run a scan to see if there are any APs available in the area.

```
$ sudo iw dev eth1 scan |grep SSID
SSID: fta_ssid
SSID: 20snh
SSID: 0B
SSID: SKYBF102
SSID: UPC1373998
```

The full scan is quite long so I extract the data for the BSS associated with the SSID *fta_ssid* only. In this case there is no security set.

```
$ sudo iw dev eth1 scan ssid fta_ssid
BSS d4:ca:6d:61:dd:8d (on eth1)
  TSF: 0 usec (0d, 00:00:00)
  freq: 2412
  beacon interval: 100
  capability: ESS ShortPreamble ShortSlotTime (0x0421)
  signal: -11.00 dBm
  last seen: 0 ms ago
  Information elements from Probe Response frame:
  SSID: fta_ssid
  Supported rates: 1.0* 2.0 5.5 11.0
  DS Parameter set: channel 1
```

Connect to the BSS with the SSID of *fta_ssid*.

```
$ sudo iw dev eth1 connect -w fta_ssid
eth1 (phy #0): connected to d4:ca:6d:61:dd:8d
```

This can be confirmed with the *iw link* command.

```
$ iw dev eth1 link
Connected to d4:ca:6d:61:dd:8d (on eth1)
  SSID: fta_ssid
  freq: 2412
```

7.2.2 Using WPA2

WPA and WPA2 are two security protocols developed by the WiFi Alliance to secure wireless computer networks as the WEP standard from the original IEEE 802.11 standard proved insecure. WPA is the implementation of the IEEE 802.11i DRAFT standard and WPA2 is the implementation of the IEEE 802.11i-2004 standard. WPA2 introduced Counter Mode Cipher Block Chaining Message Authentication Code Protocol (CCMP), a new Advanced Encryption Standard (AES) based encryption mode with strong security. WPA2 became mandatory for all new devices since March 2006.

7.2.3 WPA Supplicant

wpa_supplicant is the WPA Supplicant implementation for GNU/Linux. It implements key negotiation with a WPA Authenticator and it controls the roaming and IEEE 802.11 authentication/association of the wlan driver.

wpa_supplicant is designed to be a daemon that runs in the background and acts as the backend component controlling the wireless connection. It supports separate frontend programs and a text-based frontend (*wpa_cli*) and a GUI (*wpa_gui*) are included with *wpa_supplicant*.

To install the *wpa_gui* utility the *wpagui* needs to be installed.

```
$ sudo apt-get install wpagui
```

With a wpa2 key added to the AP rerun the scan on the BSS *fta_ssid*. Note that the security information shows that a Pre Shared Key (PSK) is set on the AP. The *CCMP ciphers* show that WPA2 is used for security.

```
$ sudo iw dev eth1 scan ssid fta_ssid
[sudo] password for alove lace:
BSS d4:ca:6d:61:dd:8d (on eth1)
  TSF: 0 usec (0d, 00:00:00)
  freq: 2412
  beacon interval: 100
  capability: ESS Privacy ShortPreamble ShortSlotTime (0x0431)
  signal: -22.00 dBm
  last seen: 0 ms ago
  Information elements from Probe Response frame:
  SSID: fta_ssid
  Supported rates: 1.0* 2.0 5.5 11.0
  DS Parameter set: channel 1
  RSN:   * Version: 1
        * Group cipher: CCMP
        * Pairwise ciphers: CCMP
        * Authentication suites: PSK
        * Capabilities: (0x0000)
```

Create a *wpa_supplicant.conf* file with the key on the AP. Use the *wpa_passphrase* utility to generate a WPA PSK from an ASCII passphrase for a SSID.

```
$ wpa_passphrase fta_ssid
# reading passphrase from stdin
mywpakey
network={
    ssid="fta_ssid"
    #psk="mywpakey"
    psk=f8af67891dd4b156041e1757b68dec57bfbcb28035486f3234aacabaae62ad6b6
}
```

```
$ sudo vi wpa_supplicant.conf
```

```
network={
    ssid="fta_ssid"
    #psk="mywpakey"
    psk=f8af67891dd4b156041e1757b68dec57bfbcb28035486f3234aacabaae62ad6b6
}
```

To run the *wpa_supplicant* the following option switches apply.

Option	Description
-B	Run daemon in the background
-c	Path to configuration file
-D	Driver to use
-d	Debugging verbosity (-dd even more)
-i	Interface to listen on, i.e. eth1

Here is a selection of drivers that are associated with the *-D* option switch. Whether to use one of these or another can usually be determined by first trying the *generic* drivers, and should they fail use *dmesg* to determine the driver that the system has used for the interface.

Driver	Description
nl80211	GNU/Linux Netlink nl80211/cfg80211 (generic)
wext	GNU/Linux wireless extensions (generic)
hostap	Host AP driver (Intersil Prism2/2.5/3)
hermes	Agere Systems Inc. driver (Hermes-I/Hermes-II)
madwifi	MADWIFI 802.11 support (Atheros, etc.)
broadcom	Broadcom wl.o driver
wired	wpa_supplicant wired Ethernet driver
roboswitch	wpa_supplicant Broadcom switch driver
bsd	BSD 802.11 support (Atheros, etc.)
ndis	Windows NDIS driver

nl80211 is the new IEEE 802.11 *netlink* interface public header. Netlink is a socket family used for the transfer of networking Inter Process Communication (IPC) between the kernel and user space processes. For example *iproute2* and *iw* use netlink to communicate with the kernel. Other older drivers are based on Input/Output ConTroL (*ioctl*) system calls. *Netlink* is designed to be more flexible than *ioctl*. *cfg80211* is the GNU/Linux IEEE 802.11 configuration API. *nl80211* is used to configure a *cfg80211* device.

```
$ sudo wpa_supplicant -B -D nl80211 -c /etc/wpa_supplicant.conf -i eth1
```

Confirm that the link is connected.

```
$ iw dev eth1 link
Connected to d4:ca:6d:70:90:d7 (on eth1)
SSID: fta_ssid
freq: 2412
```

Now that the connection is made an IP address needs to be applied to the newly created interface. This can be done by using the *ip addr* command for a static address or the *dhclient* command to get an IP address from a DHCP Server.

```
$ sudo dhclient eth1
```

Confirm the address has been established on the interface.

\$ ip addr list

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST> mtu 1500 qdisc pfifo_fast state DOWN qlen 1000
    link/ether 28:d2:44:19:83:95 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 1c:3e:84:ed:99:0b brd ff:ff:ff:ff:ff:ff
    inet 78.143.141.175/24 brd 192.168.25.255 scope global eth1
        valid_lft forever preferred_lft forever
    inet6 fe80::1e3e:84ff:feed:990b/64 scope link
        valid_lft forever preferred_lft forever
```

This page is intentionally blank

8. Virtual Private Networks (VPN)

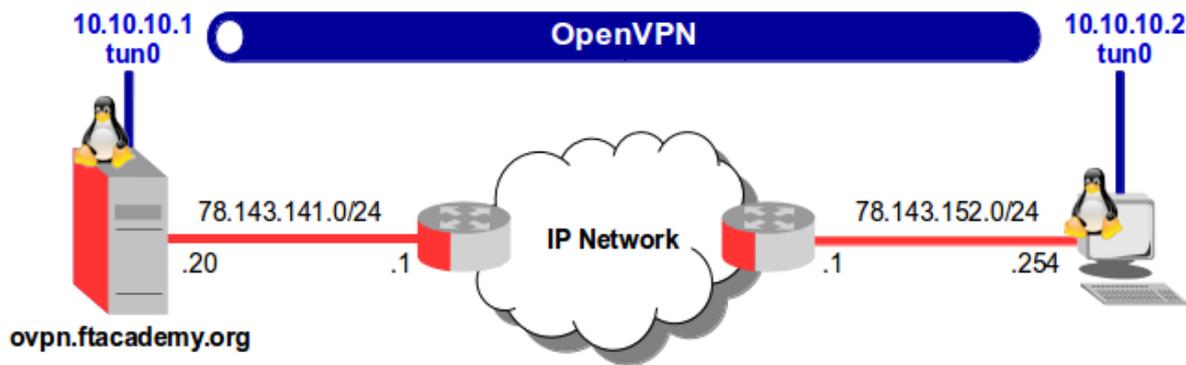
A VPN is a network that uses Internet to transport data, but stops any external members from accessing that data.

This means that a virtual network with connected VPN nodes tunnelled through another network, through which the traffic passes and with which no one outside the VPN can interact. It is used when remote users wish to access a corporate network to maintain the security and privacy of the data. Various methods can be used to configure a VPN, such as SSH, Secure Sockets Layer (SSL)/Transport Layer Security (TLS)/, Crypto IP Encapsulation (CIPE), IP Security (IPSec), Point to Point Tunnelling Protocol (PPTP) and Layer 2 Tunnelling Protocol (L2TP).

In order to perform the configuration tests in this section, OpenVPN, will be used. OpenVPN is a solution based on SSL VPN and can be used for a wide range of solutions, for example, remote access, VPN point to point, secure WiFi networks or distributed corporate networks. OpenVPN implements OSI layer 2 or 3 using SSL/TLS protocols and supports authentication based on certificates, smart cards and other confirmation methods. OpenVPN is not a proxy applications server and does not operate through a web browser.

In order to analyse it, there is an option in OpenVPN called OpenVPN for Static key configurations, which provides a simple method for configuring a VPN that is ideal for tests or point-to-point connections. The advantages are the simplicity and the fact that it is not necessary to have a X509 public key infrastructure (PKI) certificate to maintain the VPN. The disadvantages are that it only permits one client and one server, as, because the public key and private key are not used, there may be the same keys as in previous sessions and there must be a text-mode key in each peer and the secret key must be previously exchanged for a secure channel.

8.1 IPv4 OpenVPN tunnel



For demonstration build a simple Point to Point (P2P) link from one end of the network to the other. Install the sever and create a working directory in the user home directory (for more permanent installations there is a system OpenVPN directory `/etc/openvpn` that can is used. Create a configuration file setting the device `dev` to be created as a tunnel (`tun`). set-up an IP scheme for the tunnel that will be created with the address `10.10.10.1` the server side of the tunnel and `10.10.10.2` the client side. Finally a shared secret key file is identified. This is not created yet, it will be created on the client and shared with the server.

8.1.1 Server set-up

```
$ sudo apt-get install openvpn
$ mkdir ~/openvpn

$ vi ~/openvpn/server.ovpn

dev tun
ifconfig 10.10.10.1 10.10.10.2
secret static.key
```

On the client edit the `/etc/hosts` file with a name for the central OpenVPN server if it is not named via DNS. Again make a directory in the users home directory `openvpn` to house the configuration and key files. Generate a key file `static.key` locally and using a secure means (in this case sftp) copy it to the corresponding directory on the server.

8.1.2 Client set-up

```
$ sudo apt-get install openvpn

$ sudo -s
# cat << OVPNIP >> /etc/hosts
> # Added for OpenVPN - `date` #
> 78.143.141.20
> OVPNIP

# exit
$ mkdir ~/openvpn
$ cd ~/openvpn
$ openvpn --genkey --secret static.key
$ cat ~/openvpn/static.key | ssh debianuser@ovpn.ftacademy.org 'cat >>
~/openvpn/static.key'
```

```
The authenticity of host 'ovpn.ftacademy.org (78.143.141.20)' can't be
established.
ECDSA key fingerprint is 63:b3:ea:33:ea:8a:34:ee:5b:43:39:94:04:62:9c:bc.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ovpn.ftacademy.org,78.143.141.20' (ECDSA) to
the list of known hosts.
debianuser@ovpn.ftacademy.org's password: myPassword
```

Create a configuration file that points to the remote OpenVPN server, in the rest of the file mirror what is configured in the client but swap the IP addresses on the *ifconfig* line.

```
$ vi ~/openvpn/ovpn.ftacademy.org.ovpn

remote ovpn.ftacademy.org
dev tun
ifconfig 10.10.10.2 10.10.10.1
secret static.key
```

8.1.3 Run the OpenVPN Server

Change the permissions on the two newly created files and then run the *openvpn* server using the configuration file created.

```
$ chmod 600 ~/openvpn/server.ovpn
$ chmod 400 ~/openvpn/static.key

$ sudo openvpn ~/openvpn/server.ovpn
Sat May 10 11:08:21 2014 OpenVPN 2.3.2 i686-pc-linux-gnu [SSL (OpenSSL)] [LZO] [EPOLL]
[PKCS11] [eurephia] [MH] [IPv6] built on Jul 12 2013
Sat May 10 11:08:21 2014 TUN/TAP device tun0 opened
Sat May 10 11:08:21 2014 do_ifconfig, tt->ipv6=0, tt->did_ifconfig_ipv6_setup=0
Sat May 10 11:08:21 2014 /sbin/ip link set dev tun0 up mtu 1500
Sat May 10 11:08:21 2014 /sbin/ip addr add dev tun0 local 10.10.10.1 peer 10.10.10.2
Sat May 10 11:08:21 2014 UDPv4 link local (bound): [undef]
Sat May 10 11:08:21 2014 UDPv4 link remote: [undef]
Sat May 10 11:09:20 2014 Peer Connection Initiated with [AF_INET]78.143.152.254:1194
Sat May 10 11:09:21 2014 Initialization Sequence Completed
```

Confirm the *tun0* interface formed.

```
$ ip addr list tun0
6: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN
qlen 100
    link/none
    inet 10.10.10.1 peer 10.10.10.2/32 scope global tun0
        valid_lft forever preferred_lft forever
```

8.1.4 Connect with the OpenVPN client

Change the permissions of the files on the client also, then run *openvpn* with the configuration file created.

```
$ chmod 600 ~/openvpn/ovpn.ftacademy.org.ovpn
$ chmod 400 ~/openvpn/static.key

$ sudo openvpn ~/openvpn/ovpn.ftacademy.org.ovpn
Sat May 10 11:09:20 2014 OpenVPN 2.3.2 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [EPOLL]
[PKCS11] [eurephia] [MH] [IPv6] built on Jul 12 2013
Sat May 10 11:09:20 2014 TUN/TAP device tun0 opened
Sat May 10 11:09:20 2014 do_ifconfig, tt->ipv6=0, tt->did_ifconfig_ipv6_setup=0
Sat May 10 11:09:20 2014 /sbin/ip link set dev tun0 up mtu 1500
Sat May 10 11:09:20 2014 /sbin/ip addr add dev tun0 local 10.10.10.2 peer 10.10.10.1
Sat May 10 11:09:20 2014 UDPv4 link local (bound): [undef]
Sat May 10 11:09:20 2014 UDPv4 link remote: [AF_INET]78.143.141.20:1194
Sat May 10 11:09:21 2014 Peer Connection Initiated with [AF_INET]78.143.141.20:1194
Sat May 10 11:09:22 2014 Initialization Sequence Completed
```

An IP tunnel is created, confirm the IP address on the new *tun0* interface and a *traceroute* to the other end of the tunnel determines that the traffic appears to get to the far side in a single hop despite the fact it traverses multiple routers. Obviously it still does but this *traceroute* is encapsulated within a tunnel that itself is transported over the multiple routers.

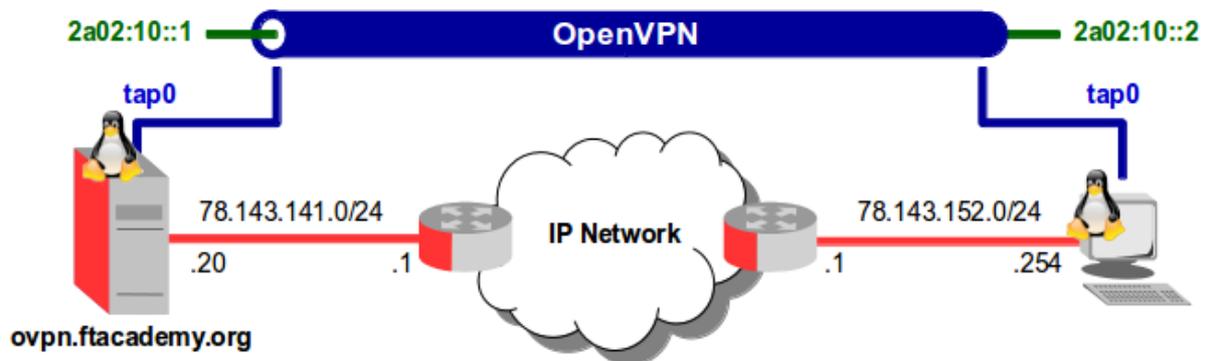
```
$ ip addr list dev tun0
4: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UNKNOWN qlen 100
    link/none
    inet 10.10.10.2 peer 10.10.10.1/32 scope global tun0
        valid_lft forever preferred_lft forever
```

```
$ traceroute 10.10.10.1
traceroute to 10.10.10.1 (10.10.10.1), 30 hops max, 60 byte packets
 1 10.10.10.1 (10.10.10.1)  1.975 ms  1.995 ms  2.003 ms
```

Reviewing an OpenVPN packet captured on the *eth0* interface below. Note that the traffic within the tunnel is hidden by OpenVPN SSL Obviously packets captured on the *tun0* interface are the actual traffic unencrypted as *tun0* is simply a separate networking interface from GNU/Linux perspective.

```
Frame : 166 bytes on wire (1328 bits) on interface eth0
Ethernet II, Src: 28:d2:44:19:83:95, Dst: d4:ca:6d:61:dd:88
Internet Protocol Version 4, Src: 78.143.152.254, Dst: 78.143.141.20
User Datagram Protocol, Src Port: openvpn (1194), Dst Port: openvpn
(1194)
OpenVPN Protocol, Opcode: 0x19, Key ID: 3
  Type: 0xcb [opcode/key_id]
    1100 1... = Opcode: 0x19
    .... .011 = Key ID: 3
  Session ID: 7496750292753433419
  HMAC: 2d886382a755288784060aced497ae9e34de64fd
  Packet-ID: 2635760
  Net Time: Apr  1, 2060 01:53:55.000000000 IST
  Message Packet-ID Array Length: 225
  Packet-ID Array
```

8.2 IPv6 OpenVPN tunnel



An alternative to creating an IP tunnel is to encapsulate the Ethernet frames at layer 2, a layer 2 tunnel.

tun devices receive raw IP packets and give them to a user space program, OpenVPN the packets are encrypted and sends to the other end of the tunnel where they get decrypted and are presented to the *tun* device on that side.

Using layer 2 *tap*, Ethernet frames are encrypted instead of IP packets. A *tap* appears as an Ethernet interface. In the following example a layer 2 VPN will be created end to end and IPv6 addresses applied at each end.

8.2.1 OpenVPN Server - tap

Leaving the infrastructure in place from the last set-up edit the server `.ovpn` files as follows. The device is now *tap* and a script is called which will apply the IPv6 addresses to the new interface. A setting of *script-security* of at least level 2 is required.

```
$ vi ~/openvpn/server.ovpn

dev tap
secret static.key
script-security 3
up /home/debianuser/openvpn/ipv6addr.sh
```

Create the `ipv6addr.sh` and make it executable.

```
$ vi ~/openvpn/ipv6addr.sh
#!/bin/bash

ip link set dev $dev up
ip -6 addr add 2a02:10::1/64 dev $dev
```

Run the server.

```
$ sudo openvpn ~/OpenVPN/server.ovpn
Sat May 10 13:00:47 2014 OpenVPN 2.3.2 i686-pc-linux-gnu [SSL (OpenSSL)] [LZO] [EPOLL]
[PKCS11] [eurephia] [MH] [IPv6] built on Jul 12 2013
Sat May 10 13:00:47 2014 NOTE: the current --script-security setting may allow this
configuration to call user-defined scripts
Sat May 10 13:00:47 2014 TUN/TAP device tap0 opened
Sat May 10 13:00:47 2014 /home/petrauser/OpenVPN/ipv6addr.sh tap0 1500 1576  init
Sat May 10 13:00:47 2014 UDPv4 link local (bound): [undef]
Sat May 10 13:00:47 2014 UDPv4 link remote: [undef]
Sat May 10 13:01:03 2014 Peer Connection Initiated with [AF_INET]78.143.152.254:1194
Sat May 10 13:01:04 2014 Initialization Sequence Completed
```

Review the *tap* interface and the assigned IPv6 address.

```
$ ip link show dev tap0
15: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UNKNOWN mode DEFAULT qlen 100
    link/ether a6:5e:47:c6:01:55 brd ff:ff:ff:ff:ff:ff

$ ip addr list dev tap0
15: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UNKNOWN qlen 100
    link/ether a6:5e:47:c6:01:55 brd ff:ff:ff:ff:ff:ff
    inet6 2a02:10::1/64 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::a45e:47ff:fec6:155/64 scope link
        valid_lft forever preferred_lft forever
```

8.2.2 OpenVPN Client - tap

Similar to the configuration on the server except the *remote* pointer to the server.

```
$ vi ~/openvpn/server.ovpn

remote ovpn.ftacademy.org
dev tap
secret static.key
script-security 3
up /home/alovelace/OpenVPN/ipv6addr.sh
```

Again like on the server create the *ipv6addr.sh* and make it executable. The only difference apart from the IPv6 address is the addition of a default route command via the IPv6 address on the server side of the tunnel.

```
$ vi ~/openvpn/ipv6addr.sh
#!/bin/bash

ip link set dev $dev up
ip -6 addr add 2a02:10::2/64 dev $dev
ip -6 route add default via 2a02:10::1
```

Connect to the server.

```
$ sudo openvpn ovpn.ftacademy.org.ovpn
[sudo] password for alove lace:
Sat May 10 13:44:21 2014 OpenVPN 2.3.2 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO]
[EPOLL] [PKCS11] [eurephia] [MH] [IPv6] built on Jul 12 2013
Sat May 10 13:44:21 2014 NOTE: the current --script-security setting may allow
this configuration to call user-defined scripts
Sat May 10 13:44:21 2014 TUN/TAP device tap0 opened
Sat May 10 13:44:21 2014 /home/alovelace/OpenVPN/ipv6addr.sh tap0 1500 1576  init
Sat May 10 13:44:21 2014 UDPv4 link local (bound): [undef]
Sat May 10 13:44:21 2014 UDPv4 link remote: [AF_INET]78.143.141.20:1194
Sat May 10 13:44:31 2014 Peer Connection Initiated with
[AF_INET]78.143.141.20:1194
Sat May 10 13:44:32 2014 Initialization Sequence Completed
```

Review the *tap* interface and the assigned IPv6 address.

```
$ ip link show dev tap0
15: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UNKNOWN mode DEFAULT qlen 100
    link/ether 72:2d:57:65:b3:9c brd ff:ff:ff:ff:ff:ff

$ ip addr list dev tap0
15: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UNKNOWN qlen 100
    link/ether 72:2d:57:65:b3:9c brd ff:ff:ff:ff:ff:ff
    inet6 2a02:10::2/64 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::702d:57ff:fe65:b39c/64 scope link
        valid_lft forever preferred_lft forever
```

Ping and traceroute the server from the client, then review the *ip -6 route list*

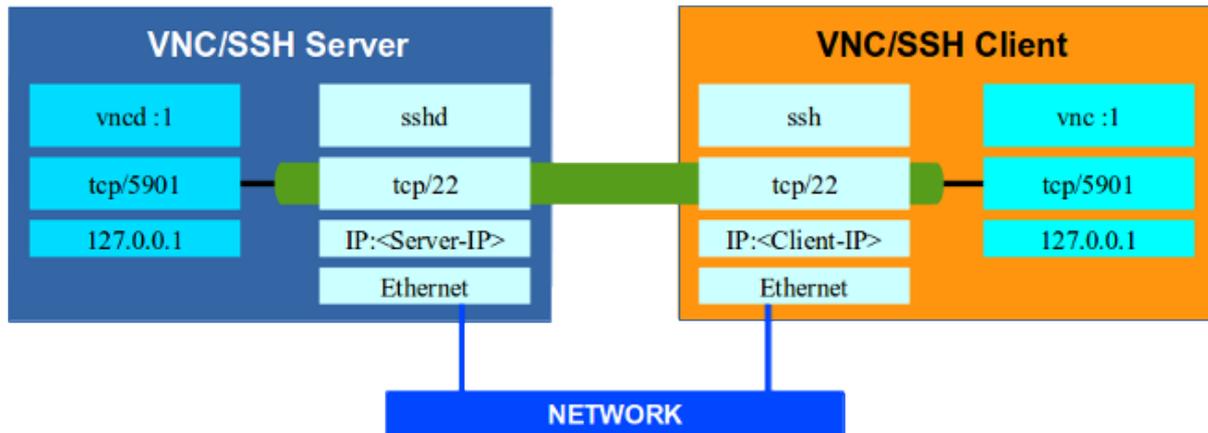
```
$ ping6 2a02:10::1
PING 2a02:10::1(2a02:10::1) 56 data bytes
64 bytes from 2a02:10::1: icmp_seq=1 ttl=64 time=2.08 ms
64 bytes from 2a02:10::1: icmp_seq=2 ttl=64 time=1.26 ms
64 bytes from 2a02:10::1: icmp_seq=3 ttl=64 time=1.24 ms

$ traceroute6 2a02:10::1
traceroute to 2a02:10::1 (2a02:10::1) from 2a02:10::2, 30 hops max, 24
byte packets
 1 2a02:10::1 (2a02:10::1)  1.416 ms  1.462 ms  0.984 ms

$ ip -6 route list
2a02:10::/64 dev tap0 proto kernel metric 256
fe80::/64 dev eth0 proto kernel metric 256
fe80::/64 dev tap0 proto kernel metric 256
default via fe80::d6ca:6dff:fe61:dd88 dev eth0 proto ra metric 1024
expires 1724sec
```

8.3 SSH VPN

SSH can be used for VPN tunnelling. SSH is not the most efficient tunnelling mechanism and is not suitable for high throughput applications. Looking at an example where a remote server is running X.org with a Desktop. To access the Desktop the Virtual Network Computing (VNC) graphical desktop sharing system is used, however over the public Internet this is a serious security risk. SSH tunnelling can be used to secure the underlying network layer.



8.3.1 Set-up VNC Server ran as localhost only

On the server install a Desktop environment and VNC.

```
$ sudo apt-get install gnome-desktop-environment
$ sudo apt-get install tightvncserver
```

Server Start the VNC Server with the option switch *-localhost*. This prevents remote VNC Viewers accessing the VNC Server.

```
$ tightvncserver -geometry 800x600 :1
```

You will require a password to access your desktops.

```
Password: <Password>
Verify: <Password>
```

```
Would you like to enter a view-only password (y/n)? n
```

Stop the TightVNC Server, so it can be configured:

```
$ tightvncserver -kill :1
```

The VNC `xstartup` file needs editing. This change sets the TightVNC Server to render the full Gnome3/Unity desktop. Copy the `xstartup` file to `xstartup.orig` and edit the `xstartup` file to replace its content as follows:

```
$ cp ~/.vnc/xstartup ~/.vnc/xstartup.orig
$ vi ~/.vnc/xstartup

#!/bin/sh
xrdb $HOME/.Xresources
xsetroot -solid grey
x-terminal-emulator -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
gnome-session &
```

Start the TightVNC Server again now that the changes are made with the desired resolution. The `-localhost` option switch prevents remote VNC Viewers accessing the VNC Server.

```
$ tightvncserver -geometry 1024x768 -localhost :1
```

8.4 VNC on the client side

Install TightVNC viewer.

```
$ sudo apt-get install xtightvncviewer
```

8.5 SSH connection and VNC connection

On the client host open an SSH connection to the server but with some special option switches.

Option	Description
-C	Enables compression of data between client and server.
-L 5901:localhost:5901	Accepts connections from the localhost on port 5901 (this equates to the local display : 1) and tunnels it to the SSH client side such that it is available locally on the client machine on its port 5901.
-v	Use when establishing the first time as it gives verbose information on connection.

```
$ ssh -C -L 5901:localhost:5901 avelace@75.144.153.182
```

On the client run TightVNC viewer to access the *local* VNC Server on port 5901 (This is in fact the tunnelled display).

```
$ xtightvncviewer -encodings "copyrect hextile" localhost:5901
```

VNC uses 'raw' pixel encoding by default as it gives better performance for local access. However in this case it is not really a local connection but a local connection redirected across an SSH tunnel.

- *-encodings "copyrect hextile"* option switch is more effective over the network, Copy Rectangle (*copyrect*) encoding is efficient when something is being moved; the only data sent is the location of a rectangle from which data should be copied to the current location. *copyrect* is also be used to efficiently transmit a repeated pattern. *hextile* encoding splits rectangles up in to 16x16 tiles, which are sent in a predetermined order. *hextile* encoding is usually the best choice for using in high-speed network environments (e.g. Ethernet local-area networks).

- The graphic shows a remote Ubuntu server being rendered in TightVNC viewer on a Linux Mint Desktop. This is performed over an SSH tunnel between the computers.



9. IP Telephony

IP Telephony (IPT) or more often called Voice over Internet Protocol (VoIP), Internet Telephony, Broadband Telephony, Broadband Phone and Voice over Broadband is the routing of voice conversations over the Internet or through any other IP based network. It is the conduct of a telephone call where the signal is carried in Internet Protocol (IP) packets, instead of over dedicated voice transmission lines. This allows the elimination of circuit switching and a reduction in bandwidth used. Instead, packet switching is used, where IP packets with voice data are sent over the network only on demand, i.e. when a caller is actually talking.

The advantages of IPT over traditional telephony include;

- Lower costs per call, especially for long-distance calls.
- Lower infrastructure costs.
- Integration with existing IT infrastructure.

Like traditional telephony, a telephone call consists of a signalling part and a bearer part. In IPT there have been a number of attempts to deal with signalling, the initial attempt was to use the International Telecommunication Union (ITU) Telecommunication Standardisation Sector (ITU-T) H.323 recommendation for call and control of audio-visual multimedia. It is based on the ITU-T Q.931 signalling used in Integrated Signalling Digital Network (ISDN). However H.323 and its family of protocols while fine for use on LANs prove difficult for signalling over WANs. A simpler text based signalling system called Session Initiation Protocol (SIP) sharing many similarities with Hypertext Transfer Protocol (HTTP) has proven to be a very successful signalling protocol over both LANs or WANs. HTTP being the IETF protocol for non real-time traffic and SIP for handling signalling for real-time traffic. SIP was proposed in 1999 as RFC 2543 with a rewrite in 2002 as RFC 3261. There are various associated supporting RFCs dealing with various SIP specifics.

SIP is an application-layer control protocol that can establish, modify, and terminate multimedia sessions such as IPT. SIP can also invite participants to already existing sessions, such as multicast conferences. Both H.323 and SIP only establish the session. Once the interactive audio and video session is established the end-to-end delivery of the real-time bearer data services is delivered by Real-time Transport Protocol (RTP). RTP was proposed as RFC 1889 in 1996 and superseded by RFC 3550 in 2003.

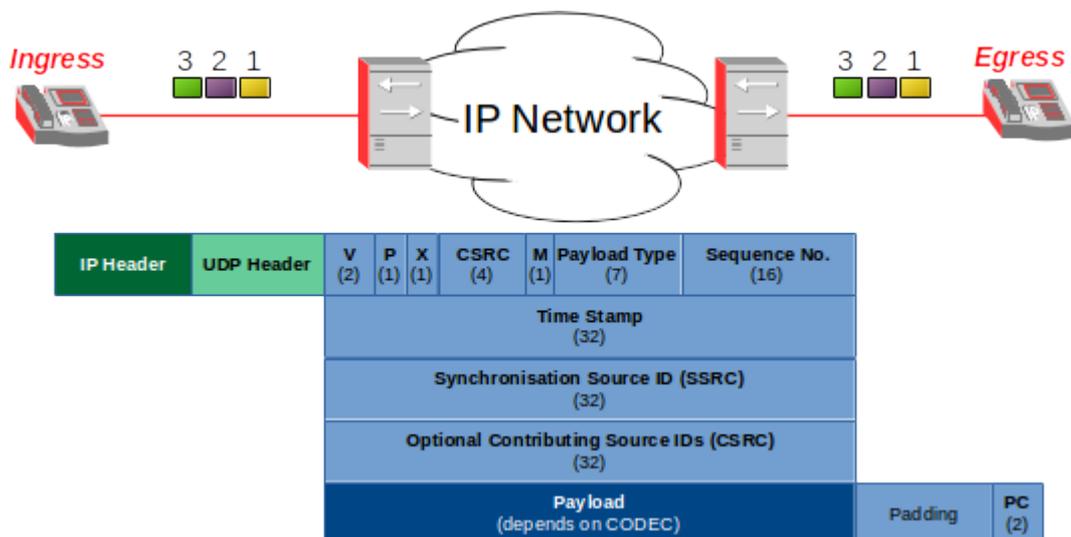
9.1 Audio Streams

Audio streams are carried using the RTP. At each end of the RTP stream Coder/DECoder (CODEC) compress and code sampled voice into data streams which are uncompressed/decoded at the receiving end. End to end audio streams are impacted by a number of conditions. These conditions must be minimised in network design to ensure voice quality.

9.2 Real-Time Transport Protocol

Real-Time Transport Protocol (RTP) developed by the Audio-Video Transport Working Group of the IETF and published on 1996 as RFC 1889. RTP was also published by the ITU-T as H.225.0. It exists as an Internet Draft Standard defined in RFC 3550 that specifies the protocol, with RFC3551 defining a specific profile for Audio and Video Conferences with Minimal Control.

RTP defines a standardised packet format for delivering audio and video over the Internet. It was originally designed as a multicast protocol, but has since been applied in many unicast applications. It is frequently used in streaming media systems either in conjunction with Real Time Streaming Protocol (RTSP), which establishes and controls single, or several time synchronised streams of continuous media such as audio and video or in video-conferencing and Softswitch systems in conjunction with SIP.



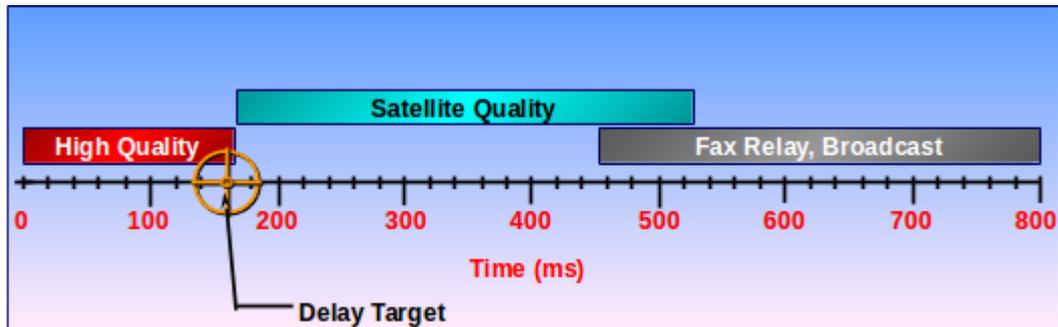
Code	Size	Meaning	Notes
V	2 bits	Version number	Currently 2
P	1 bit	Padding	1 = Padding to make packet a fixed length
X	1 bit	Extension	1 = Additional 2 octet extension
CC	4 bits	CSRC Count	Number of Content Source Identifiers
M	1 bit	Marker	Indicates start of new video frame or audio talk spurt
PT	7 bits	Payload type	Defines codec in use. Matches SDP profile
SN	16 bits	Sequence No.	Incremented for each RTP packet sent
T	32 bits	Timestamp	Indicates relative timing of payload sample
SSRCI	32 bits	Sync Source ID	Unique number to identify a participant
CSRC	32 bits	Contrib. Source ID	Up to 15 (defined by CC) additional participants. Only used if multiple streams are mixed by a conference bridge for instance

Here is a packet example from a IP Telephone call.

```

Frame 10070
Ethernet II, Src: 28:d2:44:19:83:95, Dst: 00:0e:08:d2:e8:2b
Internet Protocol Version 4, Src: 192.168.1.11, Dst: 192.168.1.12
User Datagram Protocol, Src Port: 22156, Dst Port: 16482
Real-Time Transport Protocol
  10.. .... = Version: RFC 1889 Version (2)
  ..0. .... = Padding: False
  ...0 .... = Extension: False
  ... 0000 = Contributing source identifiers count: 0
  0... .... = Marker: False
  Payload type: GSM 06.10 (3)
  Sequence number: 20871
  Timestamp: 1567201601
  Synchronization Source identifier: 0xf67ed7d3 (4135507923)
  Payload: ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff...
```

9.3 Delay

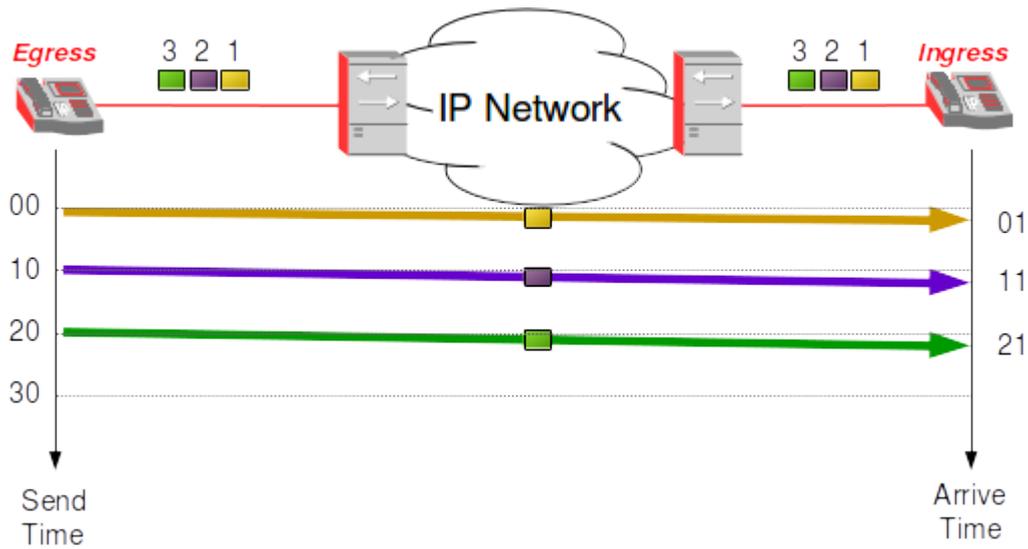


ITU's G.114 Recommendation = 0 – 150 ms 1 - way delay

Delay is the time required for a signal to traverse the network. Excessive delay can cause pauses in conversation and even causes the listener to misunderstand due to language inflection and tones. Pauses in the conversation can carry so much of the meaning in themselves as non-verbal speech. According to the ITU-T recommendation G.114, 150 ms is the maximum allowable one-way delay. In reality in most IPT networks a delay of up to 250 ms can be tolerated before the call becomes degraded to be noticeable. The 100 ms difference in these two values is simply a safety net which prevents unperceived congestion problems from reducing service quality or even bringing it to a halt. Normally the delay would be composed of a number of parts:

- Network delay
- CODEC latency
- Jitter
- Packet loss

For the purpose of the example consider that each packet holds 10 ms of audio and the transfer delay across the network is constant and equal 1 ms.



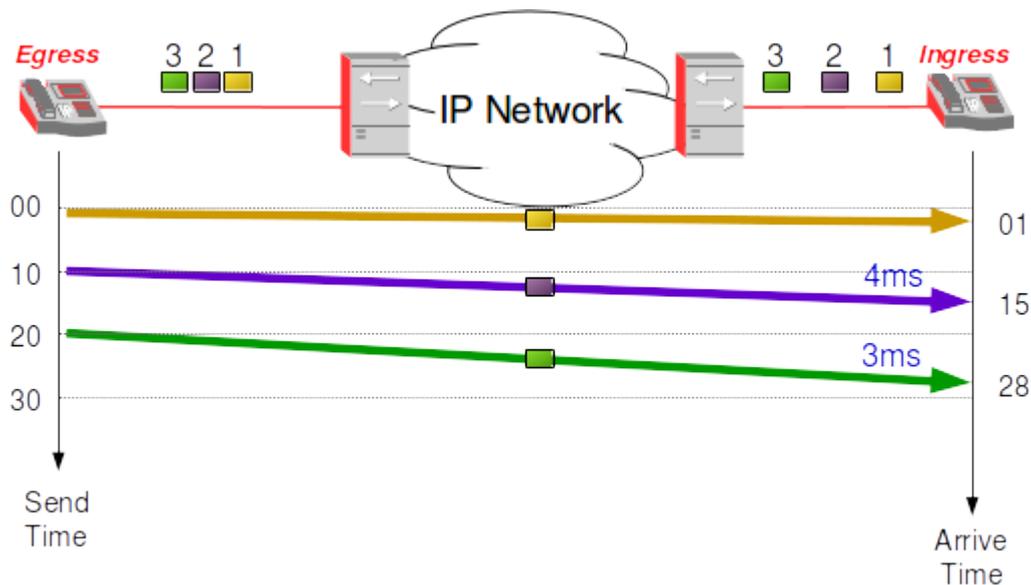
9.3.1 Network Delay

This is the delay incurred by a packet traversing the IP Network. To minimise this delay, minimise the number of routers that need to be traversed. Alternatively, specify a higher priority for voice traffic using prioritisation schemes such as Weighted Fair Queuing (WFQ) or flow control mechanisms such as MPLS (Multi-protocol Label Switching).

9.3.2 CODEC Latency

Each compression algorithm has certain built in delays. G.723 for example adds a fixed delay of 30 ms.

9.3.3 Jitter



Jitter buffering is the effect of network delay on packet arrival at the Ingress. Packets transmitted at equal intervals from the Egress arrive at irregular intervals at the Ingress. Excessive Jitter makes voice choppy and difficult to understand. Egress and Ingress Packet intervals should be nearly equal with High Quality Voice. Jitter buffers can be used to smooth out network fluctuations.

Jitter is used to overcome packet delays experienced in IP Networks by controlling the regularity in which voice packets arrive at the receiving end. In an IP Network there is no guarantee that the packets sent will arrive at equal intervals or that they will arrive in sequence for that matter. A jitter buffer at the receiving end can re-sort the packets and present the packets at equal intervals to the decompression algorithm based on the time-stamp within the RTP header. The jitter buffer must strike a balance between delay and quality. If the buffer is too small latency will be reduced but voice quality will be degraded causing audible effects in the received voice. If its too big voice quality will be improved but latency will increase, having the effect of turning a two-way conversation into a half-duplex one.

Jitter Buffer Depth

The jitter buffer is configurable and can be optimised for given network conditions to compensate for fluctuations in those conditions. However, the downside is that it can add significant delay.

$$D = ED + \beta EV$$

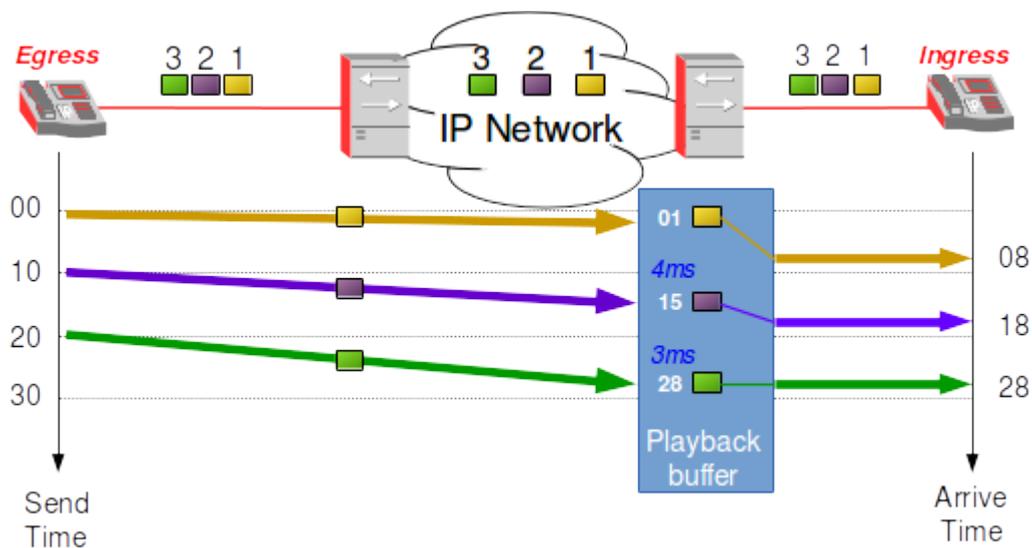
where:

D – Play-out delay (set for each talk spurt)

ED – Estimated average packet delay

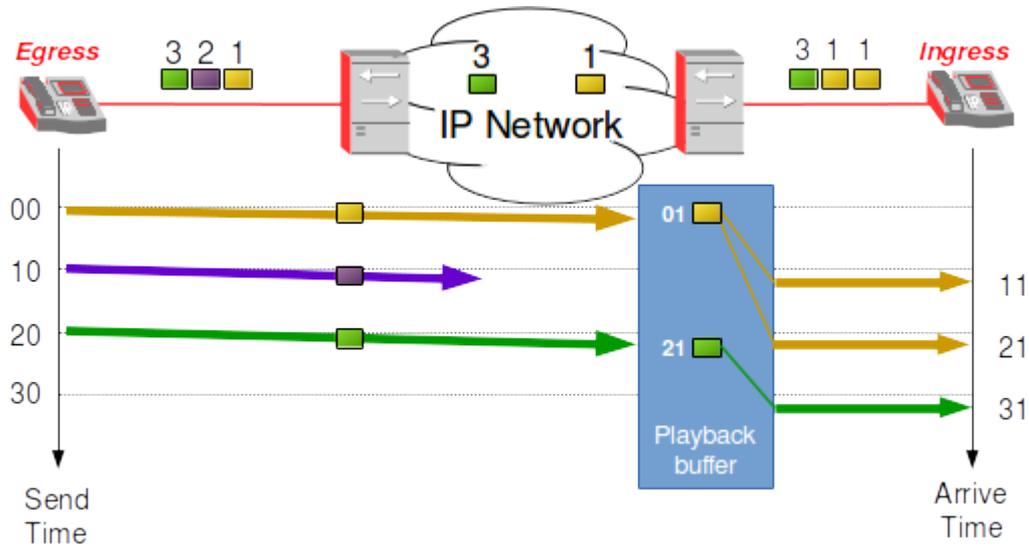
EV – Estimated average packet delay variation

β – safety factor (usually $\beta = 4$)



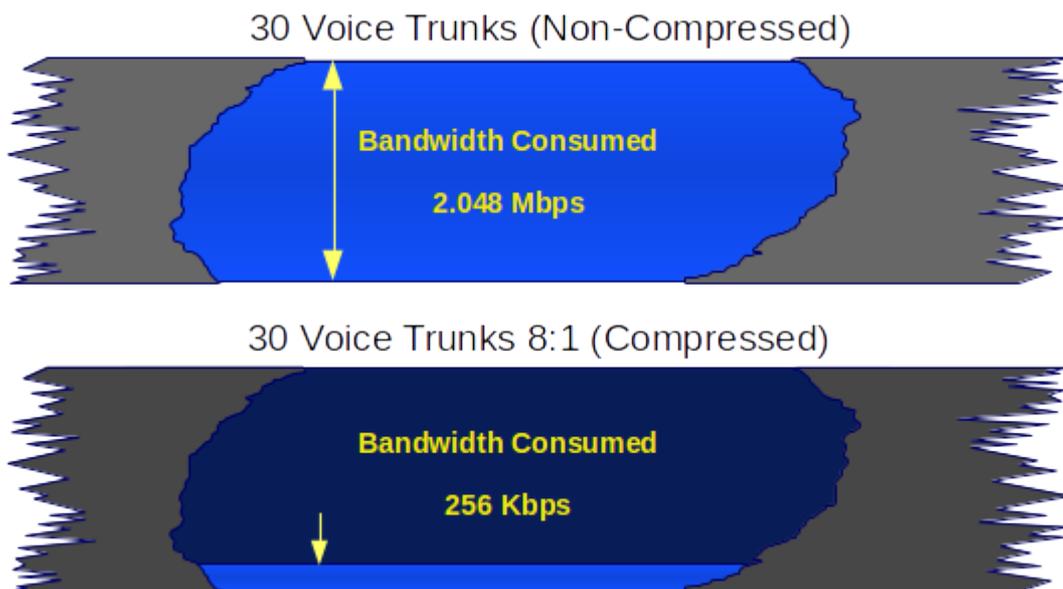
The diagram shows for the three packets how the effect of Jitter is compensated. The buffer delays sending the packets by a time greater than the sum of the delays. In this case the delays are 4 and 3 ms respectively, so transmitting after a delay of 8 ms. The receiver hears the audio at the same speed as it is created, therefore the constant transfer delay is immaterial.

9.3.4 Packet Loss



Packet loss is a normal phenomenon on packet networks. Loss can be caused by many different reasons: overloaded links, excessive collisions on a LAN, physical media errors and others. Transport layers such as TCP account for loss and allow packet recovery under reasonable loss conditions. CODEC's can also accommodate a certain amount of packet lost but when it becomes excessive then the clarity of the voice call becomes degraded. Typical mitigation for packet loss involves replaying the previous packet in place of the lost packet.

9.3.5 Voice Compression



Voice compression is an end-to-end technology which reduces the required bandwidth to carry a voice signal when compared to the standard 64 Kb/s channel required for uncompressed voice (G.711). Whatever is compressed, must be decompressed at the other end using the identical technology. The major disadvantage of voice compression is that the process itself adds delay and the more compression, the more delay is induced. Voice compression also reduces voice quality.

9.4 CODEC

A CODEC is a device or program capable of performing transformations on a data stream or signal. CODEC's can both put the stream or signal into an encoded form (often for transmission, storage or encryption) and retrieve, or decode that form for viewing or manipulation in a format more appropriate for these operations.

The choice of CODEC has an affect on voice quality when compression is utilised. Compression in a CODEC has the result of removing redundant or less important information from the receiving waveform in an effort to reduce bandwidth requirements for transmission.

The table below lists a number of popular voice CODEC's:

Standard	MOS	End to end delay	Bit rate (kb/s)	Compression algorithm	Voice quality
G.711	4.4	<< 1 ms	64	PCM	PSTN standard
G.728	4.2	<< 2 ms	16	LD-CELP	Good
G.726	4.2	1 ms	16, 24, 32, 40	ADPCM	Good 40, fair 24
G.723.1	3.5	67 - 97 ms	5.3, 6.4	ACELP	Good 6.4, fair 5.3
G.729A	4.2	25 - 30 ms	8	CS-ACELP	Good

CODEC Latency

Each compression algorithm has certain built in delays. G.723 for example adds a fixed delay of 30 ms.

9.4.1 RTP Audio & Video Payloads

RFC 3551 describes the RTP Profiles for Audio and Video Conferences with Minimal Control used in RTP version 2.

PT	Name	Clock rate (Hz)	Description
0	PCMU	8,000	ITU G.711 PCM u-Law Audio 64 kb/s
1	1016	8,000	CELP Audio 4.8 kb/s
2	G721	8,000	ITU G.721 ADPCM Audio 32 kb/s
3	GSM	8,000	European GSM Audio 13 kb/s
5	DVI4	8,000	DVI ADPCM Audio 32 kb/s
6	DVI4	16,000	DVI ADPCM Audio 64 kb/s
7	LPC	8,000	Experimental LPC Audio
8	PCMA	8,000	ITU G.711 PCM A-Law Audio 64 kb/s
9	G722	8,000	ITU G.722 Audio
10	L16	44,100	Linear 16 bit Audio 705.6 kb/s
11	L16	44,100	Linear 16 bit stereo Audio 1411.2 kb/s
14	MPA	90,000	MPEG-I or MPEG-II Audio only
15	G728	8,000	ITU G.728 Audio 16 kb/s
18	G729	8000	ITU G.729 Audio 8 kb/s
dyn	G726	8000	ITU G.726 Audio 40 kb/s
dyn	G729	8000	ITU G.726 Audio 40 kb/s
25	CELB	90,000	CelB Video
26	JBEG	90,000	JBEG Video
28	NV	90,000	nv Video
31	H261	90,000	ITU H.261 Video
32	MPV	90,000	MPEG-I and MPEG-II Video
33	MP2T	90,000	MPEG-II transport stream Video
34	H263	90,000	MPEG-4 transport stream Video

Note: Entries with Payload Type *dyn* have no static payload type assigned and are only used with a dynamic payload type.

9.5 Other Voice Quality Factors

9.5.1 *Silence Suppression*

Silence suppression takes advantage of prolonged periods of silence in conversations to reduce the number of packets transmitted. In a normal interactive conversation, each speaker typically listens for about half the time, so it is not necessary to transmit packets carrying the speaker's silence. However, it can inadvertently introduce clarity degradation by removing (clipping) parts of the speech utterances.

9.5.2 *Echo*

Echo is caused by the signal reflections of the speaker's voice from the far end telephony equipment back into the speaker's ear. Echo becomes a significant problem when the round trip delay becomes greater than 50 ms, which is always the case when dealing with an IPT network. To compensate for echo on the IPT network, echo cancellers are used.

9.6 Voice Quality Measurements

9.6.1 *P.800 MOS*

With all the factors affecting voice quality, how can one measure it. The ITU addresses this issue through two important recommendations. P.800 MOS deals with defining a method to derive a Mean Opinion Score (MOS) of voice quality. The test involves recording several pre-selected voice samples over the desired transmission media and then playing them back to a mixed group of men and women under controlled conditions. The scores given by this group are then weighed to give a single MOS score ranging between 1 (worst) and 5 (best). A MOS of 4 is considered *toll-quality* voice which is the equivalent of a voice call on the 64 kb/s G.711 channel.

9.6.2 P.861 PSQM

Perceptual Speech Quality Measurement (PSQM) tries to automate this process by defining an algorithm through which a computer can derive scores that have a close correlation to the MOS scores. While PSQM is useful, many people have voiced concerns over the suitability of this recommendation to packetised voice networks. It seems that PSQM was designed for the circuit-switched network and does not take into effect important parameters such as jitter and frame loss that are only relevant to IPT. As a result of PSQM limitations, researchers are trying to come up with alternative objective ways to measure voice quality. One such proposal is the Perceptual Analysis/Measurement System (PAMS) developed by British Telecom (BT). Tests conducted by BT have shown good correlation between automated PAMS scoring and manual MOS results.

9.7 The SIP Protocol and Server Functions

SIP (RFC 3261) as already described is an application-layer signalling or control protocol for the creation, modification, and termination of multimedia sessions with one or more participants. These sessions include Internet telephone calls, multimedia distribution, and multimedia conferences. SIP invitations used to create sessions carry session descriptions using the Session Description Protocol (SDP) defined in RFC 2327 that allow participants to agree on a set of compatible media types.

SIP makes use of elements called Proxy Servers to:

- Help route requests to the user's current location.
- Authenticate and authorise users for services.
- Implement service providers call-routing policies.
- Provide features of interest to users.

SIP also provides a registration function that allows users to upload their current locations for use by Proxy Servers. SIP can run over the top of both TCP and UDP transport protocols, while RTP runs on the UDP transport protocol or both can run over Stream Control Transmission Protocol (SCTP).

While the primary goal of SIP is to initiate a session, a secondary goal is to deliver a description of the session that the user is being invited to attend. SIP conveys information about the protocol used to describe the session. It uses a single request to send all required information, as opposed to the extensive call set-up process used with H.323. SIP and H.323 both use RTP and UDP for transport of real voice traffic.

SIP Addressing is URL like;

- E.164 XXX-XXX-XXXX @ 111.111.111.111.
- user@host or SIP: avelace@ftacademy.org

9.7.1 Session Description Protocol

SDP is a format for describing streaming media initialisation parameters. It has been published by the IETF in RFC 4566. SDP started off as a component of the Session Announcement Protocol (SAP), but found other uses in conjunction with Real Time Streaming Protocol (RTSP), SIP and just as a standalone format for describing multicast sessions.

Many of the SDP fields have no meaning in a SIP environment but remain within the standard for those other uses. SDP was not designed for media negotiation and there have been other proposals. SIP uses SDP to offer possible media types which can be used for a call. The receiving User Agent (UA) responds with the media it wants to use.

Field	Name	Mand./Opt.
v=	Protocol version number	m
o=	Owner/creator & session identity	m
s=	Session name	m
I=	Session information	o
u=	Uniform Resource Identifier	o
e=	Email address	o
p=	Phone number	o
c=	Connection information	m
b=	Bandwidth information	o
t=	Time session starts and stops	m
r=	Repeat times	o
z=	Time zone corrections	o
k=	Encryption key	o
a=	Attribute lines	o
m=	Media information	m
a=	Media attributes	o

Note: Field order is significant

9.7.2 SIP Redirect (Proxy) Server

SIP network servers behave as proxy or redirect servers. A SIP Proxy Server forwards requests from a UA to the next SIP Server UA within the network. A SIP Proxy Server also retains information for billing/accounting purposes. SIP Proxy Servers respond to client requests and inform them of the requested server's address. Numerous hops can take place to reach the final destination. SIP is very flexible and allows servers to contact external location servers to determine user or routing policies. It does not bind the user into only one scheme to locate users.

In addition, SIP servers can either maintain state-full information or forward requests in a state-less fashion.

- **State-less:** processes the message and forgets everything else in regard to the call, until the arrival of next message.
 - Deployed in the core of the networks, optimised for performance.
 - Could be integrated in IP routers.
- **State-full:** holds information in regard to the set-up and tear-down of the call.
 - Maintain state of SIP transaction, but may not see all transactions. Associated with a session, e.g. BYE.
 - Can perform *forking* and provide other services to UAs.
- **Call State-full:** Proxy Servers
 - Are in the call path from set up to tear down.
 - Can provide information about the call to user or service provider.

9.7.3 SIP Registrar

A registrar is a server that accepts REGISTER method requests and places the information it receives in those requests into the location service for the domain it manages.

9.7.4 Location Server

This is a call control Database which records the details of previously registered SIP addresses.

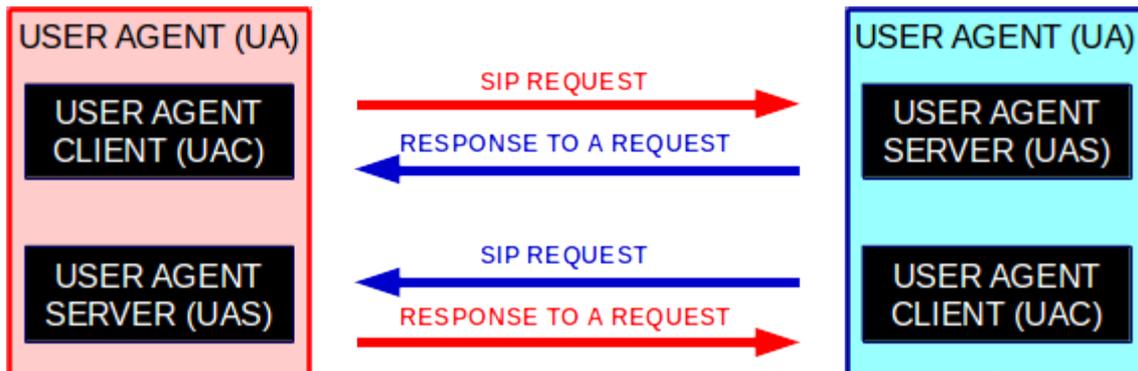
9.7.5 User Agent Client (UAC)

A user agent client is a logical UA entity that creates a new request and then uses the client transaction state machine to send it. The role of UAC lasts only for the duration of that transaction. In other words, if a piece of software initiates a request, it acts as a UAC for the duration of that transaction. If it receives a request later, it assumes the role of a User Agent Server (UAS) for the processing of that transaction.

9.7.6 User Agent Server

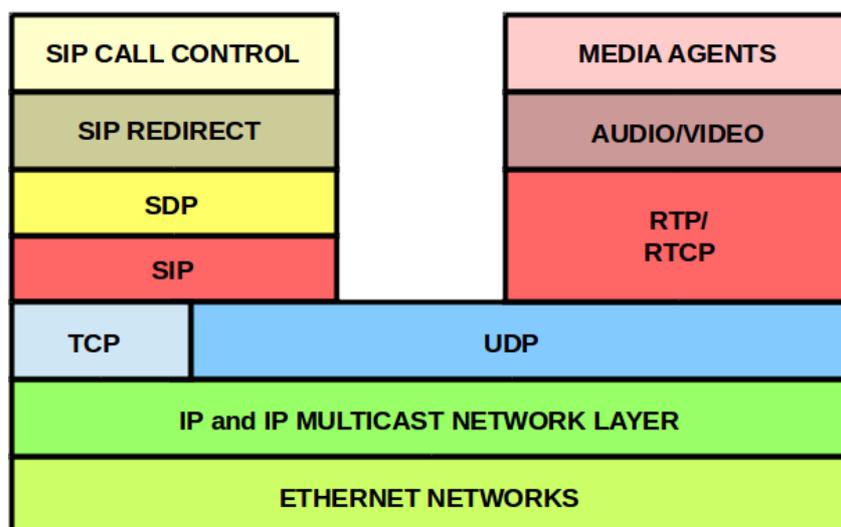
A UAS is a logical entity that generates a response to a SIP request. The response accepts, rejects, or redirects the request. This role lasts only for the duration of that transaction. In other words, if a piece of software responds to a request, it acts as a UAS for the duration of that transaction. If it generates a request later, it assumes the role of a UAC for the processing of that transaction.

9.7.7 SIP UA and Server Roles



The role of the UAC and UAS, as well as the proxy and redirect servers, is defined on a transaction-by-transaction basis. For example, the UA initiating a call acts as a UAC when sending the initial INVITE method request and as a UAS when receiving a BYE method request from the caller. Similarly, the same software can act as a proxy server for one request and as a redirect server for the next request.

9.7.8 SIP Multimedia Protocol Stack



SIP utilises both UDP, TCP or SCTP. UDP allows the application to more carefully control the timing of messages and their retransmission, to perform parallel searches without requiring TCP connection state for each outstanding request, and to use multicast. Routers can more readily snoop SIP UDP packets.

TCP allows easier passage through existing firewalls. When TCP is used, SIP can use one or more connections to attempt to contact a user or to modify parameters of an existing conference. Different SIP requests for the same SIP call may use different TCP connections or a single persistent connection, as appropriate.

9.7.9 *SIP Commands and Responses*

SIP defines several methods: *REGISTER*, *INVITE*, *ACK*, *OPTIONS*, *BYE*, *CANCEL*, *SUBSCRIBE*, *PUBLISH* and *NOTIFY*. Methods that are not supported by a proxy or redirect server are treated by that server as an *OPTIONS* method and forwarded accordingly. Methods that are not supported by a UAS or registrar cause a *501 (Not Implemented)* response to be returned.

- ***REGISTER*** is used by a SIP device to register with a SIP Server.
- ***INVITE*** is used by a registered SIP device to request a connection with another device or service.
- ***ACK*** and ***CANCEL*** are used when setting up sessions.
- ***BYE*** is used to terminate sessions.

The SIP events framework provides an extensible facility for requesting notification of certain events from other SIP UAs.

- ***SUBSCRIBE*** is used to allow a client application subscribe to a service like voice-mail or other more specific services like remote patient medical device for example.
- ***PUBLISH*** allows a service to report an event, for example a voice-mail service reporting that it has received a new voice-mail, or in the medical device example might report pre-programmed information like the customers personal and medical details in response to an event.
- ***NOTIFY*** is used with a Presence Server. A presence server is a broker that devices subscribe services to, the broker manages the relationship with the services. If it received a *PUBLISH* from a service, it checks its database for subscribed devices to the service and generates *NOTIFY* messages for each device on the list.

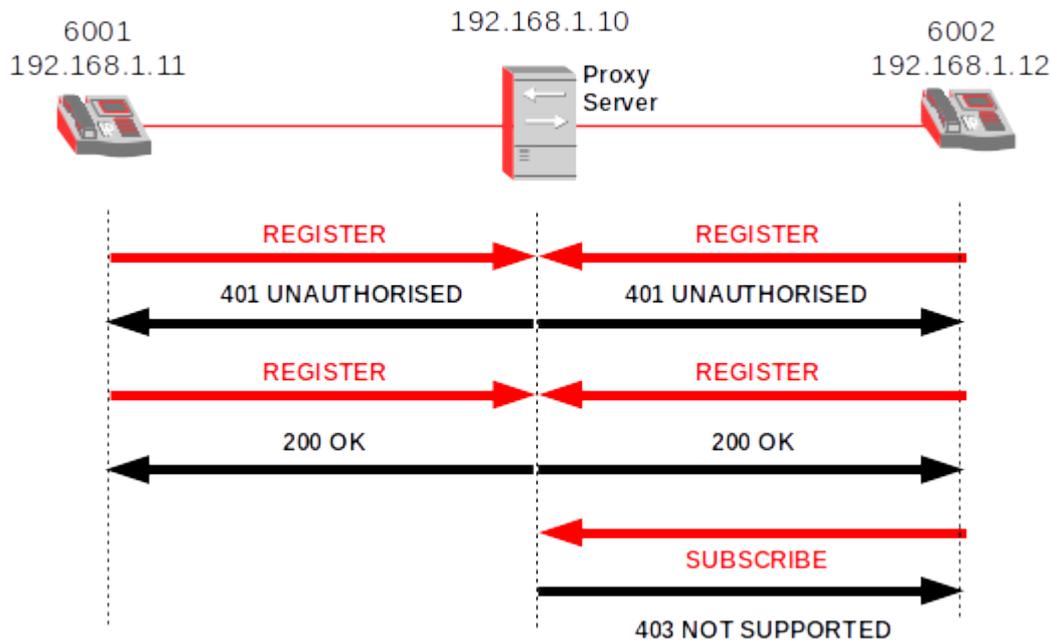
SIP Headers Used in Requests and Responses

Call-ID	Used to uniquely identify a call between two user agents
Contact	Used to convey URL of original resource requested or request originator
CSeq	Command Sequence identifies out of sequence requests & retransmissions
From	Identifies originator of request
To	Indicates recipient of request
Subject	Optional header indicating subject of media session
Content-Length	Number of octets in the message body
Content-Type	Indicates Internet media type. If not present application/SDP is assumed
User Agent	Provides additional information about the user agent e.g. manufacturer
Server	Provides additional information about the User Agent Server
Via	Records the route taken by a request and used to route response
Record-Route	Used to force all requests between User Agents to be routed via a Proxy
Route	Forces routing through a path extracted from a Record-Route header
Authorization	Carries credentials of user agent to a server
Encryption	Used to specify the portion of a SIP message that has been encrypted
Hide	Requests next hop proxy to encrypt the Via headers
Priority	Allow the user agent to set the priority of a request: e.g. urgent, emergency
Supported	List one more options implemented in a user agent or server
Unsupported	Indicates features that are not supported by the server

SIP Responses and Error Codes

SIP responses are distinguished from requests by having a *Status-Line* as their start-line. A Status-Line consists of the protocol version followed by a numeric Status-Code and its associated textual phrase, with each element separated by a single SP character. The first digit of the Status-Code defines the class of response. The last two digits do not have any categorisation role. For this reason, any response with a status code between 100 and 199 is referred to as a "1xx response", any response with a status code between 200 and 299 as a "2xx response", and so on.

9.7.10 SIP Registration



Device registration entails sending a *REGISTER* method request to a special UAS known as a registrar. A registrar acts as the front end to the location service for a domain, reading and writing mappings based on the contents of *REGISTER* requests. This location service is then typically consulted by a proxy server that is responsible for routing requests for that domain. The registrar and proxy server are logical roles that can be played by a single device in a network.

The UAC of each terminal sends a SIP *REGISTER* method to declare itself to the network. The register assigns the information to the location server and returns a *200 OK* message or a *401 UNAUTHORISED* if it is going to present an authentication challenge. The terminal at that stage may attempt to subscribe to additional services with a SIP *SUBSCRIBE* method. If this is allowed a *200 OK* response is returned otherwise the terminal will receive a *403 NOT SUPPORTED* message.

Example Registration

SIP device 6001 sends a SIP *REGISTER* method to the SIP server. The request includes the user's contact list. This SIP Server sends a *401 UNAUTHORISED* message to SIP device 6001 with a challenge to SIP device 6001. The device encrypts the user information according to the challenge issued by the SIP server and sends the response to the SIP server. The SIP server validates the SIP device 6001 credentials and registers the user in its contact database. It returns a *200 OK* response in acknowledgement to the SIP device 6001.

Frame 1 REGISTER | 6001 -> SIP Server

```
REGISTER sips:192.168.1.10 SIP/2.0
Via: SIP/2.0/TLS 192.168.1.11:5061;branch=z9hG4bKnashds7
Max-Forwards: 70
From: 6001 <sips:6001@192.168.1.11>;tag=a73kszlf1
To: 6001 <sips:6001@192.168.1.11>
Call-ID: fg3j56k79d3DK09@192.168.1.11
CSeq: 1 REGISTER
Contact: <sips:6001@192.168.1.11>
Content-Length: 0
```

Frame 2 401 UNAUTHORISED | SIP Server -> 6001

```
SIP/2.0 401 Unauthorised
Via: SIP/2.0/TLS 192.168.1.11:5061;branch=z9hG4bKnashds7
;received=192.168.1.11
From: 6001 <sips:6001@192.168.1.11>;tag=a73kszlf1
To: 6001 <sips:6001@192.168.1.11>;tag=4543334455
Call-ID: fg3j56k79d3DK09@192.168.1.11
CSeq: 1 REGISTER
WWW-Authenticate: Digest realm="sip_realm.ftacademy.org", qop="auth",
nonce="6b9c8e4adf84f1ca94341a89cbe5acd4",
opaque="", stale=FALSE, algorithm=MD5
Content-Length: 0
```

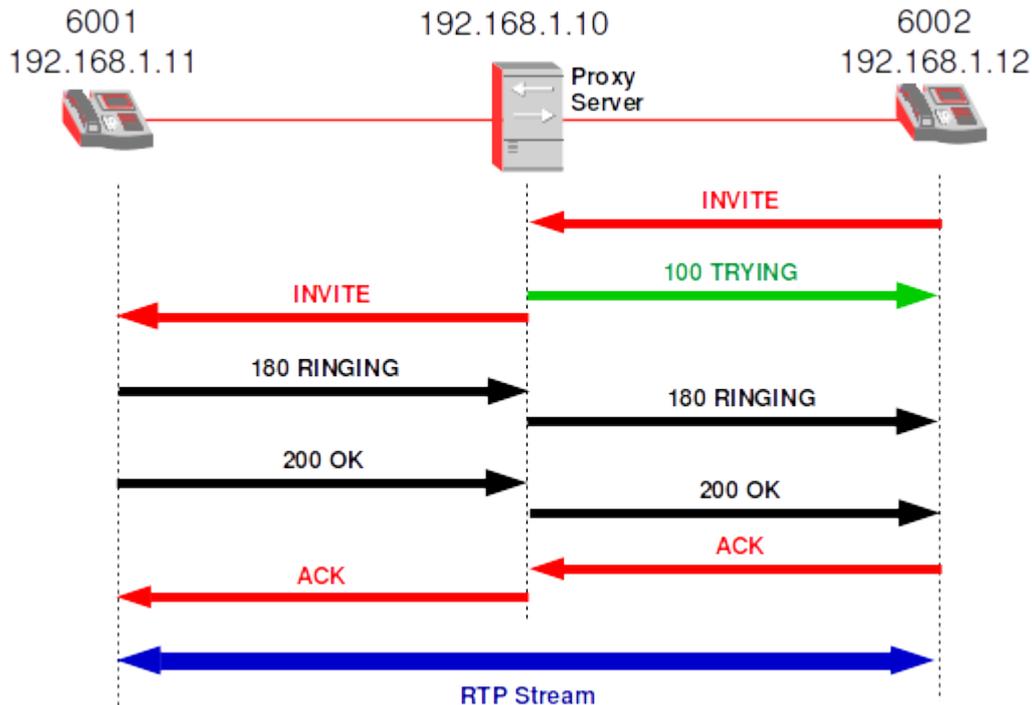
Frame 3 REGISTER | 6001 -> SIP Server

```
REGISTER sips:192.168.1.10 SIP/2.0
Via: SIP/2.0/TLS 192.168.1.11:5061;branch=dh78H9Ng2FDd3
Max-Forwards: 70
From: 6001 <sips:6001@192.168.1.11>;tag=ghY7k9h34fP9
To: 6001 <sips:6001@192.168.1.11>
Call-ID: fg3j56k79d3DK09@192.168.1.11
CSeq: 2 REGISTER
Contact: <sips:6001@192.168.1.11>
Authorization: Digest username="6001", realm="sip_realm.ftacademy.org"
nonce="6b9c8e4adf84f1ca94341a89cbe5acd4", opaque="",
uri="sips:192.168.1.10",
response="4adf8be5ca9436b9acd44f1c8e41a89"
Content-Length: 0
```

Frame 4 200 OK | SIP Server -> 6001

```
SIP/2.0 200 OK
Via: SIP/2.0/TLS 192.168.1.11:5061;branch=dh78H9Ng2FDd3
;received=192.168.1.11
From: 6001 <sips:6001@192.168.1.11>;tag=ghY7k9h34fP9
To: 6001 <sips:6001@192.168.1.11>;tag=ft65hJ21FD
Call-ID: fg3j56k79d3DK09@192.168.1.11
CSeq: 2 REGISTER
Contact: <sips:6001@192.168.1.11>;expires=3600
Content-Length: 0
```

9.7.11 SIP Call Setup



The SIP Proxy Server controls calls between the SIP devices within the IPT network. Any SIP Device wishing to call another will send the SIP Proxy Server a SIP *INVITE* method which will contain the proposed call parameters in an SDP message. The SIP Proxy Server will acknowledge this with a *100-TRYING* message. The SIP Proxy Server forwards the *INVITE* method to the called SIP Device but will insert an additional *Via:* header putting in its own details to ensure the return message comes *via* itself. If the called device is available the SIP Proxy Server will receive a *180-RINGING* message. It forwards it to the originator after stripping out the top *Via:* header line. When the called SIP Device answers it sends a *200-OK* message to the SIP Proxy Server which in turn forwards it after inserting a *Via:* header. Similarly an *ACK*nowledgement method will pass to the called device from the originating device via the SIP Proxy Server and a Real Time Protocol (RTP) media stream will be established based on that negotiated in the SDP information in the SIP messages.

Note: This flow is somewhat simplified. It is typical for a *401 UNAUTHORISED* to be sent back from the SIP Proxy Server in response to the initial *INVITE* method as the request requires user authentication.

Frame 1 **INVITE** | **6002** -> **SIP Proxy Server**

```
INVITE
Message Header
  CSeq: 1 INVITE
  Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG1ea92;rport User-Agent: Ekiga
  From: "6002" <sip:6002@192.168.1.12>;tag=7e5de992
  Call id: 9e60e992@192.168.1.12
  To: <sip:6001@192.168.1.10>
  Contact: "6002" <sip:6002@DEVICE 6002:5060>
  Allow: INVITE,ACK,OPTIONS,BYE,CANCEL,SUBSCRIBE,NOTIFY,REFER,MESSAGE,INFO,PING,PRACK
  Content length: 868
  Content-Type: : application/sdp
  Max-Forwards: 70
Message Body
Version=0
Owner=6002 2890844526 2890844526 IN IP4 192.168.1.11
Session name=-
Connection info=IN IP4 192.168.1.11
Time=0 0
Media info=audio 49172 RTP/AVP 0
Media attributes=rtpmap:0 PCMU/8000
```

Frame 2 **100 Trying** | **SIP Proxy Server** -> **6002**

```
SIP/2.0 100 Trying
Message Header
  Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG1ea92;rport User-Agent: Ekiga
  From: 6002 <sip:6002@192.168.1.10>;tag=9fxced76sl
  To: 6001 <sip:6001@192.168.1.10>
  Call-ID: 4553838437475765942@192.168.1.11
  CSeq: 2 INVITE
  Content-Length: 0
```

Frame 3 **INVITE** | **SIP Proxy Server** -> **6001**

```
INVITE
Message Header
  CSeq: 1 INVITE
  Via: SIP/2.0/UDP DEVICE SIP_Proxy:5060;branch=z9hG1ea92;rport User-Agent: Asterisk
  Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG1ea92;rport User-Agent: Ekiga
  From: "6002" <sip:6002@192.168.1.12>;tag=7e5de992
  Call id: 9e60e992@192.168.1.12
  To: <sip:6001@192.168.1.10>
  Contact: "6002" <sip:6002@DEVICE 6002:5060>
  Allow: INVITE,ACK,OPTIONS,BYE,CANCEL,SUBSCRIBE,NOTIFY,REFER,MESSAGE,INFO,PING,PRACK
  Content length: 868
  Content-Type: : application/sdp
  Max-Forwards: 70
Message Body
Version=0
Owner=6002 2890844526 2890844526 IN IP4 192.168.1.11
Session name=-
Connection info=IN IP4 192.168.1.11
Time=0 0
Media info=audio 49172 RTP/AVP 0
Media attributes=rtpmap:0 PCMU/8000
```

Frame 4 **180 RINGING** | **6001** -> **SIP Proxy Server**

```
SIP/2.0 180 RINGING
Message Header
  Via: SIP/2.0/UDP DEVICE SIP_Proxy:5060;branch=z9hG1ea92;rport User-Agent: Asterisk
  Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG1ea92;rport User-Agent: Ekiga
  From: 6002 <sip:6002@192.168.1.10>;tag=9fxced76sl
  To: 6001 <sip:6001@192.168.1.10>
  Call-ID: 4553838437475765942@192.168.1.11
  CSeq: 2 INVITE
  Content-Length: 0
```

Frame 5 **180 RINGING** | SIP Proxy Server -> 6002

SIP/2.0 180 RINGING

Message Header

Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG1ea92;rport User-Agent: Ekiga
 From: 6002 <sip:6002@192.168.1.10>;tag=9fxced76sl
 To: 6001 <sip:6001@192.168.1.10>
 Call-ID: 4553838437475765942@192.168.1.11
 CSeq: 2 INVITE
 Content-Length: 0

Frame 6 **200 OK** | 6001 -> SIP Proxy Server

SIP/2.0 200 OK

Message Header

Via: SIP/2.0/UDP DEVICE SIP_Proxy:5060;branch=z9hG1ea92;rport User-Agent: Asterisk
 Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG1ea92;rport User-Agent: Ekiga
 From: 6002 <sip:6002@192.168.1.10>;tag=9fxced76sl
 To: 6001 <sip:6001@192.168.1.10>
 Call-ID: 4553838437475765942@192.168.1.11
 CSeq: 2 INVITE
 Content-Length: 0

Frame 7 **200 OK** | SIP Proxy Server -> 6002

SIP/2.0 200 OK

Message Header

Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG1ea92;rport User-Agent: Ekiga
 From: 6002 <sip:6002@192.168.1.10>;tag=9fxced76sl
 To: 6001 <sip:6001@192.168.1.10>
 Call-ID: 4553838437475765942@192.168.1.11
 CSeq: 2 INVITE
 Content-Length: 0

Frame 8 **ACK** | 6002 -> SIP Proxy Server

ACK

Message Header

Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG1ea92;rport User-Agent: Ekiga
 From: 6002 <sip:6002@192.168.1.10>;tag=9fxced76sl
 To: 6001 <sip:6001@192.168.1.10>
 Call-ID: 4553838437475765942@192.168.1.11
 CSeq: 2 ACK
 Content-Length: 0

Frame 9 **ACK** | SIP Proxy Server -> 6001

SIP/2.0 200 OK

Message Header

Via: SIP/2.0/UDP DEVICE SIP_Proxy:5060;branch=z9hG1ea92;rport User-Agent: Asterisk
 Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG1ea92;rport User-Agent: Ekiga
 From: 6002 <sip:6002@192.168.1.10>;tag=9fxced76sl
 To: 6001 <sip:6001@192.168.1.10>
 Call-ID: 4553838437475765942@192.168.1.11
 CSeq: 2 ACK
 Content-Length: 0

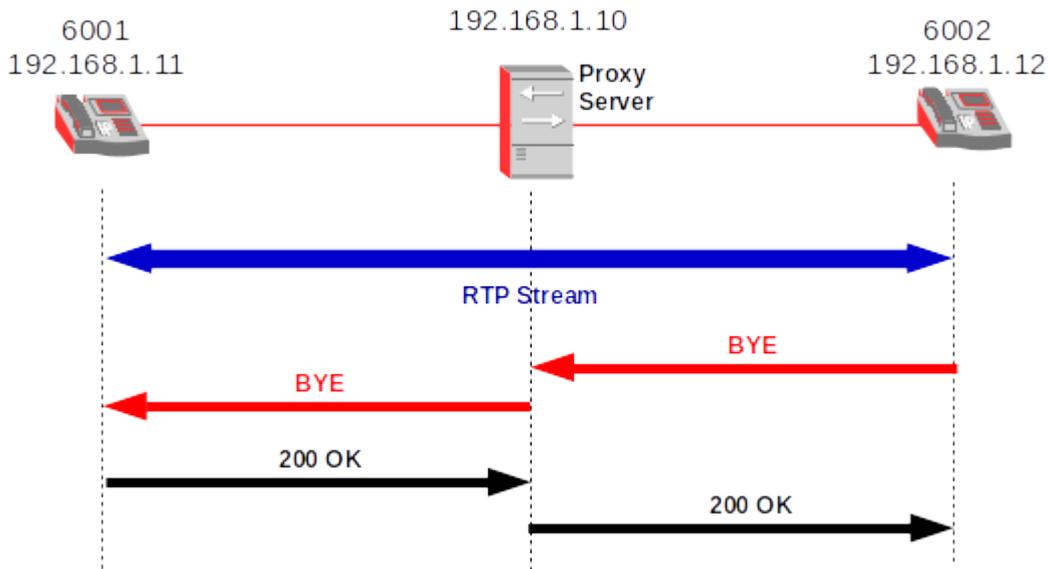
The RTP Stream establishes between SIP device 6002 and SIP device 6001.

Frame 123

Real-Time Transport Protocol

10.. = Version: RFC 1889 Version (2)
 ..0. = Padding: False
 ...0 = Extension: False
 0000 = Contributing source identifiers count: 0
 0... = Marker: False
 Payload type: GSM 06.10 (3)
 Sequence number: 20871
 Timestamp: 1567201601
 Synchronization Source identifier: 0xf67ed7d3 (4135507923)
 Payload: ffffffff...

9.7.12 SIP Call Terminate



The call can be terminated by a BYE message from either phone. If a terminate recording is required the SIP Proxy Server can impose a terminate via the SIP Proxy in the original set-up messages.

Frame 1 **BYE | 6002 -> SIP Proxy Server**

```

BYE
Message Header
Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG1ea92;rport User-Agent: Ekiga
From: 6002 <sip:6002@192.168.1.10>;tag=9fxced76sl
To: 6001 <sip:6001@192.168.1.10>
Call-ID: 4553838437475765942@192.168.1.11
CSeq: 2 BYE
Content-Length: 0

```

Frame 2 **BYE | SIP Proxy Server -> 6001**

```

BYE
Message Header
Via: SIP/2.0/UDP DEVICE SIP_Proxy:5060;branch=z9hG1ea92;rport User-Agent: Asterisk
Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG1ea92;rport User-Agent: Ekiga
From: 6002 <sip:6002@192.168.1.10>;tag=9fxced76sl
To: 6001 <sip:6001@192.168.1.10>
Call-ID: 4553838437475765942@192.168.1.11
CSeq: 2 BYE
Content-Length: 0

```

Frame 3 **200 OK | 6001 -> SIP Proxy Server**

```

SIP/2.0 200 OK
Message Header
Via: SIP/2.0/UDP DEVICE SIP_Proxy:5060;branch=z9hG1ea92;rport User-Agent: Asterisk
Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG1ea92;rport User-Agent: Ekiga
From: 6002 <sip:6002@192.168.1.10>;tag=9fxced76sl
To: 6001 <sip:6001@192.168.1.10>
Call-ID: 4553838437475765942@192.168.1.11
CSeq: 2 BYE
Content-Length: 0

```

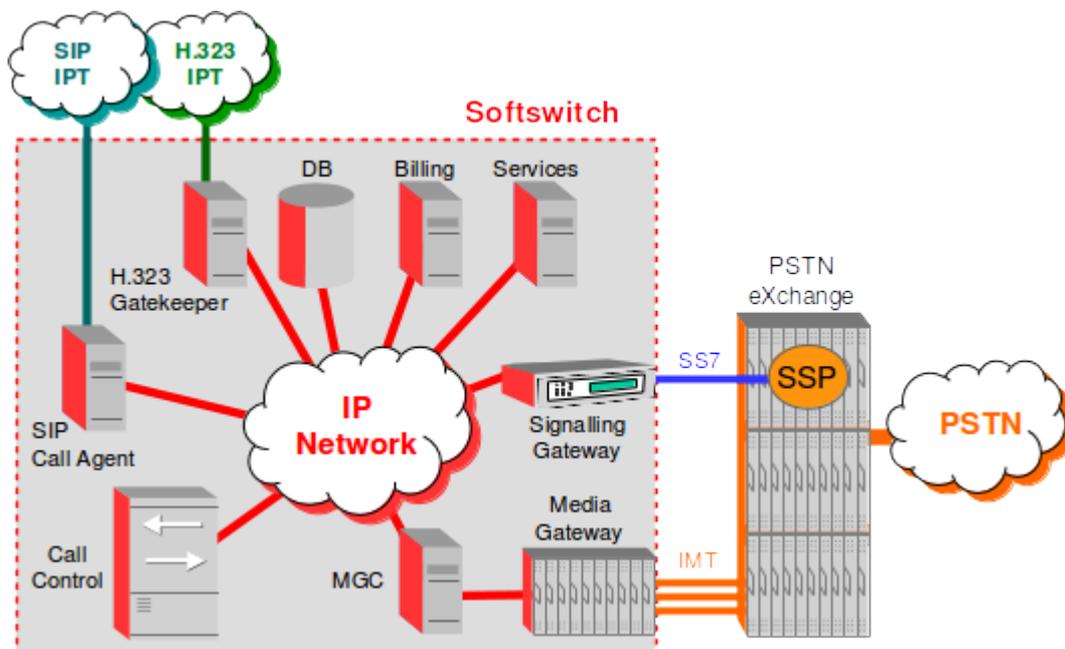
Frame 4 200 OK | SIP Proxy Server -> 6002

```
SIP/2.0 200 OK
Message Header
Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG1ea92;rport User-Agent: Ekiga
From: 6002 <sip:6002@192.168.1.10>;tag=9fxced76s1
To: 6001 <sip:6001@192.168.1.10>
Call-ID: 4553838437475765942@192.168.1.11
CSeq: 2 BYE
Content-Length: 0
```

9.8 IPT and the PSTN

IPT networks must interact with the PSTN either over Plain Old Telephone Service (POTS) lines, ISDN or with Inter Machine Trunks (IMT) for bearer channels and SS7 interface to an Service Switching Point (SSP) for signalling.

9.8.1 Softswitch



A Softswitch is the central device in an IPT. It includes the SIP Proxy Server functions already described plus Call Control, H.323 Gatekeeper, Gateways that allow for interconnection to the Public Switched Telephony Networks (PSTN) as well as back-end services like Billing and Service Delivery platforms.

A Softswitch can control connections at the junction point between circuit and packet networks. A single device containing both the switching logic and the switching fabric can be used for this purpose; however, modern technology has led to a preference for decomposing this device into;

- Call Agent (CA)
- H.323 Gatekeeper
- Signalling Gateway (SG)
- MG Controller (MGC)
- Media Gateways (MG)

9.8.2 Call Agent

The Call Agent takes care of functions like billing, call routing, signalling, call services and so on and is the *brain* of the device. A Call Agent may control several different MG's in geographically dispersed areas over a TCP/IP link.

The call agent functions include:

- Billing
- Call routing
- Signalling
- Call services

A CA may control several different MG's in geographically dispersed areas over a TCP/IP links.

9.8.3 Media Gateway

The MG connects different types of digital media stream together to create an end-to-end path for the media (voice and data) in the call. It may have interfaces to connect to traditional PSTN networks like E1, E3 or STM1, it may have interfaces to connect to ATM and IP networks and in the modern system will have Ethernet interfaces to connect IPT calls. The call agent will instruct the MG to connect media streams between these interfaces to connect the call, all transparently to the end-users.

Looking towards the end users from the switch, the MG may be connected to several access devices. These access devices can range from small Analogue Telephone Adaptors (ATA) which provide just one telephone plug to an Integrated Access Device (IAD) or Private Automatic Branch eXchange (PABX) which may provide several hundred telephone connections.

9.9 MG Controllers

MGs provide a conversion point between the audio signals carried in RTP streams and other network bearer channels. They are controlled by MGCs using the Media Gateway Control Protocol (MGCP). MGCP was first proposed as RFC 2705 in 1999 and updated by RFC 3435 in 2003.

Another protocol Megaco (H.248) H.248 or Gateway Control Protocol (GCP) is a joint recommendation from the ITU-T and is also the subject of IETF RFC 3525. It is an implementation of the Media Gateway Control Protocol Architecture from RFC 2805 which defines protocols that are used between elements of a physically decomposed multimedia gateway. The protocol controls the elements of a physically decomposed multimedia gateway, which enables separation of the call control logic from media processing logic.

Megaco (H.248) performs the same functions as MGCP, it uses a different syntax, commands and processes and supports a broader range of networks.

9.9.1 Signalling Gateway

The SG provides Transparent inter-working of signalling between circuit-switched and IP network. It may terminate SS7 signalling or translate and relay messages over an IP network to a MGC or another SG. It provides a full duplex inter-networking function between SS7 and IP protocols.

The SG communicate with:

- SS7 network over A-Links to SSPs
- MG over TCP/IP
- MGC which communicates with multiple MG's via MGCP or H.248 Megaco

9.10 Services

Custom services can be created by accessing subroutines in the application servers using Application Program Interfaces (API). When service modules are used in combination the service possibilities are vast. Here are some common protocols and APIs used to deliver services on SIP networks.

Call Processing Language (CPL): is an XML-based scripting language for describing and controlling call services.

SIP Common Gateway Interface (CGI): is almost identical to HTTP CGI and is particularly suitable as a web service creation environment.

The **Java APIs for Integrated Networks (JAIN):** are specified as a community extension to the Java platform. By providing a new level of abstraction and associated Java interfaces for service creation across circuit switched and packet networks, JAIN bridges IP and IN protocols to create an open market.

OneAPI: is a set of application programming interfaces (APIs) that expose network capabilities over IP networks. OneAPI complements existing client-side and Web APIs by providing access to network capabilities and information, regardless of operator. The OneAPI supersedes a series of earlier protocols called *Parlay X APIs*. Basically it allows an application to authenticate new users against a service providers subscriber identity data. Applications are build to operate with one service provider, but the identity feature will work across all carriers' networks as OneAPI acts as a universal bridge between different operators.

9.11 FOSS Implementations

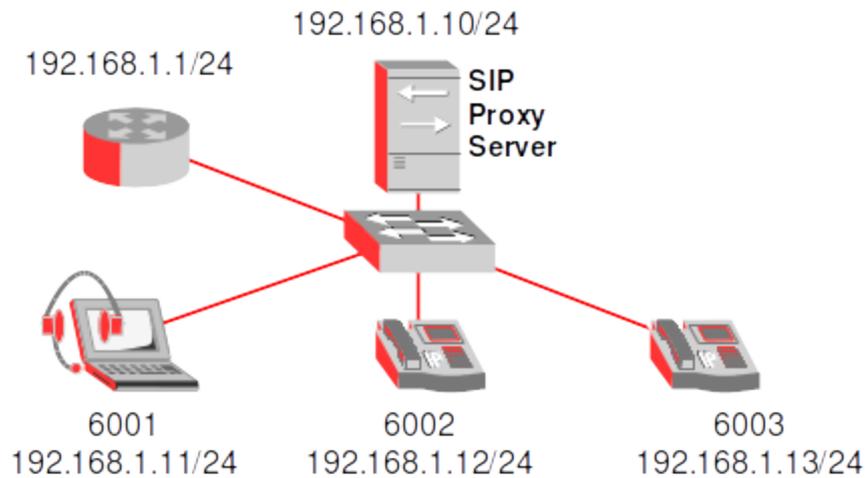
There are a number of FOSS licensed SIP switch products available.

- Asterisk
- Cipango SipServlets 1.1 application server
- Elastix
- Enterprise Communications System sipXecs
- FreePBX
- FreeSWITCH
- GNU SIP Witch
- Kamailio, formerly OpenSER
- Mobicents Platform
- Mysipswitch
- OpenSIPS, fork of OpenSER
- SailFin
- SIP Express Router (SER)
- Yate
- YXA, based on Erlang

Additionally there is lots of choice for SIP Client devices.

- Blink, for Linux and Windows
- Ekiga, formerly named GnomeMeeting
- Empathy, using GTK+ libraries and Telepathy framework
- Jitsi, a Java VoIP and Instant Messaging client with ZRTP encryption, for FreeBSD, Linux, Mac OS X, Windows
- KPhone, using Qt libraries, for Linux
- Linphone, with a core/UI separation, the GUI is using GTK+ libraries, for Linux, Mac OS X, Windows, and mobile phones
- MicroSIP, lightweight softphone, using PJSIP stack, for Windows
- PhoneGaim, based on Pidgin
- QuteCom, formerly named OpenWengo, using Qt libraries, for Windows, Mac, and Linux
- SFLphone, with GTK+/Qt GUI, also supports IAX2 protocol, for Linux
- Telephone, Mac OS X softphone written in Cocoa/Objective-C
- Twinkle, using Qt libraries, for Linux
- Yate client, using Qt libraries

9.12 Test network



To build the network in the diagram the document describes the process using an Asterisk Server, an Ekiga or SFLphone soft client and 2 LinkSys SPA IP phones.

9.12.1 Asterisk Server

Prepare a GNU/Linux server which will fulfil the role of an Asterisk Server with a fixed IP address, a netmask and a default gateway. (hint - edit the `/etc/network/interfaces` file).

```
$ ip addr | grep inet
  inet 127.0.0.1/8 scope host lo
  inet6 ::1/128 scope host
  inet 192.168.1.10/24 brd 192.168.1.255 scope global eth1
  inet6 fe80::ba27:ebff:fe95:4739/64 scope link
```

Update the system and install the Asterisk Server.

```
$ sudo apt-get update
```

```
$ sudo apt-get install asterisk
```

```
- Configuring libvpb0
```

```
- ITU-T telephone code: 353, Click OK # i.e. 353 is the Ireland CC
```

The files that are required for a basic set-up are in the `/etc/asterisk/` directory. These are:

- sip.conf
- extensions.conf
- voicemail.conf

SIP Channel configuration – sip.conf

Backup the `/etc/asterisk/sip.conf` file

```
$ sudo mv /etc/asterisk/sip.conf /etc/asterisk/sip.conf.orig
```

Configure three SIP channels as shown in the network diagram.

General

This section defines a *context* in which the extensions will be configured later, a port number for SIP messaging and an IP address to bind to, this is the IP address on the interface that messages are expected on, i.e. the listening address. The default of 0.0.0.0 means *listen on all network interfaces*.

Channels

Each channel is labelled, the extension number is used in the example i.e. `[6001]`. The channel type can be `<peer | user | friend>`.

Channel type

- **peer**: is a SIP entity to which Asterisk sends calls, like another SIP provider. The peer must authenticate at registration.
- **user**: is a SIP entity which can make calls through the Asterisk switch but cannot receive calls.
- **friend**: is an entity which is both a user and a peer, i.e. an IP Phone, Soft-client etc. Asterisk actually creates two objects, a peer and a user, both with the same name.

Context

Defines a context in the extensions list.

Secret

Shared secret that the SIP Client device must also have.

Host

IP address of the SIP Client. In the case of it being set to *dynamic* then the SIP Client will register its IP address at the time of registration. This allows a DHCP Server to be used for example to supply IP addresses to the SIP Client devices.

```
$ sudo vi /etc/asterisk/sip.conf
```

```
[general]
context=internal
port=5060
bindaddr=0.0.0.0
```

```
[6001]
type=friend
context=internal
secret=1234
host=dynamic
```

```
[6002]
type=friend
context=internal
secret=1234
host=dynamic
```

```
[6003]
type=friend
context=internal
secret=1234
host=dynamic
```

SIP Dialplan – extensions.conf

Now backup the `/etc/asterisk/extensions.conf` file.

```
$ sudo mv /etc/asterisk/extensions.conf /etc/asterisk/extensions.conf.orig
```

Now adding the three entries to the dialplan for SIP Devices as shown in the network diagram.

General

Set the *static* option to yes, this relates to the *dialplan save* command. The *writeprotect* set to *no* and *static=yes*, then the current dialplan can be saved with the *dialplan save* command overwriting the existing *extensions.conf*. *Clearglobalvars* if set will clear global variables in the event of an Asterisk reload. In this case it is set to ensure that global variables will be persistent in the case of a reload.

Context

The context is where the extensions are defined in a dialplan priority ordered list in the following format:

```
exten = extension,priority,Command(parameters)
```

So for example the line below means this line is the first priority for extension 6001. The command *Dial* tells the Asterisk Switch to place a call of type SIP to `${EXTEN}`. `${EXTEN}` simply replaces itself with the extension field, in this case 6001. So it is possible to rewrite this line with 6001 in place of `${EXTEN}`. There are many other types (for example):

zap	Regular telephone line card in the Asterisk Switch (line card)
mgcp	MGCP
misdn	ISDN channel
h.323	H.323 network device
iax / iax2	Inter Asterisk eXchange protocol
skinny	Cisco Skinny client-only
sip	SIP
gtalk	Google Talkchannel
ss7	SS7

```
exten => 6001,1,Dial(SIP/${EXTEN})
```

The Hangup line is the second priority and should the call fail then this is the next step in the dial plan. This causes a hang up of the calling channel. A cause code like *404 NOT FOUND* can be put in the parenthesis is given the channel's hangup cause is set to the given value.

```
exten => 6001,2,Hangup()
exten => 6001,2,Hangup(404 NOT FOUND)
```

Configure the dialplan to accommodate the three extensions in the diagram.

```
$ sudo vi /etc/asterisk/extensions.conf

[general]
static=yes
writeprotect=no
clearglobalvars=no

[internal]
exten => 6001,1,Dial(SIP/${EXTEN})
exten => 6001,2,Hangup()

exten => 6002,1,Dial(SIP/${EXTEN})
exten => 6002,2,Hangup()

exten => 6003,1,Dial(SIP/${EXTEN})
exten => 6003,2,Hangup()
```

Restart Asterisk Server and use the *rasterisk* CLI interface to review the SIP peers, users and dialplan.

```
$ sudo service asterisk restart

$ sudo rasterisk

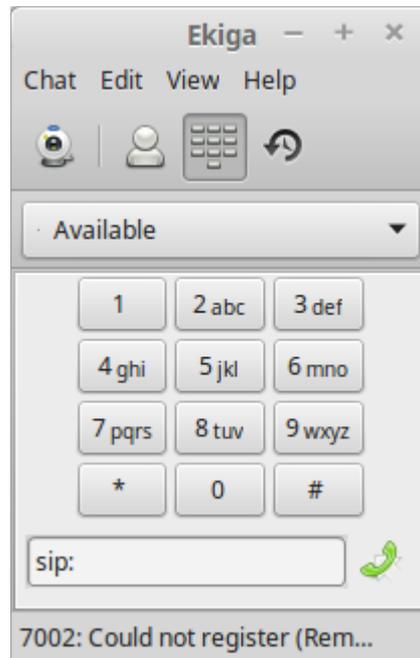
NetDev01*CLI> sip set debug on
NetDev01*CLI> sip reload
NetDev01*CLI> sip show users

NetDev01*CLI> sip show peers
Name/username      Host                Dyn Forcerport ACL Port      Status
6001/6001          192.168.1.11       D   N                5060 Unmonitored
6002/6002          192.168.1.12       D   N                5060 Unmonitored
6003/6003          192.168.1.13       D   N                5060 Unmonitored

NetDev01*CLI> sip show users
Username    Secret    Accountcode    Def.Context    ACL    ForcerPort
6003        1234     1234           default        No    Yes
6002        1234     1234           default        No    Yes
6001        1234     1234           default        No    Yes

NetDev01*CLI> dialplan show internal
[ Context 'internal' created by 'pbx_config' ]
'6001' =>          1. Dial(SIP/6001)           [pbx_config]
                  2. Hangup()                 [pbx_config]
'6002' =>          1. Dial(SIP/6002)           [pbx_config]
                  2. Hangup()                 [pbx_config]
'6003' =>          1. Dial(SIP/6003)           [pbx_config]
                  2. Hangup()                 [pbx_config]
```

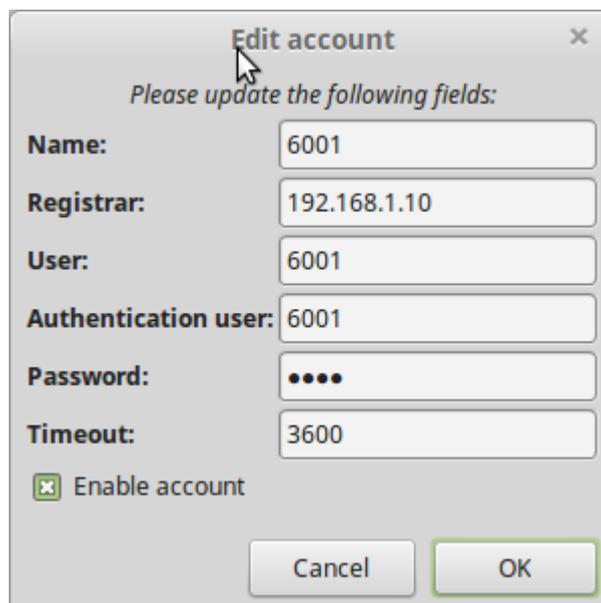
9.12.2 SIP Softphone Client



On the computer with the soft-client which is acting as a SIP Device, i.e. *192.168.1.11* install the SIP Ekiga and the SFLphone Client software.

```
$ sudo apt-get install ekiga
$ sudo apt-get install sflphone-gnome
```

Run the *Ekiga* application, select *Edit -> Accounts*, select *Accounts -> Add a SIP Account*.

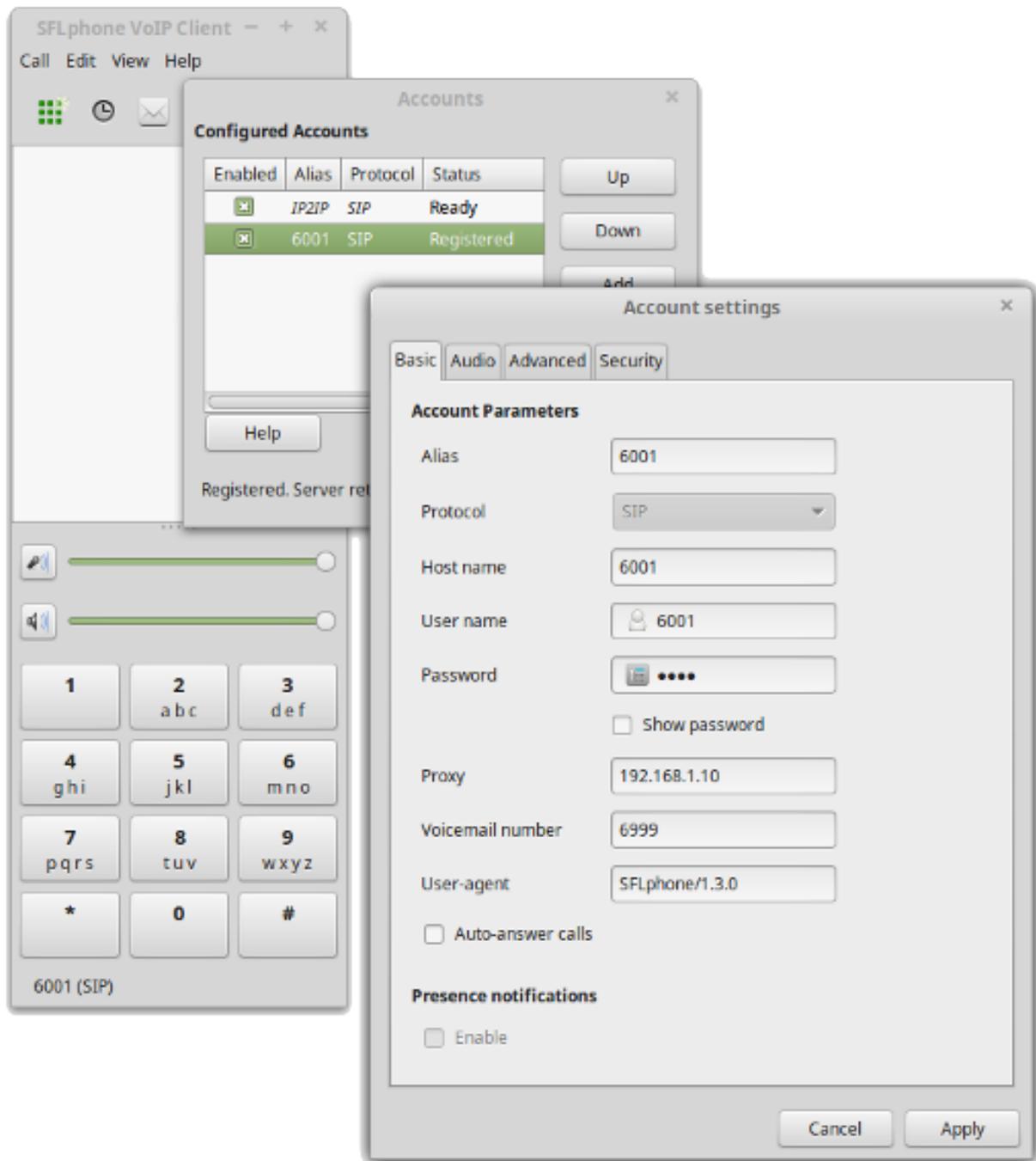


Fill in the table as shown, in this case for 6001 and click *OK*. The account should register with the Asterisk Server.

Another option is the *SPL Softphone*.

Run the application and select *Edit -> Accounts*, then select *Add*.

Fill-out the *Account Parameters* similar to what is shown below as select *Apply*. The Status should change to *Registered*.



9.12.3 SIP Phone

There are many manufacturers of SIP phones like LinkSys, Yealink to name a few. Configure 2 devices for *192.168.1.12* and *192.168.1.13*. The following needs to be configured:

- Network settings
 - Host-name: 6001
 - IP Address: 192.168.1.11
 - IP net-mask: 255.255.255.0
 - IP Gateway: 192.168.1.1
- SIP settings
 - SIP Port: 5060
- Proxy and Registration
 - SIP Proxy: 192.168.1.10
 - Make Call Without Reg: no
 - Answer Call Without Reg: no
- Subscriber information
 - Display name: 6001
 - Authentication ID: 6001
 - User ID: 6001
 - Use Authentication ID: Yes
- Audio Configuration
 - Preferred Codec: G711u
 - Silence Suppression Enable: no

9.12.4 Configuring voice-mail

Asterisk includes a working voice-mail module and it is configured via */etc/asterisk/voicemail.conf* file. Backup the default file.

```
$ sudo mv /etc/asterisk/voicemail.conf /etc/asterisk/voicemail.conf.orig
```

General

For the simple setup it is just necessary to establish the file format to use for voice-mails. Options are `< wav | wav49 | gsm | g723sf | >`. For quality *wav* is a good selection, for small files use either *wav49* or *gsm*.

Default

Voice-mail boxes are setup under the *[default]* context heading.

Entries for each voice-mail box should follows this format:

```
<nnnn> => <pppp>,<name>, <email>
```

where *nnnn* is the SIP Device identity, *pppp* is the voice-mail PIN.

```
$ sudo vi /etc/asterisk/voicemail.conf
```

```
[general]
format=wav
```

```
[default]
6001 => 2345,Ada Lovelace,alovelace@ftacademy.org
6002 => 9876,Charles Babbage,cbabbage@ftacademy.org
6003 => 6789,Luigi Menabrea,lmenabrea@ftacademy.org
```

Dialplan

To complete, re-modify the */etc/asterisk/extensions.conf* file to add the voice-mail to each extension. The *20* in the *Dial()* refers to a time-out delay of 20 seconds before moving to the next priority. i.e. 2. If the letter *u* is present in the *voicemail()* command will cause the unavailable message to be played.

The line *exten => 6999,1,VoiceMailMain(\${CALLERID(num)})* establishes number 6999 as the voice-mail number. When it is rang the number of the device is passed to the voice-mail module, it requests a pass-code before allowing access to the voice-mails. The pass-code is the number in the *voicemail.conf* corresponding to the SIP device ID. i.e. 9876 for 6002.

```
$ sudo vi /etc/asterisk/extensions.conf
```

```
[general]
static=yes
writeprotect=no
clearglobalvars=no
```

```
[internal]
exten => 6001,1,Dial(SIP/${EXTEN},20)
exten => 6001,2,voicemail(${EXTEN},u)

exten => 6002,1,Dial(SIP/${EXTEN},20)
exten => 6002,2,voicemail(${EXTEN},u)

exten => 6003,1,Dial(SIP/${EXTEN},20)
exten => 6003,2,voicemail(${EXTEN},u)

exten => 6999,1,VoiceMailMain(${CALLERID(num)})
```

To disable the password request when ringing in to retrieve voice-mails add the option `s` at the end of the line as follows:

```
exten => 6999,1,VoiceMailMain(${CALLERID(num)},s)
```

Reload the voice-mail settings and confirm setup.

```
NetDev01*CLI> voicemail reload
Reloading voice-mail configuration...

NetDev01*CLI> voicemail show users
Context    Mbox  User                Zone      NewMsg
default    6001  Ada Lovelace        Zone      0
default    6002  Charles Babbage    Zone      0
default    6003  Luigi Menabrea     Zone      0
3 voice-mail users configured.
```

9.13 Testing the configuration

Ensure the user is in the `/etc/group` entry for *wireshark*. Unplug all phones so that the SIP Proxy Server is the only device on the network.

```
$ sudo vi /etc/group
...
wireshark:x:127:alovelace
...
```

On the SIP Proxy Server either run *Wireshark* or its command-line derivative *tshark* to capture all traffic on port 5060 or

```
$ tshark -V -i eth1 -S "-----" -f 'port 5060' | tee /tmp/sipCapture.txt

- -V - Detailed output
- -i <interface> - Interface
- -S - Page separator
- -f <filter> - Filter traffic of this type.
tee - Read from standard input and write to standard output and files.
```

9.13.1 Registration test - IP Phone

Power up one of the IP Phones, say 6002 and the device, a *Linksys/SPA941* sends a SIP registration message to the SIP Proxy Server.

```
Frame 1
Ethernet II, Src: 00:0e:08:d2:e8:2b, Dst: b8:27:eb:95:47:39
Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.10
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Request-Line: REGISTER sip:192.168.1.10 SIP/2.0
  Message Header
    Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG4bK-f50830af
    From: "6002" <sip:6002@192.168.1.10>;tag=b8433b6f238a190400
    To: "6002" <sip:6002@192.168.1.10>
    Call-ID: 6b81461c-88494544@192.168.1.12
    CSeq: 4157 REGISTER
    Max-Forwards: 70
    Contact: "6002" <sip:6002@192.168.1.12:5060>;expires=3600
    User-Agent: Linksys/SPA941
    Content-Length: 0
    Allow: ACK, BYE, CANCEL, INFO, INVITE, NOTIFY, OPTIONS, REFER
    Supported: replaces
```

The SIP Proxy Server, an *Asterisk PBX* responds with the *401 UNAUTHORISED* message inviting the IP Phone to authenticate using the scheme: *Digest* with the *MD5* hashing algorithm on the realm *asterisk* and it supplies a number to be used once (nonce) *752fc98f*.

```
Frame 2
Ethernet II, Src: b8:27:eb:95:47:39, Dst: 00:0e:08:d2:e8:2b)
Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.1.12
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Status-Line: SIP/2.0 401 Unauthorised
  Message Header
    Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG4bK;
      received=192.168.1.12;rport=5060
    From: "6002" <sip:6002@192.168.1.10>;tag=b8433b6f238a190400
    To: "6002" <sip:6002@192.168.1.10>;tag=as092cae5e
    Call-ID: 6b81461c-88494544@192.168.1.12
    CSeq: 4157 REGISTER
    Server: Asterisk PBX
    Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY,
      INFO, PUBLISH
    Supported: replaces, timer
    WWW-Authenticate: Digest algorithm=MD5, realm="asterisk", nonce="752fc98f"
    Content-Length: 0
```

The IP Phone responds with a username *6002*, on the realm *asterisk*, the nonce *752fc98f*, a uri *sip:192.168.1.10*, the algorithm *MD5* and a response to the authentication query *661a4caa478660bcde036ebbb0853cdf*.

```

Frame 3
Ethernet II, Src: 00:0e:08:d2:e8:2b), Dst: b8:27:eb:95:47:39
Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.10
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Request-Line: REGISTER sip:192.168.1.10 SIP/2.0
  Message Header
    Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG4bK-b287c76b
    From: "6002" <sip:6002@192.168.1.10>;tag=b8433b6f238a190400
    To: "6002" <sip:6002@192.168.1.10>
    Call-ID: 6b81461c-88494544@192.168.1.12
    CSeq: 4158 REGISTER
    Max-Forwards: 70
    Authorization: Digest username="6002",realm="asterisk",nonce="752fc98f",
                  uri="sip:192.168.1.10",algorithm=MD5,
                  response="661a4caa478660bcde036ebbb0853cdf"
    Contact: "6002" <sip:6002@192.168.1.12:5060>;expires=3600
    User-Agent: Linksys/SPA941
    Content-Length: 0
    Allow: ACK, BYE, CANCEL, INFO, INVITE, NOTIFY, OPTIONS, REFER
    Supported: replaces

```

The SIP Proxy Server responds to the successful authentication with a *200 OK* message to indicate that the IP Phone has been registered with a contact URI User Part of: *6002*, a contact Uniform Resource Identifier (URI) Host Part of: *192.168.1.12* and a contact URI Host Port of: *5060*.

```

Frame 4
Ethernet II, Src: b8:27:eb:95:47:39, Dst: 00:0e:08:d2:e8:2b)
Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.1.12
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Status-Line: SIP/2.0 200 OK
  Message Header
    Via: SIP/2.0/UDP 192.168.1.12:5060;
        branch=z9hG4bK-b287c76b;received=192.168.1.12;rport=5060
    From: "6002" <sip:6002@192.168.1.10>;tag=b8433b6f238a190400
    To: "6002" <sip:6002@192.168.1.10>;tag=as092cae5e
    Call-ID: 6b81461c-88494544@192.168.1.12
    CSeq: 4158 REGISTER
    Server: Asterisk PBX
    Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY,
          INFO, PUBLISH
    Supported: replaces, timer
    Expires: 3600
    Contact: <sip:6002@192.168.1.12:5060>;expires=3600
    Date: Wed, 08 Apr 2015 21:09:10 GMT
    Content-Length: 0

```

9.13.2 Registration test - Softphone

Run up the Ekiga Softphone on a laptop and monitor the process, it mirrors that of the IP Phone.

```
Frame 1
Ethernet II, Src: 28:d2:44:19:83:95, Dst: b8:27:eb:95:47:39
Internet Protocol Version 4, Src: 192.168.1.11, Dst: 192.168.1.10
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Request-Line: REGISTER sip:192.168.1.10 SIP/2.0
  Message Header
    CSeq: 1 REGISTER
    Via: SIP/2.0/UDP 192.168.1.51:5060;
      branch=z9hG4bK68548735-ae2-e411-8843-28d244198395;rport
    User-Agent: Ekiga Softphone
    From: <sip:6001@192.168.1.10>;tag=6e088735-ae2-e411-8843-28d244198395
    Call-ID: 36fa8635-ae2-e411-8843-28d244198395@riomhaire-0B
    To: <sip:6001@192.168.1.10>
    Contact: <sip:6001@192.168.1.51:5060>;q=1
    Allow: INVITE,ACK,OPTIONS,BYE,CANCEL,SUBSCRIBE,NOTIFY,REFER,MESSAGE,
      INFO,PING,PRACK
    Expires: 3600
    Content-Length: 0
    Max-Forwards: 70

Frame 2
Ethernet II, Src: b8:27:eb:95:47:39, Dst: 28:d2:44:19:83:95
Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.1.11
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Status-Line: SIP/2.0 401 Unauthorised
  Message Header
    Via: SIP/2.0/UDP 192.168.1.51:5060;
      branch=z9hG4bK68548735-ae2-e411-8843-28d244198395;
      received=192.168.1.51;rport=5060
    From: <sip:6001@192.168.1.10>;tag=6e088735-ae2-e411-8843-28d244198395
    To: <sip:6001@192.168.1.10>;tag=as2ef2ea7f
    Call-ID: 36fa8635-ae2-e411-8843-28d244198395@riomhaire-0B
    CSeq: 1 REGISTER
    Server: Asterisk PBX
    Allow: INVITE,ACK,CANCEL,OPTIONS,BYE,REFER,SUBSCRIBE,NOTIFY,
      INFO,PUBLISH
    Supported: replaces,timer
    WWW-Authenticate: Digest algorithm=MD5, realm="asterisk", nonce="3ca33900"
    Content-Length: 0
```

Frame 3

Ethernet II, Src: 28:d2:44:19:83:95, Dst: b8:27:eb:95:47:39
Internet Protocol Version 4, Src: 192.168.1.11, Dst: 192.168.1.10
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol

Request-Line: REGISTER sip:192.168.1.10 SIP/2.0

Message Header

CSeq: 2 REGISTER
Via: SIP/2.0/UDP 192.168.1.51:5060;
branch=z9hG4bK1e078835-ae2-e411-8843-28d244198395;rport
User-Agent: Ekiga Softphone
Authorization: Digest username="6001", realm="asterisk",
nonce="3ca33900", uri="sip:192.168.1.10", algorithm=MD5,
response="eae51c6b4a0caa137b8872d1bf4ff2b"
From: <sip:6001@192.168.1.10>;tag=6e088735-ae2-e411-8843-28d244198395
To: <sip:6001@192.168.1.10>
Contact: <sip:6001@192.168.1.51:5060>;q=1
Allow: INVITE,ACK,OPTIONS,BYE,CANCEL,SUBSCRIBE,NOTIFY,REFER,MESSAGE,
INFO,PING,PRACK
Expires: 3600
Content-Length: 0

Frame 4

Ethernet II, Src: b8:27:eb:95:47:39, Dst: 28:d2:44:19:83:95
Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.1.11
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol

Status-Line: SIP/2.0 200 OK

Message Header

Via: SIP/2.0/UDP 192.168.1.51:5060;
branch=z9hG4bK1e078835-ae2-e411-8843-28d244198395;
received=192.168.1.51;rport=5060
From: <sip:6001@192.168.1.10>;tag=6e088735-ae2-e411-8843-28d244198395
To: <sip:6001@192.168.1.10>;tag=as2ef2ea7f
Call-ID: 36fa8635-ae2-e411-8843-28d244198395@riomhaire-0B
CSeq: 2 REGISTER
Server: Asterisk PBX
Allow: INVITE,ACK,CANCEL,OPTIONS,BYE,REFER,SUBSCRIBE,NOTIFY,
INFO,PUBLISH
Supported: replaces, timer
Expires: 3600
Contact: <sip:6001@192.168.1.51:5060>;expires=3600
Date: Wed, 08 Apr 2015 21:39:47 GMT
Content-Length: 0

Following the registration the Ekiga Softphone attempts to PUBLISH a service. The service is defined as eXtensible Markup Language (XML).

```
Frame 5
Ethernet II, Src: 28:d2:44:19:83:95, Dst: b8:27:eb:95:47:39
Internet Protocol Version 4, Src: 192.168.1.11, Dst: 192.168.1.10
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
Request-Line: PUBLISH sip:6001@192.168.1.10 SIP/2.0
Message Header
  CSeq: 3 PUBLISH
  Via: SIP/2.0/UDP 192.168.1.51:5060;
      branch=z9hG4bK3a538935-ae2-e411-8843-28d244198395;rport
  User-Agent: Ekiga Softphone
  From: <sip:6001@192.168.1.10>
  Call-ID: f8448935-ae2-e411-8843-28d244198395@riomhaire-0B
  To: <sip:6001@192.168.1.10>
  Expires: 300
  Event: presence
  Content-Length: 486
  Content-Type: application/pidf+xml
  Max-Forwards: 70
Message Body
eXtensible Markup Language
  <?xml
    version="1.0"
    encoding="UTF-8"
  ?>
  <presence
    xmlns="urn:ietf:params:xml:ns:pidf"
    xmlns:dm="urn:ietf:params:xml:ns:pidf:data-model"
    xmlns:rpid="urn:ietf:params:xml:ns:pidf:rpid"
    entity="pres:6001@192.168.1.10">
    <tuple
      id="TB799B8A0">
      <status>
        <basic>
          open
        </basic>
      </status>
      <contact
        priority="1">
        sip:6001@192.168.1.10
      </contact>
      <note>
        I&apos;m available using Ekiga
      </note>
      <timestamp>
        2015-04-16T14:57:37+01:00
      </timestamp>
    </tuple>
  </presence>
```

The SUBSCRIBE message is used to connect to a published service.

```
Frame 6
Ethernet II, Src: 28:d2:44:19:83:95, Dst: b8:27:eb:95:47:39
Internet Protocol Version 4, Src: 192.168.1.11, Dst: 192.168.1.10 (192.168.1.10)
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Request-Line: SUBSCRIBE sip:6001@192.168.1.10 SIP/2.0
    Method: SUBSCRIBE
    Request-URI: sip:6001@192.168.1.10
  Message Header
    CSeq: 1 SUBSCRIBE
    Via: SIP/2.0/UDP 192.168.1.51:5060;
        branch=z9hG4bKf4578935-ae2-e411-8843-28d244198395;rport
    User-Agent: Ekiga/4.0.1
    From: <sip:6001@192.168.1.10>;tag=94538935-ae2-e411-8843-28d244198395
    Call-ID: ce3c8935-ae2-e411-8843-28d244198395@riomhaire-0B
    To: <sip:6001@192.168.1.10>
    Accept: application/simple-message-summary
    Contact: <sip:6001@192.168.1.51:5060>
    Allow: INVITE,ACK,OPTIONS,BYE,CANCEL,SUBSCRIBE,NOTIFY,REFER,MESSAGE,
        INFO,PING,PRACK
    Expires: 3600
    Event: message-summary
    Content-Length: 0
    Max-Forwards: 70
```

9.13.3 Voice call test

The IP Phone, a *Linksys/SPA941* number *6002* sends a SIP INVITE method to the SIP Proxy Server requesting a session with *6003*. It is willing to establish the session for any of the CODECs:

- PCMU/8000
- G726-32/8000
- G723/8000
- PCMA/8000
- G729a/8000
- G726-40/8000
- G726-24/8000
- G726-16/8000

```
Frame 1
Ethernet II, Src: 00:0e:08:d2:e8:2b, Dst: b8:27:eb:95:47:39
Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.10
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Request-Line: INVITE sip:6003@192.168.1.10 SIP/2.0
  Message Header
    Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG4bK-8b01725e
    From: "6002" <sip:6002@192.168.1.10>;tag=3996f9ed38e5b1100
    To: "6003" <sip:6003@192.168.1.10>
    Call-ID: d36d581d-f5a812a1@192.168.1.12
    CSeq: 101 INVITE
    Max-Forwards: 70
    Contact: "6002" <sip:6002@192.168.1.12:5060>
    Expires: 240
    User-Agent:
    Content-Length: 391
    Allow: ACK, BYE, CANCEL, INFO, INVITE, NOTIFY, OPTIONS, REFER
    Supported: replaces
    Content-Type: application/sdp
  Message Body
    Session Description Protocol
      Session Description Protocol Version (v): 0
      Owner/Creator, Session Id (o): - 8603 8603 IN IP4 192.168.1.12
      Session Name (s): -
      Connection Information (c): IN IP4 192.168.1.12
      Time Description, active time (t): 0 0
      Media Description, name and address (m): audio 16454 RTP/AVP 0 2 4
      Media Attribute (a): rtpmap:0 PCMU/8000 8 18 96 97 98 101
      Media Attribute (a): rtpmap:2 G726-32/8000
      Media Attribute (a): rtpmap:4 G723/8000
      Media Attribute (a): rtpmap:8 PCMA/8000
      Media Attribute (a): rtpmap:18 G729a/8000
      Media Attribute (a): rtpmap:96 G726-40/8000
      Media Attribute (a): rtpmap:97 G726-24/8000
      Media Attribute (a): rtpmap:98 G726-16/8000
      Media Attribute (a): rtpmap:101 telephone-event/8000
      Media Attribute (a): fmp:101 0-15
      Media Attribute (a): pt:30
      Media Attribute (a): sendrecv
```

The SIP Proxy Server, an Asterisk PBX responds with a *401 UNAUTHORISED* message and an MD5 challenge for the nonce *0b543a15*.

```
Frame 2
Ethernet II, Src: b8:27:eb:95:47:39, Dst: 00:0e:08:d2:e8:2b
Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.1.12
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Status-Line: SIP/2.0 401 Unauthorised
  Message Header
    Via: SIP/2.0/UDP 192.168.1.12:5060;
      branch=z9hG4bK-8b01725e;received=192.168.1.12;rport=5060
    From: "6002" <sip:6002@192.168.1.10>;tag=3996f9ed38e5b1100
    To: "6003" <sip:6003@192.168.1.10>;tag=as1ad8b26f
    Call-ID: d36d581d-f5a812a1@192.168.1.12
    CSeq: 101 INVITE
    Server: Asterisk PBX
    Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY,
      INFO, PUBLISH
    Supported: replaces, timer
    WWW-Authenticate: Digest algorithm=MD5, realm="asterisk", nonce="0b543a15"
    Content-Length: 0
```

The IP Phone responds with an ACK method.

```
Frame 3
Ethernet II, Src: 00:0e:08:d2:e8:2b, Dst: b8:27:eb:95:47:39
Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.10
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Request-Line: ACK sip:6003@192.168.1.10 SIP/2.0
  Message Header
    Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG4bK-8b01725e
    From: "6002" <sip:6002@192.168.1.10>;tag=3996f9ed38e5b1100
    To: "6003" <sip:6003@192.168.1.10>;tag=as1ad8b26f
    Call-ID: d36d581d-f5a812a1@192.168.1.12
    CSeq: 101 ACK
    Max-Forwards: 70
    Contact: "6002" <sip:6002@192.168.1.12:5060>
    User-Agent: Linksys/SPA941
    Content-Length: 0
```

The IP Phone then re-sends the INVITE method including a response to the authentication challenge in Frame 2.

```
Frame 4
Ethernet II, Src: 00:0e:08:d2:e8:2b, Dst: b8:27:eb:95:47:39
Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.10
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Request-Line: INVITE sip:6003@192.168.1.10 SIP/2.0
  Message Header
    Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG4bK-db213ea6
    From: "6002" <sip:6002@192.168.1.10>;tag=3996f9ed38e5b1100
    To: "6003" <sip:6003@192.168.1.10>
    Call-ID: d36d581d-f5a812a1@192.168.1.12
    CSeq: 102 INVITE
    Max-Forwards: 70
    Authorization: Digest username="6002",realm="asterisk",nonce="0b543a15",
                  uri="sip:6003@192.168.1.10",algorithm=MD5,
                  response="e65f2c2bcac5a1f58148bb2362d0a88b"
    Contact: "6002" <sip:6002@192.168.1.12:5060>
    Expires: 240
    User-Agent: Linksys/SPA941
    Content-Length: 391
    Allow: ACK, BYE, CANCEL, INFO, INVITE, NOTIFY, OPTIONS, REFER
    Supported: replaces
    Content-Type: application/sdp
  Message Body
    Session Description Protocol
      Session Description Protocol Version (v): 0
      Owner/Creator, Session Id (o): - 8603 8603 IN IP4 192.168.1.12
      Session Name (s): -
      Connection Information (c): IN IP4 192.168.1.12
      Time Description, active time (t): 0 0
      Media Description, name and address (m): audio 16454 RTP/AVP 0 2 4
      Media Attribute (a): rtpmap:0 PCMU/8000          8 18 96 97 98 101
      Media Attribute (a): rtpmap:2 G726-32/8000
      Media Attribute (a): rtpmap:4 G723/8000
      Media Attribute (a): rtpmap:8 PCMA/8000
      Media Attribute (a): rtpmap:18 G729a/8000
      Media Attribute (a): rtpmap:96 G726-40/8000
      Media Attribute (a): rtpmap:97 G726-24/8000
      Media Attribute (a): rtpmap:98 G726-16/8000
      Media Attribute (a): rtpmap:101 telephone-event/8000
      Media Attribute (a): fmp:101 0-15
      Media Attribute (a):ptime:30
      Media Attribute (a): sendrecv
```

As the authentication passes the SIP Proxy Server sends a *100 TRYING* message to the originating IP Phone.

```
Frame 5
Ethernet II, Src: b8:27:eb:95:47:39, Dst: 00:0e:08:d2:e8:2b
Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.1.12
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Status-Line: SIP/2.0 100 Trying
  Message Header
    Via: SIP/2.0/UDP 192.168.1.12:5060;
        branch=z9hG4bK-db213ea6;received=192.168.1.12;rport=5060
    From: "6002" <sip:6002@192.168.1.10>;tag=3996f9ed38e5b1100
    To: "6003" <sip:6003@192.168.1.10>
    Call-ID: d36d581d-f5a812a1@192.168.1.12
    CSeq: 102 INVITE
    Server: Asterisk PBX
    Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY,
          INFO, PUBLISH
    Supported: replaces, timer
    Contact: <sip:6003@192.168.1.10:5060>
    Content-Length: 0
```

The SIP Proxy forwards the INVITE method to the destination IP Phone 6003, replacing the *Via*: with its own line. (In some implementations the SIP Proxy Server adds a *Via*: header above the original *Via*: header).

```
Frame 6
Ethernet II, Src: b8:27:eb:95:47:39, Dst: 00:0e:08:d2:e8:2a
Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.1.13
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Request-Line: INVITE sip:6003@192.168.1.13:5060 SIP/2.0
  Method: INVITE
  Request-URI: sip:6003@192.168.1.13:5060
  Message Header
    Via: SIP/2.0/UDP 192.168.1.10:5060;branch=z9hG4bK5c5878cc;rport
    Max-Forwards: 70
    From: "6002" <sip:6002@192.168.1.10>;tag=as2bb1ea4b
    To: <sip:6003@192.168.1.13:5060>
    Contact: <sip:6002@192.168.1.10:5060>
    Call-ID: 2f6514fc5ee9128b21d27e8d2ca276cd@192.168.1.10:5060
    CSeq: 102 INVITE
    User-Agent: Asterisk PBX
    Date: Wed, 08 Apr 2015 22:05:04 GMT
    Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY,
          INFO, PUBLISH
    Supported: replaces, timer
    Content-Type: application/sdp
    Content-Length: 297
  Message Body
    Session Description Protocol
      Session Description Protocol Version (v): 0
      Owner/Creator, Session Id (o): root 689995268 689995268
                                   IN IP4 192.168.1.10
      Session Name (s): Asterisk PBX
      Connection Information (c): IN IP4 192.168.1.10
      Time Description, active time (t): 0 0
      Media Description, name and address (m): audio 12968 RTP/AVP 0 3 8 101
      Media Attribute (a): rtpmap:0 PCMU/8000
      Media Attribute (a): rtpmap:3 GSM/8000
      Media Attribute (a): rtpmap:8 PCMA/8000
      Media Attribute (a): rtpmap:101 telephone-event/8000
      Media Attribute (a): fmtp:101 0-16
      Media Attribute (a):ptime:20
      Media Attribute (a):sendrecv
```

The IP Phone 6003 responds with a *100 TRYING* message which is followed by a *180 RINGING* message when the IP Phone starts ringing.

```
Frame 7
Ethernet II, Src: 00:0e:08:d2:e8:2a, Dst: b8:27:eb:95:47:39
Internet Protocol Version 4, Src: 192.168.1.13, Dst: 192.168.1.10
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Status-Line: SIP/2.0 100 Trying
  Message Header
    To: <sip:6003@192.168.1.13:5060>
    From: "6002" <sip:6002@192.168.1.10>;tag=as2bb1ea4b
    Call-ID: 2f6514fc5ee9128b21d27e8d2ca276cd@192.168.1.10:5060
    CSeq: 102 INVITE
    Via: SIP/2.0/UDP 192.168.1.10:5060;branch=z9hG4bK5c5878cc
    Server: Linksys/SPA941
    Content-Length: 0
```

When answered the IP Phone 6003 sends a *200 OK* message to indicate the receiver has gone off hook and it receives a *ACK* method in acknowledgement.

```
Frame 8
Ethernet II, Src: 00:0e:08:d2:e8:2a, Dst: b8:27:eb:95:47:39
Internet Protocol Version 4, Src: 192.168.1.13, Dst: 192.168.1.10
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Status-Line: SIP/2.0 200 OK
  Message Header
    To: <sip:6003@192.168.1.13:5060>;tag=20b726cc77a8f284i0
    From: "6002" <sip:6002@192.168.1.10>;tag=as2bb1ea4b
    Call-ID: 2f6514fc5ee9128b21d27e8d2ca276cd@192.168.1.10:5060
    CSeq: 102 INVITE
    Via: SIP/2.0/UDP 192.168.1.10:5060;branch=z9hG4bK5c5878cc
    Contact: "6003" <sip:6003@192.168.1.13:5060>
    Server: Linksys/SPA941
    Content-Length: 208
    Allow: ACK, BYE, CANCEL, INFO, INVITE, NOTIFY, OPTIONS, REFER
    Supported: replaces
    Content-Type: application/sdp
  Message Body
    Session Description Protocol
      Session Description Protocol Version (v): 0
      Owner/Creator, Session Id (o): - 2332769 2332769 IN IP4 192.168.1.13
      Session Name (s): -
      Connection Information (c): IN IP4 192.168.1.13
      Time Description, active time (t): 0 0
      Media Description, name and address (m): audio 16398 RTP/AVP 0 101
      Media Attribute (a): rtpmap:0 PCMU/8000
      Media Attribute (a): rtpmap:101 telephone-event/8000
      Media Attribute (a): fmtp:101 0-15
      Media Attribute (a):ptime:30
      Media Attribute (a): sendrecv
```

```
Frame 9
Ethernet II, Src: b8:27:eb:95:47:39, Dst: 00:0e:08:d2:e8:2a
Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.1.13
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Request-Line: ACK sip:6003@192.168.1.13:5060 SIP/2.0
  Message Header
    Via: SIP/2.0/UDP 192.168.1.10:5060;branch=z9hG4bK67b7e50e;rport
    Max-Forwards: 70
    From: "6002" <sip:6002@192.168.1.10>;tag=as2bb1ea4b
    To: <sip:6003@192.168.1.13:5060>;tag=20b726cc77a8f284i0
    Contact: <sip:6002@192.168.1.10:5060>
    Call-ID: 2f6514fc5ee9128b21d27e8d2ca276cd@192.168.1.10:5060
    CSeq: 102 ACK
    User-Agent: Asterisk PBX
    Content-Length: 0
```

The SIP Proxy Server forwards the *200 OK* message to the originator IP Phone *6002* to confirm the other IP Phone *6003* has gone off hook and received an *ACK* method in receipt. A RTP media stream can now flow between *6002* and *6003*.

```
Frame 10
Ethernet II, Src: b8:27:eb:95:47:39, Dst: 00:0e:08:d2:e8:2b
Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.1.12
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Status-Line: SIP/2.0 200 OK
  Message Header
    Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG4bK-
db213ea6;received=192.168.1.12;rport=5060
    From: "6002" <sip:6002@192.168.1.10>;tag=3996f9ed38e5b1100
    To: "6003" <sip:6003@192.168.1.10>;tag=as6f48c732
    Call-ID: d36d581d-f5a812a1@192.168.1.12
    CSeq: 102 INVITE
    Server: Asterisk PBX
    Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY,
      INFO, PUBLISH
    Supported: replaces, timer
    Contact: <sip:6003@192.168.1.10:5060>
    Content-Type: application/sdp
    Content-Length: 276
  Message Body
    Session Description Protocol
      Session Description Protocol Version (v): 0
      Owner/Creator, Session Id (o): root 1936169497 1936169497
        IN IP4 192.168.1.10
      Session Name (s): Asterisk PBX
      Connection Information (c): IN IP4 192.168.1.10
      Time Description, active time (t): 0 0
      Media Description, name and address (m): audio 16186 RTP/AVP 0 8 101
      Media Attribute (a): rtpmap:0 PCMU/8000
      Media Attribute (a): rtpmap:8 PCMA/8000
      Media Attribute (a): rtpmap:101 telephone-event/8000
      Media Attribute (a): fmp:101 0-16
      Media Attribute (a):ptime:20
      Media Attribute (a): sendrecv

Frame 11
Ethernet II, Src: 00:0e:08:d2:e8:2b, Dst: b8:27:eb:95:47:39
Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.10
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Message Header
    Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG4bK-7ae9b810
    From: "6002" <sip:6002@192.168.1.10>;tag=3996f9ed38e5b1100
    To: "6003" <sip:6003@192.168.1.10>;tag=as6f48c732
    Call-ID: d36d581d-f5a812a1@192.168.1.12
    CSeq: 102 ACK
    Max-Forwards: 70
    Authorization: Digest username="6002", realm="asterisk", nonce="0b543a15",
      uri="sip:6003@192.168.1.10", algorithm=MD5,
      response="e65f2c2bcac5a1f58148bb2362d0a88b"
    Contact: "6002" <sip:6002@192.168.1.12:5060>
    User-Agent: Linksys/SPA941
    Content-Length: 0
```

9.13.4 Hangup test

Place the IP Phone 6002 receiver on-hook so the device sends a *BYE* method to the SIP Proxy Server for Call ID *2f6514fc5ee9128b21d27e8d2ca276cd*.

```

Frame 1
Ethernet II, Src: 00:0e:08:d2:e8:2b, Dst: b8:27:eb:95:47:39
Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.10
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Request-Line: BYE sip:6003@192.168.1.10:5060 SIP/2.0
  Message Header
    Via: SIP/2.0/UDP 192.168.1.12:5060;branch=z9hG4bK-55a96bae
    From: "6002" <sip:6002@192.168.1.10>;tag=3996f9ed38e5b1100
    To: "6003" <sip:6003@192.168.1.10>;tag=as6f48c732
    Call-ID: d36d581d-f5a812a1@192.168.1.12
    CSeq: 103 BYE
    Max-Forwards: 70
    Authorization: Digest username="6002",realm="asterisk",nonce="0b543a15",
                  uri="sip:6003@192.168.1.10:5060",algorithm=MD5,
                  response="8cb8308cde00ab5b64d60e754fe70d0a"
    User-Agent: Linksys/SPA941
    Content-Length: 0

```

The SIP Proxy Server acknowledges this by sending a *200 OK* message back in response.

```

Frame 2
Ethernet II, Src: b8:27:eb:95:47:39, Dst: 00:0e:08:d2:e8:2b
Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.1.12
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Status-Line: SIP/2.0 200 OK
  Message Header
    Via: SIP/2.0/UDP 192.168.1.12:5060;
        branch=z9hG4bK-55a96bae;received=192.168.1.12;rport=5060
    From: "6002" <sip:6002@192.168.1.10>;tag=3996f9ed38e5b1100
    To: "6003" <sip:6003@192.168.1.10>;tag=as6f48c732
    Call-ID: d36d581d-f5a812a1@192.168.1.12
    CSeq: 103 BYE
    Server: Asterisk PBX
    Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY,
          INFO, PUBLISH
    Supported: replaces, timer
    Content-Length: 0

```

The SIP Proxy Server sends the *BYE* method to the IP Phone 6003 to inform it that 6002 has gone on-hook.

```
Frame 3
Ethernet II, Src: b8:27:eb:95:47:39, Dst: 00:0e:08:d2:e8:2a
Internet Protocol Version 4, Src: 192.168.1.10, Dst: 192.168.1.13
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Request-Line: BYE sip:6003@192.168.1.13:5060 SIP/2.0
  Message Header
    Via: SIP/2.0/UDP 192.168.1.10:5060;branch=z9hG4bK0ff44bda;rport
    Max-Forwards: 70
    From: "6002" <sip:6002@192.168.1.10>;tag=as2bb1ea4b
    To: <sip:6003@192.168.1.13:5060>;tag=20b726cc77a8f284i0
    Call-ID: 2f6514fc5ee9128b21d27e8d2ca276cd@192.168.1.10:5060
    CSeq: 105 BYE
    User-Agent: Asterisk PBX
    X-Asterisk-HangupCause: Normal Clearing
    X-Asterisk-HangupCauseCode: 16
    Content-Length: 0
```

The IP Phone 6003 acknowledges this to the SIP Proxy and goes on-hook. Call is completed.

```
Frame 4
Ethernet II, Src: 00:0e:08:d2:e8:2a, Dst: b8:27:eb:95:47:39
Internet Protocol Version 4, Src: 192.168.1.13, Dst: 192.168.1.10
User Datagram Protocol, Src Port: sip (5060), Dst Port: sip (5060)
Session Initiation Protocol
  Status-Line: SIP/2.0 200 OK
  Message Header
    To: <sip:6003@192.168.1.13:5060>;tag=20b726cc77a8f284i0
    From: "6002" <sip:6002@192.168.1.10>;tag=as2bb1ea4b
    Call-ID: 2f6514fc5ee9128b21d27e8d2ca276cd@192.168.1.10:5060
    CSeq: 105 BYE
    Via: SIP/2.0/UDP 192.168.1.10:5060;branch=z9hG4bK0ff44bda
    Server: Linksys/SPA941
    Content-Length: 0
```

9.14 Asterisk GUI

For more complex configurations Asterisk has a configuration ready-made GUI.

9.14.1 Install Asterisk-gui

The GNU/Linux subversion (SVN) version control system client is required to download the latest version of the *asterisk-gui* from the Digium website.

```
$ sudo apt-get install subversion
```

Download the latest source to the source directory on the asterisk server.

```
$ cd /usr/src
```

```
$ sudo svn checkout http://svn.digium.com/svn/asterisk-gui/trunk asterisk-gui
```

Run the *configure* script to prepare for the system.

```
$ cd /usr/src/asterisk-gui
$ sudo ./configure
```

```

                .$$$$$$$$$$$$$$$$$=..
                .7$7$7..           .7$7$7:..
                .$$:..             ,7$7.7
                .7$.           7$$$$$           .$$$77
                ..$$           $$$$$$           .$$$7
                ..7$ .?. $$$$$$ .?. 7$$$$.
                $.$. .$$$7. $$$$7 .7$$$$. .$$$
                .777. .$$$$$77$$$$77$$$$$7. $$$,
                $$$~ .7$$$$$$$$$$$$$7. .$$$
                .$$7           .7$$$$$$$$$7:   ?$$$
                $$$           ?7$$$$$$$$$$$$I   .$$$7
                $$$           .7$$$$$$$$$$$$$$$$ :$$$
                $$$           $$$$$$7$$$$$$$$$$$$ .$$$
                $$$           $$$ 7$$$$7 .$$$ .$$$
                $$$           $$$$7           .$$$
                7$$$$7           7$$$$$           7$$$
                $$$$$           $$$           $$$
                $$$$7.           $$ (TM)
                $$$$$$.           .7$$$$$$$ $$
                $$$$$$$$$$$$$7$$$$$$$$$. $$$$$$
                $$$$$$$$$$$$$$$$$$.

```

```
configure: Package configured for:
configure: OS type : linux-gnu
configure: Host CPU : armv7l
```

Use the *make* utility to determine what needs to be recompiled and recompile as required. The *make install* copies files into the appropriate directories on the system.

```
$ sudo make
```

```
+----- Asterisk-GUI Build Complete -----+
+ Asterisk-GUI has successfully been built, +
+ and can be installed by running:         +
+                                           +
+             make install                  +
+-----+
```

```
$ sudo make install
```

```
+---- Asterisk GUI Installation Complete ---+
+
+   YOU MUST READ THE SECURITY DOCUMENT   +
+
+ Asterisk-GUI has successfully been      +
+ installed.                               +
+                                           +
+-----+
+
+           BEFORE THE GUI WILL WORK      +
+
+ Before the GUI will run, you must perform +
+ some modifications to the Asterisk      +
+ configuration files in accordance with  +
+ the README file. When done, you can    +
+ check your changes by doing:           +
+                                           +
+             make checkconfig            +
+-----+
```

9.14.2 Asterisk configuration files

Edit the file Asterisk *httpd.conf* to uncomment the *enabled*, *enabledstatic* as well as change the *bindaddr=127.0.0.1* to *bindaddr=0.0.0.0*.

```
$ sudo vi /etc/asterisk/http.conf
enabled=yes
enablestatic=yes

bindaddr=0.0.0.0
```

Edit the Asterisk *manager.conf* file to look like this.

```
$ sudo vi /etc/asterisk/manager.conf

[general]
enabled = yes
webenabled = yes
port = 5038
bindaddr = 0.0.0.0

#include "manager.d/*.conf"
```

Create a configuration file for the admin user.

```
$ sudo -s
# cat << EOM >> /etc/asterisk/manager.d/admin.conf
[admin]
secret = voippass
read = system,call,log,verbose,command,agent,config,read,write,originate
write = system,call,log,verbose,command,agent,config,read,write,originate
EOM
# exit
$
```

Map the */var/lib/asterisk/static-http/* directory, the location of the *index.html* file to the */usr/share/asterisk/* directory which is the root directory of the webserver. Change the ownership of the Asterisk library files to that of *asterisk* and group *asterisk*.

```
$ sudo rm -rf /usr/share/asterisk/static-http/
$ sudo ln -s /var/lib/asterisk/static-http/ /usr/share/asterisk/
$ cd /var/lib/asterisk
$ sudo chown -R asterisk.asterisk static-http
$ sudo chown -R asterisk.asterisk scripts
$ sudo chown -R asterisk.asterisk gui_backups
```

Give ownership of the Asterisk configuration files to user *asterisk*, group *asterisk* also.

```
$ sudo chown asterisk:asterisk /etc/asterisk/*.conf
$ sudo chmod 644 /etc/asterisk/*.conf
$ sudo chown asterisk:asterisk /etc/asterisk/manager.d/*.conf
$ sudo chmod 644 /etc/asterisk/manager.d/*.conf
```

Restart the Asterisk server to force a re-read of the configuration files.

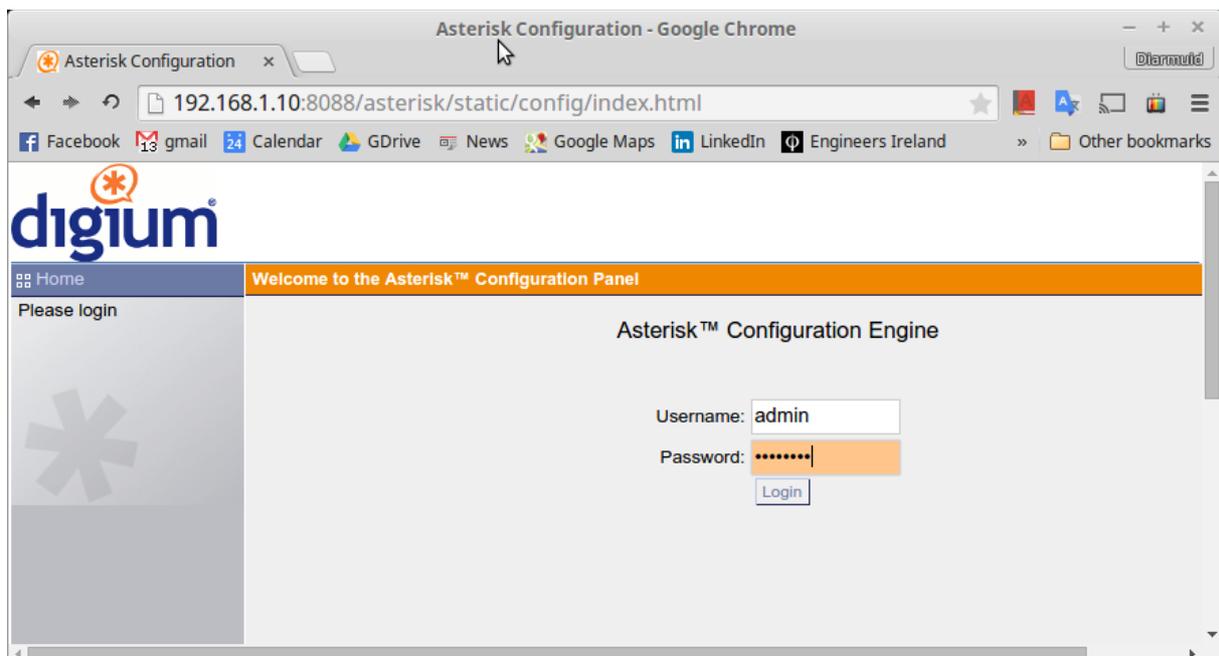
```
$ sudo service asterisk restart
Stopping Asterisk PBX: asterisk.
Starting Asterisk PBX: asterisk.
```

9.14.3 Connect to the Asterisk Server GUI

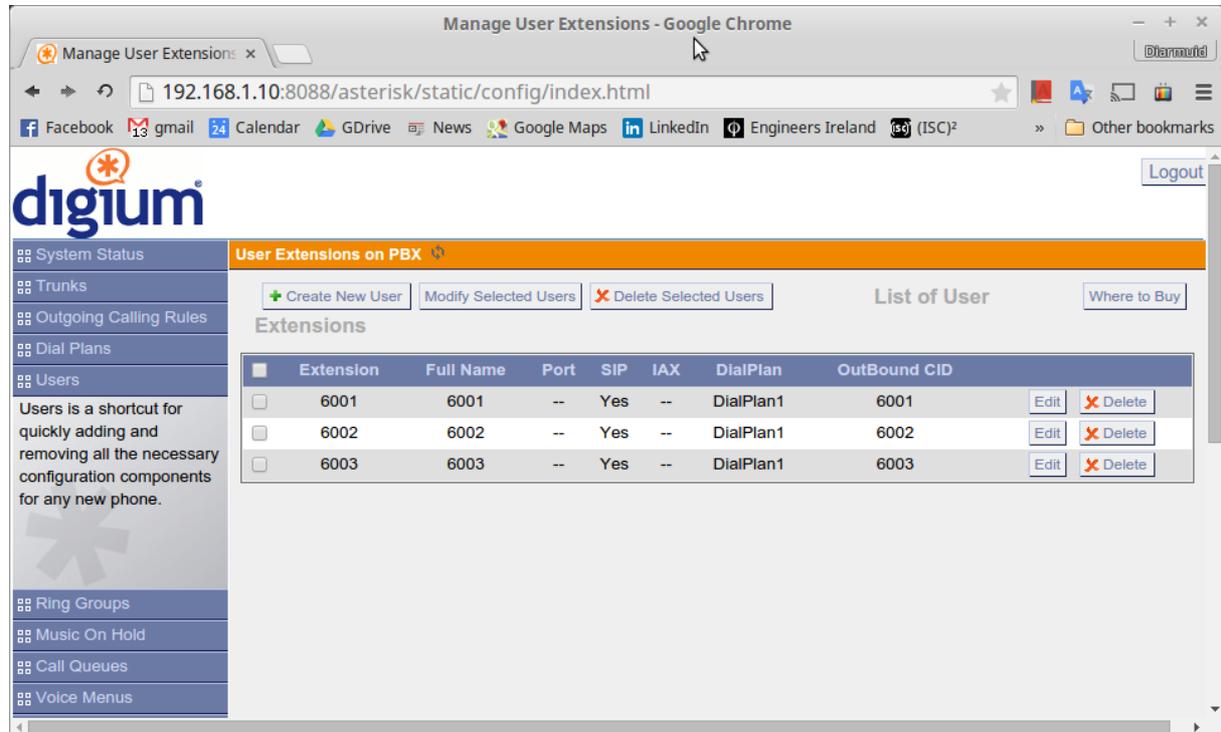
<http://<Server IP Address>:8088/asterisk/static/config/index.html>

Username: **admin**

Password: **voippass**



The configuration of the Asterisk Server using the GUI and the full range of its capabilities is outside the scope of this document. The diagram below shows the configuration of 3 user extensions similar to that established manually in the example.



The screenshot shows the 'Manage User Extensions' interface in a Google Chrome browser. The address bar displays '192.168.1.10:8088/asterisk/static/config/index.html'. The page features the Digium logo and a sidebar with navigation options: System Status, Trunks, Outgoing Calling Rules, Dial Plans, Users, Ring Groups, Music On Hold, Call Queues, and Voice Menus. The main content area is titled 'User Extensions on PBX' and includes buttons for 'Create New User', 'Modify Selected Users', and 'Delete Selected Users'. Below these is a table of extensions:

Extension	Full Name	Port	SIP	IAX	DialPlan	OutBound CID
6001	6001	--	Yes	--	DialPlan1	6001
6002	6002	--	Yes	--	DialPlan1	6002
6003	6003	--	Yes	--	DialPlan1	6003

9.15 Conclusion

IPT and Softswitch is a branch of Networking Engineering in its own right. It not alone requires skills in IP networking but an understanding of traditional telecommunications. This serves as an introduction to the topic, for a more complete understanding it is necessary to look further afield. The following websites are great places to explore the topic further.

<http://www.asterisk.org/>

<http://www.voip-info.org/>

<http://www.digium.com/>

10. IP Services

10.1 Configuration of `inetd` or `xinetd`

The next step in the configuration of the network is to configure the servers and services that will allow another user to access the local machine or its services. The server programs will use the ports to listen to the requests from the clients, which will be sent to this service as *IP:port*. The servers may work in two different ways: standalone (in which the service listens to the assigned port and is always active) or through the Internet Service Daemon (*inetd*).

inetd often called the Internet Super Server daemon controls and manages the network connections of the services specified in the `/etc/inetd.conf` file. *inetd* is a server of servers. It executes rarely used servers on demand, to conserve system resources by avoiding to fork a lot of processes which might lie dormant for most of their lifetime.. When a service request is made, starts up the appropriate server and transfers the request. After some concerns regarding security the *eXtended InterNET Daemon* (*xinetd*) evolved as an alternative with the configuration files located in `/etc/xinetd.conf`.

Two important files must be configured: `/etc/services` and `/etc/inetd.conf` or `/etc/xinetd.conf`. In the first file, the services, the ports and the protocol are associated, and in the second, the server programs that will respond to a request to a determined port. The `/etc/services` format is *name port/protocol aliases*, where the first field is the service name, the second is the port where the service is attended and the protocol that it uses, and the next field is an alias of the name. There is a series of default pre-configured services.

`/etc/services`

```

tcpmux      1/tcp                # TCP port service multiplexer
echo        7/tcp
echo        7/udp
discard     9/tcp          sink null
discard     9/udp          sink null
sysstat     11/tcp         users
daytime     13/tcp
daytime     13/udp
netstat     15/tcp
qotd        17/tcp          quote
msp         18/tcp          # message send protocol
msp         18/udp
chargen     19/tcp          ttytst source
chargen     19/udp          ttytst source
ftp-data    20/tcp
ftp         21/tcp
fsp         21/udp          fspd
ssh         22/tcp          # SSH Remote Login Protocol
ssh         22/udp
telnet      23/tcp
smtp        25/tcp          mail
time        37/tcp          timserver
time        37/udp          timserver
...

```

The `/etc/inetd.conf` file is the configuration for the master network service (`inetd` server daemon). Each line contains seven fields separated by spaces: `service socket_type proto flags user server_path server_args`, where:

- `service`: is the service described in the first column in `/etc/services`
- `socket_type`: is the type of socket (possible values are `stream`, `dgram`, `raw`, `rdm`, or `seqpacket`)
- `proto`: is the protocol that is valid for this input (it must match that in `/etc/services`)
- `flags`: indicates the action that should be taken when there is a new connection on a service that is attending another connection, (`wait` tells `inetd` not to start up a new server or `nowait` means that `inetd` must start up a new server)
- `user`: will be the local user-name with which the client that has started up the service is identified
- `server_path`: is the directory where the server is located
- `server_args`: are possible arguments that will be passed to the server

An example of some `/etc/inetd.conf` lines is (`#` is a comment, so if a service has `#` before the name, it means that it is not available):

`/etc/inetd.conf`

```
...
telnet stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.telnetd
ftp stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.ftpd
# fsp dgram udp wait root /usr/sbin/tcpd /usr/sbin/in.fspd
shell stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.rshd
login stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.rlogind
# exec stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.rexecd...
...
```

The `/etc/xinetd.conf` configuration file essentially calls in the files in the `/etc/xinetd.d` directory with the line `includedir /etc/xinetd.d`. Each file is a description of a single service and take the form:

```
service <service_name>
{
    <attribute> <assign_op> <value> <value> ...
    ...
}
```

In the example the default Debian GNU/Linux *xinetd* files are displayed with the *time* service file expanded.

```
$ ls /etc/xinetd.d
```

```
chargen  daytime  discard  echo  time
```

```
$ cat /etc/xinetd.d/time
```

```
# default: off
# description: An RFC 868 time server. This protocol provides a
# site-independent, machine readable date and time. The Time service sends back
# to the originating source the time in seconds since midnight on January first
# 1900.
# This is the tcp version.

service time
{
    disable          = yes
    type             = INTERNAL
    id               = time-stream
    socket_type      = stream
    protocol         = tcp
    user             = root
    wait            = no
}

# This is the udp version.
service time
{
    disable          = yes
    type             = INTERNAL
    id               = time-dgram
    socket_type      = dgram
    protocol         = udp
    user             = root
    wait            = yes
}
```

10.2 Other network services

Apart from the *inetd/xinetd* configuration, the typical configuration of network services in a desktop or basic server environment might also include (some of these services will be examined in the chapter on servers):

- *ssh*: secure interactive connection to replace telnet that includes two configuration files */etc/ssh/ssh_config* (for the client) and */etc/ssh/sshd_config* (for the server)
- *exim*: Mail Transfer Agent (MTA), includes configuration files: */etc/exim/exim.conf*, */etc/mailname*, */etc/aliases*, */etc/email-addresses*.
- *fetchmail*: daemon for downloading the mail from a POP3 account, */etc/fetchmailrc*
- *procmail*: program for filtering and distributing local mail, *~/.procmailrc*
- *tcpd*: Filtering services for enabled and disabled machines and domains for connecting to the server (wrappers): */etc/hosts.allow*, */etc/hosts.deny*
- *DHCP*: Service for managing (server) or obtaining an IP (client), */etc/dhcp3/dhclient.conf* (client), */etc/default/dhcp3-server* (server), */etc/dhcp3/dhcpd.conf* (server)
- *CVS*: system for managing concurrent versions, */etc/cvs-cron.conf*, */etc/cvs-pserver.conf*
- *NFS*: network file system, */etc/exports*
- *Samba*: network file system and sharing printers in Windows networks, */etc/samba/smb.conf*
- *lpr*: daemon for the printing system, */etc/printcap* (for the Ipr system -not CUPS-)
- *Apache2*: Web Server, */etc/apache2/**
- *squid*: Server proxy-cache, */etc/squid/**

10.2.1 Additional configuration: protocols and networks

There are other configuration files that are hardly ever used, but that can be interesting. The */etc/protocols* is a file that shows the protocol identifiers with the protocol names; in this way, programmers can specify the protocols by their names in the programs.

/etc/protocols

```
ip 0 IP # internet protocol, pseudo protocol number
#hopopt 0 HOPOPT # IPv6 Hop-by-Hop Option [RFC1883]
icmp 1 ICMP # internet control message protocol
igmp 2 IGMP # Internet Group Management
ggp 3 GGP # gateway-gateway protocol
ipencap 4 IP-ENCAP # IP encapsulated in IP (officially ``IP'')
```

...

The */etc/networks* file has a function similar to */etc/hosts*, but where the networks are concerned, it shows the network names in relation to its IP address (the *route* command will show the name of the network and not its address in this case).

/etc/networks

default	0.0.0.0
loopback	127.0.0.0
link-local	169.254.0.0

10.2.2 Security aspects

It is important to take into account the security aspects in network connections, as a significant amount of attacks occur through the network. We will discuss this subject in more detail in the unit on security; however, there are some basic recommendations that should be taken into account in order to minimise the risks immediately before and after configuring the network in our computer:

- Do not activate services in */etc/inetd.conf*, or include service files in *xinetd.d* that will not be used, insert an *#* before the name to avoid sources of risk
- Modify the */etc/ftpusers* file to deny access to certain users who may have an FTP connection to your machine
- Modify the */etc/securetty* file which lists ttys from which root can log in, to indicate from which terminals (a name per line), for example: *tty1 tty2 tty3 tty4*, it will be possible for the root superuser to connect. The root superuser will not be able to connect from any of the remaining terminals
- Use the *tcpd* program. This server is a wrapper that makes it possible to allow/deny a service from a given node and it is placed in */etc/inetd.conf* as a service intermediary. The *tcpd* verifies certain access rules in two files: */etc/hosts.allow* and */etc/host.deny*
 - If the connection is accepted, it starts up an appropriate service passed as an argument (for example, the FTP service line shown earlier in *inetd.conf*:

```
ftp stream tcp nowait root /usr/sbin/tcpd/usr/sbin/in.ftpd
```

tcpd first search */etc/hosts.allow* and then inside of */etc/hosts.deny*. The *hosts.deny* file contains the rules on which nodes do not have access to a service within this machine. A restrictive configuration is **ALL : ALL**, as it will only allow access to the services from the nodes declared in */etc/hosts.allow*

- The */etc/hosts.equiv* file permits access to this machine without having to enter the password. Using this mechanism is not recommended; users should be advised not to use the equivalent from the user account, through the *.rhosts* file.

- In Debian GNU/Linux, it is important to configure */etc/security/access.conf*, the file that indicates the rules on who and from where it is possible to log in to this machine. This file has a line by command with three fields separated by a `:` of the permission type: Users: origin. The first will be an `+o-` (allow or deny), the second a user name/user names, group or `user@host`, and the third will be the name of a device, node, domain, node or networks addresses or ALL.

Example 6.21. Example of *access.conf*

This command does not permit root logins over `tty1`:

```
ALL EXCEPT root:tty1 ...
```

-

It permits access to `u1`, `u2`, `g1` and all those in the `ftacademy.org` domain:

```
+:u1 u2 g1 .ftacademy.org:ALL
```

-

10.2.3 IP Options

IP Forwarding

IP Forwarding is a kernel option that by default is disabled. To check the *net.ipv4.ip_forward* register with the *sysctl* utility use the following command. *sysctl* is the command used to read and configure kernel parameters at runtime.

```
$ sysctl net.ipv4.ip_forward
net.ipv4.ip_forward = 0
```

```
$ sysctl net.ipv6.conf.all.forwarding
net.ipv6.conf.all.forwarding = 0
```

So IP forwarding is disabled for IPv4 and IPv6. To temporarily enable IP forwarding use the following commands. These in effect changes the value in */proc/sys/net/ipv4/ip_forward* and */proc/sys/net/ipv6/conf/all/forwarding* to 1.

```
$ sudo sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

```
$ sudo sysctl -w net.ipv6.conf.all.forwarding=1
net.ipv6.conf.all.forwarding=1
```

This is a temporary solution and upon the next reboot the option will be back to 0 in the register. To make it permanent edit the options in the */etc/sysctl.conf* file. Note that to enable IPv6 packet forwarding disables SLAAC for this host.

```
...
net.ipv4.ip_forward=1
net.ipv6.conf.all.forwarding=1
...
```

To reload the changed values run the `sysctl` command with the `-p` option switch.

```
$ sudo sysctl -p /etc/sysctl.conf
```

Other widely used are: `ip_default_ttl`, which is the lifetime for an IP packet (64 milliseconds, by default), `ip_bootp_agent` logical variable (BOOLEAN) which accepts packets (or not) with the origin address of the `0.b.c.d` type and the destination of this node, broadcast or multicast.

10.2.4 Commands for solving problems with the network

If there are problems in the configuration of the network, begin by verifying the output of the following commands to obtain an initial idea:

```
$ ip link show
$ ip addr list
$ cat /proc/interrupts
$ dmesg | less
```

In order to verify the network connection, we can use the following commands (`netkit-ping`, `traceroute`, `dnsutils`, `iptables` and `net-tools` must be installed):

```
$ ping ftacademy.org           # verifies the Internet connection
$ traceroute ftacademy.org     # scans IP packets
$ ip route list                # verifies the routing configuration
$ dig [@ftacademy.org] www.ftacademy.org # verifies the registries in
                                     # on the dns.ftacademy.org server.
$ sudo iptables -L -n |less    # verifies packet filtering (kernel >=2.4)
$ ss -A all                    # shows all the open ports
$ ss -l --inet                 # shows all the listening sockets
$ ss -l -A tcp                  # shows the listening tcp ports (number)
$ ss -l -A udp                  # shows the listening udp ports (number)
```

10.3 DHCP Configuration

DHCP. It is very simple to configure and it is useful because, instead of having to configure each node in a network individually, this can be done in a centralised manner and administering it is therefore easier. The configuration of a client is very easy, as we only have to install one of the following packages:

- `dhclient` (Internet Systems Consortium)
- `dhcp3-client` (version 3, Internet Software Consortium)
- `dhcpcd` (Yoichi Hariguchi and Sergei Viznyuk)
- `pump` (Red Hat)

For a basic configuration edit the `/etc/network/interfaces`, which defines the interfaces of the host. If the interface `eth0` is to be configured through DHCP during boot up, add/edit the `eth0` entry in the file where:

- `auto`: the interface should be configured during boot time
- `inet`: interface uses TCP/IP networking
- `dhcp`: the interface can be configured through DHCP

```
$ sudo vi /etc/network/interfaces
```

```
auto eth0
iface eth0 inet dhcp
```

10.3.1 DHCP Server

Configuring the server requires more care, but it is not especially complicated. Install as follows:

```
$ sudo apt-get install isc-dhcp-server
```

Configuration

With root permissions edit the `/etc/default/isc-dhcp-server` file,

```
$ sudo -s
# vi /etc/default/isc-dhcp-server
```

```
INTERFACES="eth0"
```

The main configuration file for the `dhcp-server` is `/etc/dhcp/dhcpd.conf`. Edit it like this and any options not mentioned below that are in the file should be left alone.

```
$ sudo vi /etc/dhcp/dhcpd.conf
```

```
option domain-name "ftacademy.org";
option domain-name-servers 78.143.141.250, 78.143.141.251;

subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.2 192.168.1.250;
    option domain-name "intranet.ftacademy.org";
    option routers 192.168.1.1;
    option broadcast-address 192.168.1.255;
    option domain-name-servers 78.143.141.250, 8.8.8.8;
    default-lease-time 600;
    max-lease-time 7200;
}
```

To make the changes effective restart the *dhcp-daemon*. Run with root privileges:

```
$ sudo service isc-dhcp-server restart
```

This creates a DHCP Server on interface *eth0* that assigns IP addresses in the range *192.168.1.2 - 192.168.1.250/24*, giving a default gateway of *192.168.1.1*.

10.4 IP Masquerade

The IP Masquerade is a resource used so that a set of machines may use a single IP address. This permits the hidden nodes on a private network, such as *198.162.10.1* to access the public network, say the Internet; but they cannot directly accept external calls or services; only through the machine that has the real IP. Traffic from the private network to the Internet must have the private source IP address replaced with the Masquerade *public* IP address. Outward connections must be tracked so incoming returning traffic can be correctly identified and the correct private IP address swapped in the packet header for the *public* IP address before forwarding to the private network. This is achievable because of a GNU/Linux feature called Connection Tracking (conntrack). While on the *public* network the source IP address is masqueraded as if it came from the GNU/Linux server. The *iproute2 ip* command incorporates the tools to do this.

```
ip route add nat <internal network>[/<mask length>] via <public addr>
```

Assuming the requirement is to NAT for the private address space *192.168.10.0/24* via the public interface *eth2* on public IP address *78.143.141.52*.

```
$ sudo ip addr add 192.168.10.1/24 dev eth0
$ sudo ip addr add 78.143.141.52/24 dev eth2
$ sudo iptables --table nat --append POSTROUTING --source 192.168.10.0/24
--out-interface eth2 --jump MASQUERADE
```

The firewall if one is implemented has to be adapted to allow traffic in and out of *eth2* for the *192.168.10.0/24* network.

```
$ sudo iptables -append FORWARD --source 192.168.10.0/24 --out-interface
eth2 -j ACCEPT
$ sudo iptables -append FORWARD --destination 192.168.10.0/24 --match
state --state ESTABLISHED,RELATED --in-interface eth2 --jump ACCEPT
```

This page is intentionally blank

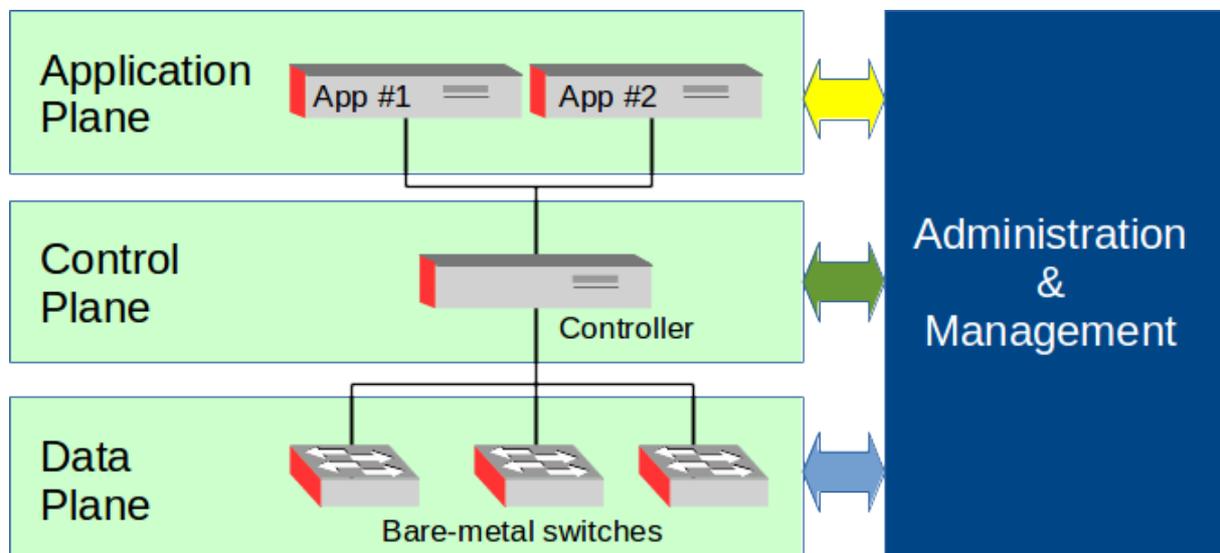
11. Software Defined Networking (SDN)

11.1 Introduction

Over the last ten years or so the landscape in computing has changed dramatically with the Cloud, large-scale Data Centres and virtualisation. However while networks have increased in speed and there has been a convergence on Ethernet as the standard for all links, to the point that the difference between LAN, MAN and Wide Area Network WAN has diminished dramatically. What has not changed however is the core switch and router function which is generally a hardware based stand-alone device that is self sufficient in terms of the data they switch or route and the control necessary to make that happen.

While the underlying networks have converged towards the all Ethernet / all Internet Protocol IP model, in some form the number of services have increased rapidly. In the past Service Providers provided Internet Access in the form of Broadband and possibly layered a voice service either as a circuit switched out of band telephone line or as a VoIP service with some packet priority mechanism to give Quality of Service (QoS). In more recent years this service is increasingly being supplemented with a Television over IP (TVoIP) service that more often than not requires a separate Set Top Box (STB) for its provision.

11.2 Software Defined Networking



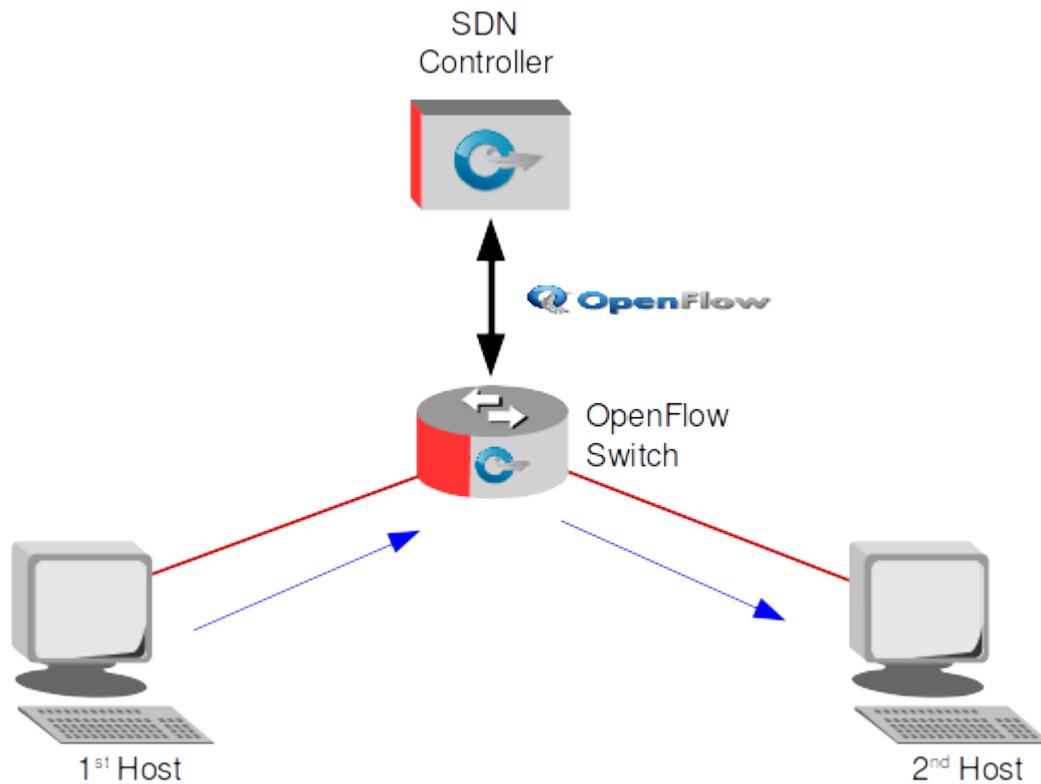
A quiet revolution started with an SDN test in Ohio State University in 2001; the idea being to remove the control functions from an Ethernet switch and leave it with only data forwarding functions with the extracted control functions migrated to a server called an SDN Controller, the SDN Controller taking control of several bare-metal or white-box switches to form an SDN Island. A bare-metal switch is built from standardised commodity parts thereby reducing the cost of manufacture as the intelligence has been extracted to commodity servers. This gives the SDN Controller a visibility of the complete range of ports across several hardware platforms over a Control - Data Plane Interface (CDPI). The SDN Controller itself is programmable, centrally managed and agile. Thus the hardware elements become known as the Data Plane while the SDN Controller is the Control Plane. Because the Controller is programmable it makes the creation of an Application Plane tier possible which gains access to the SDN Controller via a North Bound Interface (NBI). To complete an SDN solution an Administration and Management tier is added to manage the overall network.

To make all of this possible it was essential that hardware vendors develop devices that could be readily controlled by an SDN Controller over a standards based protocol. To this end the OpenFlow protocol was developed at Stanford University. This new protocol would allow the SDN Controller access and manipulate the data forwarding plane of the hardware. This protocol evolved into an open standard whose adoption, development and promotion was vested in the Open Networking Foundation (ONF) in 2011.

Over the years hardware manufacturers of switches and routers were making increasingly larger profits as the demand for networking devices sky-rocketed. This was particularly so due the development of large scale data centres. The new SDN model is ideally suited to meet the needs of Data Centres while at the same time allow them to make significant reduction in costs.

11.3 SDN operation

In order to understand how SDN switching works consider a traditional switch. A frame arrives at a switch port, the switch inspects the frame header and determines if it has a record for the destination MAC address. If it doesn't, then the frame is forwarded on all ports and the source port MAC is recorded upon which the port the frame was received. If it does then the frame is only forwarded to the known port associated with the destination MAC. All these decisions are made in the individual switch.



In an SDN Network when as is shown in detail in the mininet example labs below, when a frame arrives at an OpenFlow Switch, it performs a table entry check and if it finds that it has a *table-miss*, which means there is no flow entry associated with this frame, it will send an OpenFlow *Packet In (OFPT_PACKET_IN)* message to the SDN Controller with a unique Buffer Identifier for a decision. The SDN Controller responds to the OpenFlow Switch using the same Buffer Identifier with the decision to *Output to switch port* on all ports.

The response from the second host arrives at the OpenFlow Switch and is given a new Buffer Identifier, again there is a *table-miss* so the OpenFlow Switch sends an OpenFlow *OFPT_PACKET_IN* message to the SDN Controller. The SDN Controller now sends an OpenFlow *Flow MOD* to the OpenFlow Switch to add an entry to the Flow Table for this now known traffic. Subsequent similar packets are then forwarded automatically by the OpenFlow Switch until the idle time-out of 60 seconds has been exceeded and then the process must be repeated.

The next packet in from the original host triggers another *OFPT_PACKET_IN*. This time the SDN Controller knows the port that the second host is connected on, so it sends an OpenFlow *Flow MOD* to the OpenFlow Switch to add an entry to the Flow Table for the traffic. Subsequent similar packets are then forwarded automatically by the OpenFlow Switch until the idle time-out of 60 seconds has been exceeded and then the process must be repeated.

In the example the frame that arrived was unmatched by the OpenFlow Switch. It is typical for the SDN Controller to *pre-load* the OpenFlow Switch with flows. It is also not simply the MAC fields in the frame header nor the IP Addresses in the packet header. Flows can be based on a multitude of values within the overall frame and its sub packet and even transport session headers:

- The port the frame arrived on
- The source Ethernet port
- The destination Ethernet port
- The source IPv4 or IPv6 address
- The destination IPv4 or IPv6 address
- IPv6 Flow Label
- IPv6 Extension Header pseudo-field
- ICMPv6 type or code
- Target IP address, source or target link layer address in IPv6 Neighbour Discovery (ND)
- VLAN Identifier (VLAN-ID)
- VLAN Priority Code Point (PCP)
- Differentiated Services (DiffServ) Code Point (DSCP)
- IP Header Explicit Congestion Notification (ECN)
- IPv4 or IPv6 Protocol number
- TCP Source, Destination port or flags
- UDP Source or Destination port
- Stream Control Transmission Protocol (SCTP) Source or Destination port
- ICMP Type or Code
- ARP Opcode
- MAC Addresses in ARP payload
- IP Addresses in ARP payload
- The LABEL, Traffic Class (TC) or Bottom of Stack (BoS) in first MPLS Shim header
- User Customer Address (UCA) field in the first Provider Backbone Bridge (PBB) instance tag

11.3.1 Flow Tables

In the forwarding instructions the controller specifies to the OpenFlow switch the group of parameters used to define individual flows and what action to carry out on frames that match the flow. The OpenFlow switch has as can be seen in the diagram multiple *Flow Tables*. In the initial OpenFlow version this was limited to a single table however the Application Specific Integrated Circuit (ASIC) hardware in switches were capable of much more so later versions of the OpenFlow protocol allowed for multiple tables which improves performance and scalability.

11.3.2 Group Tables

OpenFlow protocol also added *Group Tables* consisting of group entries. A flow entry can be pointed to a group, which enables OpenFlow to have additional methods of forwarding:

- *SELECT*: for load sharing and redundancy
- *ALL*: for multicast or broadcast forwarding
- *INDIRECT*: which allows for multiple flow entries to point to a common group ID
- *FAST FAILOVER*: which enables the switch to change forwarding without requiring communication with the SDN Controller in the event of a port failure

11.3.3 Meter Tables

The Meter Tables consists of *per-flow meters* used by OpenFlow to implement QoS. Each *per-flow meter* measures the rate of frames assigned to it and controls the rate of those frames. Each meter consists of one or more meter bands which specify the rate at which the band applies and how frames are processed. Each meter band is identified by its rate and contains:

- *Band type*: defines how packet are processed
- *rate*: defines the lowest rate at which the band can apply
- *burst*: defines the granularity of the meter band
- *counters*: updated when packets are processed by a meter band
- *type specific arguments*
 - *drop*: discard the packet. Can be used as a rate limiter band
 - *dscp remark*: increase the drop precedence of the DSCP field in the IP header. Can act a simple DiffServ policer

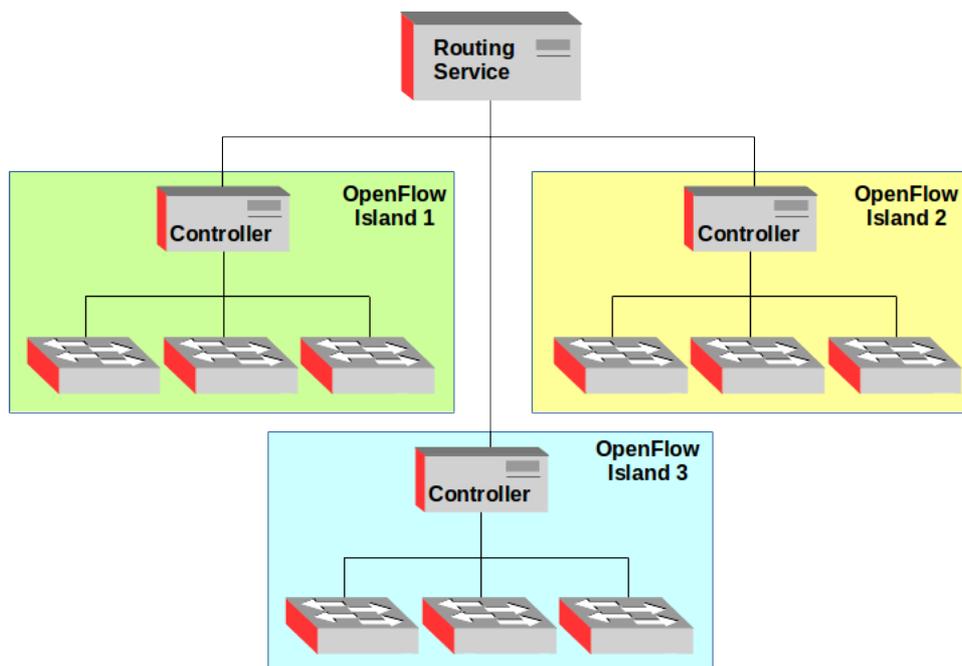
11.4 SDN Controllers

There are a number of SDN Controllers, chief among them are some Free and OpenSource (FOSS) projects. Here are three of the most popular SDN Controllers however there are many more.

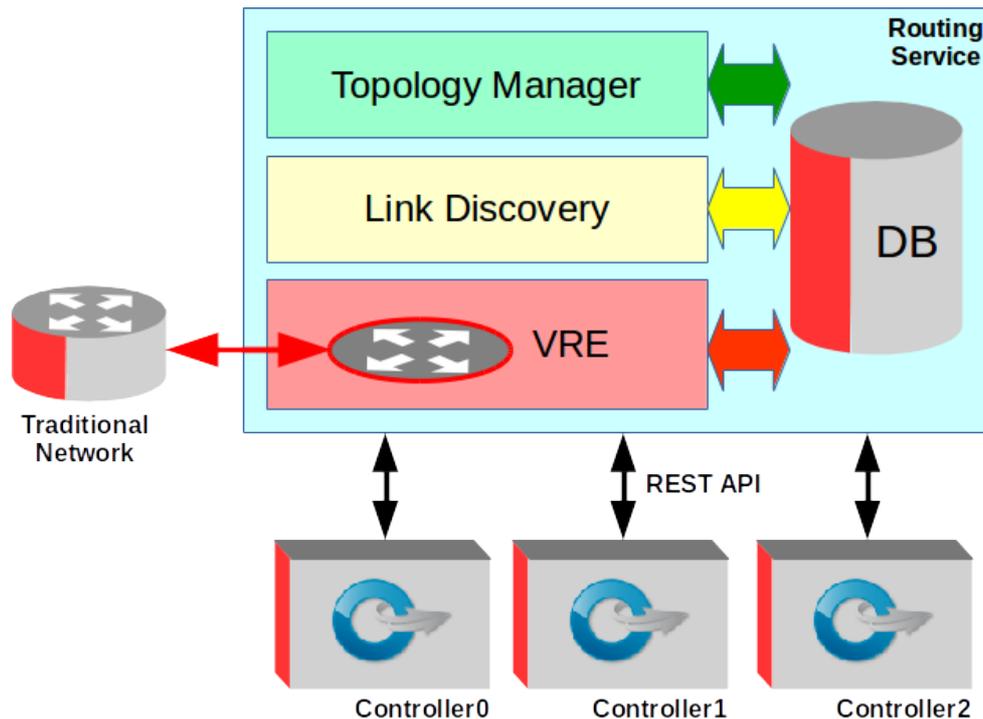
- *NOX* is a general purpose OpenFlow based SDN Controller written in C/C++.
- *POX* is a general OpenFlow based SDN controller built using Python. POX supports an SDN API with virtualisation support.
- *Floodlight* is a FOSS, Java based OpenFlow Controller. It incorporates a REpresentational State Transfer (REST) Web Services API as the recommended interface to develop applications.

11.5 SDN Applications

11.5.1 SDN Routing Service



As we have seen the SDN Controller manages switches on the CDPI using OpenFlow protocol. On the NBI the SDN Controller interfaces using REST API with application services. In traditional networks networks are linked by routers. In an SDN Network groups of switches are managed by a controller and this is called an *OpenFlow Island*. In an SDN Network the *Flows* that the SDN Controller send to the individual switches are controlled by SDN Applications. One typical example is the routing service.



Routing in this case is an SDN Application and consists of:

- Link Discovery Module (LDM)
- Topology Manager
- Virtual Routing Engine (VRE)

11.6 Link Discovery Module

The LDM discovers and maintains the status of all physical links on the network. When OpenFlow Switches discover other switches via Link Layer Discovery Protocol (LLDP) this information is passed to the Link Discovery Module (LDM). Additionally when unknown traffic is discovered by an OpenFlow Switch as described above the SDN Controller also passes this to the LDM. In this way the LDM derives the information to build a picture of the overall network topology as a Neighbour Database.

11.7 Topology Manager

The Topology Manager builds the topology from the Neighbour Database. It generates the logical OpenFlow Islands and determines the shortest path between OpenFlow nodes. From this the Topology Manager can build the individual topology databases for the controllers which contain the shortest paths plus alternate paths to each OpenFlow node or hosts connected to them.

11.8 Virtual Routing Engine (VRE)

The function of the VRE is to allow SDN networks interoperate with traditional networks. It builds a virtual networking topology to represent the SDN network to the traditional networks using traditional routing protocols like Open Shortest Path First (OSPF) and Border Gateway Protocol (BGP).

While it is essential for SDN to have an application like the Routing Service to handling routing within the SDN and to interact with traditional networks, the SDN architecture lends itself readily to newer SDN Applications that can interact with the Controller over the REST API and thereby influence the OpenFlow Switches in new and imaginative ways not possible in today's traditional networks.

11.9 Using Mininet to experiment with SDN

Mininet is a project that creates a virtual network on a computer, a *network emulator*. On it it is possible to develop a network of hosts, switches, routers and links based on a single GNU/Linux kernel. Mininet uses Linux Containers (LXC) lightweight virtualisation to allow for experimentation with SDNs and SDN Controllers. For example a SDN Controller can be given a network of devices to work with and because they are based on the GNU/Linux kernel behave exactly as a standalone GNU/Linux device.

To allow for experimentation lets set-up a Mininet VM image to work with. Install Oracle VirtualBox as a hypervisor first (<https://www.virtualbox.org>).

11.10 Set-up a guest VM with the mininet image

Download VM from <http://mininet.org/download/>

- Run the Oracle VirtualBox Manager and click the *New* button to create a new virtual machine.
 - Name: mininet
 - Type: Linux
 - Version: Linux 2.6 / 3.x (64 bit)
- Memory size: 1024 MB
- Hard drive
 - Use an existing virtual hard drive file
 - Select *mininet-vm-x86_64.vmdk*, click *Open*, then *Create*
- Mininet VM settings
 - General → Advanced → Shared Clipboard: Bidirectional
 - Network → Adapter 1
 - Attached to: Bridged Adapter
 - Name: wlan0 (or whatever interface the host is using)
- Power on VM

Login to the image and update and upgrade the image.

```
mininet-vm login: mininet
Password: mininet
```

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Get the IP address of the mininet VM.

```
$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default
    qlen 1000
    link/ether 08:00:27:e6:cb:e2 brd ff:ff:ff:ff:ff:ff
    inet 192.168.22.83/24 brd 192.168.25.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fee6:cbe2/64 scope link
        valid_lft forever preferred_lft forever
```

From the host computer SSH to the VM guest.

```
$ ssh mininet@192.168.22.83
mininet@192.168.22.83's password: mininet
mininet@mininet:~$
```

11.10.1 Add rights to wireshark for user

For experimentation purposes it is important that *Wireshark* is installed to allow for the capture of packets on the mininet VM.

```
$ apt-get install wireshark tshark
$ sudo dpkg-reconfigure wireshark-common
```

Package configuration

```
| Configuring wireshark-common |
Dumpcap can be installed in a way that allows members of the "wireshark" system group to capture packets.
This is recommended over the alternative of running Wireshark/Tshark directly as root, because less of
the code will run with elevated privileges.

For more detailed information please see /usr/share/doc/wireshark-common/README.Debian.

Enabling this feature may be a security risk, so it is disabled by default. If in doubt, it is suggested
to leave it disabled.

Should non-superusers be able to capture packets?

<Yes>                                <No>
```

Add the *mininet* user to the *wireshark* group.

```
$ sudo usermod -a -G wireshark mininet
```

Allow users access to *dumpcap*.

```
$ sudo setcap 'CAP_NET_RAW+eip CAP_NET_ADMIN+eip' /usr/bin/dumpcap
```

Logout and log back in as user *mininet*.

Confirm that *wireshark* can access the Ethernet interface.

```
$ tshark -i eth0
Capturing on 'eth0'
  1  0.000000 Routerbo_c0:4d:fa → Broadcast      ARP 60 Who has
192.168.25.177? Tell 192.168.25.254
  2  0.183036 HewlettP_f3:08:3a → Broadcast      ARP 60 Who has
192.168.25.147? Tell 192.168.25.161
```

11.11 Confirm Wireshark works over SSH

wireshark can be run on the *mininet* VM guest and viewed on the host via *ssh* which forwards the graphical output via X11 forwarding to the host display via the *-X* option.

```
$ ssh -X mininet@192.168.22.83 'wireshark -i lo'
```

11.12 Build a mininet test network

Look at the startup options for *mininet*. When connecting to the *mininet* VM guest via SSH use the *-X* option to enable X11 forwarding. This is useful for opening shells for different hosts.

```
host~$ ssh -X mininet@192.168.22.83
```

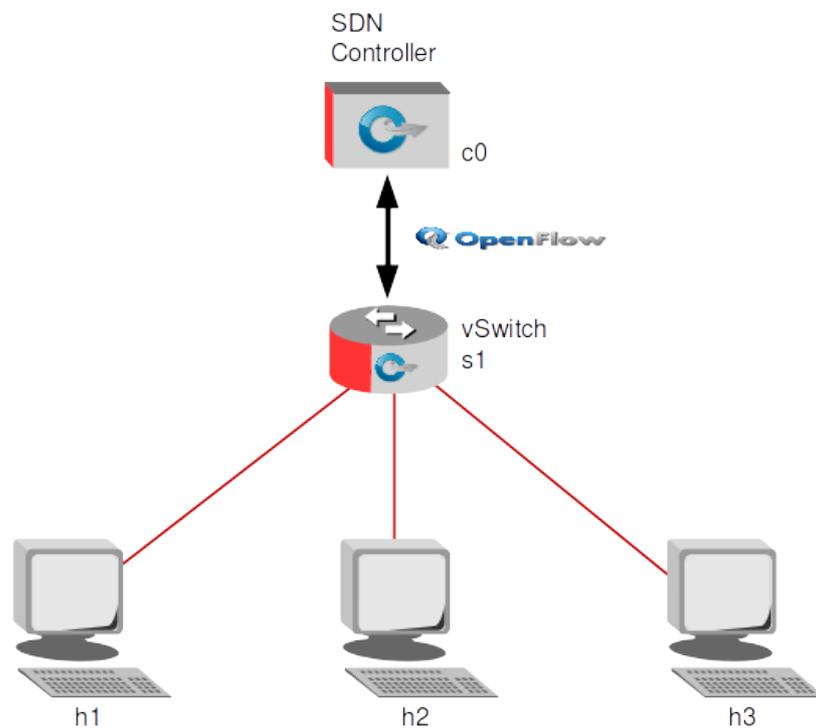
\$ sudo mn -h

Usage: mn [options]
 (type mn -h for details)

The mn utility creates Mininet network from the command line. It can create parametrized topologies, invoke the Mininet CLI, and run tests.

Options:

-h, --help	show this help message and exit
--switch=SWITCH	default ivs lxb ovs ovsbr ovsk ovsl user[,param=value...]
--host=HOST	cfs proc rt[,param=value...]
--controller=CONTROLLER	default none nox ovsc ref remote ryu[,param=value...]
--link=LINK	default tc[,param=value...]
--topo=TOPO	linear minimal reversed single torus tree[,param=value...]
-c, --clean	clean and exit
--custom=CUSTOM	read custom classes or params from .py file(s)
--test=TEST	cli build pingall pingpair iperf all iperfudp none
-x, --xterms	spawn xterms for each node
-i IPBASE, --ipbase=IPBASE	base IP address for hosts
--mac	automatically set host MACs
--arp	set all-pairs ARP entries
-v VERBOSITY, --verbosity=VERBOSITY	info warning critical error debug output
--innamespace	sw and ctrl in namespace?
--listenport=LISTENPORT	base port for passive switch listening
--noistenport	don't use passive listening port
--pre=PRE	CLI script to run before tests
--post=POST	CLI script to run after tests
--pin	pin hosts to CPU cores (requires --host cfs or --host rt)
--nat	adds a NAT to the topology that connects Mininet hosts to the physical network. Warning: This may route any traffic on the machine that uses Mininet's IP subnet into the Mininet network. If you need to change Mininet's IP subnet, see the --ipbase option.
--version	prints the version and exits
--cluster=server1,server2...	run on multiple servers (experimental!)
--placement=block random	node placement for --cluster (experimental!)



Establish a basic network with an SDN Controller (*c0*) an Open vSwitch (OVS) and three hosts.

Options:

- Switch
 - *ivs* - Indigo Virtual Switch
 - *lxbr* - Linux Bridge
 - *ovs* - Open vSwitch
 - *ovsbr* - Open vSwitch in standalone/bridge mode
 - *ovsk* - OpenFlow 1.3 switch
 - *ovsl* - Open VSwitch legacy kernel-space switch using *ovs-openflowd*
- Controller
 - *nox* - Nicira Networks OpenFlow controller
 - *ovsc* - Open vSwitch controller
 - *ref* - OpenFlow reference controller
 - *remote* - Controller running outside of mininet (i.e. OpenDaylight for example)
 - *ryu* - RYU Network Operating System
- topo
 - *linear* - Linear topology of *k* switches, with *n* hosts per switch
 - *minimal* - Single switch and two hosts
 - *reversed* - Single switch connected to *k* hosts, with reversed ports, the lowest-numbered host is connected to the highest-numbered port
 - *single* - Single switch connected to *k* hosts
 - *torus* - 2-D Torus mesh interconnect topology
 - *tree* - a tree network with a given depth and fanout
- *mac* - automatically set host MACs

Create a basic network to start with from the *mn* mininet launch command.

```
$ sudo mn --topo tree,depth=1,fanout=3 --switch ovsk --controller ref --mac
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(s1, h1) (s1, h2) (s1, h3)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1
*** Starting CLI:
```

Review the network elements and the links between them.

```
mininet> hosts
*** Unknown command: hosts
mininet> nodes
available nodes are:
c0 h1 h2 h3 s1

mininet> links
s1-eth1<->h1-eth0 (OK OK)
s1-eth2<->h2-eth0 (OK OK)
s1-eth3<->h3-eth0 (OK OK)

mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=7960>
<Host h2: h2-eth0:10.0.0.2 pid=7963>
<Host h3: h3-eth0:10.0.0.3 pid=7965>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None pid=7970>
<Controller c0: 127.0.0.1:6633 pid=7952>
```

To run commands on the hosts, use the hostname followed by the command. For example look at the IP address on *h1* and route table on *h2*.

```
mininet> h2 ip addr show | grep eth0
159: h2-eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    inet 10.0.0.2/8 brd 10.255.255.255 scope global h2-eth0

mininet> h2 ip route
10.0.0.0/8 dev h2-eth0 proto kernel scope link src 10.0.0.2
```

Test connectivity from one host to another.

Test options in the cli are:

- build
- pingall - Ping between all hosts
- pingallfull - Ping between all hosts, including times
- pingpair - Ping between first two hosts
- iperf <host1> <host2> - Run TCP iperf between two hosts
- iperfudp <bw i.e. 100M> <host1> <host2> - UDP iperf

```
mininet> h1 ping -c1 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=4.66 ms

--- 10.0.0.3 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 4.664/4.664/4.664/0.000 ms
```

Look at network links between elements.

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0
c0
```

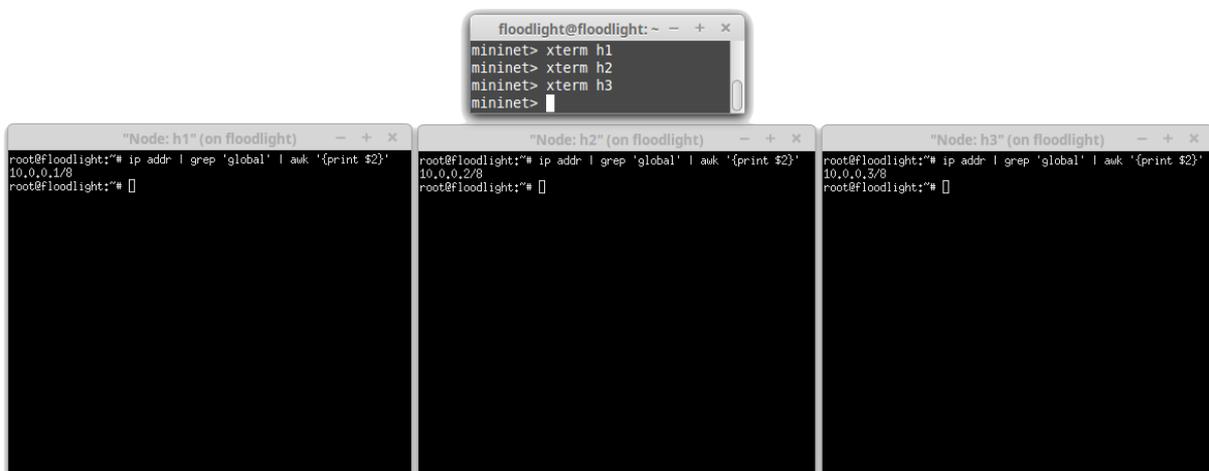
The mininet command *pingall* is useful for checking connectivity between hosts.

```
mininet> pingall
*** Ping: testing ping reachability
h1 → h2 h3
h2 → h1 h3
h3 → h1 h2
*** Results: 0% dropped (6/6 received)
```

iperf can be ran from the mininet prompt to test bandwidth between links. Here is an example between *h1* and *h2*.

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['18.0 Gbits/sec', '18.0 Gbits/sec']
```

Running commands on hosts can be done directly from the mininet command shell as seen above or individual xterms can be ran for hosts. In the example see the IPv4 addresses for each host.



The image shows a terminal window titled 'floodlight@floodlight: ~ - + x' with the following output:

```
mininet> xterm h1
mininet> xterm h2
mininet> xterm h3
mininet>
```

Below this are three separate terminal windows, each titled 'Node: h1 (on floodlight)', 'Node: h2 (on floodlight)', and 'Node: h3 (on floodlight)'. Each window shows the command 'root@floodlight:~# ip addr | grep 'global' | awk '{print \$2}'' and its output: '10.0.0.1/8', '10.0.0.2/8', and '10.0.0.3/8' respectively. Each window also shows the prompt 'root@floodlight:~# '.

11.12.1 Exiting mininet

To exit mininet use the *exit* command.

```
mininet> exit
*** Stopping 1 controllers
c0
*** Stopping 1 switches
s1 ...
*** Stopping 3 links

*** Stopping 3 hosts
h1 h2 h3
*** Done
completed in 3.536 seconds
```

It is a good idea to follow this up with *sudo mn -c* to tidy up before running another network.

```
$ sudo mn -c
*** Removing excess controllers/ofprotocols/ofdatapaths/pings/noxes
killall controller ofprotocol ofdatapath ping nox_core lt-nox_core ovs-
openflowd
ovs-controller udpbwtest mnexec ivs 2> /dev/null
killall -9 controller ofprotocol ofdatapath ping nox_core lt-nox_core
ovs-openflowd ovs-controller udpbwtest mnexec ivs 2> /dev/null
pkill -9 -f "sudo mnexec"
*** Removing junk from /tmp
rm -f /tmp/vconn* /tmp/vlogs* /tmp/*.out /tmp/*.log
*** Removing old X11 tunnels
*** Removing excess kernel datapaths
ps ax | egrep -o 'dp[0-9]+' | sed 's/dp/nl:/'
*** Removing OVS datapathsovs-vsctl --timeout=1 list-br
ovs-vsctl --timeout=1 list-br
*** Removing all links of the pattern foo-ethX
ip link show | egrep -o '([-_.[:alnum:]]+-eth[[:digit:]]+)'
*** Killing stale mininet node processes
pkill -9 -f mininet:
*** Shutting down stale tunnels
pkill -9 -f Tunnel=Ethernet
pkill -9 -f .ssh/mn
rm -f ~/.ssh/mn/*
*** Cleanup complete.
```

11.13 Configuring hosts

Recreate the same network but this time specifying the bandwidth of links and adding delay. The options:

- host
 - cfs - Completely Fair Scheduler (CFS)
 - proc
 - rt, - Real Time (RT)
 - cpu=<positive fraction, or -1> - CPU bandwidth limit

Note: RT_GROUP_SCHED must be enabled in the kernel to change the *rt,cpu*.

```
$ sudo mn --topo tree,depth=1,fanout=3 --switch ovsk --controller ref
--mac --host rt,cpu=0.25
```

11.14 Configuring links

Recreate the same network but this time for traffic control (tc) specify the bandwidth of links and adding delay. The options:

- link tc
 - bw=<value> - Value in Mb/s
 - delay=<value> - Time unit expressed as '5ms', '50us' or '1s'
 - max_queue_size=<x> - Queue size in packets
 - loss=<0 - 100> - Percentage loss
 - use_htb=<True | False> - Hierarchical Token Bucket (HTB) rate limiter

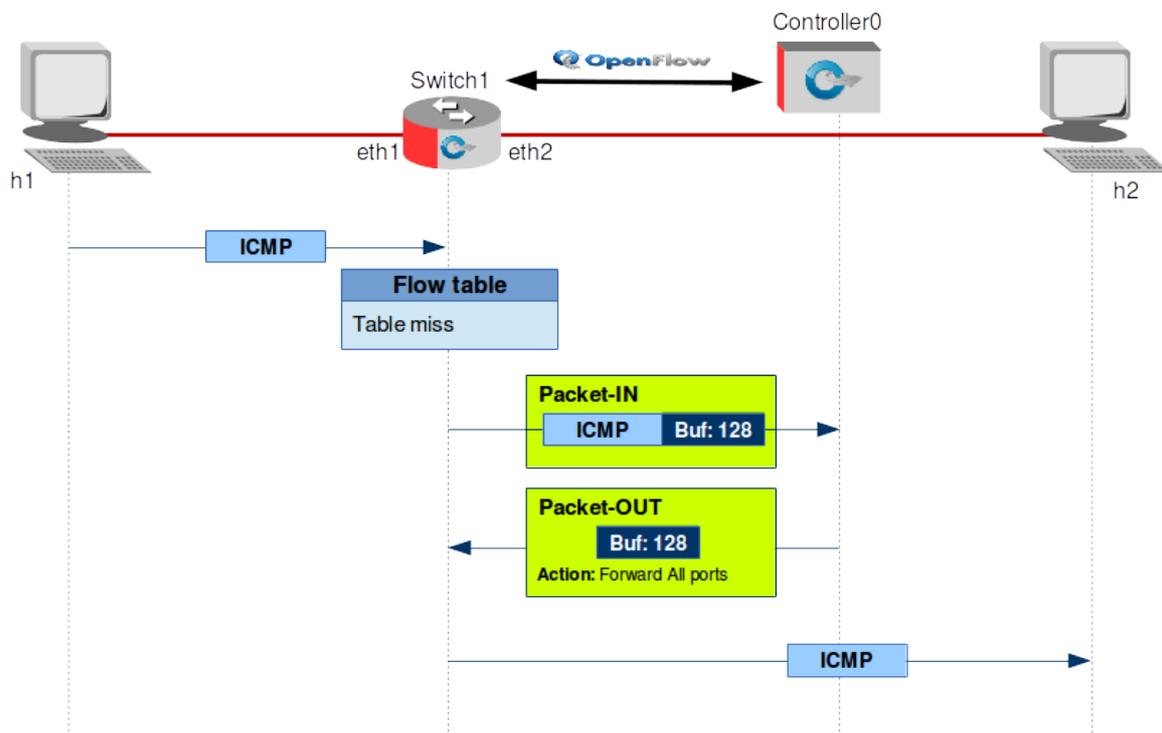
```
$ sudo mn --topo tree,depth=1,fanout=3 --switch ovsk --controller ref
--mac --link tc,bw=100,delay=20ms
[sudo] password for mininet:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(100.00Mbit 20ms delay) (100.00Mbit 20ms delay) (s1, h1) (100.00Mbit 20ms
delay) (100.00Mbit 20ms delay) (s1, h2) (100.00Mbit 20ms delay)
(100.00Mbit 20ms delay) (s1, h3)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 (100.00Mbit 20ms delay) (100.00Mbit 20ms delay) (100.00Mbit 20ms
delay)
*** Starting CLI:
```

Repeat the *iperf* test between *h1* and *h2*. Note the difference in results from the earlier test.
 *** Results: ['18.0 Gbits/sec', '18.0 Gbits/sec'].

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['85.7 Mbits/sec', '99.3 Mbits/sec']
```

11.15 Reviewing OpenFlow traffic

The OpenFlow traffic between the Controller *c0* and the Open vSwitch *s1*. As the controller and switch share the same VM guest they control channel is via the loopback interface, so monitoring the loopback *lo0* interface will give access to this messaging. In this case the messaging is using OpenFlow v1.0 so using the filter *of* we can see communications between the devices.

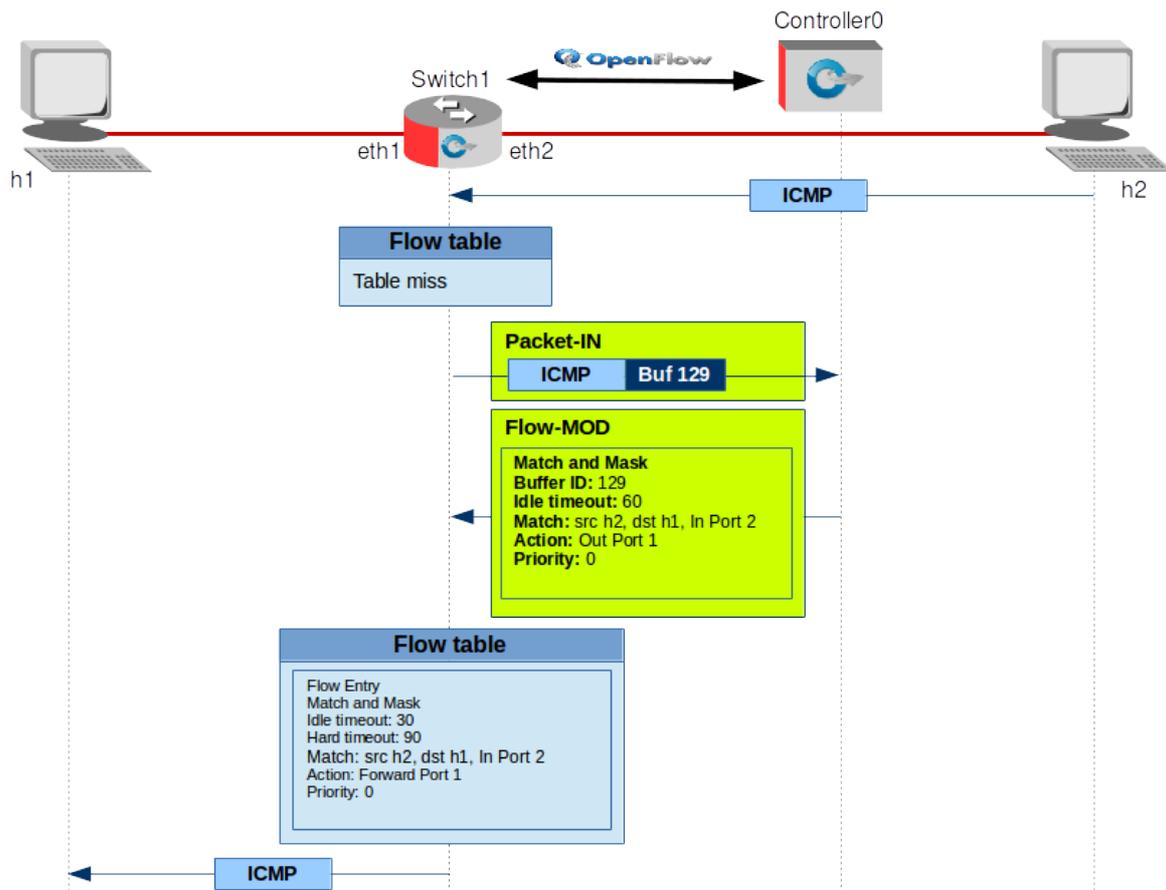


An Internet Control Message Protocol (ICMP) arrives at *s1*. It does a table entry check and finds that it has a *table-miss* no flow entry. The vSwitch *s1* sends an OpenFlow *Packet In* (*OFPT_PACKET_IN*) message to the controller *c0* with a unique Buffer Identifier *128* for a decision.

```
Frame 1: OFPT_PACKET_IN
Ethernet II, Src: 00:00:00_00:00:00, Dst: 00:00:00_00:00:00
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 60348, Dst Port: 6633, Seq: 9,
Ack: 9, Len: 60
OpenFlow
  version: 1
  type: OFPT_PACKET_IN (10)
  length: 60
  xid: 0
  buffer_id: 128
  total_len: 42
  in_port: 1
  reason: OFPR_NO_MATCH (0)
  Ethernet packet
```

The controller *c0* responds to the vSwitch *s1* using the Buffer Identifier *128* with the decision to *Output to switch port* on all ports.

```
Frame 2: OFPT_PACKET_OUT
Ethernet II, Src: 00:00:00_00:00:00, Dst: 00:00:00_00:00:00
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 6633, Dst Port: 60348, Seq: 9,
Ack: 69, Len: 24
OpenFlow
  version: 1
  type: OFPT_PACKET_OUT (13)
  length: 24
  xid: 0
  buffer_id: 128
  in_port: 1
  actions_len: 8
  of_action list
    of_action_output
      type: OFPAT_OUTPUT (0)
      len: 8
      port: 65531
      max_len: 0
```



The response from the host *h2* arrives at the Open vSwitch *s1* and is given a Buffer IDentifier *129*, again there is a table-miss so the vSwitch *s1* sends an OpenFlow *OFPT_PACKET_IN* message to the controller *c0*.

```

Frame 3: OFPT_PACKET_IN
Ethernet II, Src: 00:00:00_00:00:00, Dst: 00:00:00_00:00:00
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 60348, Dst Port: 6633, Seq: 69,
Ack: 33, Len: 60
OpenFlow
  version: 1
  type: OFPT_PACKET_IN (10)
  length: 60
  xid: 0
  buffer_id: 129
  total_len: 42
  in_port: 2
  reason: OFPR_NO_MATCH (0)
  Ethernet packet

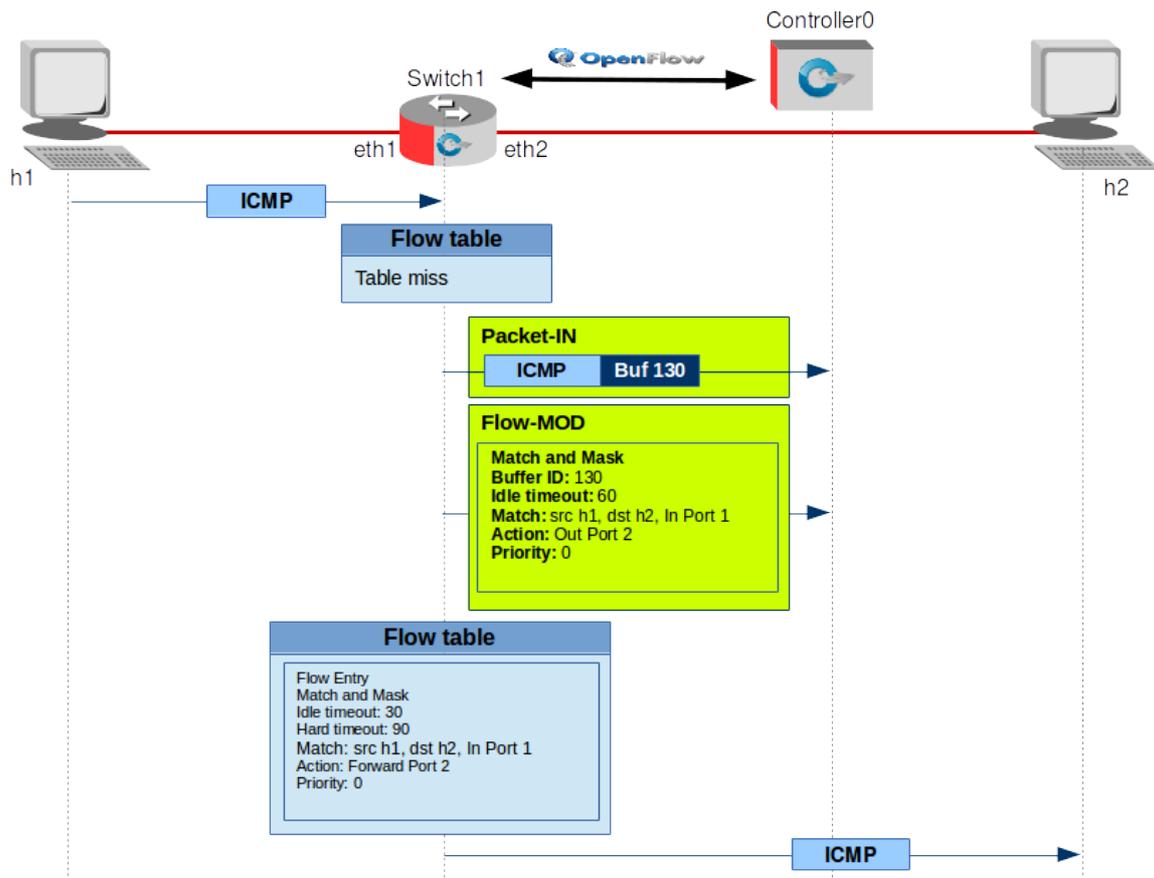
```

In this case the controller *c0* sends an OpenFlow *Flow MOD* to the Open vSwitch *s1* to add an entry to the Flow Table for traffic from 10.0.0.2 → 10.0.0.1 on Ethernet port 2 → port 1. Subsequent similar packets are then forwarded automatically by the Open vSwitch until the idle time-out of 60 seconds has been exceeded and then the process must be repeated.

```

Frame 4: OPENFLOW FLOW MODIFICATION
Ethernet II, Src: 00:00:00_00:00:00, Dst: 00:00:00_00:00:00
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 6633, Dst Port: 60348, Seq: 33,
Ack: 129, Len: 80
OpenFlow
  version: 1
  type: OFPT_FLOW_MOD (14)
  length: 80
  xid: 0
  of_match
    wildcards: 0x0000000000000000
    in_port: 2
    eth_src: 00:00:00_00:00:02
    eth_dst: 00:00:00_00:00:01
    vlan_vid: 65535
    vlan_pcp: 0
    eth_type: 2054
    ip_dscp: 0
    ip_proto: 2
    ipv4_src: 10.0.0.2
    ipv4_dst: 10.0.0.1
    tcp_src: 0
    tcp_dst: 0
  cookie: 0
  _command: 0
  idle_timeout: 60
  hard_timeout: 0
  priority: 0
  buffer_id: 129
  out_port: 0
  flags: Unknown (0x00000000)
  of_action list
    of_action_output
      type: OFPAT_OUTPUT (0)
      len: 8
      port: 1
      max_len: 0

```



The next packet in from *h1* triggers another *OFPT_PACKET_IN*.

```

Frame 5: OFPT_PACKET_IN
Ethernet II, Src: 00:00:00_00:00:00, Dst: 00:00:00_00:00:00
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 60348, Dst Port: 6633, Seq: 129,
Ack: 113, Len: 116
OpenFlow
  version: 1
  type: OFPT_PACKET_IN (10)
  length: 116
  xid: 0
  buffer_id: 130
  total_len: 98
  in_port: 1
  reason: OFPR_NO_MATCH (0)
  Ethernet packet

```

This time Controller *c0* knows the port that *h2* is on so it sends an OpenFlow *Flow MOD* to the vSwitch *s1* to add an entry to the Flow Table for traffic from 10.0.0.1 → 10.0.0.2 on Ethernet port 1 → port 2. Subsequent similar packets are then forwarded automatically by the Open vSwitch until the itle time-out of 60 seconds has been exceeded and then the process must be repeated.

```
Frame 6: OPENFLOW FLOW MODIFICATION
Ethernet II, Src: 00:00:00_00:00:00, Dst: 00:00:00_00:00:00
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 6633, Dst Port: 60348, Seq: 113,
Ack: 245, Len: 80
OpenFlow
  version: 1
  type: OFPT_FLOW_MOD (14)
  length: 80
  xid: 0
  of_match
    wildcards: 0x0000000000000000
    in_port: 1
    eth_src: 00:00:00_00:00:01
    eth_dst: 00:00:00_00:00:02
    vlan_vid: 65535
    vlan_pcp: 0
    eth_type: 2048
    ip_dscp: 0
    ip_proto: 1
    ipv4_src: 10.0.0.1
    ipv4_dst: 10.0.0.2
    tcp_src: 8
    tcp_dst: 0
  cookie: 0
  _command: 0
  idle_timeout: 60
  hard_timeout: 0
  priority: 0, ip=[controller IP]
  buffer_id: 130
  out_port: 0
  flags: Unknown (0x00000000)
  of_action list
    of_action_output
      type: OFPAT_OUTPUT (0)
      len: 8
      port: 2
      max_len: 0
```

Ping is not the only command you can run on a host. Mininet hosts can run any command or application that is available to the underlying Linux system and its file system. You can also enter any bash command, including job control (&, jobs, kill, etc..)

Next, run a simple HTTP server on *h1*, making a request from *h3*, then shut down the web server.

11.16 Webserver test

Install the *lynx* text based web client on the mininet VM.

```
$ sudo apt-get install lynx
```

Check the IP addresses of the *h1* and *h3* hosts. Confirm connectivity between them.

```
mininet> h1 ip addr | grep "inet.*eth0"
    inet 10.0.0.1/8 brd 10.255.255.255 scope global h1-eth0
```

```
mininet> h3 ip addr | grep "inet.*eth0"
    inet 10.0.0.3/8 brd 10.255.255.255 scope global h3-eth0
```

```
mininet> h1 ping -c1 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=4.46 ms
```

xterm *h1* and *h3* so you have access to individual shells for each host.

```
mininet> xterm h1
mininet> xterm h3
```

Run a webserver on *h1* xterm.

```
root@mininet-vm:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Use *lynx* on the *h3* xterm to access the webserver.

```
root@mininet-vm:~# lynx 10.0.0.1
```

```
Directory listing for /
Directory listing for / (p1 of 2)
-----
* .bash_history
* .bash_logout
* .bashrc
* .cache/
* .gitconfig
* .mininet_history
* .profile
* .rnd
* .ssh/
* .wireshark/
* .Xauthority
* install-mininet-vm.sh
* loxigen/
* mininet/
* oflops/
* oftest/
* openflow/
-- press space for next page --
Arrow keys: Up and Down to move. Right to follow a link; Left to go back.
H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

On the webserver xterm on *h1* the following message pops up.

```
10.0.0.3 - - [18/Mar/2015 01:11:34] "GET / HTTP/1.0" 200 -
```

An alternative is to run the server in the mininet console as a background process and the use lynx to view it.

```
mininet> h1 python -m SimpleHTTPServer 80 &
Serving HTTP on 0.0.0.0 port 80 ...
```

```
mininet> h2 lynx h1
```

```

Directory listing for /
Directory listing for / (p1 of 2)
-----
* .bash_history
* .bash_logout
* .bashrc
* .cache/
* .gitconfig
* .mininet_history
* .profile
* .rnd
* .ssh/
* .wireshark/
* .Xauthority
* install-mininet-vm.sh
* loxigen/
* mininet/
* oflops/
* oftest/
* openflow/
-- press space for next page --
Arrow keys: Up and Down to move. Right to follow a link; Left to go back.
H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

Kill the webserver on host *h1*.

```
mininet> h1 ps -ef | grep SimpleHTTPServer
root      2624  1406  0 01:17 pts/4    00:00:00 python -m SimpleHTTPServer 80
root      2669  1406  0 01:19 pts/4    00:00:00 grep SimpleHTTPServer

mininet> h1 kill 2624
```

So what happened ?

From → to	Version	Length	Type	BufID	Reason/Action
S1 → C0	OF 1.0	158	OFPT_PACKET_IN	342	Reason: OFPR_NO_MATCH 10.0.0.3 → 10.0.0.1 TCP SYN 39109 → 80 In pt: 3
C0 → S1	OF 1.0	90	OFPT_PACKET_OUT	342	Action: OFPAT_OUTPUT In pt: 3 Out pt: 65531
S1 → C0	OF 1.0	158	OFPT_PACKET_IN	343	Reason: OFPR_NO_MATCH 10.0.0.1 → 10.0.0.3 TCP SYN, ACK 80 → 39109 In pt: 1
C0 → S1	OF 1.0	146	OFPT_FLOW_MOD	343	Action: OFPR_FLOW_MOD 10.0.0.1 → 10.0.0.3 TCP SRC: 80 TCP DST: 39109 In pt: 1 Out pt:3
S1 → C0	OF 1.0	150	OFPT_PACKET_IN	344	Reason: OFPR_NO_MATCH 10.0.0.3 → 10.0.0.1 TCP ACK 39109 → 80 In pt: 3
C0 → S1	OF 1.0	146	OFPT_FLOW_MOD	344	Action: OFPR_FLOW_MOD 10.0.0.3 → 10.0.0.1 TCP SRC: 39109 TCP DST: 80 In pt: 3 Out pt:1

h3 tried to send an Ethernet frame containing a TCP SYN message to port 80 on the *h1* webserver.

The Open vSwitch *s1* does not have a flow for this in its flow table so it encapsulated the message in an OpenFlow *OFPT_PACKET_IN* message with Buffer ID 342 and a Reason code of *OFPR_NO_MATCH*.

Controller *C0* responded with an *Output to switch port (OFPAT_OUTPUT)* message telling the Open vSwitch *s1* to send on all its ports.

When the responding SYN, ACK is received the Open vSwitch *s1* has no match for the return path either so it encapsulates in an OpenFlow *OFPT_PACKET_IN* message with Buffer ID 343 and a Reason code of *OFPR_NO_MATCH*.

This time the Ethernet port for the destination is known as a result of the earlier message so the Controller *c0* instructs the Open vSwitch *s1* with a OpenFlow Flow Modification message to map HTTP traffic for 10.0.0.1 → 10.0.0.3 in on port 1 to be forwarded to port 3.

The next response from *h3* will again be forwarded as an *OFPT_PACKET_IN* message with Buffer ID 344 and a Reason code of *OFPR_NO_MATCH* to Controller *c0*.

As the Ethernet port for *h1* is now known an OpenFlow Flow Modification message to map HTTP traffic for 10.0.0.3 → 10.0.0.1 in on port 3 to be forwarded to port 1 is sent to the Open vSwitch *s1* from the Controller *c0*.

All similar traffic to/from *h1* to *h3* will now be handled by the Open vSwitch *s1* with need for recourse to the Controller *c0* until the idle timeout of 60 seconds has passed.

11.17 Custom Topologies

The topologies created thusfar have been defined by the *mn* command options and these are limited. It will become necessary to create more customised topologies and this can be achieved using Python scripts. Mininet has example scripts in:

```
/home/mininet/mininet/examples
```

and custom scripts are created in:

```
/home/mininet/mininet/custom.
```

```
$ cd /home/mininet/mininet/custom
/home/mininet/mininet/custom$ ls
README  topo-2sw-2host.py
```

```
$ cat README
```

```
This directory should hold configuration files for custom mininets.
```

See `custom_example.py`, which loads the default minimal topology. The advantage of defining a mininet in a separate file is that you then use the `--custom` option in `mn` to run the CLI or specific tests with it.

To start up a mininet with the provided custom topology, do:

```
sudo mn --custom custom_example.py --topo mytopo
```

An example is given for a two switch solution with a host in each.

```
/home/mininet/mininet/custom$ cat topo-2sw-2host.py
```

```
"""Custom topology example
```

```
Two directly connected switches plus a host for each switch:
```

```
host --- switch --- switch --- host
```

Adding the 'topos' dict with a key/value pair to generate our newly defined

```
topology enables one to pass in '--topo=mytopo' from the command line.
"""
```

```
from mininet.topo import Topo
```

```
class MyTopo( Topo ):
```

```
    "Simple topology example."
```

```
    def __init__( self ):
```

```
        "Create custom topo."
```

```
        # Initialize topology
```

```
        Topo.__init__( self )
```

```
        # Add hosts and switches
```

```
        leftHost = self.addHost( 'h1' )
```

```
        rightHost = self.addHost( 'h2' )
```

```
        leftSwitch = self.addSwitch( 's3' )
```

```
        rightSwitch = self.addSwitch( 's4' )
```

```
        # Add links
```

```
        self.addLink( leftHost, leftSwitch )
```

```
        self.addLink( leftSwitch, rightSwitch )
```

```
        self.addLink( rightSwitch, rightHost )
```

```
topos = { 'mytopo': ( lambda: MyTopo() ) }
```

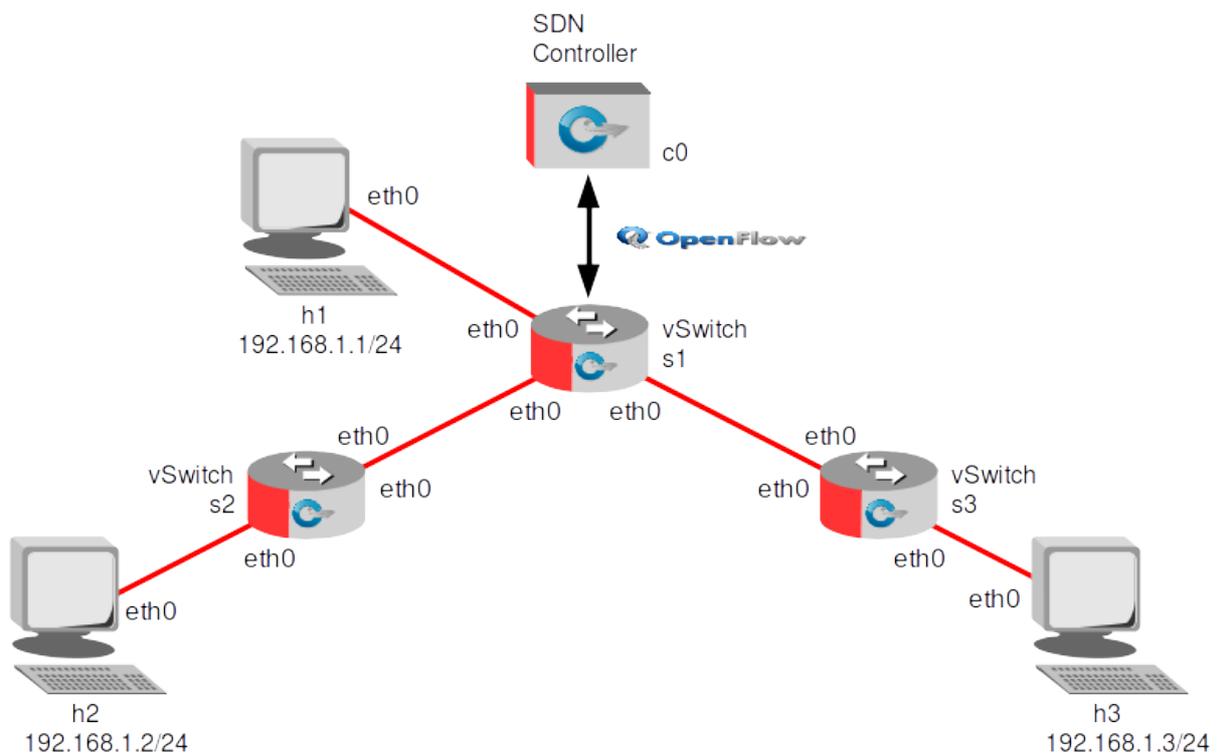
Run this example and confirm it is working.

```
$ sudo mn --custom topo-2sw-2host.py --topo mytopo
```

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s3 s4
*** Adding links:
(h1, s3) (s3, s4) (s4, h2)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 2 switches
s3 s4
*** Starting CLI:

mininet> net
h1 h1-eth0:s3-eth1
h2 h2-eth0:s4-eth2
s3 lo: s3-eth1:h1-eth0 s3-eth2:s4-eth1
s4 lo: s4-eth1:s3-eth2 s4-eth2:h2-eth0
c0
```

11.17.1 Create custom topology



Taking this diagram as an example to build. Use the existing file as a template to build the required custom topology.

```
-----Custom-OvS.py-----

#!/usr/bin/python

"""
Custom topology example

Three directly connected switches plus a host attached to each
switch with a controller (c0):

      c0
     /|\
    /  |  \
   /   |   \
  /----|----\
 h1 --- s1 | s3 --- h3
   \----|----/
    \   |   /
     \  |  /
      s2 --- h2

"""
from mininet.net import Mininet
from mininet.node import Controller
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.topo import Topo

def customNet():

    "Create a customNet and add devices to it."

    net = Mininet( controller=Controller )

    # Add controller
    info( 'Adding controller\n' )
    net.addController( 'c0' )

    # Add hosts
    info( 'Adding hosts\n' )
    h1 = net.addHost( 'h1' )
    h2 = net.addHost( 'h2' )
    h3 = net.addHost( 'h3' )

    # Add switches
    info( 'Adding switches\n' )
    s1 = net.addSwitch( 's1' )
    s2 = net.addSwitch( 's2' )
    s3 = net.addSwitch( 's3' )

    # Add links
    info( 'Adding switch links\n' )
    net.addLink( s1, s2 )
    net.addLink( s2, s3 )

    info( 'Adding host links\n' )
    net.addLink( h1, s1 )
    net.addLink( h2, s2 )
    net.addLink( h3, s3 )

    info( '*** Starting network\n' )
    net.start()
```

```

    info( '*** Running CLI\n' )
    CLI( net )

    info( '*** Stopping network' )
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    customNet()

```

Now run the new custom topology.

```

/home/mininet/mininet/custom$ sudo ./custom-OvS.py
Adding controller
Adding hosts
Adding switches
Adding switch links
Adding host links
*** Starting network
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3
*** Running CLI
*** Starting CLI:
mininet>

```

Reviewing the new network with the *dump*, *net*, *pingall*, *iperf* and *dpctl dump-flows* commands.

```

mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=1290>
<Host h2: h2-eth0:10.0.0.2 pid=1293>
<Host h3: h3-eth0:10.0.0.3 pid=1295>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=1300>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None pid=1303>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=1306>
<Controller c0: 127.0.0.1:6633 pid=1282>

mininet> net
h1 h1-eth0:s1-eth2
h2 h2-eth0:s2-eth3
h3 h3-eth0:s3-eth2
s1 lo: s1-eth1:s2-eth1 s1-eth2:h1-eth0
s2 lo: s2-eth1:s1-eth1 s2-eth2:s3-eth1 s2-eth3:h2-eth0
s3 lo: s3-eth1:s2-eth2 s3-eth2:h3-eth0
c0

mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)

```

```

mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['18.5 Gbits/sec', '18.5 Gbits/sec']

mininet> iperf h1 h3
*** Iperf: testing TCP bandwidth between h1 and h3
*** Results: ['17.8 Gbits/sec', '17.8 Gbits/sec']

mininet> iperf h2 h3
*** Iperf: testing TCP bandwidth between h2 and h3
*** Results: ['18.1 Gbits/sec', '18.1 Gbits/sec']

mininet> dpctl dump-flows
*** s1
-----
NXST_FLOW reply (xid=0x4):
*** s2
-----
NXST_FLOW reply (xid=0x4):
*** s3
-----
NXST_FLOW reply (xid=0x4):

mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=2.81 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=2.82 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.595 ms

--- 10.0.0.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.595/2.078/2.826/1.048 ms

mininet> dpctl dump-flows
*** s1
-----
NXST_FLOW reply (xid=0x4):
  cookie=0x0, duration=3.854s, table=0, n_packets=3, n_bytes=294,
  idle_timeout=60, idle_age=1,
  priority=65535,icmp,in_port=1,vlan_tci=0x0000,dl_src=b2:a6:82:06:f5:0b,dl_dst=da:
  f0:ca:b8:ca:79,nw_src=10.0.0.2,nw_dst=10.0.0.1,nw_tos=0,icmp_type=0,icmp_code=0
  actions=output:2
  cookie=0x0, duration=2.856s, table=0, n_packets=2, n_bytes=196, idle_timeout=60,
  idle_age=1,
  priority=65535,icmp,in_port=2,vlan_tci=0x0000,dl_src=da:f0:ca:b8:ca:79,dl_dst=b2:
  a6:82:06:f5:0b,nw_src=10.0.0.1,nw_dst=10.0.0.2,nw_tos=0,icmp_type=8,icmp_code=0
  actions=output:1
*** s2 -----
NXST_FLOW reply (xid=0x4):
  cookie=0x0, duration=2.859s, table=0, n_packets=2, n_bytes=196, idle_timeout=60,
  idle_age=1,
  priority=65535,icmp,in_port=1,vlan_tci=0x0000,dl_src=da:f0:ca:b8:ca:79,dl_dst=b2:
  a6:82:06:f5:0b,nw_src=10.0.0.1,nw_dst=10.0.0.2,nw_tos=0,icmp_type=8,icmp_code=0
  actions=output:3
  cookie=0x0, duration=3.86s, table=0, n_packets=3, n_bytes=294, idle_timeout=60,
  idle_age=1,
  priority=65535,icmp,in_port=3,vlan_tci=0x0000,dl_src=b2:a6:82:06:f5:0b,dl_dst=da:
  f0:ca:b8:ca:79,nw_src=10.0.0.2,nw_dst=10.0.0.1,nw_tos=0,icmp_type=0,icmp_code=0
  actions=output:1
*** s3 -----
NXST_FLOW reply (xid=0x4):

```

11.18 OpenDaylight

Project OpenDaylight is a Linux Foundation Collaborative Project. The software combines SDN components including a fully pluggable controller, interfaces, protocol plug-ins and applications to create a framework for SDN and Network Functions Virtualisation (NFV) solutions. The current release is *Helium*. In this case OpenDaylight will be used as an SDN Controller.

Create a second VirtualBox VM and install a GNU/Linux distribution like Debian GNU/Linux, Ubuntu or CentOS. Create a user called *opendaylight* with a password *opendaylight* with sudoer rights. Under the VM settings, set the network to Bridged Adapter for the interface to the network, in this way the VM will get an IP address from the network DHCP Server.

11.18.1 Install OpenDaylight

Install the following software that OpenDaylight requires.

- **maven** - Java software project management and comprehension tool.
- **git** - Distributed revision control system.
- **openjdk-7-jre** - OpenJDK Java runtime.
- **openjdk-7-jdk** - OpenJDK Development Kit.

```
$ sudo apt-get update
$ sudo apt-get install maven git openjdk-7-jre openjdk-7-jdk
```

Get the source code for the Project Daylight from:

<http://www.opendaylight.org/software/downloads>

```
$ wget
https://nexus.opendaylight.org/content/groups/public/org/opendaylight/integration/distribution-karaf/0.2.3-Helium-SR3/distribution-karaf-0.2.3-Helium-SR3.tar.gz
```

```
Resolving nexus.opendaylight.org (nexus.opendaylight.org)... 23.253.119.7
Connecting to nexus.opendaylight.org (nexus.opendaylight.org)|23.253.119.7|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 229105921 (218M) [application/x-gzip]
Saving to: `distribution-karaf-0.2.3-Helium-SR3.tar.gz'

51% [=====>] 118,466,283 210K/s eta 7m 14s
```

Build the OpenDaylight Controller and for simplicity change the directory name to *opendaylight*.

```
$ tar -xvzf distribution-karaf-0.2.3-Helium-SR3.tar.gz
$ mv distribution-karaf-0.2.3-Helium-SR3 opendaylight
```

Create JAVA_HOME.

```
$ cat << JAVARC >> ~/.bashrc

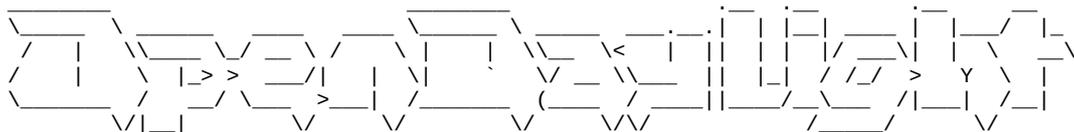
> #JAVA_HOME for OpenDaylight
> export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64
> JAVARC

$ export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64
```

11.18.2 Running OpenDaylight

Run OpenDaylight.

```
$ ~/opendaylight/bin/karaf
```



```
Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.
```

```
opendaylight-user@root>
```

11.18.3 OpenDaylight User Experience (DLUX)

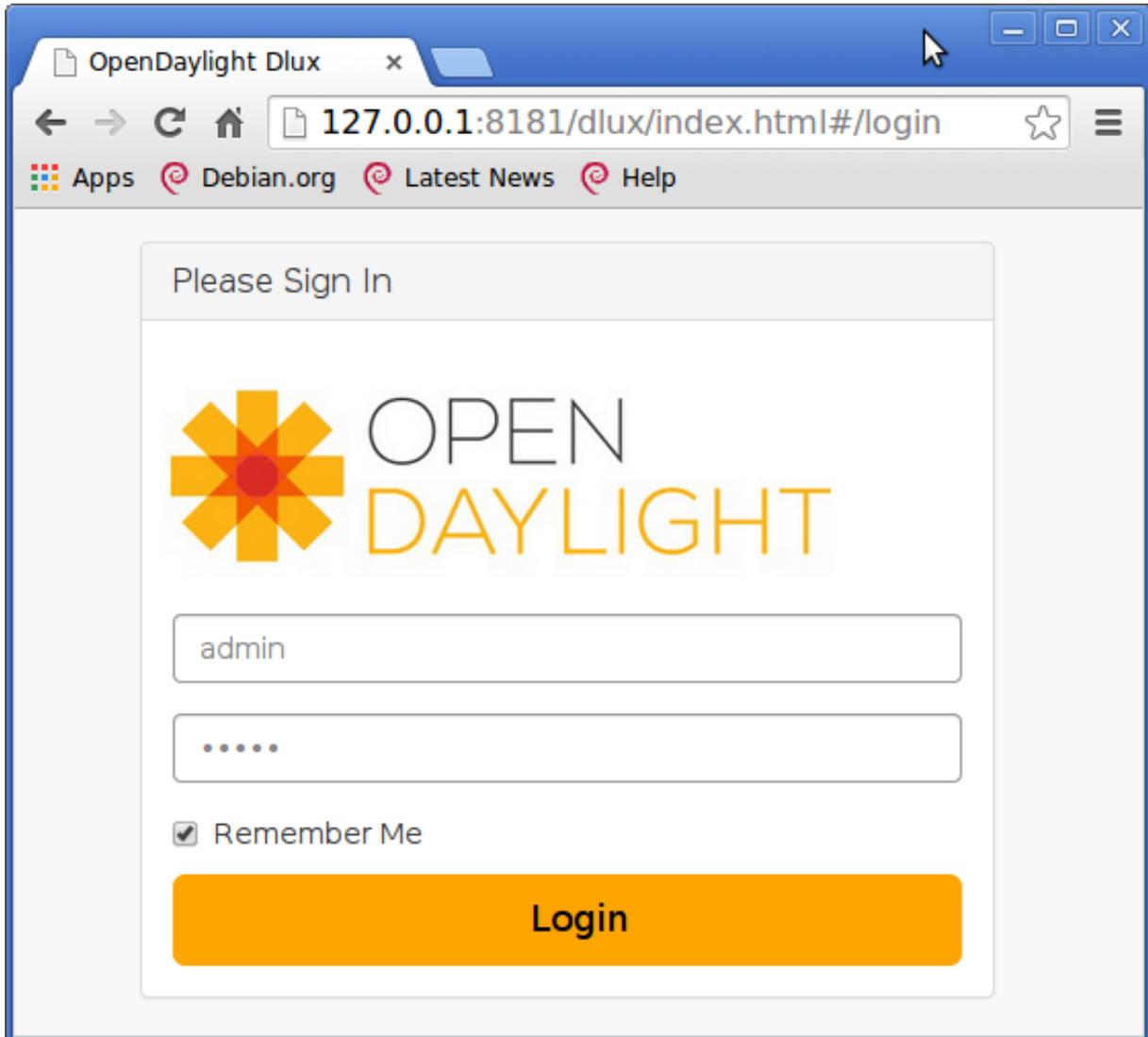
On the *karaf* shell install DLUX.

- odl-restconf -
- odl-l2switch-switch -
- odl-mdsal-apidocs -
- odl-dlux-core -

```
opendaylight-user@root> feature:install odl-restconf odl-l2switch-switch
odl-mdsal-apidocs odl-dlux-core
```

To access the DLUX dashboard a local browser is required as it is only accessible from the localhost. To access it remotely it is possible to use X11 forwarding. Install a web browser on the remote machine and run it. Then link to `//http://localhost:8181/index.html//`.

```
$ ssh -X opendaylight@192.168.25.111  
$ sudo apt-get install chromium
```



11.18.4 Start mininet network

Start a mininet network on the mininet computer. In this case point to the remote OpenDaylight remote controller on port 6633, the standard OpenFlow port. The following python script is a variant of the previous script except the controller now points to the remote Open Daylight controller.

```
-----Custom-RemoteODL.py-----

#!/usr/bin/python

"""
Custom topology example

Three directly connected switches plus a host attached to each switch
with a remote ODL SDN Controller (c0):

      c0
      /|\
ODL   /|\ 192.168.25.111
...../|\.....
Mininet /|\ 192.168.25.83
      /|\
h1 --- s1 --- s3 --- h3
      \|\
      s2 --- h2

"""

from mininet.net import Mininet
from mininet.node import Controller, RemoteController
from mininet.cli import CLI
from mininet.log import setLogLevel, info
#from mininet.topo import Topo

# OpenDayLight controller
ODL_CONTROLLER_IP='192.168.25.111'
ODL_CONTROLLER_PORT=6633

# Define remote OpenDaylight Controller

print 'OpenDaylight IP Addr:', ODL_CONTROLLER_IP
print 'OpenDaylight Port:', ODL_CONTROLLER_PORT

def customNet():

    "Create a customNet and add devices to it."

    net = Mininet( topo=None, build=False )
```

```
# Add controller
info( 'Adding controller\n' )
net.addController( 'c0',
                  controller=RemoteController,
                  ip=ODL_CONTROLLER_IP,
                  port=ODL_CONTROLLER_PORT
                )

# Add hosts
info( 'Adding hosts\n' )
h1 = net.addHost( 'h1' )
h2 = net.addHost( 'h2' )
h3 = net.addHost( 'h3' )

# Add switches
info( 'Adding switches\n' )
s1 = net.addSwitch( 's1' )
s2 = net.addSwitch( 's2' )
s3 = net.addSwitch( 's3' )

# Add links
info( 'Adding switch links\n' )
net.addLink( s1, s2 )
net.addLink( s2, s3 )

info( 'Adding host links\n' )
net.addLink( h1, s1 )
net.addLink( h2, s2 )
net.addLink( h3, s3 )

info( '*** Starting network ***\n' )
net.start()

info( '*** Running CLI ***\n' )
CLI( net )

info( '*** Stopping network ***' )
net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    customNet()
```

Run the script.

```
$ sudo ./Custom-RemoteODL.py
OpenDaylight IP Addr: 192.168.25.111
OpenDaylight Port: 6633
Adding controller
Adding hosts
Adding switches
Adding switch links
Adding host links
*** Starting network ***
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3
*** Running CLI ***
*** Starting CLI:
```

Review the topology.

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=10598>
<Host h2: h2-eth0:10.0.0.2 pid=10601>
<Host h3: h3-eth0:10.0.0.3 pid=10603>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=10608>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None pid=10611>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None pid=10614>
<RemoteController c0: 192.168.25.111:6633 pid=10591>

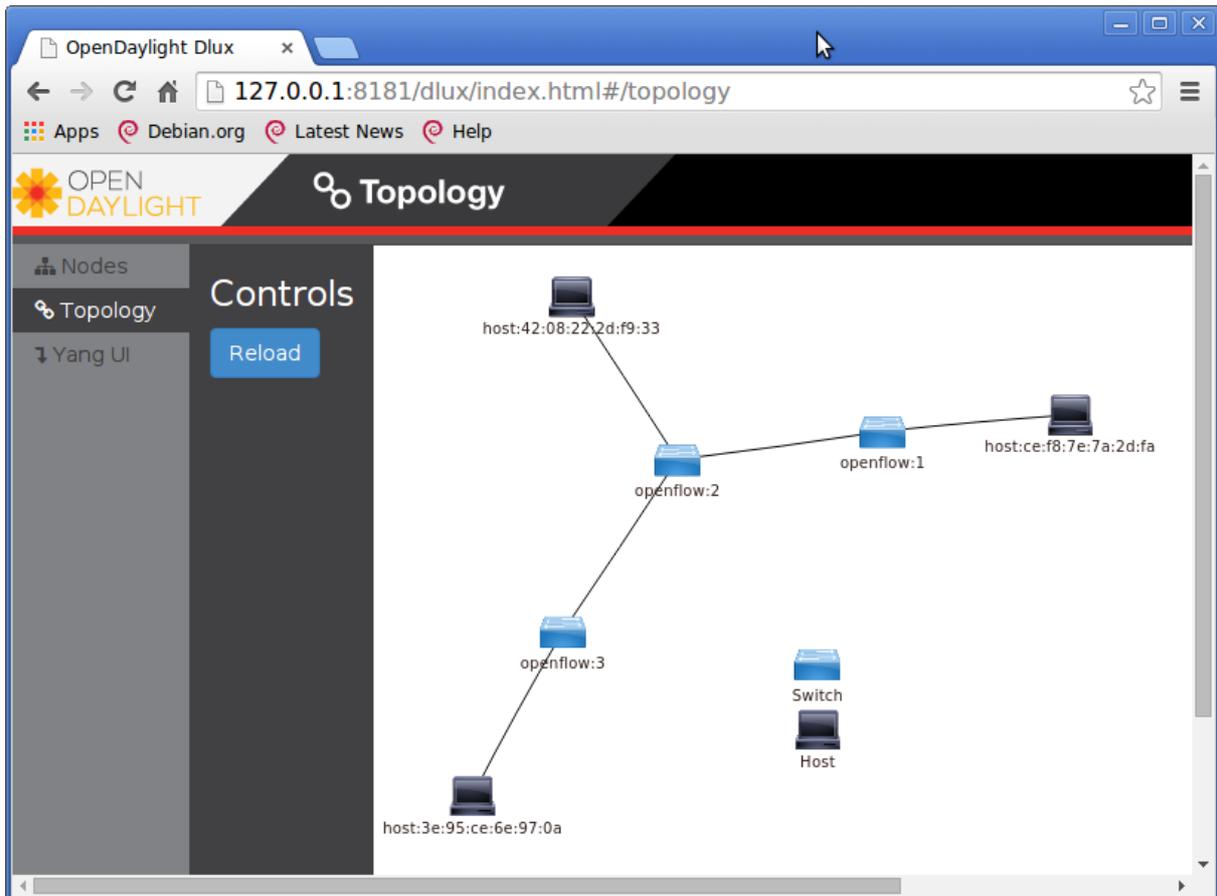
mininet> net
h1 h1-eth0:s1-eth2
h2 h2-eth0:s2-eth3
h3 h3-eth0:s3-eth2
s1 lo: s1-eth1:s2-eth1 s1-eth2:h1-eth0
s2 lo: s2-eth1:s1-eth1 s2-eth2:s3-eth1 s2-eth3:h2-eth0
s3 lo: s3-eth1:s2-eth2 s3-eth2:h3-eth0
c0
```

Test the topology.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)

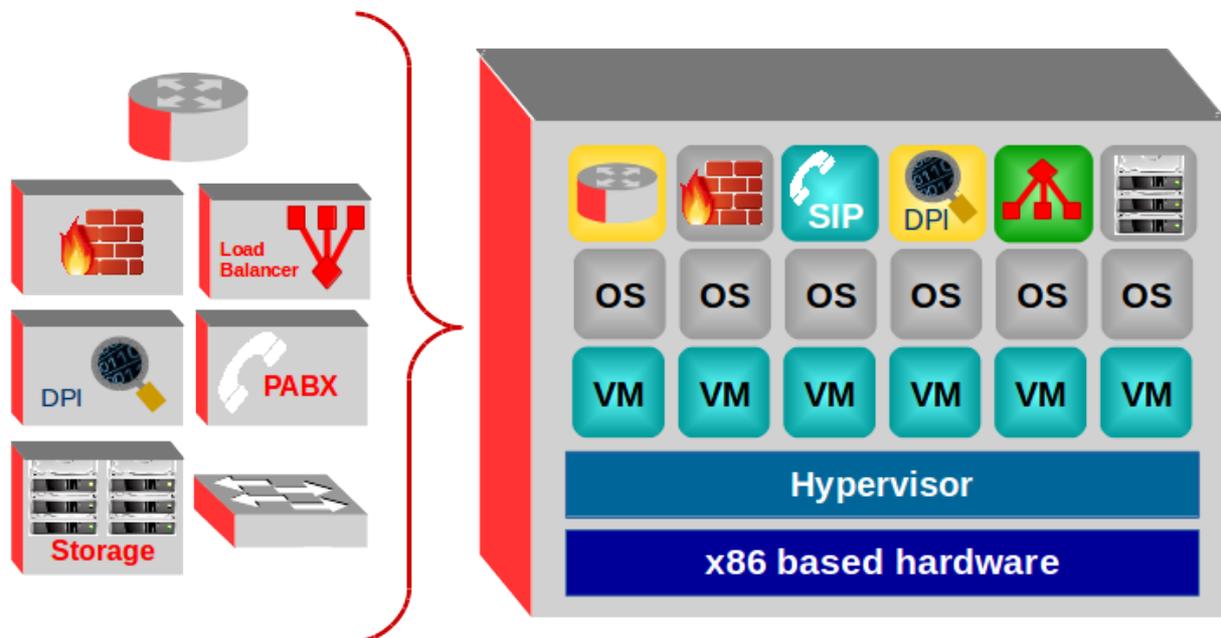
mininet> iperf h1 h3
*** Iperf: testing TCP bandwidth between h1 and h3
Waiting for iperf to start up...*** Results: ['227 Mbits/sec', '234 Mbits/sec']
```

Now look at the network in DLUX Topology dashboard.



This is a basic introduction to mininet and SDN. Further work in this area should be undertaken. To do so will require a knowledge of the Python programming language.

12. Networks Function Virtualisation (NFV)



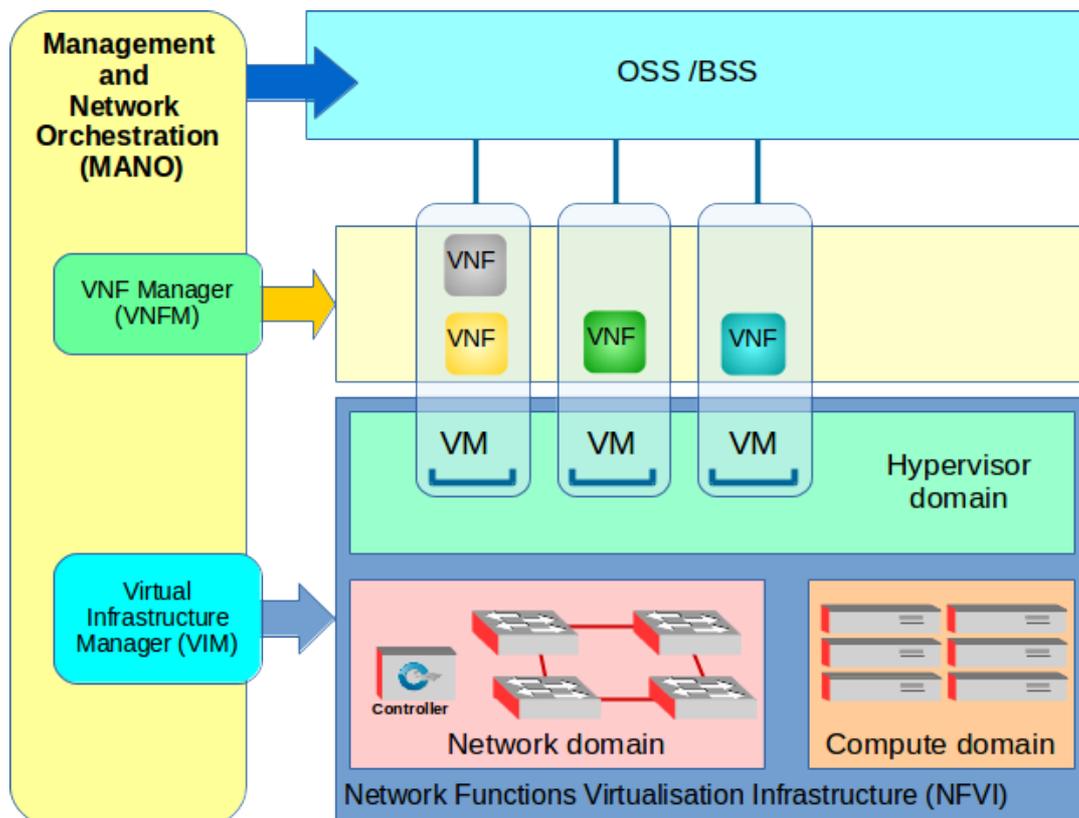
At an SDN & OpenFlow World Congress in Darmstadt, Germany in October 2012 a group of Tier 1 Service Providers launched an initiative called NFV. These operators could see that Virtualisation and Cloud computing could evolve the way services are delivered on networks by consolidation and virtualisation of network equipment on industry standard high volume servers as can be seen in the NFV concept figure above. Functions could also be migrated to centralised virtualised infrastructure while also offering the facility to push virtualisation of functions right out to the end user premises.

While SDN and NFV are complimentary to each-other they are not as yet inter-dependent and can therefore be operated either together, or independently. Obviously moving functions that were heretofore based on specialist hardware presents a number of challenges, such as;

- the portability to a virtualised system and interoperability with existing infrastructure
- the performance trade-off between standards based hardware and that of specialised, function specific hardware.
- the interaction of the Management and Network Orchestration (MANO) of the distributed functions with the network. Using the benefits of automation to achieve the transformational aspects of NFV.
- the integration of functions into the overall NFV ecosystem and its coexistence with legacy systems.
- the new challenges in terms of security and stability have evolved as a result of cloud computing and virtualisation.

These challenges and newer security challenges will evolve from this new networking system. The benefits of NFV however make the case for migration so compelling that it without doubt will form the core of services to be offered by Service Providers into the future. Hardware-based appliances have a specific life, which is getting shorter and shorter with the rapid pace of development, and they need regular replacement. This complicates maintenance procedures and customer support with no financial benefit to the Service Provider.

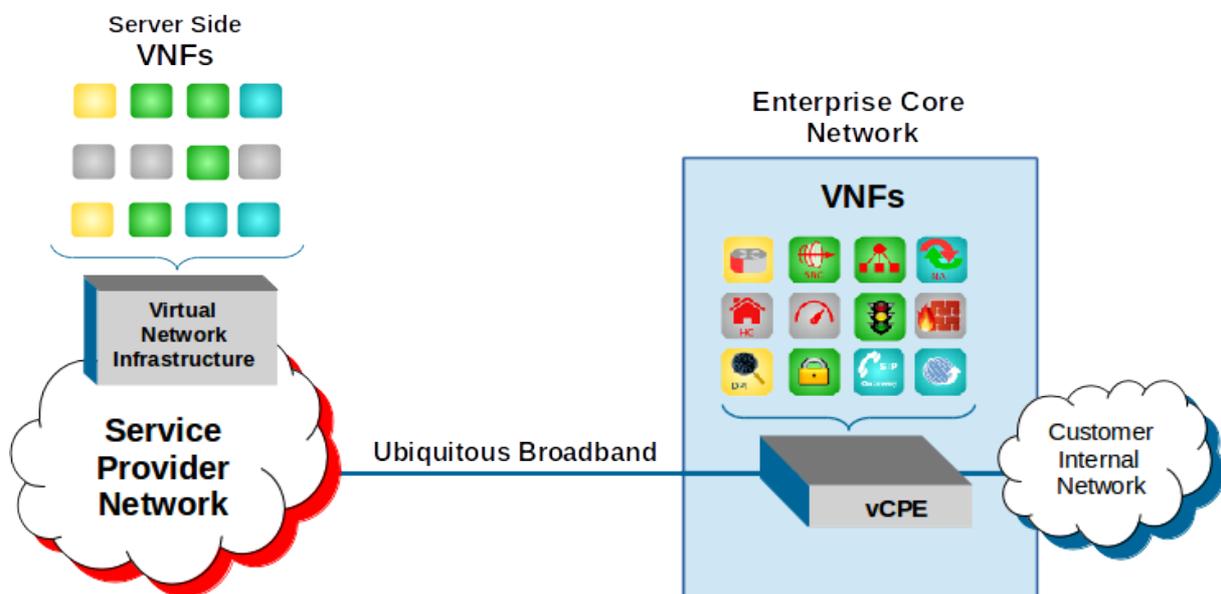
NFV will transform the design of the network to implement these functions in software, many of these will process centrally thereby allowing for their operation to be migrated and backed up as needed. This will reduce equipment costs and reduce power consumption due to power management features in standard servers and storage, while eliminating the need for specific hardware. Services can be scaled up and down in a similar fashion to that provided by cloud services today. IT MANO mechanisms familiar today in cloud services will facilitate the automatic installation and scaling of capacity by building Virtual Machines (VM) or Containers to meet demand. In this way traffic patterns and service demand can be met in an automated and managed fashion. As a result the Service Provider can increase the speed to market of both existing NFVs but also decrease the time it takes to innovate new services and deliver them on the virtualised infrastructure.



The graphic on the previous page shows the overall NFV ecosystem. The underlying infrastructure collectively called the Network Functions Virtualisation Infrastructure (NFVI) consists of three domains, Network, Compute and Hypervisor/Virtualisation. The *Network Domain* consists of islands of switches with SDN Controllers or a traditional routed and switched network we have today. The *Compute Domain* consists of the computing hardware and storage necessary to support the upper layers. The final domain in the NFVI is the Hypervisor/Virtualisation Domain which is the virtualisation hypervisors and VMs. This can be built using Hypervisors like Xen, VMWare or using Container technology like Docker. These NFVI domains are managed by a Virtual Infrastructure Manager (VIM).

A Virtual Network Function Manager (VNFM) controls the building of individual Virtual Network Functions (VNF) on the VMs. MANO performs the overall management of the VIM, VNFM and Operations Support Systems (OSS) / Business Support System (BSS) and allows the Service Provider to quickly deploy and scale VNF services as well as provide and scale resources for VNFs. This system reduces administrator workloads and removes the need for manual administration type tasks. It also offers APIs and other tooling extensions to integrate with existing environments.

12.1 Providing NFV to the customer



The roll-out of ubiquitous high speed broadband offers the Service Provider the ability to supply a Virtual Customer Premises Equipment (vCPE) to the customer upon which VNFs can be offered.

Current services that can be converted into NFV style services are:

- Router
- Session Border Controller (SBC)
- Load Balancer
- Network Address Translation (NAT)
- Home Gateway (HG)
- Application Acceleration
- Traffic Management
- Firewall
- Deep Packet Inspection (DPI)
- Bulk Encryption
- Content Caching
- Session Initiation Protocol Gateway (SIP-GW)

This however is just the beginning, these services already exist on traditional deployment mechanisms. The fact that virtualisation will now be available in the vCPE at the customer premises means that a Service Provider can deploy new services not envisaged as yet and deploy services on a trial basis, all without equipment changes.

12.2 NFV Standards

After the initial White Paper from the Darmstadt-Germany Call for Action in 2012 it was decided to form an Industry Specification Group (ISG) under ETSI. Phase 1 of this group was to *drive convergence on network operator requirements for NFV to include applicable standards, where they already exist, into industry services and products to simultaneously develop new technical requirements with the goal of stimulating innovation and fostering an open ecosystem of vendors* (ETSI, 2012). They issues a progress White Paper in October 2013 and a final paper in October 2014 which drew attention to the second release of ETSI NFV ISG documents that were subsequently published in Jan 2015. Dec 2014 was considered to be the end of phase 1 and phase 2 was launched. This will see some reorganisation of the ISG NFV working groups, to focus less on requirements and more on adoption.

The key areas that will be addressed include:

- Stability, Interoperability, Reliability, Availability, Maintainability
- Intensified collaboration with other bodies
- Testing and validation to encourage interoperability and solidify implementations
- Definition of interfaces
- Establishment of a vibrant NFV ecosystem
- Performance and assurance considerations
- Security

12.3 OPNFV

The Linux Foundation established a Collaborative Project called '*Open Platform NFV (OPNFV)*' in October 2014. The project intent is to provide an FOSS platform for the deployment NFV solutions that leverage's investments from a community of developers and solution providers. Considering many of the elements of the future platform already exist in

The initial focus of the OPNFV will be the NFVI and VIM. In reality this means the OPNFV will focus on building interfaces between existing FOSS projects like those listed below. Creating these interfaces between what are essentially existing elements to create a functional reference platform will be a major win for the technology and certainly contribute to the goals of phase 2 of the ETSI NFV ISG.

- *Virtual Infrastructure Management*: OpenStack, Apache CloudStack, etc.
- *Network Controller and Virtualization Infrastructure*: OpenDaylight, etc.
- *Virtualization and hypervisors*: KVM, Xen, libvirt, LXC, etc.
- *Virtual forwarder*: Open vSwitch (OVS), Linux bridge, etc.
- *Data-plane interfaces and acceleration*: Dataplane Development Kit (DPDK), Open Dataplane (ODP), etc.
- *Operating System*: Linux, etc.

12.4 Conclusion

The networking industry did not change significantly during the last decade, innovation was confined to port speed increase and the migration to an all-Ethernet environment. At the customer end the roll-out of ubiquitous broadband is progressing steadily and Service Providers have migrated from ATM core to all IP networks. Over the same period there was a revolution in terms of computing with the roll-out of cloud based services driven by advances in virtualisation. The innovation that brought about the cloud is about to enter networking and telecommunications in the form of SDN and NFV. Initial penetration of SDN has started in the Data Centres where the cost base of the traditional Ethernet switches and routers have driven the adoption of the new technology. As phase 2 of the NFV ISG provides working solutions over the next two years I predict that they will be adopted by the Service Providers who can deliver new services, cost effectively. This will provide a better and more functionally rich service to their customers and particularly in the case of SMEs the ability to offload existing functions to their provider rather than managing them within their own IT departments.

13. Abbreviations

6over4	IPv6 Domains via IPv4
ACK	ACKnowledge
AES	Advanced Encryption Standard
AH	Authentication header
AP	Access Point
API	Application Programming Interface
ARP	Address Resolution Protocol
AS	Autonomous System
ASCII	American Standard Code for Information Interchange
ASIC	Application Specific Integrated Circuit
ATM	Asynchronous Transfer Mode
BDR	Backup Designated Router
BGP	Border Gateway Protocol
BGP4+	BGP4 plus
bgpd	BGP Daemon
BGPv4	Border Gateway Protocol version 4
BID	Bridge ID
BoS	Bottom of Stack
BPDU	Bridge Protocol Data Unit
BSS	Basic Service Set
BSS	Business Support System
CAT	Category
CCMP	Counter Mode with Cipher Block Chaining Message Authentication Code Protocol
CDN	Content Distribution Network
CIDR	Classless Inter-Domain Routing
CORE	Common Open Research Emulator
COTS	Commercial-off-the-Shelf
CPU	Central Processing Unit
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
DHCP	Dynamic Host Configuration Protocol
DHCPv6	DHCP version 6
DiffServ	Differentiated Services
DNS	Domain Name Service
DOCSIS	Data Over Cable Service Interface Specification

DoD	Department of Defence
DPDK	Data Plane Development Kit
DPI	Deep Packet Inspection
DR	Designated Router
DSCP	DiffServ Code Point
DSL	Digital Subscriber Line
DSSS	Direct Sequence Spread Spectrum
EAP	Extensible Authentication Protocol
EAP-TLS	EAP – Transport Layer Security
ECN	Explicit Congestion Notification
EGP	External Gateway Protocol
EIGRP	Enhanced Interior Gateway Routing Protocol
EIRP	Effective Isotropic Radiated Power
ESP	Encapsulation security payload
ESS	Extended Service Set
ETSI	European Telecommunications Standards Institute
EUI	Extended Unique Identifier
FCAPS	Fault, Configuration, Accounting, Performance, and Security Management
FCC	Federal Communications Commission
FCS	Frame Check Sequence
FE	FastEthernet
FHSS	Frequency Hopping Spread Spectrum
FOSS	Free and Open Source Software
FTP	File Transfer Protocol
FWA	Fixed Wireless Access
GbE	Gigabit Ethernet
GHz	Gigahertz
GPL	GNU General Public License
GRE	Generic Routing Encapsulation
GTK	Group Temporal Key
GUI	Graphical User Interface
HDMI	High-Definition Multimedia Interface
HG	Home Gateway
HTTP	Hypertext Transfer Protocol
HV	Hypervisor
I/O	Input/Output
IBSS	Independent Basic Service Set
ICANN	Internet Corporation for Assigned Names and Numbers

ICMP	Internet Control Message Protocol
ICMPv6	ICMP version 6
ID	Identifier
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IGMP	Internet Group Membership Protocol
IGP	Interior Gateway Protocol
IGRP	Interior Gateway Routing Protocol
IMAP	Interim Mail Access Protocol
IP	Internet Protocol
IPng	IP Next Generation
IPS	Intrusion Prevention System
IPSec/SSL	IP Security/ Secure Sockets Layer
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IS-IS	Intermediate System to Intermediate System
ISATAP	Intra-Site Automatic Tunnel Addressing Protocol
ISG	Industry Specification Group
ISP	Internet Service Providers
ISSU	In Service Software Upgrade
IT	Information Technology
IV	Initialisation Vector
LAN	Local Area Network
LB	Load Balancer
LDAP	Lightweight Directory Access Protocol
LDM	Link Discovery Module
LED	Light Emitting Diode
LLDP	Link Layer Discovery Protocol
LSA	Link State Advertisement
LSDB	Link State Database
LTE	Long Term Evolution
M2M	Machine-to-Machine communications
MAC	Medium Access Control
MAN	Metropolitan Area Networks
MANO	Management and Network Orchestration
MD5	Message Digest 5
MIC	Message Integrity Check

MIMO	Multiple In, Multiple Out
MMF	Multi-Mode Fibre
MNDP	MikroTik Neighbour Discovery Protocol
MPLS	Multi-Protocol Label Switching
MU-MIMO	Multi-user MIMO
NACK	No ACK
NAT	Network Address Translation
NAT-PT	Network Address Translation/Protocol Translation
ND	Neighbour Discovery
NDP	Neighbour Discovery Protocol
NF	Network Function
NFS	Network File System
NFV	Network Functions Virtualisation
NFVI	Network Functions Virtualisation Infrastructure
NIC	Network Interface Controller
NIC	Network Interface Card
NIS	Network Information Service
Nonce	Random, arbitrary number used one time only
NSD	Network Service Descriptors
NSSA	Not So Stubby Area
ODP	Open Data Plane
OFDM	Orthogonal Frequency-Division Multiplexing
ONF	Open Networking Foundation
opdfd	OSPF Daemon
OpenFlow	Specifications developed by the Open Networking Foundation
OPNFV	Open Platform NFV
OS	Operating System
OSI	Open Standards Interconnect
OSPF	Open Shortest Path First
ospf6d	OSPF IPv6 Daemon (OSPFv3)
ospfd	OSPF Daemon (OSPFv2)
OSPFv2	Open Shortest Path First version 2
OSPFv3	Open Shortest Path First version 3
OSS	Operations Support System
PBB	Provider Backbone Bridge
PBKDF2	Password-Based Key Derivation Function version 2
PCP	Priority Code Point
PDH	Plesiochronous Digital Hierarchy

PMK	Pairwise Master Key
PON	Passive Optical Network
POP3	Post Office Protocol version 3
PSK	Pre Shared Key
PTK	Pairwise Transient Key
Q-in-Q	Queue in Queue (also QinQ)
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
RA	Router Advertisement
RADIUS	Remote Access Dialin User Service
RARP	Reverse ARP
RC4	Rivest Cipher 4
RFC	Request For Comment
RIP	Routing Internet Protocol
RIPE	European IP Research
RIPng	RIP Next Generation
RIPv2	RIP version 2
RPC	Remote Procedure Call
RS	Router Solicitation
RSN	Robust Security Network
RSTP	Rapid Spanning Tree Protocol
SA	Smart Antenna
SBC	Session Border Controller
SCTP	Stream Control Transmission Protocol
SD	Secure Digital flash memory cards
SDH	Synchronous Digital Hierarchy
SDN	Software Defined Network
SFTP	Secure FTP
SIIT	state-less IP/ICMP Translation Algorithm
SIP	Session Initiation Protocol
SIP-GW	SIP Gateway
SLA	Service Level Agreement
SLAAC	StateLess Address Auto Configuration
SMF	Single-Mode Fibre
SNMP	Simple Network Management Protocol
SONET	Synchronous Optical Networking
SPF	Shortest Path First
SPT	Shortest Path Tree

SSH	Secure Shell
SSID	Service Set Identifier
STA	Spanning Tree Algorithm
STP	Spanning Tree Protocol
SVN	SubVersioN
TC	Traffic Class
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
TKIP	Temporal Key Integrity Protocol
TSSG	Telecommunications Systems & Software Group
UCA	User Customer Address
UDP	User Datagram Protocol
vCPE	Virtual Customer Premises Equipment
VGA	Video Graphics Array
VID	VLAN ID
VLAN	Virtual Local Area Network
VLD	Virtual Link Descriptors
VM	Virtual Machine
VMWare	Proprietary Hypervisor
VNA	Virtualised Network Appliance
VNF	Virtual Network Function
VNFD	VNF Descriptors
VNFFGD	VNF Forwarding Graph Descriptors
VRE	Virtual Routing Engine
WAN	Wide Area Network
WAP	Wireless Access Point
WEP	Wireless Encryption Protocol
WiFi	Wireless Fidelity
WLAN	Wireless Local Area Network
WPA	Wi-Fi Protected Access
WPA2	Wi-Fi Protected Access version 2
Xen	Proprietary Hypervisor
YP	Yellow Pages
zebra	Quagga kernel interface
ACELP	Algebraic Code Excited Linear Prediction
ADPCM	Adaptive Differential PCM
API	Application Program Interface
ATA	Analogue Telephone Adaptors

BT	British Telecom
CODEC	COder/DECoder
CPL	Call Processing Language
CS-ACELP	Conjugate Structure ACELP
DID	Direct Inward Dialling
E1	2.048 Mb/s link with 30 64 kb/s bearer, 64 kb/s signalling and framing timeslots.
E3	34 Mb/s mutiplexed frame of 16 E1s
H.323	Call control for packet-based multimedia communications systems
HTTP	Hypertext Transfer Protocol
IAD	Integrated Access Device
IETF	Internet Engineering Task Force
IPT	IP Telephony
ITU	International Telecommunication Union
ITU-T	Telecommunication Standardisation Sector
LD-CELP	Low Delay Code Excited Linear Prediction
MG	Media Gateway
MGC	Media Gateway Controller
MGCP	Media Gateway Control Protocol
MOS	Mean Opinion Score
MPLS	Multi-protocol Label Switching
OneAPI	Services API for multimedia where subscriber ID us used to authenticate
PABX	Private Automatic Branch eXchange
PAMS	Perceptual Analysis/Measurement System
PBX	Private Branch eXchange
PCM	Pulse Code Modulation
PSQM	Perceptual Speech Quality Measurement
PSTN	Public Switched Telephony Networks
RTP	Real-Time Transport Protocol
RTSP	Real Time Streaming Protocol
SAP	Session Announcement Protocol
SDP	Session Description Protocol
SG	Signalling Gateway
SGCP	Simple Gateway Control Protocol
SIGTRAN	Signalling Transport
SIP	Session Initiation Protocol
SS7	Signalling System No. 7
SSP	Service Switching Point
STCP	Stream Control Transmission Protocol

STM1	155.52 Mbit/s Synchronous Transport Module 1
UA	User Agent
UAC	UA Client
UAS	UA Server
URI	Uniform Resource Identifier
VoIP	Voice over Internet Protocol
WFQ	Weighted Fair Queuing
XML	eXtensible Markup Language

14. Bibliography

(2012). Network Functions Virtualisation - An Introduction, Benefits, Enablers, Challenges & Call for Action (White Paper). Darmstadt-Germany. 22-24 Oct 2012. Available: http://portal.etsi.org/NFV/NFV_White_Paper.pdf [28 Mar 2015].

Project Mininet. (2015). Available: <http://mininet.org> [accessed: 31 Mar 2015].

Simon, D., Aboba, B., Hurst, R. (2008). RFC 5216. The EAP-TLS Authentication Protocol. Available: <http://tools.ietf.org/html/rfc5216> [accessed: 12 Apr 2014].

(2013). Network Functions Virtualisation (NFV) - Network Operator Perspectives on Industry Progress (Update white paper). Frankfurt-Germany. 15-17 Oct 2013. Available: http://portal.etsi.org/NFV/NFV_White_Paper2.pdf [accessed: 28 Mar 2015].

(2014). Network Functions Virtualisation (NFV) - Network Operator Perspectives on Industry Progress (Update white paper). Frankfurt-Germany. 14-17 Oct 2014. Available: https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf [accessed: 28 Mar 2015].

Aboba, B., Blunk, L., Vollbrecht, J. Carlson, J., Levkowitz, H. (2004). RFC 3748. Extensible Authentication Protocol (EAP). Available: <http://tools.ietf.org/html/rfc3748> [accessed: 8 Mar 2015].

Andreasen, F. and Foster, B. (2003). RFC 3435. Media Gateway Control Protocol (MGCP) Version 1.0. Available: <http://tools.ietf.org/rfc/rfc3435.txt> [accessed 09 Apr 2015].

Aoun, C., Davies, E. (2007). RFC 4966 - Reasons to Move the NAT-PT to Historic Status. Available: <http://tools.ietf.org/html/rfc4966> [accessed: 8 Mar 2015].

Arango, M. Dugan, A., Elliott, I., Huitema, C. and Pickett, S. (1999). RFC 2705. Media Gateway Control Protocol (MGCP) Version 1.0. Available: <http://tools.ietf.org/rfc/rfc2705.txt> [accessed 09 Apr 2015].

Bates, T., Rekhter, Y., Chandra, R., Katz, D. (2000). RFC 2858 - Multiprotocol Extensions for BGP-4. Available: <http://tools.ietf.org/html/rfc2858> [accessed: 23 Mar 2014].

Cisco. (2006). Understanding Rapid Spanning Tree Protocol (802.1w). Available: <http://www.cisco.com/c/en/us/support/docs/lan-switching/spanning-tree-protocol/24062-146.html> [accessed: 3 Aug 2014].

Coltun, R., Ferguson, D., Moy, J. and Lindem, A. (2008). RFC5340. OSPF for IPv6. Standards Track. Available: <http://tools.ietf.org/html/rfc5340> [accessed: 10 Mar 2015].

Deering, S., Hinden, R. (1998). RFC 2460 - Internet Protocol, Version 6 (IPv6) Specification. Available: <https://tools.ietf.org/rfc/rfc2460.txt> [accessed: 8 Mar 2015].

Dell Inc. (2013). Software-Defined Networking (SDN) Deployment Guide v1.0. 28 Feb 2013. Available: https://www.force10networks.com/CSPortal20/KnowledgeBase/DOCUMENTATION/InstallGuidesQuickrefs/SDN/SDN_Deployment_1.0_28-Feb-2013.pdf [accessed: 29 Mar 2015]

Docker Project. (2015). Available: <https://www.docker.com> [accessed: 29 Mar 2015].

Droms, R. (1997). Dynamic Host Configuration Protocol. Available: <http://www.ietf.org/rfc/rfc2131.txt> [accessed: 31 Feb 2014].

Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., Carney, M. (2003). Dynamic Host Configuration Protocol for IPv6 (DHCPv6). Available: <http://www.ietf.org/rfc/rfc3315.txt> [accessed: 23 Mar 2014].

ETSI NFV (2015). <http://www.etsi.org/technologies-clusters/technologies/nfv> [accessed: 29 Mar 2015].

Faughnan, L. (2013). Software Defined Networking. 1 May 2013. Available: <http://www.techcentral.ie/software-defined-networking> [accessed: 28 Mar 2015].

Fredrich, T. (2013). RESTful Service Best Practices - Recommendations for Creating Web Services. Pearson eCollege.

Giraud, J.P. (2012). WPA support in Debian. <https://wiki.debian.org/WPA> [accessed: 24 May 2014].

Greene, N., Ramalho, M., Rosen, B. (2000). RFC 2805: Media Gateway Control Protocol Architecture and Requirements. Available: <https://www.ietf.org/rfc/rfc2805.txt> [accessed 14 Apr 2015].

Groves, C., Pantaleo, M., Anderson, T., Taylor, T. (2003). RFC 3525: Gateway Control Protocol Version 1. Available: <https://www.ietf.org/rfc/rfc3525.txt> [accessed 14 Apr 2015].

Handley, M., Jacobson, V., Perkins, C. (2006). RFC 4566: SDP: Session Description Protocol. Available: <https://www.ietf.org/rfc/rfc4566.txt> [accessed 14 Apr 2015].

Handley, M., Schulzrinne, H., Schooler, E and Rosenberg, J. (1999). RFC 2543. SIP: Session Initiation Protocol. Available: <http://tools.ietf.org/rfc/rfc2543.txt> [accessed 09 Apr 2015].

Hertzog, R., Mas, R. (2013). The Debian Administrator's Handbook. Chapter 10. Network Infrastructure. ISBN 9791091414029. Available: <http://debian-handbook.info/browse/stable/network-infrastructure.html> [accessed: 15 May 2014].

Hertzog, R., Mas, R. (2013). The Debian Administrator's Handbook. Chapter 10. Network Infrastructure. ISBN 9791091414029. Available: <http://debian-handbook.info/browse/stable/network-infrastructure.html> [accessed: 16 May 2014].

Hinden, R., Deering, S. (2006). RFC 4291 - IPv6 Addressing Architecture. Available: <http://tools.ietf.org/html/rfc4291> [accessed: 8 Mar 2015].

Hinden, R., Haberman, B. (2005). RFC 4193 - Unique Local IPv6 Unicast Addresses. Available: RFC 4193 <http://tools.ietf.org/html/rfc4193> [accessed: 8 Mar 2015].

IEEE. (1997). 802.11-1997. Wireless Local Area Networks.

IEEE. (1997). IEEE 802.11 - Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.

IEEE. (1998). 802.1D-1998 - Standard for Local Area Network MAC (Media Access Control) Bridges.

- IEEE. (1999). 802.11a-1999. 54 Mbit/s, 5 GHz WiFi standard.
- IEEE. (1999). 802.11b-1999. Enhancements to 802.11 to support 5.5 and 11 Mbit/s
- IEEE. (1999). IEEE 802.11a - High-speed Physical Layer in the 5 GHz band.
- IEEE. (1999). IEEE 802.11b - Higher Speed Physical Layer Extension in the 2.4 GHz band.
- IEEE. (2001). 802.1X - Port Based Network Access Control.
- IEEE. (2003). 802.11g-2003. 54 Mbit/s, 2.4 GHz standard.
- IEEE. (2003). IEEE 802.11g - Further Higher Data Rate Extension in the 2.4 GHz Band.
- IEEE. (2004). 802.11i-2004 - WLAN MAC and Physical Layer (PHY) specifications, MAC Security Enhancements.
- IEEE. (2004). 802.11i: WiFi Enhanced security.
- IEEE. (2004). 802.1D-2004 - Standard for Local and metropolitan area networks: Media Access Control (MAC) Bridges.
- IEEE. (2004). 802.1W-2004 - Rapid Reconfiguration of Spanning Tree (incorporated into 802.1D-2004).
- IEEE. (2005). 802.1ad-2005 - Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks - Amendment 4: Provider Bridges.
- IEEE. (2007). 802.11-2007. Rollup of previous standards.
- IEEE. (2009). 802.11n-2009. Higher throughput improvements using Multiple Input, Multiple Output (MIMO) antennas.
- IEEE. (2009). IEEE 802.11n - Information technology — Local and metropolitan area networks.
- IEEE. (2011). 802.1Q-2011 - Standard for Local and metropolitan area networks - Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks.
- IEEE. (2012). 802.11-2012. Rollup of previous standards.
- IEEE. (2013). 802.11ac-2013 - Very High Throughput with improved modulation scheme, wider channels and multi user MIMO.
- IEEE. (2013). IEEE 802.11ac - Telecommunications and information exchange between systems — Local and metropolitan area networks.
- Ishiguro, K. et al. (2013). Quagga. A routing software package for TCP/IP networks. Available: <http://www.nongnu.org/quagga/docs/quagga.pdf> [accessed: 3 Aug 2014].

ITU-T. (2010). 05/1998 Q.931: ISDN user-network interface layer 3 specification for basic call control. Geneva, ITU-T.

ITU-T. (2010). 12/2009 H.323: Packet-based multimedia communications systems. Geneva, ITU-T.

ITU-T. (2013). 03/2013 H.248: Gateway control protocol: Version 3. Geneva, ITU-T.

Khetrapal, G., Sharma, S.K. (2013). Demystifying Routing Services in SDN. Aricent Group. Available: <http://www.aricent.com/sites/default/files/pdfs/Aricent-Demystifying-Routing-Services-SDN-Whitepaper.pdf> [accessed: 29 Mar 2015].

Lennox, J., Schulzrinne, H., Rosenberg, J. (2001). RFC 3050: Common Gateway Interface for SIP. Available: <https://www.ietf.org/rfc/rfc3050.txt> [accessed 14 Apr 2015].

Linux Foundation. (2014). OPNFV - An open platform to accelerate NFV. 30 Oct 2014. Available: https://www.opnfv.org/sites/opnfv/files/pages/files/opnfv_whitepaper_103014.pdf [accessed: 28 Mar 2015].

Linux Foundation. (2014). OPNFV - An open platform to accelerate NFV. 30 Oct 2014. https://www.opnfv.org/sites/opnfv/files/pages/files/opnfv_whitepaper_103014.pdf [28 Mar 2015].

LitePoint. (2013). IEEE 802.11ac: What Does it Mean for Test? Available: http://litepoint.com/whitepaper/80211ac_Whitepaper.pdf [accessed: 12 Apr 2014].

Mahy, R. (2004). RFC 3680: A Session Initiation Protocol (SIP) Event Package for Registrations. Available: <https://www.ietf.org/rfc/rfc3680.txt> [accessed 14 Apr 2015].

Mahy, R. (2004). RFC 3842: A Message Summary and Message Waiting Indication Event Package for the Session Initiation Protocol (SIP). Available: <https://www.ietf.org/rfc/rfc3842.txt> [accessed 14 Apr 2015].

Malinen, J. (2013). Linux WPA/WPA2/IEEE 802.1X Supplicant. Available: http://w1.fi/wpa_supplicant/ [accessed: 24 May 2014].

Marques, P., Dupont, F. (1999). RFC 2545 - Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing. Available: <http://tools.ietf.org/html/rfc2545> [accessed: 23 Mar 2014].

Moy, J. (1991). RFC1247. Open Shortest Path First Version 2. Draft Standard. Available: <http://tools.ietf.org/html/rfc1247> [accessed: 31 Feb 2014].

Moy, J. (1994). RFC1583. Open Shortest Path First Version 2. Standards Track. Available: <http://tools.ietf.org/html/rfc1583> [accessed: 31 Feb 2014].

Moy, J. (1997). RFC2178. Open Shortest Path First Version 2. Standards Track. Available: <http://tools.ietf.org/html/rfc2178> [accessed: 22 Mar 2014].

Moy, J. (1998). RFC2328. Open Shortest Path First Version 2. Standards Track. Available: <http://tools.ietf.org/html/rfc2328> [accessed: 22 Mar 2014].

Nessjøen, H. (2007). Open source MAC Telnet server and client based on MNDP. Available: <https://github.com/haakonnessjoen/MAC-Telnet> [accessed: 8 Mar 2015].

- Niemi, A., Willis, D. (2010). RFC 5839: An Extension to Session Initiation Protocol (SIP) Events for Conditional Event Notification. Available: <https://www.ietf.org/rfc/rfc5839.txt> [accessed 14 Apr 2015].
- NIST. (2001). FIPS PUB 197 - Advanced Encryption Standard (AES).
- Olivé, E. P. (2010). Open Networks. First Edition. Unoversitat Oberta de Catalunya, Free Technology Academy.
- Open Networking Foundation (2012). Software-Defined Networking: The New Norm for Networks (White paper). 13 April 2012. Available: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf> [accessed: 28 Mar 2015].
- Open Networking Foundation (2013). What is the ONF. Available: <https://www.opennetworking.org/images/stories/downloads/about/onf-what-why.pdf> [accessed: 28 Mar 2015].
- Open Networking Foundation (2015). Software-Defined Networking (SDN) Definition. [online] <https://www.opennetworking.org/sdn-resources/sdn-definition> [accessed: 28 Mar 2015].
- Open Networking Foundation. (2014). OpenFlow Switch Specification v1.5. 19 Dec 2014. Available: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf> [accessed: 29 Mar 2015].
- OpenSSH Project. (2015). Available: <https://openssh.com> [accessed: 31 Jul 2014].
- OpenVPN Project. (2015). Available: <https://openvpn.net> [accessed: 31 Jul 2014].
- OPNFV. (2015). Available: <http://opnfv.org> [accessed: 28 Mar 2015].
- OPNFV. (2015). OPNFV - An open platform to accelerate NFV. 30 Oct 2014. Linux Foundation. Available: https://www.opnfv.org/sites/opnfv/files/pages/files/opnfv_whitepaper_103014.pdf [accessed: 28 Mar 2015].
- Oracle VirtualBox. (2015). <https://www.virtualbox.org> [accessed: 29 Mar 2015].
- Project Floodlight. (2015). Available: <http://www.projectfloodlight.org/floodlight/> [accessed: 28 Mar 2015].
- RealVNC Software. (2014). Available: <https://www.realvnc.com/> [accessed: 31 Jul 2014].
- Rosen, R. (2014). Linux Kernel Networking: Implementation and Theory. Apress. ISBN-13: 978-1430261964.
- Rosenberg, J. (2004). RFC 3856: A Presence Event Package for the Session Initiation Protocol (SIP). Available: <https://www.ietf.org/rfc/rfc3856.txt> [accessed 14 Apr 2015].
- Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R. and Handley, M. (2002). RFC 3261. SIP: Session Initiation Protocol. Available: <http://tools.ietf.org/rfc/rfc3261.txt> [accessed 09 Apr 2015].
- Rosenberg, J., Schulzrinne, H., Camarillo, G. (2005). RFC 4168: The Stream Control Transmission Protocol (SCTP) as a Transport for the Session Initiation Protocol (SIP). Available: <https://www.ietf.org/rfc/rfc2805.txt> [accessed 14 Apr 2015].

Schulzrinne, H. and Casner, S. (2003). RFC 2551. RTP Profile for Audio and Video Conferences with Minimal Control. Available: <https://www.ietf.org/rfc/rfc3551.txt> [accessed 09 Apr 2015].

Schulzrinne, H. Casner, S., Frederick, R. and Jacobson, V. (1996). RFC 1889. RTP: A Transport Protocol for Real-Time Applications. Available: <http://tools.ietf.org/rfc/rfc1889.txt> [accessed 09 Apr 2015].

Schulzrinne, H. Casner, S., Frederick, R. and Jacobson, V. (2003). RFC 3550. RTP: A Transport Protocol for Real-Time Applications. Available: <http://tools.ietf.org/rfc/rfc3550.txt> [accessed 09 Apr 2015].

Schulzrinne, H., Rao, A., Lanphier, R. (1998). RFC 2326: Real Time Streaming Protocol (RTSP). Available: <https://www.ietf.org/rfc/rfc2326.txt> [accessed 14 Apr 2015].

Schulzrinne, H., Rao, A., Lanphier, R. (1998). RFC 2326: Real Time Streaming Protocol (RTSP). Available: <https://www.ietf.org/rfc/rfc2326.txt> [accessed 14 Apr 2015].

Simmons, G. (2013). Bridging Network Connections. Available: <https://wiki.debian.org/BridgeNetworkConnections> [accessed: 24 May 2014].

Smith, A. (2010). iproute2: Life after ifconfig. Available: <http://andys.org.uk/bits/2010/02/24/iproute2-life-after-ifconfig/> [accessed: 24 Mar 2014].

Thompson, S., Huitema, C., Ksinant, V., Souissi, M. (2003). RFC 3596 - DNS Extensions to Support IP Version 6. Available: <http://tools.ietf.org/html/rfc3596> [accessed: 8 Mar 2015].

Thompson, S., Narten, T., Jinmei, T. (2007). RFC 4862 - IPv6 Stateless Address Autoconfiguration. Available: <http://www.ietf.org/rfc/rfc4862.txt> [accessed: 10 Mar 2015].

TightVNC Software. (2014). Available: <http://www.tightvnc.com> [accessed: 31 Jul 2014].

US Naval Research Laboratory (2015). Common Open Research Emulator (CORE) 4.7 Documentation. Available: <http://downloads.pf.itd.nrl.navy.mil/docs/core/core-html/> [accessed: 20 Apr 2015].

Van Styn, H. (2011). VLAN Support in Linux. Linux Journal. ([accessed: 3/8/2014]. Available: <http://www.linuxjournal.com/article/10821> [accessed: 22 Apr 2014].

VMWare Inc. (2015). Available: <http://www.vmware.com> [accessed: 29 Mar 2015].

Xen Project. (2015). Available: <http://www.xenproject.org> [accessed: 29 Mar 2015].

15. GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <<http://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a

Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another

language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a

complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as

given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may

be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the

present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.