Implementation and Evaluation of Virtual Network Functions Performance in the Home Environment

Clive Burke

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden

This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of MSc in EE with focus on Telecommunication Systems. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author: Clive Burke
E-mail: clbu05@students.bth.se

University advisor:
Kurt Tutschku
Department of Communication Systems (DIKO)

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

ABSTRACT

Networks Functions Virtualization in a Home environment is being discussed and trialed extensively. People mention that it is a “game changer”. The main purpose of this thesis is to prove virtualization works and is better than existing home network environments.

In the Related Works section we explore other topics which relate to what we set out to achieve. There are some interesting related works which we describe here, mainly SoftEther and OpenWRT. Taking both subjects together, try to get them working with each other to achieve a Virtual Dynamic Host Control Protocol, VDHCP server. We also explore another important topic on how to tether a Linux Customer Premises Equipment, CPE, for IP Service Delivery. A thesis completed in BTH was also reviewed. This discusses the various different VPN solutions available to use in today’s Internet.

The next chapter in my thesis, Methodology, will describe how we designed, implemented and configured a system to achieve Virtualizing a Network Function in the Home Environment, more specifically DHCP. We start with an introduction of what we want to achieve and how to achieve it. We then go on to detail what NFV Architecture is made up of. Following this, we will detail how we designed the system this was and why. We also explain some Key Functions of the system and why they play an important part of the overall setup. The final part of the method will explain how the system was implemented and also configured.

The methods chapter is followed by the results we gathered using various monitoring and test simulation software. We detail different iterations and scenarios to explain if a Virtualized Network Function is better in the cloud or not.

The final chapter of this thesis will be conclusions. This section will discuss the advantages and disadvantages of using NFV in the Home environment. From the results we achieve with my measurements and monitoring it is described how better to achieve NFV in this scenario. We will also discuss future work, how to improve the system for the home user and what other areas could be explored to enhance the user experience with their home internet environment. We will detail how one could also benefit on working with a more secure, reliable and scalable network.

Keywords: NFV, DHCP, Residential, Virtualization

ABBREVIATIONS

The below details important abbreviations mentioned in this thesis:

QoS	Quality of Service
PSTN	Public Switched Telephone Network
TCP	Transmission Control Protocol
IPSec	Internet Protocol Security
FTP	File Transport Protocol
NAT	Network Address Translation
ISP	Internet Service Provider
MPLS	Multiprotocol Label Switching
UDP	User Datagram Protocol
HTTP	Hypertext Transfer Protocol
VDHCP	Virtual Dynamic Host Control Protocol
CPE	Customer Premises Equipment
NFV	Network Functions Virtualization
TLS	Transport Layer Security
VLAN	Virtual Local Area Network
L2TP	Layer 2 Tunneling Protocol
URL	Uniform Resource Locator
NFVC	Network Function Virtualization Component
CAPEX	Capital Expenditure
OPEX	Operating Expenditure
COTS	Commercial off the Shelf
DNS	Domain Name System
SSP	Service Switching Point
SDN	Software Defined Networking
POC	Proof of Concept
GUI	Graphical User Interface
LuCI	Lua Unified Configuration Interface
EC2	Elastic Compute Cloud
BOOTP	Bootstrap Protocol
NFVI	Network Function Virtualization Infrastructure
SGSN	Serving GPRS Support Node
GGSN	Gateway GPRS Support Node
HLR	Home Location Register
PDN	Packet Data Network
EMS	Element Management System
SSTP	Secure Socket Tunneling Protocol
SSH	Secure Shell
UNIX	Uniplex Information and Computing System
AWS	Amazon Web Service
LAN	Local Area Network
WAN	Wide Area Network
VPNC	Virtual Private Network Cascade
AMI	Amazon Machine Image
AP	Access Point

CONTENTS

1. Introduction
- 1.1 Background……3
- 1.2 Aims and Objectives……4
- 1.3 Research Questions……4
- 1.4 Expected Contribution……5

2. Related Work
- 2.1 Parents Controlled Home Internet Cafes……6
- 2.2 Design and Implementation of SoftEther VPN……6
- 2.3 Investigation of different VPN Solutions ……7
- 2.4 Tethered Linux CPE for IP Service Delivery……7

3. Methodology
- 3.1 Introduction……9
- 3.2 NFV Architecture & Background……10
- 3.3 System Design…… 12
  - 3.3.1 Firmware and Router Selection……12
  - 3.3.2 SoftEther……14
  - 3.3.3 OpenWRT……15
  - 3.3.4 Amazon EC2……17
  - 3.3.5 Test Tools……17
- 3.4 Key Function Design……19
  - 3.4.1 Wireless Relay……19
  - 3.4.2 Cascading……20
  - 3.4.3 SSL VPN……21
- 3.5 Implementation……23
  - 3.5.1 Development Environment……23
  - 3.5.2 Amazon EC2……23
  - 3.5.3 OpenWRT……25
  - 3.5.4 SoftEther……30
  - 3.5.5 DHCP Relay……31
- 3.6 Configuration……32
  - 3.6.1 SoftEther on OpenWRT……32
  - 3.6.2 SoftEther on Windows……33

4. Results
- 4.1 Virtual DHCP leases……35
- 4.2 DHCP testing……37

5. Analysis
- 5.1 NFV Traffic Analysis……40
- 5.2 Virtual Network Operation……40

6. Conclusion and Future Work
- 6.1 Conclusion……41
- 6.2 Discussion and contributions……41
- 6.3 Further work……42
  - 6.3.1 Better reliability……42
  - 6.3.2 Improvements in System Performance……42
  - 6.3.3 Improved Fault Management……43

1. Introduction

1.1 Background

Network Functions Virtualization, NFV, aims to transform the way network environments are designed by evolving the current virtualization techniques. There are 4 main types of virtualization techniques, Guest Operating System Virtualization, Shared Kernel Virtualization, Kernel Level Virtualization and Hypervisor Virtualization. A VNF can consist of one or more virtual machines running different software and processes, running on top of high volume servers, storage, routers and switches, in our case cloud computing infrastructure, instead of having proprietary hardware devices for each network function. This process includes network functions and consolidation of many physical network components onto industry standard high volume servers, switches and storage. The aforementioned components would have usually been located in datacenters and end user premises. The service quality delivered by a Network Function Virtualization Component, NFVC, to end users is dependent on the service quality of the compute, network and other resources delivered by NFV infrastructure [1].

The high level objectives of NFV include rapid service innovation through software based deployment and operationalization of network functions and end to end services. It provides improved operational efficiency resulting from common automation and operating procedures. Reduced power usage can be achieved by the migration of workloads and reducing the number of servers in operation at any one time, only powering on servers when needed due to capacity. Interoperability is a key player in the deployment with greater flexibility in assigning VNFs to hardware. Reduced CAPEX and OPEX will also be achieved.

The separation of network functions away from physical hardware yields many benefits for the Service Provider and also the customer. Some of the benefits include [2]

- Reduction of space needed for network hardware
- Reduction of power consumption
- Reduction of maintenance costs
- Easier and more flexible upgrades and new feature rollouts
- Longer equipment life cycles
- Reduced maintenance and hardware costs

Today's CPE, Customer Premises Equipment, in the home typically involves a router which connects you to the Internet and also carries out many network functions. When this is virtualised it will move these functions into the cloud [3].

In this thesis, we describe Network Functions Virtualizations Architecture and implement NFV to compare with the physical equivalent. This thesis aims to propose virtualized home network architecture, modelled and implemented through a Virtual Appliance. This is a thesis focused on experimental research where we implement Proof of Concept to demonstrate the virtualization of a Network Function, DHCP.

In an NFV environment it a requirement to provide service continuity and seamless failover in the event of any failure and thus it is necessary to minimize the impact on end-to-end user services. Any service in the virtual environment has the ability to match or improve security in the physical environment.

For this implementation thesis we am going to try to stick with the Robustness principle [4], *"Be conservative in what you do, be liberal in what you accept from others*". This

can be defined as code that sends commands or data to other machines (or to other programs on the same machine) should conform completely to the specifications, but code that receives input should accept non-conformant input as long as the meaning is clear.

Postel's Law, as it's also known, was inspired by an Internet pioneer Jon Postel, who also wrote an early specification of Transmission Control Protocol.

The motivation behind this thesis is to create a prototype to ease the issues that arise when configuring, maintaining and deploying Home Routers and services that run on these routers. Existing routers based on propitiatory hardware and software are one of the more expensive parts of a service providers infrastructure. The can use up limited control plane resources before they use up data plane resources hence the reason why it would be preferable to virtualize the control plane functions into the cloud to improve scalability and improve costs.

This thesis produced an extremely functional mobile NFV device that has many other potential functions. One for example is an instant secure Local Area Network that is created once our router is plugged in. This has many other worthy uses which we will detail throughout the thesis.

1.2 Aims and Objectives

The main objective of this thesis is to replace proprietary hardware and operating systems with a virtual infrastructure built onto Commercial off the Shelf, COTS, hardware, employing standard hypervisors and virtual machines [5]. Any function in the virtual environment should equal or improve that in the physical environment. It will be important to show the benefits of using NFV. This will consist of backing up our claims of the benefits by providing accurate measurements that prove this.

The main objectives of the thesis will accommodate:

- Modelling of a physical home network infrastructure.
- Modelling of a virtualized home network infrastructure.
- The function that will be virtualized are Dynamic Host Configuration Protocol, DHCP.
- Implementation of both models in the physical and virtual environment respectively.
- Relocation of a VM, which executes that VNF, to another host system to demonstrate redundancy and diversity.
- Validating the model by analysing network traffic and to compare latency and packet loss.
- Analysis of what Functions can be moved into the virtual world and what functions are necessary to remain in the Residential Gateway.

1.3 Research Questions

1. How do you design a Virtual Network Function efficiently to maximize performance in order to replace their physical equivalent CPE, Customer Premises Equipment?

2. In what way can a Virtualized Network Function perform the same or better than the physical equivalent?
3. What Functions are better optimized virtually and what are better to remain in the home?

1.4 Expected Contribution

The expected outcomes of this thesis are:

- To design and implement a working Network Function in a virtualized environment.
- To obtain meaningful results from the 1st iteration of experiments and more accurate results from iterations following this after a circular approach model.
- Evidence that a Virtualized Network Function can perform the same or better that its equivalent physical function.
- To prove that having a virtualized home gateway is preferable than the currently deployed physical one.
- To demonstrate the benefits of NFV in the home environment

2. Related Work

2.1 Parents Controlled Home Internet Cafes[6]

This is thesis work carried out by Yunpeng Han in the University of Agder. The motivation behind this thesis is the requirement for a method to control family members' Internet usage. This Proof of Concept thesis is based on using OpenWRT to achieve a free and user friendly Internet access control system will be very helpful to parents. It is known as 'Home Internet cafe'. Some of the features created with this implementation thesis are listed below.

- Set up the timescales that a specific user is allowed to access the Internet
- Set up the total Internet access limitation time of target user
- Block some services or access to unwanted internet gaming or websites
- Website blocking by URL Address or keyword.

This work relates to our thesis in the fact that it uses OpenWRT as its prototype. It differs to our thesis as we will be virtualizing DHCP rather than using OpenWRT to achieve Parental Control.

2.2 Design and Implementation of SoftEther VPN[7]

SoftEther literally translates to Software Ethernet. This thesis was written by Daiyuu Nobori in January 2013. . He actually started at the university in 2003 and started up a company, SoftEther Corporation in 2004. He is currently a PhD student at the university. SoftEther VPN is a cross-platform multi-protocol VPN program. He made it free to the public to download, and released its source code in 2014. As this is OpenSource, it is keeping in line with what we set out to achieve, to use OpenSource for as much as we can in the thesis.

To give some background on how SoftEther evolved, we will explain. In 2003, SoftEther 1.0 was written. The first version up for sale was released in August 2004. SoftEther 2.0 was released in December 2005, and the name was changed to PacketiX 2.0. In March 2010, PacketiX VPN 3.0 was released with new features to include, for example, support for IPv6, 802.1Q VLAN and TLS 1.0. IN July 2013 the name was again changed, this time to SoftEther VPN, reverting to what it was known originally, but adding VPN to the end of the title. And most recently, on 4th January, 2014, SoftEther Corporation the source code was released as OpenSource.

SoftEther is highly interoperable and can be installed on Operating Systems such as Linux, Mac OS FreeBSD and Solaris. Mobile platforms running iOS, Windows Phone and Android can be used with the client installed using L2TP or IPsec protocols. It can also be installed on routers manufactured by Cisco, Linksys and Juniper. And also any embedded device with OpenWRT running.

The main components of SoftEther VPN are the VPN Server, VPN Client, VPN Bridge, VPN Server Manager, VPN Client Manager and the VPN Command-Line Admin Utility. We will go into detail to explain each component later on in the thesis.

In our thesis we will be utilizing the VPN bridge component of SoftEther to achieve our goal of virtualizing DHCP. When the bridge is installed onto the router any device that connects to the router will automatically receive an IP address from the virtualized DHCP without any modifications or installations needed on the device.

2.3 Investigation of different VPN Solutions And Comparison of MPLS, IPSec and SSL based VPN Solutions [8]

This thesis was written by Sheikh Riaz Ur Rehman. As we utilize a VPN to achieve the Virtualized Network Function this thesis was selected as a related work due to the detailed comparison of the various VPNs that are available to use. By definition, a Virtual Private Network, VPN, is a network in which connectivity between various sites is established on a shared network with the same security as a private network. Every user on the VPN can share the same resources. This thesis gives the reader a good understanding of how a VPN works and how the different types of VPNs available compare with each other. In Chapter 5 Sheikh describes how SSL VPNs are useful to secure web based VPN. As we will be using SSL VPN technology to implement a Virtualized Network Function, this chapter gives an understanding into SSP VPN architecture and how it functions.

This thesis gives us a good understanding of the various VPN methods available and helps us decide which is right for our prototype.

2.4 Tethered Linux CPE for IP Service Delivery [9]

In this paper, written by Fernando Sánchez and David Brazewell focuses on the virtualization of the Home Residential gateway and moving it to the cloud, leaving only a minimalist Customer Premises Equipment, CPE. It details the modelling, motivation and Implementation considerations for such a migration. It is designed around a lightweight CPE that is controlled from the cloud utilizing virtualized software images. Their design utilizes the latest releases in the Linux Networking stack running in the Kernel space. This is complemented by using Software-Defined Networking, SDN, and Network Functions Virtualization, NFV. When this entire infrastructure is in place, the virtualization of the CPE has been achieved, while maintaining the performance and security of the traditional physical CPE. It also adds to the home infrastructure by providing monitoring your network and maintaining it from an easily accessed cloud datacentre.

Another motivator behind this design is to enable only minimalist equipment to the customer, cheaply. This can be related to my own work as we also want to provide the customer with the latest applications quickly but with little cost. In this way, the CPE becomes a passive device with most of the functions being carried out in the cloud.

Just like in our thesis, this paper achieves the relocation of the control plane from the home environment into the cloud freeing up valuable resources from the home device. It is related to our work in the way that we are both trying to achieve smaller, less resource intensive elements that can provide the same or more functions that a home router usually provides.

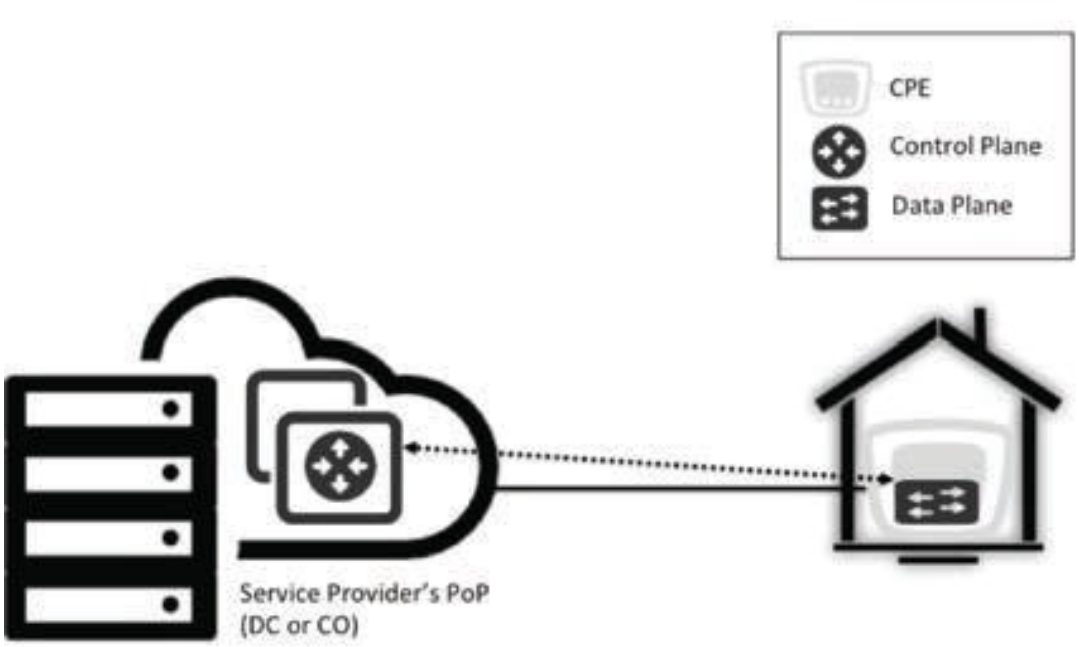

Figure 1: CPE with virtualized stack [6]

As you can see from the above figure, the separation of the control plane from the customer's router can free up limited control plane resources before they use up data plane resources. To improve scalability and costs this virtualization will be beneficial to both the customer and the service provider. It is beneficial to the customer as they can allow the maintenance of their services be carried out by the service provider. They will also have a quicker turnaround time for delivered new services. It is beneficial to the service provider by reducing CAPEX and also allowing for the development and testing of new services offline with no impact to the customer services.

3. Methodology

3.1 Introduction

The experimental research carried out in this thesis will be to find a way to replicate the physical functions that are part of the hardware and to reproduce it on a virtual machine. The experimental testing will follow a pattern of complexity, observation, decision and change. A few iterations will be needed to be carried out in the experimental phase. This will ensure a more robust result.

A literature research will be conducted to help with any relevant published data and experiences carried out previously relating to NFV and Virtualization in general. Within the literature review we will aim to uncover topics relating to NFV measurements, Use Case scenarios with NFV and whatever outcomes from POC, Proof of Concepts, that have been already carried out.

Statistical analysis of both the virtual and physical machines measuring packet delay and latency will be implemented. The results will be presented in a manner which will be easily understandable to the reader. Data collection from the monitoring of the performance will help us to understand the differences between the various designs.

Designing the experiments will help in deciding what hardware and software will fit best for our needs. Choosing the correct tools to generate and observe traffic to get a worthy result will be an essential part of the design phase.

Physical implementation will consist of installing OpenWRT [9] onto a standard home router, D-link 505, using a firmware image Barrier Breaker 14.07.The version number is very important when choosing the router. Some previous versions are incompatible with some routers. This is a fully writable file system with package management. This build comes without a GUI, Graphic User Interface. OpenWRT is based on the Linux kernel, and is primarily used on embedded devices to route network traffic. Other Network Functions which could be useful in OpenWRT are the stateful firewall, NAT, DNS and port forwarding. However, for our thesis, we will be virtualizing DHCP.

I plan to utilize LuCI Essentials with OpenWRT. LuCI is essentially a web user interface for embedded systems. It uses the Lua programming language. It splits the interface up into logical parts like models and views and uses object-orientated libraries and templates. This ensures better performance, smaller installation size, faster runtimes and simple maintainability. It does have some dependencies which will also be needed to be installed.

Virtual implementation will involve the use of SoftEther which will be installed on a Windows 2008 host operating system running on Amazons EC2 cloud server. It will be installed as an application. We will use Ubuntu as the OS to compile SoftEther for installation onto OpenWRT. Ubuntu is running on top of VirtualBox which is installed onto MacBook Pro due to superior CPU power available from this device. Other machines I own, older machines, were tried with Ubuntu OS but proved too slow for the compiling of the package. The MacBook Pro compile time took roughly 30 minutes with 4 CPUs available. When I used an Ubuntu install on an older machine the compiling was let to run for 3 hours before I gave up and stopped it.

I will be virtualizing a Network Function, DHCP. In order to virtualize this function we will be using a package called dhcrelay. This package basically provides a means of relaying DHCP and BOOTP requests from a subnet to which no DHCP server is directly connected to one or more DHCP servers on other subnets. The DHCP Relay agent will listen for queries

from clients on a specified interface, passing them onto the virtualized DHCP server running in “The Cloud”. A DHCP relay agent will be installed onto the Residential Gateway; our OpenWRT router, which also has SoftEther installed to enable us to pass the relayed DHCP request out onto the internet and towards our EC2 server running our DHCP server. This setup relies exclusively on the use of a VPN tunnel to enable an Ethernet bridge and accomplish our relocation of DHCP onto the cloud.

For my measurements a traffic generating tool known as IPNetMonitorX [11] will be used. It can gather accurate latency and packet loss metrics for DHCP. The tool will simulate typical Internet traffic so we can benchmark the infrastructure. Features include improvement of the precision of latency measurements and allow for per packet per query reporting,. This tool will be installed on Mac OS client. This tool measures the DHCP lease assignments to client computers by ramping up lease assignment over time to determine the maximum performance profile to benchmark against.

In my testing the goal will be to measure the maximum reply rate achieved by the DHCP servers, both the physical one and the virtual one. The tool was created to push the server to its limit with a high load of requests. The server will then try to reply to every request and answer them within a specific timeframe. This will be the way to benchmark each server’s performance.

3.2 NFV Architecture and Background

NFV entities and components offer diverse opportunities for software developers and vendors. There are significant differences in the way the networks are structured with virtualization technologies as compared with non-virtualized infrastructure. Since software is decoupled from hardware, network elements are no more dependent on software and hardware integration i.e. software can grow and progress independently. The scalability of network functions is more flexible and dynamic.

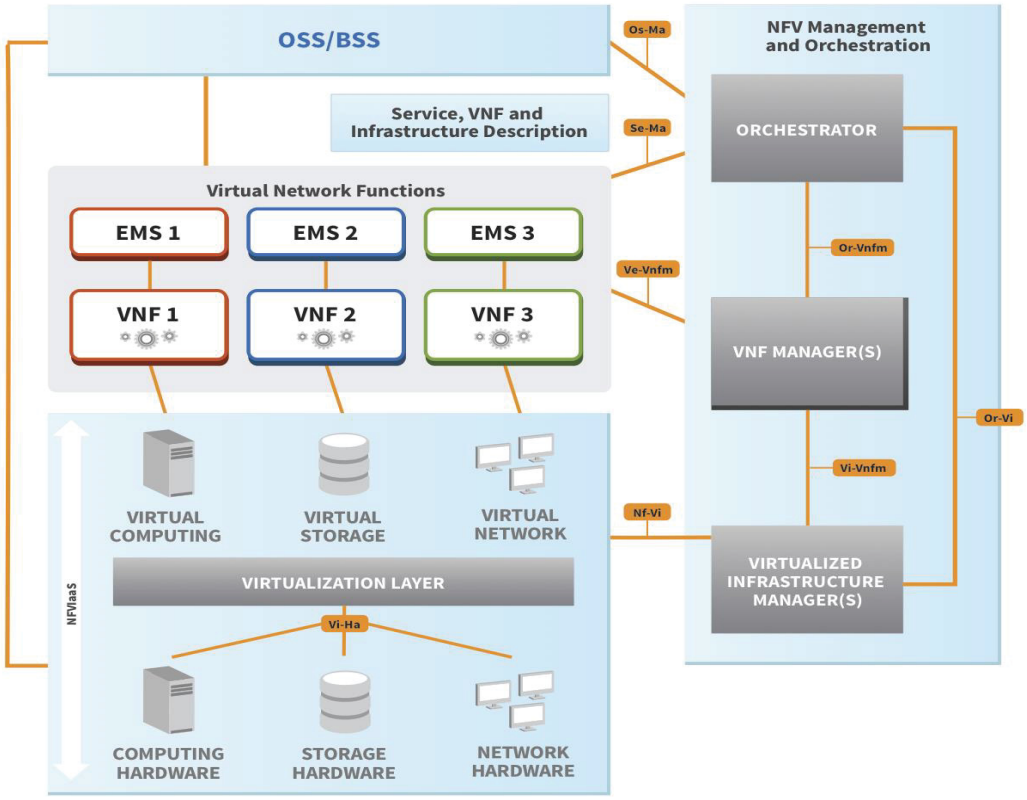

Figure 2: NFV Architecture.

All resources of network elements are available in a granular and abstract way to external control modules using well defined interfaces. This also extends to the simultaneous existence of different control elements and it could be deployed as a centralized controller, thus improving the efficiency of end to end control and optimization of the network.

The architectural framework is shown in Figure 2 [4]. It has been divided into three major domains. Network service providers and network elements vendors are free to develop and implement if they need to provide NFV compatible products.

VNFs are the network functionalities defined entirely by software instances and running over physical resources. It is compatible and capable of implementing over NFVI. The VNF replaces non visualized network nodes, which are produced, software oriented and isolated from the hardware dependency [12]. Possible functionalities could be 3GPP core network elements e.g. SGSN, Serving GPRS Support Node, GGSN, Gateway GPRS Support Node , HLR, PDN, Home Location Register, Gateway, DHCP Servers, web servers and firewalls etc. Every VNF instance is accompanied by Element Management System (EMS) modules. These are required to run and manage individual VNF and its peculiarities.

The NFV Infrastructure is the totality of all hardware and software components building up the environment to run virtualized network elements [6]. NFVI functionality could be distributed at several locations. NF software entities run over the NFV Infrastructure. It provides the diversity of physical resources and their virtualization procedures. The physical hardware resources can have computing, storage and communication network functions processing through virtualized layer. It consists of Commercial-Off-The-Shelf (COTS)

hardware, supporting components and software platform necessary to run infrastructure. Computing and storage resources are shared assets. Networks can have switching devices like routers, wired and wireless links [6].

The virtualization layer decouples VNF instances from underlying hardware. It's where the hypervisor functionalities are assumed. This layer is responsible for abstraction of hardware resources, implementation of VNFs to use virtualized environment and also responsible for the provisioning of the virtualized components.

NFV Management and Orchestration is responsible for the orchestration and life cycle management of physical and virtual parts. It manages the VNF instances and communicates with EMSs. M&O covers the virtualization specific tasks necessary to run NFV framework. Virtualization Infrastructure Managers perform two major tasks, resource management and operation management. Resource management includes inventory of software, computing, storage and network resources. It also takes care of the allocation of virtualization enablers and management of the infrastructure resources. From an operational perspective it does root cause analysis of performance issues and collects information for capacity planning and optimization [14].

The Orchestrator is in charge of the orchestration and management of NFVI and software resources, and realization of network services on NFVI.

A VNF manager is responsible for life cycle management of VNF which includes instantiate, update, query, scaling and termination. A VNF manager can take care of a single VNF instance. In practice multiple managers could be deployed.

3.3 System Design

3.3.1 Firmware and Router Selection

When you can connect to multiple wireless connections, some establishments will only provide wired Ethernet connection or a poor Wi-Fi connection. You will find similar setups in conference rooms and other corporate locations. Using a wire to connect to a standard laptop that has an Ethernet port is acceptable. However, a lot of employees are leaving the laptop at home and bringing their iPads and smart phones – all of which only have a wireless connection. Some of the newer notebooks don't even have an Ethernet port, and this will continue with a push towards a complete wireless environment.

The D-Link DIR-505 [15], displayed, in figure 3 below aims to help with this wireless connection issue. The DIR505 is a small form factor router, very portable and can fit into your pocket. The power plug is useful in the fact that you don't need to remember to bring a separate power supply every time you travel. At less than 300 SEK, it packs a lot into a value router. It is compatible with OpenWRT and with 8MB flash it can install SoftEther and Luci essentials with a little room left over for some monitoring tools. It has CPU speed of 400MHz and also 64MB RAM. It uses an Atheros based processor. Its small form factor shown in the image below shows its convenience to be brought and installed practically anywhere.

Figure 3: Dlink 505 router

TP-Link TL-WR710N was another router tested. The firmware was generated via Buildroot and installed. However, when the firmware was installed it bricked the router and made it unusable. Again, this is the importance of using specific versions of OpenWRT and not untested Beta versions. A list of compatible routers on which OpenWRT can be installed is available online. This information also includes the version of router which upgraded successfully. This only problem you have at this stage is that you do not know the exact version of the router until you actually purchase it and check the board by opening the device. Sometimes the version number can be found on the outside of the router but this is not always the case. To give you an example, version 1.1 could be built with a different processor as version 1.2

Other routers we have tried to flash but were unsuccessful due to various reasons are detailed below. We installed OpenWRT onto TP-Link WA830RE and this was successful. But there was no space left...This was the first device we tried to get working. We did succeed in so far as installing OpenWRT onto it. When it came to the stage of installing SoftEther onto it we discovered it didn't have enough memory, it had 4MB. For the SoftEther package to install without issues you need at least 8MB of memory.
The next router purchased and tried was DGND3800B. This is a router built by NetGear and is supported for OpenWRT however this router is for the German market only. It runs DSL on Annex B so we opted against it. Annex B is the way your internet is presented to your home. With Annex A, which is what we have in Sweden, ADSL is presented over Public Switched Telephone Network, PSTN. In the case of Annex B, ADSL is presented over ISDN. So again, this router was returned and deemed unworthy for my setup.

3.3.2 SoftEther Architecture

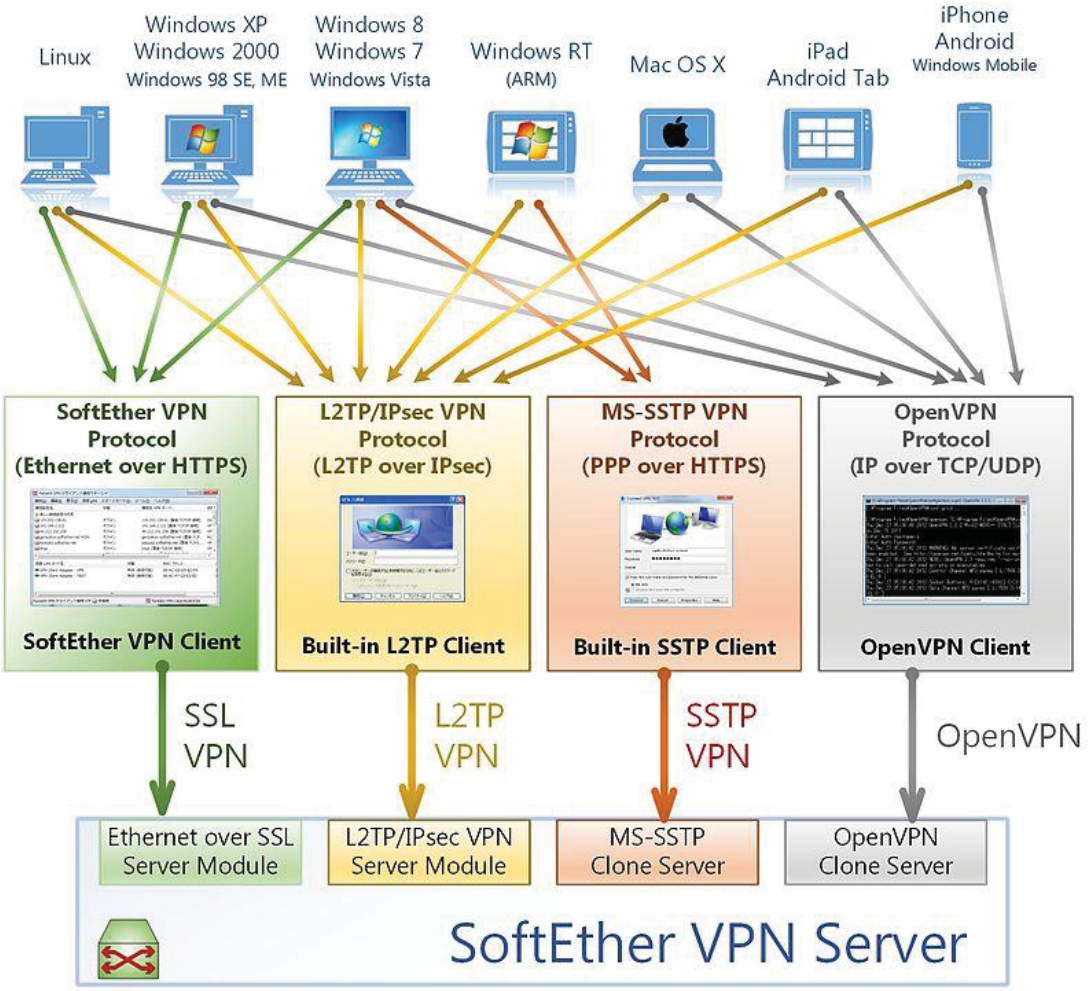

Figure 4: SoftEther Architecture [6]

The VPN Server component of SoftEther starts the VPN server task to listen for and accept connections from a client or bridge with many different VPN protocols. The server consists of protocols including Ethernet over SSL, L2TP/IPsec VPN, MS-SSTP and a clone of OpenVPN. The range of protocols included in the server enables SoftEther server to work on Linux, Windows, FreeBSD, and Mac OS.

The server can have different Virtual Hubs and Virtual Layer-3 Switches. Just like a physical Ethernet switch, a virtual hub has full layer-2 Ethernet packet-switching. This can also function as a layer 3 switch, mirroring the activities of a physical layer 3 hardware switch. For this thesis we will be using the layer 2 function.

The VPN Server also has localized bridges. The bridge acts as the middleware layer 2 switching between a physical Ethernet network-adapter and a Virtual Hub. When a connection is made, the transmission encapsulates the Ethernet frame. It uses a TCP connection for the transmission.

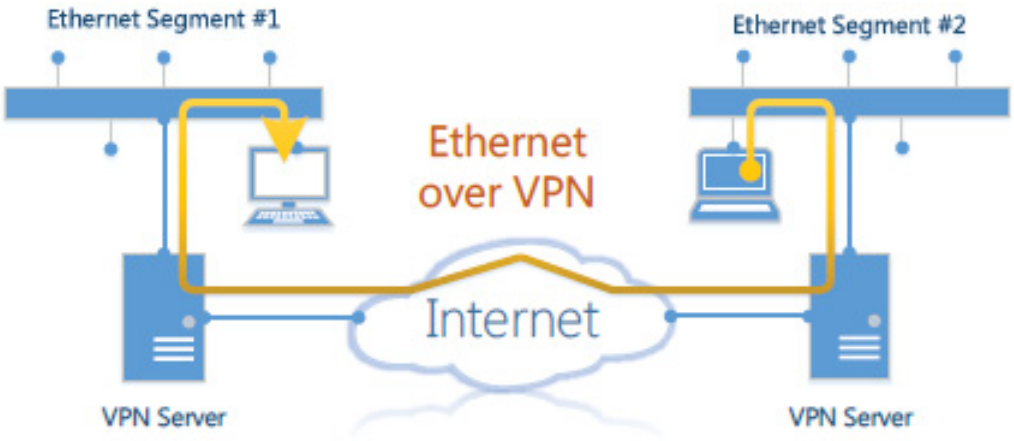

Figure 5: Ethernet over VPN [7]

SoftEther Bridge is a program which has the virtualized function of an Ethernet network adapter. During setup the program creates a startup registration point in the Operating System in order to boot automatically when any user starts the OS. Also during installation the software adds a Windows Service designed to always run in the background. The software is designed to connect to the Internet and then add an exception to the firewall. The inclusion of multiple clients enables SoftEther to work on not only the OS mentioned in the server section above but also enables the client to be installed on Android phones, IPhones, and embedded devices that run OpenWRT. The protocols that facilitate this are L2TP/IPsec and OpenVPN clone.

SoftEther Bridge is installed on an Operating System to enable site to site connectivity. When a VPN client connects to a remote network using SoftEther bridge, it is assigned an IP address that is part of the remote physical Ethernet subnet and is then able to communicate with other machines on the remote network as if it were all connected locally. A bridged VPN passes IP broadcasts enabling the remote DHCP server act as the only DHCP server on the network. A VPN Bridge connects to the main remote VPN Server by using cascade connection between the two sites. A cascade connection is a virtualized version of an uplink connection between two physical Ethernet switches.

3.3.3 OpenWRT

OpenWRT is basically a custom built firmware for your home router. It has a very modular architecture which allows it to build easily for many different devices and makes the various package selection simple. Its interoperability to work with practically any home router out there is one of its key benefits. OpenWRT is a small Linux distribution mostly installed on embedded devices, for example our residential gateway. It is built on top of the Linux kernel and includes a mixture of various software packages. Installation of packages utilizes the opkg Package management system. OpenWRT can use the command-line interface when you are working with a minimal install to save space on the embedded device. It also includes an optional web-based GUI interface known as Luci. For this thesis, we will be using both.

It has been one of the key drivers behind the Wi-Fi revolution. Some people like to refer to it as “Wireless Freedom”. It is also open source. You can install different versions of OpenWRT, depending on what is supported for the router you decide to use. We will be using the latest version, Barrier Breaker 14.07. The version is important because some devices are incompatible with earlier version.

OpenWRT removes the boundaries and restrictions of proprietary firmware in order to make it highly customizable and dynamic. A wide range of options is open to both the average Joe customer and also customers with a professional background in IT, giving the experts the freedom of development and handing the average user the freedom of customization. Figure 6 shows the structure of OpenWRT and we will summarize the purpose of each component. [19]

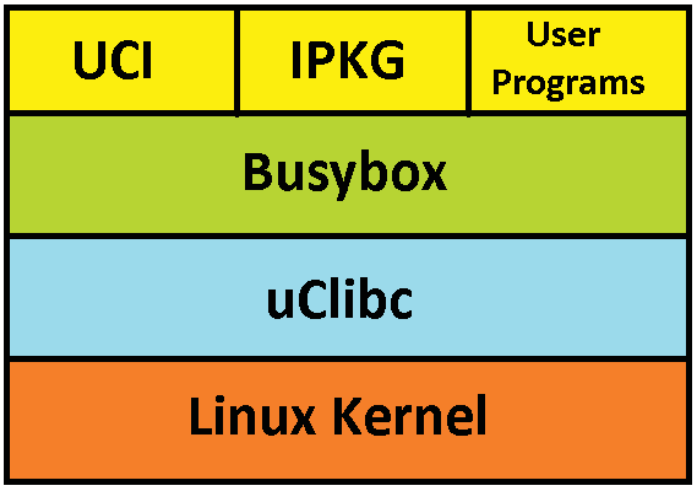

Figure 6: OpenWRT Structure

Unified Configuration Interface, UCI, is intended to centralize the configuration of OpenWRT. Simplification is one of the main motivations behind OpenWRT project. UCI is an interface to a series of text files, easy-to-read configuration files that are all centrally located, making it much simpler to configure. UCI provides a common way of getting and setting configuration for many programs at the command line.

Ipkg packages are the standard way of installing software for OpenWRT. It stands for Itsy packager. The opkg utility is a packager specific to OpenWRT. It is a lightweight package manager used to download and install OpenWRT packages from local package repositories or packages located on the Internet. When you install a package with opkg it tries to resolve any dependencies with packages in the repositories. This package manager provides a simple way to include additional features not already in the OpenWRT firmware. Ipkg is a command line program, and you need to use putty to SSH into your router.

User programs that run in the user space only have access only to a limited part of memory .User space processes can only access a small part of the kernel. These are known as system calls. If a program performs a system call, a software interrupt is sent to the kernel via an interface which then sends the appropriate interrupt handler and then continues its work after the handler has completed.

BusyBox consists of small versions of various common UNIX utilities packaged into a single small executable. It is known as the “Swiss Army Knife of Embedded Linux". It is open source and combines 300 common commands.

uClibc is a C library for developing embedded Linux systems. This library provides the basic routines for allocating memory, pattern matching, opening and closing files, searching directories, reading and writing files, string handling and arithmetic.

The Linux kernel is the process that manages input/output requests from software and translates them into data processing instructions for the central processing unit and other electronic components of an embedded system.

3.3.4 Amazon EC2

Amazons Elastic Compute Cloud, EC2, is a section of Amazon.com's cloud computing platform, named Amazon Web Services, AWS. EC2 allows users to use virtual computers on which to run their own computer applications. EC2 allows the scalable development and deployment of applications by utilizing a web service through which a user can start an Amazon Machine Image to create a virtual machine, this is known as an instance, containing any software desired. A user can start, pause, and terminate server instances as needed, paying by the hour for active servers, hence the term "elastic". The server space in Amazons datacenter is rented by the user. EC2 provides users with control over the geographical location of instances that allows for latency optimization and high levels of redundancy. This instance functions as a virtual private server. The size of these instances are based on Elastic Compute Units. Specifications for the EC2 instance I choose are detailed as Microsoft Windows 2008 R2 SP1 Datacenter edition and 64-bit architecture, 1 vCPUs, 2.5 GHz, Intel Xeon Family and 1 GB memory. This is the best available with the free package.

As EC2 works as a rental, the power, size and performance of your instance depends on what you signed up for. The offer we signed up for was Free Tier. This option provides you the opportunity to run a miniaturized server, storage, EBS, and bandwidth for one year.

Amazon's elastic IP address feature functions to the user’s perspective just like a static IP address in traditional data centers. There is a fundamental difference though. A member can configure the mapping of an elastic IP address to any one of their virtual machine instance without any assistance from an Amazon EC2 administrator. In this way an Elastic IP Address is owned by the member and not to the actual virtual machine. This IP address binding will stay in place unless it is disconnected by the user. The mapping will remain associated with the account even while it is associated with no instance.

3.3.5 Test Tools

The main measurements we set out to achieve is DHCP performance, comparing the virtual server against the standard physical server. We measured latency and also failure rate. Latency is a key measurement due to the dependency on the timeout of the various DHCP clients. This measurement is also used to show the performance comparison when using the virtualized function rather than the physical one. DHCP latency can be the time required to obtain an IP address from the server. If DHCP latency appears to be an issue then an increase in the DHCP timeout value would be necessary.

To measure this latency we used a tool called IPNetMonitorX which runs on MacOS. To monitor the DHCP process we used Wireshark. We will explain IPNetMonitorX first and then go on to detail Wireshark.

IPNetMonitorX is a network troubleshooting application developed by Sustainable Softworks. They specialize in networking and communications and have been developing tools for MacOS since 1996. It is actually a suite of networking tools consisting of 23 different packages for debugging Internet service problems and optimizing performance. Some of the key benefits of this suite of tools are detailed below.

- Easy access to the tools is provided by a floating tool palette allowing you to launch each tool quickly and easily.

- A display window allowing you to see the testing being run in real-time.
- Multiple tools can be used simultaneously

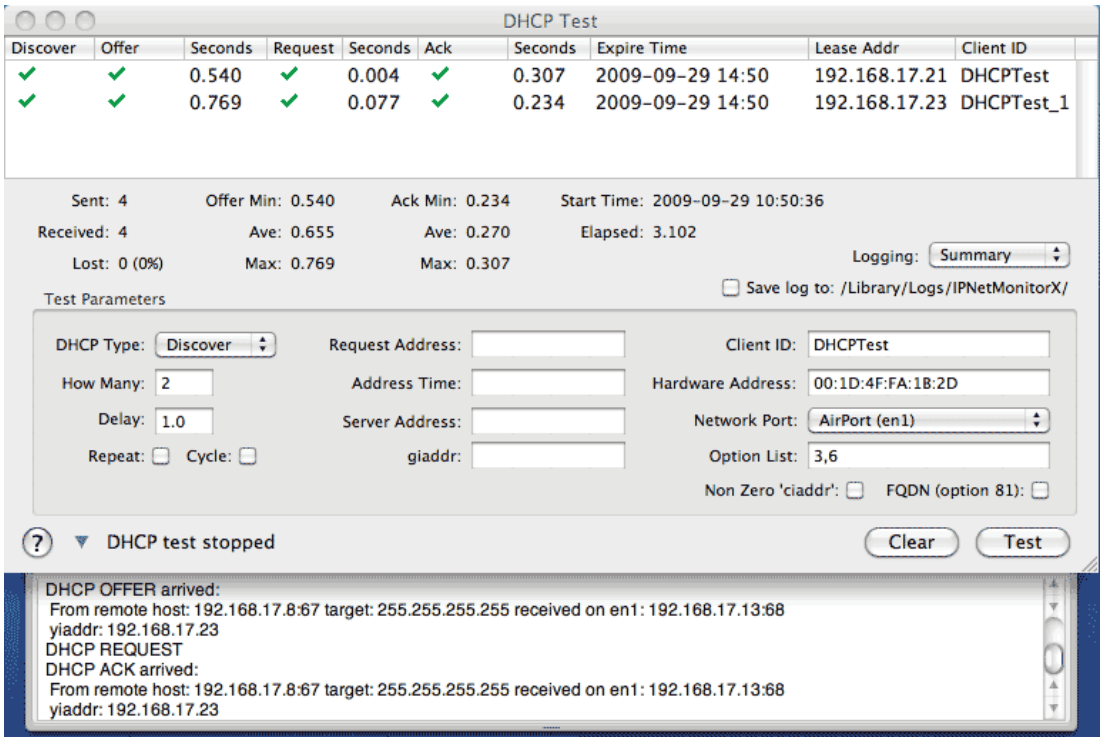

Figure 7: IPNetMonitorX DHCP Test Tool

For our DHCP measurements we will be using one of the tools known as DHCP test. This is shown in figure 7 above. This tool allows you to simulate DHCP transactions and exercise a DHCP Server. The dropdown menu for DHCP type allows selection of 7 different types of DHCP packet to generate.

When you select “Discover” the tool broadcasts messages on the network subnet using the destination address 255.255.255.255 or the specific subnet broadcast address. If a DHCP OFFER is received from any DHCP server, the tool will then send a DHCP REQUEST message to request a lease. It will then wait for the corresponding "ACK" message. This test simulates one or more DHCP clients starting up with no previous lease information. The number of clients to be simulated can be chosen. When "Verify" is selected, the tool sends a DHCP REQUEST message, the same as when a client is in the INIT-REBOOT stage in order to verify an existing lease. When “Renew” is selected the tool sends a DHCP REQUEST message, the same if the client is searching to renew an existing lease. When "Rebind" is selected from the dropdown menu the tool sends a DHCP REQUEST message as if the client is trying to rebind. When “Inform” is selected the tool sends a DHCP INFORM message. When “Release” is selected, the tool sends a DHCP RELEASE message using any IP addresses previously discovered to release the corresponding lease bindings. When “BootP” is selected the tool sends a request that does not include a DHCP message type in order to simulate a BootP client.

In order to generate enough packets to get some good measurements the “How many” option will also be utilized to generate 1000 messages on both the physical DHCP server and the virtual one. Each DHCP transaction following the initial transaction will use a specific Client_ID generated by appending the initial transaction number to the prefix of the Client_ID. The first test uses the actual Client_ID entered by the tester including the

hardware address and IP address of the machine being used for the testing. The Client_IDs used in this prototype are Clive's House and Clive's EC2 Virtual Hub. They are for reference only.

3.4 Key Function Design

3.4.1 Wireless Relay

When computers and attached servers are not connected to the same IP network, a DHCP relay agent can be deployed to transfer DHCP requests and acknowledgements between each other. The DHCP relay agent functions as the buffer between DHCP clients and the server. It listens for client messages and adds important configuration information, such as the client's link data, which is utilized by the server to allocate the IP address for the client. When the DHCP server acknowledges, the DHCP relay agent pushes the acknowledgement back to the DHCP client.

When DHCP client starts up and is initialized, it will request packets broadcast on the local network configuration. If the local network has a DHCP server, you can directly request IP settings without the need to use a DHCP relay. If the local network does not have a DHCP server, DHCP relay feature will be utilized. After you have received the broadcast packets they will be processed and forwarded to the relevant DHCP server on the other network. From the beginning to the end of the DHCP transmission additional interactions will follow the packet exchange process in the below Figure.

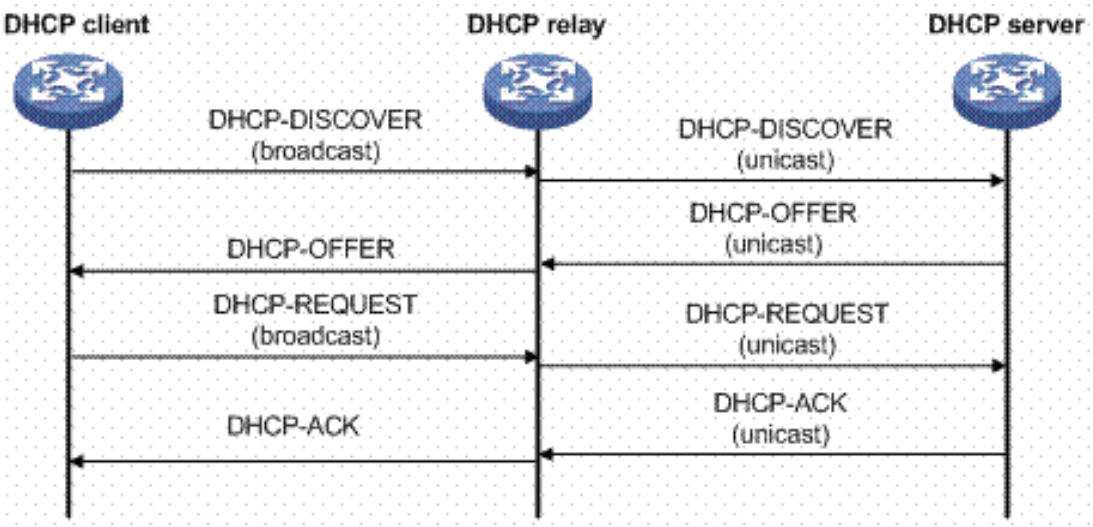

Figure 8: DHCP Relay Message Transmissions

The DHCP relay device modifies the relevant fields in the message, the DHCP server broadcast packets into unicast packets and is responsible for the conversion between the server and the client.

When you are dealing with a multi segment network it gets a little more complex. This is because the DHCP broadcast messages cannot, traditionally, cross the different routers. Different administrators use different methods to get around this. The first method would be to put a DHCP server on each network. This would be a good option if you are dealing with a relatively small network with few segments. If you were to use this method in a global enterprise, installing a DHCP server on each separate network can increase expenses and require more intervention when maintaining and administering.

When dealing in a situation where you do have a big network, a more useful method is to consolidate the DHCP servers and centralize them in one or two datacenters. To solve the problem of DHCP broadcast requests, routers can be configured to forward BOOTP messages selectively. This is referred to as BOOTP Relay.

The idea of BOOTP Relay needs some more explaining. It gets even more confusing when the term BOOTP Forwarding is used. This is because the concepts for Relay and Forwarding are quite different. Forwarding implies that the message is forwarded from one interface to another, without any modification to the packet. Relay implies that the message is modified or processed, which usually means modifications being carried out to the original packet.

DHCP Relay Agent, dhcrelay, comes from the Internet Systems Consortium. It can be installed onto OpenWRT and offers a way of relaying DHCP and BOOTP messages from a subnet to which no DHCP server is directly connected to one or more DHCP servers on other networks.

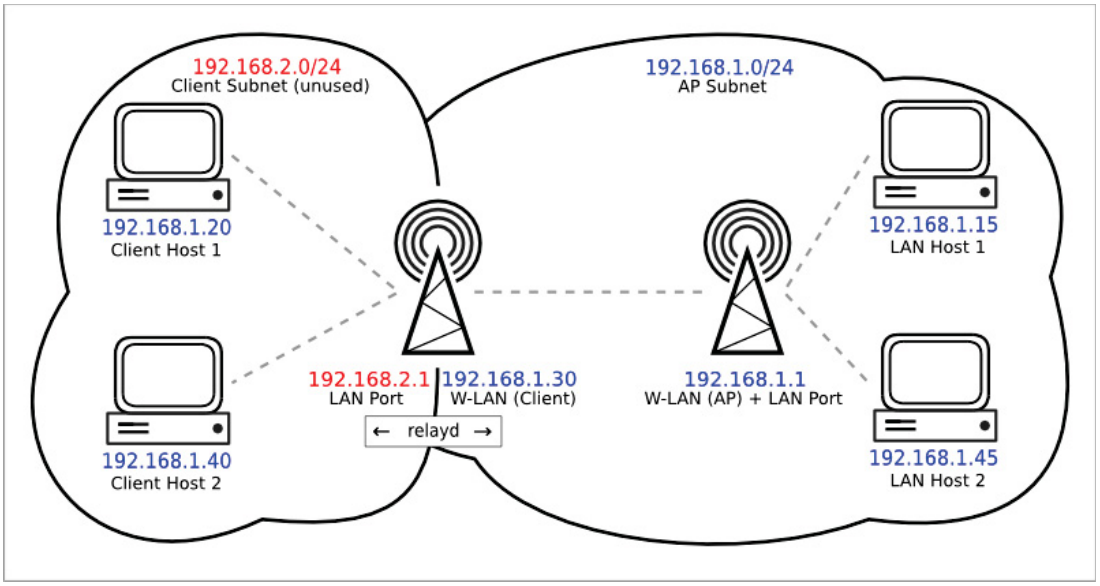

Figure 9: DHCP Relay Infrastructure

Relayd is a daemon to relay and redirect incoming connections dynamically to a host machine. Its key operations are to run as an application layer gateway, load-balancer, or transparent proxy. This relay daemon is able to observe clusters of hosts for availability, this is discovered by checking for a specific service common to a host group. When availability is discovered, layer 3 forwarding is set up by relayd.

3.4.2 Cascading

The definition of cascading in networks is to expand your wireless or wired network with an Ethernet cable. This type of connection is used in situations where there is a need to improve the performance of the network without removing the existing router. Many different devices can be cascade connected. Using cascading also helps to isolate the network traffic. This setup typically involved a primary router, our SoftEther VPN server, and a secondary router, our OpenWRT Softether Bridge installed onto the wireless router. We will be using a LAN to LAN cascade rather than a LAN to WAN cascade. The type of cascade you choose all depends on whether you want to have the segments in different IP ranges or part of the same range. LAN to LAN cascading provides the prospect to share files and resources within the network. The wireless equivalent of Layer 2 cascading is called bridging.

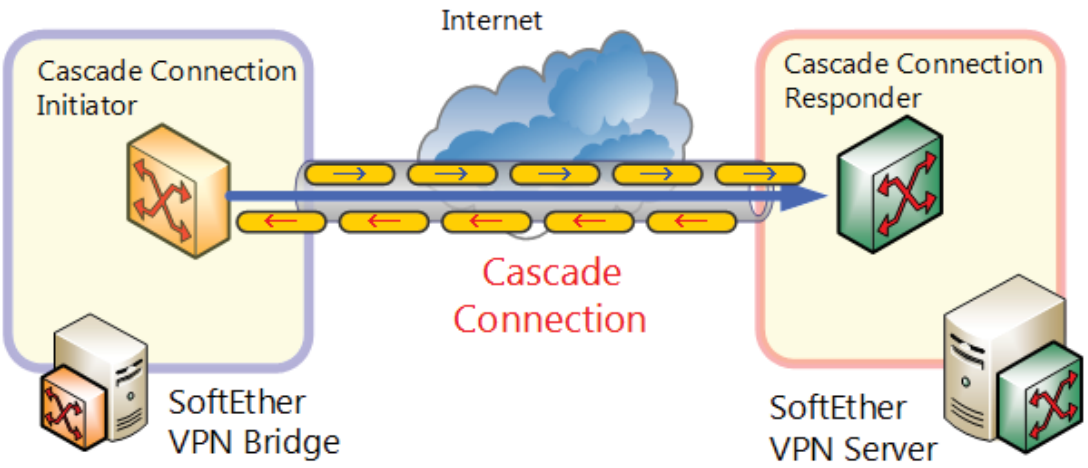

Figure 10: Cascade Connection Operation [21]

A Virtual Private Network Cascade, VPNC, is a method of establishing a multi-tunnel encrypted tunnel within a tunnel through various trusted servers across the unsecured public network.

The methods for creating VPN cascade connections are important in creating a LAN to LAN VPN using SoftEther VPN. Using the VPN cascade enables the cascade connection of a Virtual Hub setup on the VPN Server to another bridge operating in the local or remote servers.

When two Virtual Hubs are running on separate segments or when they are running on the same segment, the virtual hubs are initially not connected to each other so they are two different separate segments from the standpoint of a layer 2 connection. Just as you would connect an Ethernet cable between two separate routers a virtual cascade connection allows the connection of two or more virtual segments with a very long network cable. However, you don't have the restriction of distance of the cable holding the network expansion back. Cascading enables, at an Ethernet level, free layer 2 traffic between a Virtual Network Adaptor connected to, for example, Virtual Hub A, and a networked computer locally bridged to A and a Network adaptor connected to B and a network computer locally bridged to B.

3.4.3 SSL-VPN

SSL-VPN is basically HTTPS, HTTP over SSL. It stands for Secure Sockets Layer Virtual Private Network. When HTTPS is used it can virtually allow transmission through many kinds of firewalls which would normally disallow IPsec-based VPNs. There are two types of SSL-VPN, SSL Portal VPN and SSL Tunnel VPN. We will be utilizing the latter type.

This type of VPN permits clients to remotely access local network functions and services through an encrypted and authenticated tunnel by securing all network traffic. This results in giving the appearance that the client is on the remote network, regardless of location. Using this type of VPN accomplishes a high level of interoperability with client platforms and configurations for remote networks and firewalls, providing a more consistent connection.

From the diagram below a computer or anything that is connected to the internet without connecting to the OpenWRT with SoftEther running accesses the Internet through the unsecured red path. The Service Provider can monitor all traffic from this IP address and the user has no access to file sharing on their LAN and no access to any shared resources of Network Functions.

The OpenWRT Router establishes the blue SoftEther SSL VPN connection.

The Client behind this VPN uses the yellow secured connection through SSL VPN provided by SoftEther to gain access the Internet. Because the traffic inside the blue tunnel is our Layer 2 Ethernet tunnel, the connection acts like any entity connected in this way can share all the same network functions, securely and from any location across the globe.

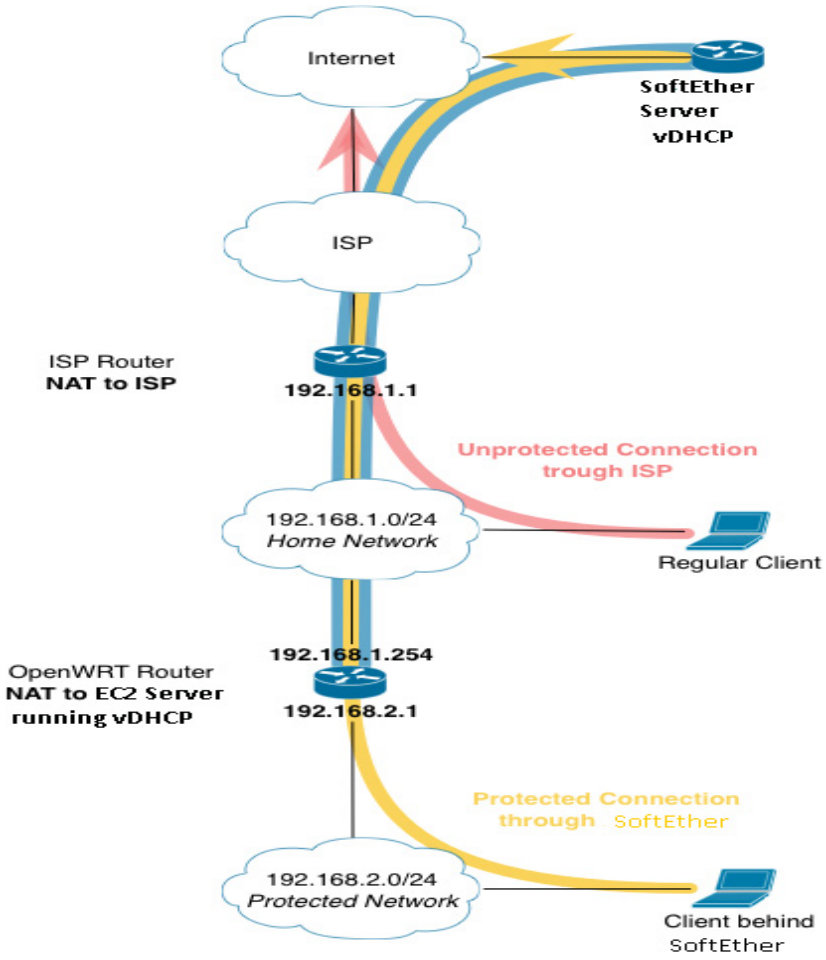

Figure 11: OpenWRT Operation Comparison

3.5 Implementation

3.5.1 Development Environment

My development environment consisted of Ubuntu OS to use as a build environment to create the SoftEther package in preparation for installation onto the home wireless router. Whether this is a virtualized machine or not, it can build and compile OpenWRT firmware with the packages you want to install for the specific embedded system or machine you want to install it onto.

In order to achieve a cloud server Amazons EC2 was utilized. This provided an environment where we tested and compiled various Virtual Instances. The main VPN server is installed onto this cloud server. Our cascaded bridge installed onto OpenWRT will connect to this public IP address.

MacOS is used as a centralized system to simulate, monitor and analyze DHCP traffic. DHCPerf installed on an Ubuntu machine was considered initially but after some evaluation and testing it was found that the MacOS testing suite was both more superior and more simplistic to use compared to the Ubuntu tool.

3.5.2 Amazon EC2

In order to use Amazon EC2 the first task you need to carry out is to create an Amazon account. As we use Amazon to purchase from their web shop this account worked for me. You will however need to provide credit card details so they can charge the card if you go over the allocated 750 free hours per month of server time. Once you have setup an account you need to log into Amazon EC2 dashboard and start loading up your servers.

From the dashboard you need to select EC2. You reach the Launch Instance page where you can select the type of server you want to deploy. In my case we deploy a Windows 2008 server. The instances are known as Amazon Machine Images, AMIs.

For security reasons you need to create a key and connect with public key authentication. You create secret key and download it to your machine. It is essential that you do nowt lose this key. It is also advisable when you first log into your machine that you change the password.

The final step in creating your instance is to setup a security group. This is similar to opening ports on a router. For my instance we added a security group that opened Remote Desktop Protocol, RDP, TCP and UDP port 3389 so we can access the machine from anywhere. we also added exceptions for ports 443, 992, 1194, and 5555 which are used by SoftEther client and server to initiate the VPN connection. A security group acts as a virtual firewall that controls the traffic for one or more instances.

Figure 12: Amazon Web Services Console

Type	Protocol	Port Range	Source
Custom TCP Rule	TCP	992	Anywhere 0.0.0.0/0
RDP	TCP	3389	Anywhere 0.0.0.0/0
HTTP	TCP	80	Anywhere 0.0.0.0/0
All ICMP	ICMP	0 - 65535	Anywhere 0.0.0.0/0
Custom TCP Rule	TCP	443	Anywhere 0.0.0.0/0
Custom TCP Rule	TCP	1194	Anywhere 0.0.0.0/0
Custom TCP Rule	TCP	5555	Anywhere 0.0.0.0/0

Figure 13: AWS Security Group

3.5.3 OpenWRT

Firstly, we need to make our SoftEther package which will be used to install onto our OpenWRT router. What we will use to achieve this is Buildroot. This is free and open-source software. It consists of build system for the Linux based OpenWRT distribution. It also works on BSD or Mac OSX systems. The recommendation is to use a Linux distribution and for our test environment we will use Ubuntu. Ubuntu desktop environment can also be utilized as a monitoring and test packet generation source. From the Buildroot wiki [22] "Buildroot is a set of Makefiles and patches that simplifies and automates the process of building a complete and bootable Linux environment for an embedded system, while using cross-compilation to allow building for multiple target platforms from one Linux-based development system. It can automatically build the required cross-compilation toolchain, create a root file system, compile a Linux kernel image, and generate a boot loader for the targeted embedded system, or it can perform any independent combination of these steps." It is no spring chicken and has been around for many years but it is simple and very flexible primarily intended to be used with small or embedded systems.

There are some prerequisites required to be installed onto the Linux platform in order to generate an installable OpenWRT firmware image file.

- 200 MB of hard disk space for OpenWRT Buildroot
- 300 MB of hard disk space for OpenWRT Buildroot + OpenWRT Feeds
- 2.1 GB of hard disk space for source packages downloaded during build from OpenWRT Feeds
- 3-4 GB of available hard disk space to build (i.e. cross-compile) OpenWRT and generate the firmware file
- 1-4 GB of RAM to build OpenWRT (build x86's img need 4GB RAM)

Due to these requirements a Virtualized environment is ideal for what we want to achieve. VirtualBox was used to run my Ubuntu host.

sudo apt-get update

sudo apt-get install -y subversion make gcc g++ libncurses5-dev libghc-zlib-dev libreadline-dev libssl-dev gawk bzip2 patch xz-utils git unzip

The ncurses library routines are a terminal-independent method of updating character screens with reasonable optimization.

The libghc-zlib-dev package provides a pure interface for compressing and decompressing streams of data represented as lazy ByteStrings. It uses the zlib C library so it has high performance. It supports the \"zlib\", \"gzip\" and \"raw\" compression formats. It provides a convenient high level API suitable for most tasks and for the few cases where more control is needed it provides access to the full zlib feature set.

The GNU readline library aids in the consistency of user interface across discrete programs that need to provide a command line interface. The GNU history library provides a consistent user interface for recalling lines of previously typed input. This package is a dependency package depending on libreadline6-dev.

libssl and libcrypto development libraries, header files and manpages. It is part of the OpenSSL implementation of SSL.

svn co svn://svn.openwrt.org/openwrt/branches/barrier_breaker

cd barrier_breaker

Cloning OpenWRT Buildroot involves downloading the OpenWRT Bleeding Edge trunk version, with svn. The svnserve program is a lightweight server, capable of speaking to clients over TCP/IP using a custom, stateful protocol. Clients contact a svnserve server by using URLs that begin with the svn:// or svn+ssh:// scheme. There are different ways of running svnserve. Clients can authenticate themselves to the server, and configure appropriate access control to their repositories. The above command creates a directory called OpenWRT, the OpenWRT Buildroot build-directory The OpenWRT toolchain "OpenWRT Buildroot" is included [23].

In order to add Softether for OpenWRT repository to OpenWRT Buildroot feeds file the below command is used.

echo "src-git softethervpn https://github.com/el1n/OpenWRT-package-softether.git" >> feeds.conf.default

The next step in the process is to update OpenWRT SDK feeds and install SoftEther into OpenWRT SDK.

./scripts/feeds update

./scripts/feeds install softethervpn

OpenWRT includes make config which compiles the package [24].

The command **make menuconfig** uses ncurses-based graphical utility, Fig 2, it's the more preferred and quicker way to make the package. It compiles a system based on the defaults for your selected hardware or virtualized hardware. The default values are sufficient for what we want to achieve in our environment.

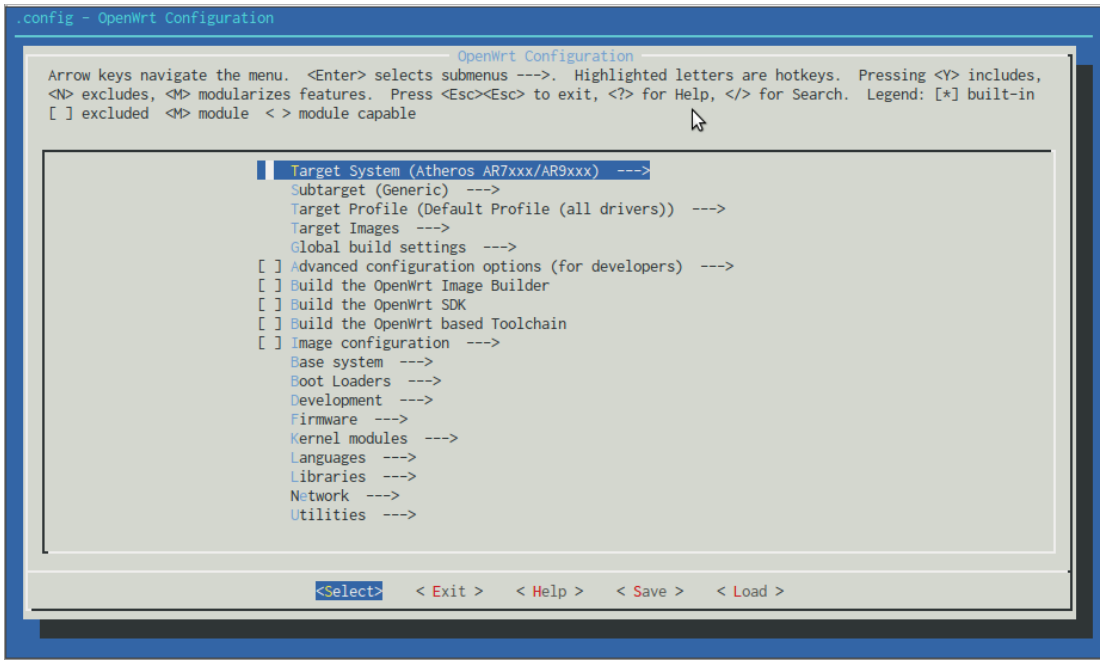

Figure 14: OpenWRT buildroot make menuconfig

The configuration options are located in the root of the kernel source tree, in a file named .config. This file can be accessed and modified directly. After any changes to the configuration, or when using an existing configuration file on a new kernel tree, you can validate and update the configuration.

The target system is changed to your specific platform in the configuration screen, followed by Network/VPN. There is an option for SoftEther which is selected in the VPN section. Any options can be changed.

Next is to compile the packages with the command

make prepare –jX

Depending on the amount of cores you have on your system, X is replaced with this amount. This command prepares the directory to download to and places the file feeds.conf into the correct place. Using the command

make package/softethervpn/compile V=99 –jX

Running **make <package>** builds and installs that particular package and its dependencies. Installed packages are stored in package/openwrt directory. If a new package is added, **menuconfig** command needs to be run every time. In my case the file was stored into */bin/x86/packages/softethervpn/* directory. The filename was softethervpn*_4.15-9538_x86.ipk*. The ipk file is generated to work with Itsy Package Management System, This is a lightweight package management system specify created to be used in embedded hardware. It was used in the Unslung operating system in OpenWRT. Unslung is an open source firmware.

Also installed was Luci package. There is also some prerequisites for this package. You first need to connect to the router using an Ethernet cable. Open the internet browser and connect to http://192.168.1.1/. This is the administration page of the router. It will have the default factory firmware installed. You have to enter the login and password. Within the maintenance interface there will be an option to update system firmware. Click on the upload button, select the OpenWRT file previously configured for this specific target and confirm your selections. The router will go through a reboot and once it comes back up again it will be running OpenWRT firmware.

Installing the SoftEther package created with rootbuild is the next stage in the process. Before this is carried out enable SSL functionality in the router. This involves web browsing to 192.168.1.1 and changing the Administrators password and selecting what specific interfaces you want to enable SSL. You can now putty into the OpenWRT server.
At this stage you have the firmware image you want to install onto the wireless router. Running from the command line of the router rather than web browsing to the Luci web interface requires the use of putty SSH shell. From the OpenWRT command prompt using the below command will install SoftEther into the root directory.

opkg install /tmp/sfotethervpn*

Note that we have ran the .ipk file from the tmp directory so the install file can be removed after the installation to save space on the embedded device Many of them only have just the minimum requirements on free space so it is essential to minimize the install by using tmp directory and only compiling the packages you need.

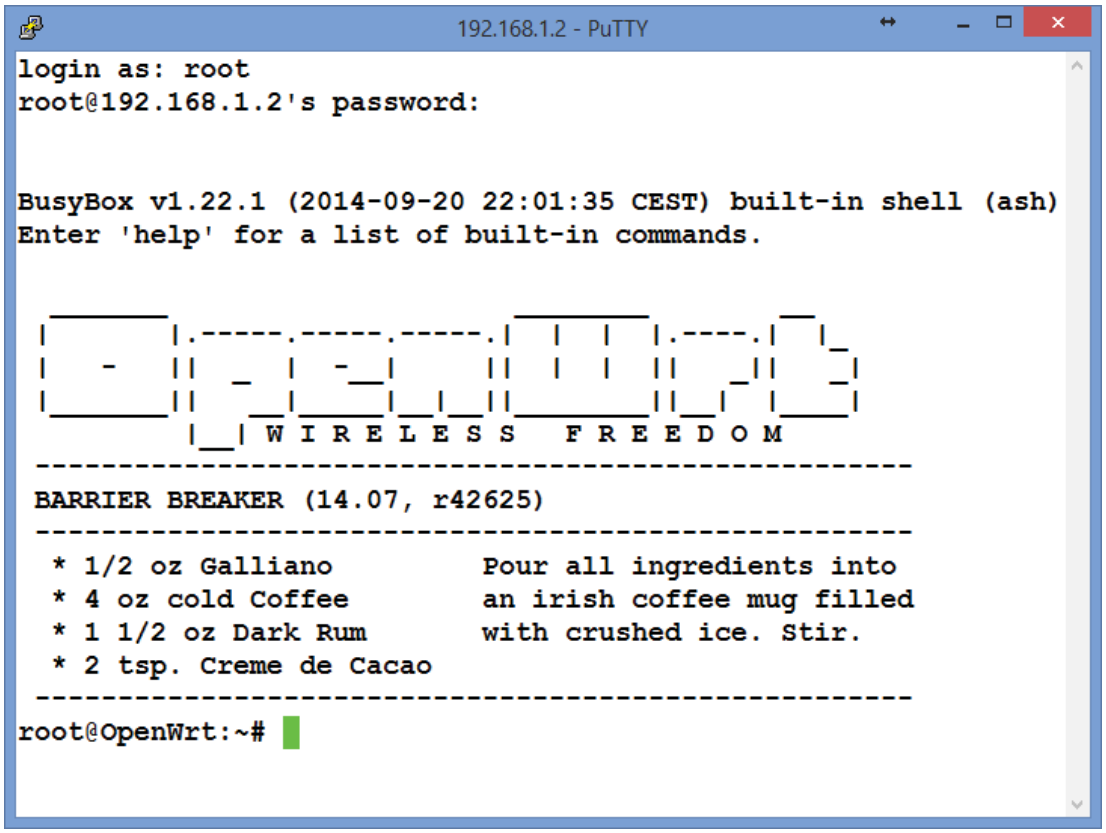

Figure 15: OpenWRT Barrier Breaker firmware on DLINK 505 router

The opkg package manager is a derivative of .ipkg. It's a lightweight package manager that can download and install OpenWRT packages from local package repositories or ones located in the Internet. You can have your own repository and a selection of previously downloaded packages. In my case we chose the option to download from the internet. The package manager attempts to resolve any dependencies with packages. If there are dependencies it will attempt to download automatically from the connected repository. If the automated update of dependencies fail then you will need to use the manual commands below.

opkg update

opkg install zlib libpthread librt libreadline libncurses libiconv-full kmod-tun libopenssl

After the Softether package is installed onto the router it is necessary to test the VPN server. For proper installation and functionality. Start up the server using command

/usr/bin/env LANG=en_US.UTF-8 /usr/bin/vpnserver start

The server will be available now to log into so we can check if everything has been compiled and installed correctly. Softether utilises a command line interface known as vpncmd to carry out some configurations.

/usr/bin/env LANG=en_US.UTF-8 /usr/bin/vpncmd

This will bring you to an administration screen where the initial server test tools can be accessed. Softether VPN Tools contains a VPN Operation Environment Check Tool. If the operation environment is operating sufficiently then it will in the majority of cases be

possible to run the VPN server smoothly. It runs 6 different checks covering the Kernel System, Memory Operation System, ANSI/Unicode string processing system, file system, Thread processing system and the network system.

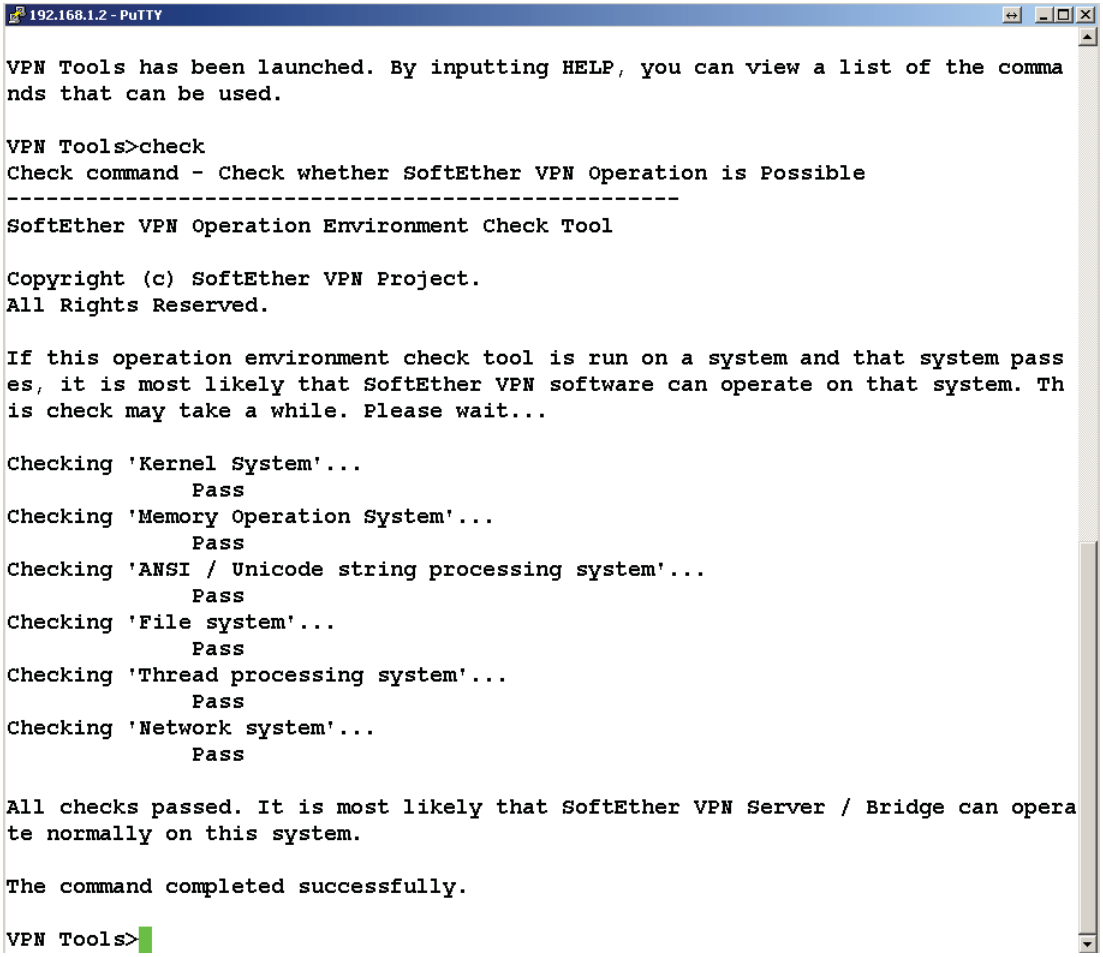

Figure 16: SoftEther VPN Operation Environment Check Tool

Previously known as SoftEther VPN Command Line Management Utility. Vpncmd package, in figure 17 below, does not require a web interface and is run in the command line. It can be used on all operating systems that support SoftEther VPN. This includes many different OS including Windows, Linux, FreeBSD, Solaris, and Mac OS X.

You can operate vpncmd using any of the following three modes.

- VPN Server / VPN Bridge Management Mode
- VPN Client Management Mode
- Use VPN Tools Command (Create Certificate or Measure Communication Throughput) Mode

VPN Server/Bridge Management Mode enables management by establishing a TCP connection on port 443 to the SoftEther VPN Server or SoftEther VPN Bridge running as a localhost or on a remote server running on a cloud based datacentre. Likewise, VPN Client Management Mode enables control by connecting to VPN Client running locally or on a remote server. The third mode, Use VPN Tools Mode, enables the use of only the test command and create certificate command only on the same platform that vpncmd is running. You cannot use this mode to connect and test to a remote VPN Server, VPN Client, or other services.

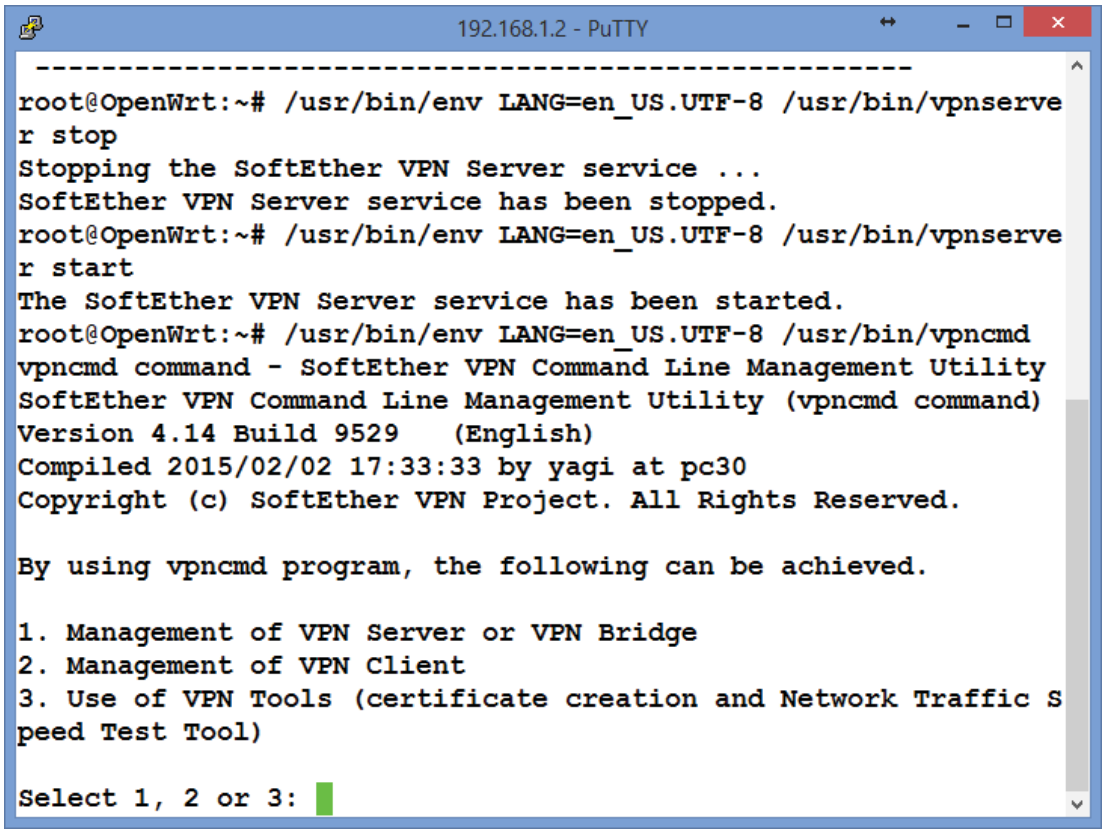

Figure 17: SoftEther VPN Command Line Management Utility

3.5.4 SoftEther VPN Server on Windows

SoftEther server needs to be installed on both a Windows server and also on the residential router. In the previous section we discussed how to implement SoftEther onto OpenWRT firmware. In this section we will cover the installation of SoftEther Server manager and Client Manager onto a Windows server. This can be installed using the setup file vpnserver-build-number-win32-x86.exe. This will start a wizard which will assist in the installation. Hard disk space and CPU speed of the server are items to consider before installation.

It is recommended that storage should be between 30 and 100GB in size. This depends on how you want to deploy the server and on how many connections you require. SoftEther logs many different system events, so depending on the size of your deployment this would be a good indication of the HDD capacity. The CPU can act as a bottleneck if you choose the incorrect speed for your network. If the CPU speed is too slow, the communication delay time may increase and data traffic may decrease [20].

As mentioned previously, the server can run in two different modes, Service Mode and User mode. The main difference between the two modes is that User mode would need to be started every time the server reboots. Service Mode will run the server in the background as a service. We will be running our system in Service Mode. The installer will run through a wizard where you enter the standard information about what directory you want to install to. In order to manage this server after installation it is necessary to install Server Manager, a GUI used to configure your SoftEther servers, no matter whether they are located locally or remotely.

The package itself is quite easy to install, just like any other application you need to install onto a Windows machine. It will install the SoftEther VPN Command Line Utility, the Server running as a service, a VPN server manager used to configure the server and also some administrative tools such as the Network Traffic Speed Test Tool. Once the installation is complete you will be prompted to start the VPN server manager where you can configure any of your servers or bridges from the same console, show in Figure 29 in the Annex. The most important part of this setup is in the implementation part which we will discuss in the next chapter.

3.5.5 Wireless Relay

The following steps need to be done for OpenWRT to install relayd onto OpenWRT router.

- Configure a managed network interface to connect to your 'upstream' Access Point
- Add a new wireless interface configured in AP mode with desired encryption
- Add a new network interface using the protocol of 'relayd' bridging the upstream and downstream networks
- Change firewall settings so that both input and output packets are allowed for both upstream and downstream networks

The guide on OpenWRT wiki page was followed and we will run through the steps below. This is all carried out by using putty to SSH into OpenWRT router and commands are run from the CLI. We also make use of VI Editor to modify the files [26]. Relayd was installed on TP-Link WA830RE.

The first step is to create an interface for the wireless router. This is carried out by modifying the file located in */etc/config/network*

```
config 'interface' 'wwan'
        option      'proto'          'static'
        option      ‘ipaddr’        '192.168.1.254'
      option        ‘netmask’ '255.255.255.0'
        option      ‘gateway’ '192.168.1.1'
```

The next file that needs to be modified is located in */etc/config/wireless.* The below lines need to be added to it.

```
config 'wifi-device' 'radio0'
      option '      ‘type'      'mac80211'
      option        'channel'   '11'
        option      ‘disabled’  '0'

config 'wifi-iface'
      option        'device'    'radio0'
      option '      network'    'wwan'
      option        'mode'      'sta'
      option        'ssid'      'Telia1’
      option        'encryption' 'wpa-psk'
        option      'key'       'secret-key'
```

The Wi-Fi network needs to be restarted, and the relayd package needs to be installed and enabled with the below commands.

```
wifi down; wifi
```

opkg update
opkg install relayd
/etc/init.d/relayd enable

The file located in /etc/config/network needs to be modified once more to declare the relay interface.

```
config 'interface' 'stabridge'
    option      'proto'     'relay'
      option    'network'    'lan wwan'
      option    'ipaddr'    '192.168.1.254'
```

The resulting interfaces of the OpenWRT router should now look like the figure below.

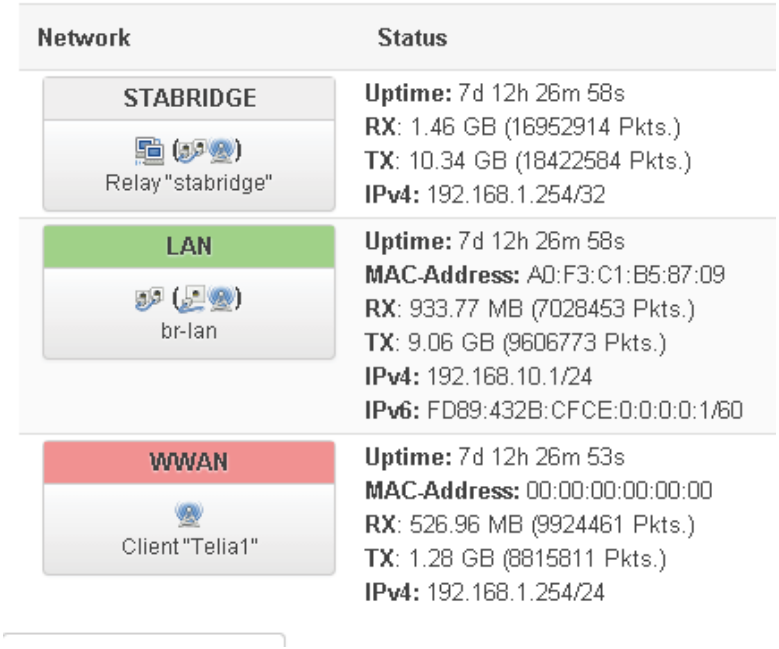

Figure 18: OpenWRT Network Connections

3.6 Configuration

3.6.1 Softether on OpenWRT

When you administer the vpn connection you need to specify the hostname or IP address of the computer that the destination VPN server or Bridge is operating on. You do have the option to specify a port but we shall use the default port 443. You also need to specify the Virtual Hub Name. In order to enable the server to start on boot the command below is used. This saves on us having to start the server after every reboot.

Port forwarding in the firewall to open the ports needed is carried out by accepting TCP ports 443, 992, 1194 and 5555. The firewall config file located in */etc/config/firewall* can be amended with the settings in portforward.txt, annex A. Once the configuration file is ready you can restart the firewall with the command below

/etc/init.d/firewall restart

You then need to use SoftEther Server manager to connect up your cascade connection. When you connect to the OpenWRT server below you need to tell it to connect to the EC2 Server via a cascade connection below. This would be the destination VPN server. When

you input the IP address a list of all available Virtual Hubs setup on the server will appear. Use the username and password setup earlier for the credentials.

In Figure 19 in the appendix we need to define the interface to use on OpenWRT router. As we are using a site-to-site VPN Ethernet Bridge. It may be the case that whatever router you use may have more than one interface so you could possibly use one interface for your connection to the Ethernet Bridge and one interface for the connection to the hosts.

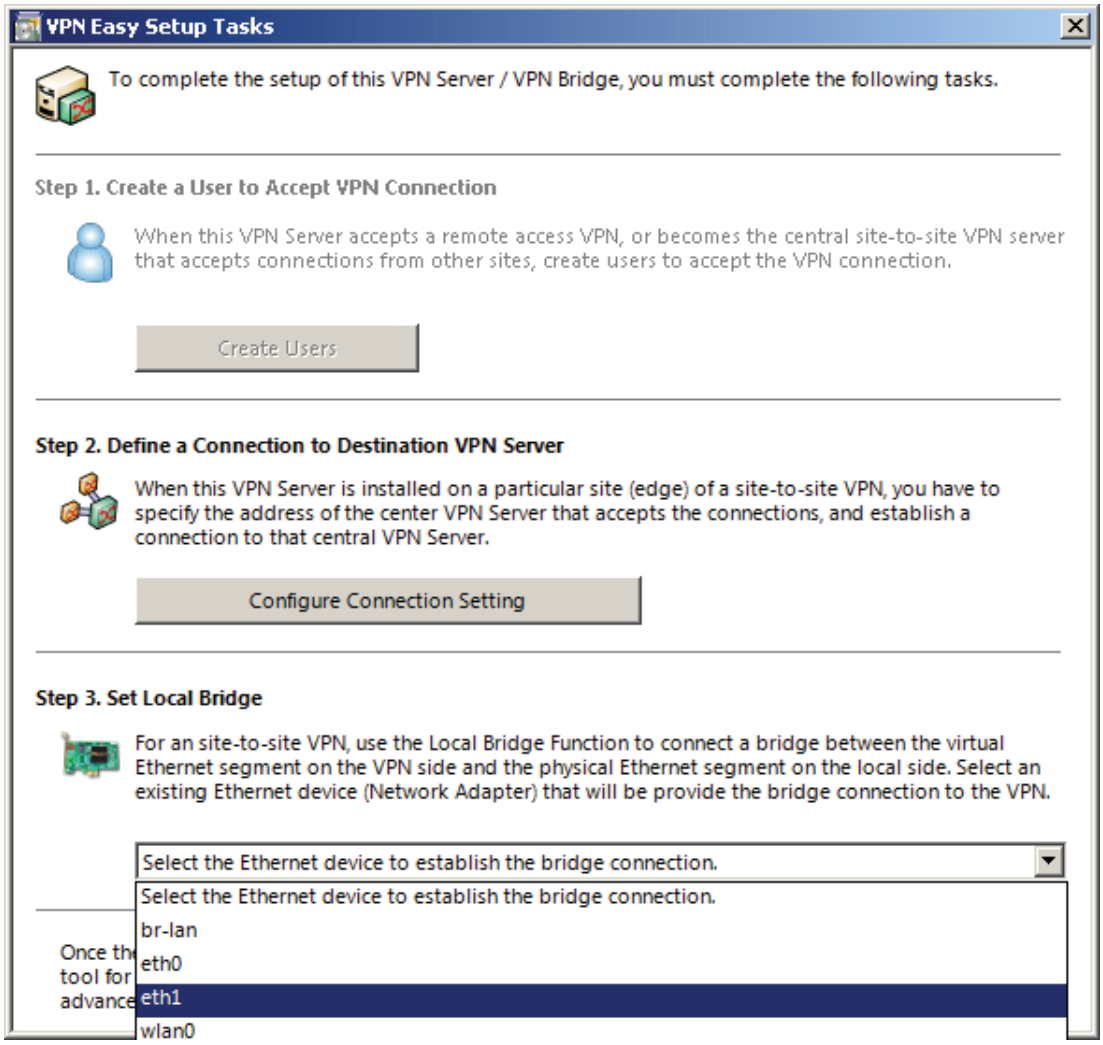

Figure 19: Interface Selection on SoftEther OpenWRT bridge

3.6.2 Softether on Windows

Once the Softether windows server is installed along with the server manager application you are in a position to start configuring the VPN server. All the configuration is carried out via the VPN Server Manager.

To connect to a specific server, open the VPN Server Manager and create a connection to that server, whether it is a cloud based server or a localhost. It can also be connected to a VPN bridge. Our connections will connect to a Softether bridge running on OpenWRT home router and also a connection to a Softether server running on Amazons EC2 and also PacketIX test lab virtual environment.

The main details needed to get the connection working are server IP address and port number. You can also select https or SOCKS proxy server setup but we will be using a direct IP connection. This method is used in an environment where only direct routing will be established between an endpoint, whether it is a client or bridge, and the VPN server computer. In our case both our VPN bridge and VPN server computer located on Amazons EC2 are both directly connected to a public IP address usable on the internet where normal NAT and a transparent firewall exists between the two entities.

I chose to also avail of the Automatic reconnect feature whereby the VPN link will try to be re-established if the link drops. The reconnection count and interval can be modified to suit a specific setup. We chose the Reconnection Endless option enabling the VPN client or Bridge to continually attempt to re-establish the VPN server.

Once a connection setting is setup and can reach the server, it is displayed the next time VPN Server Manager is started. The setup we desire is displayed in fig below.

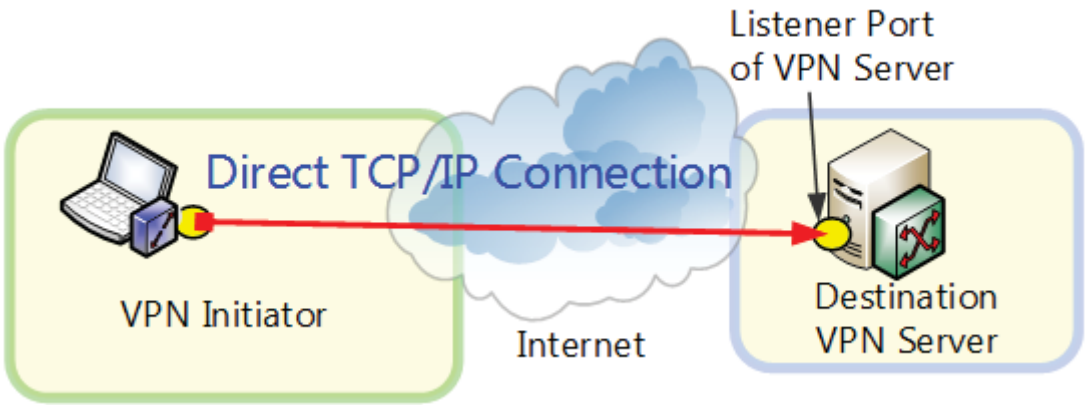

Figure 20: VPN Initiation

When the cascade connection is in place, all IP addresses, user names, passwords and virtual hubs setup it will be time to initiate the Virtual DHCP Server. In Figure 22 it is shown where to configure the DHCP server the way you want it.

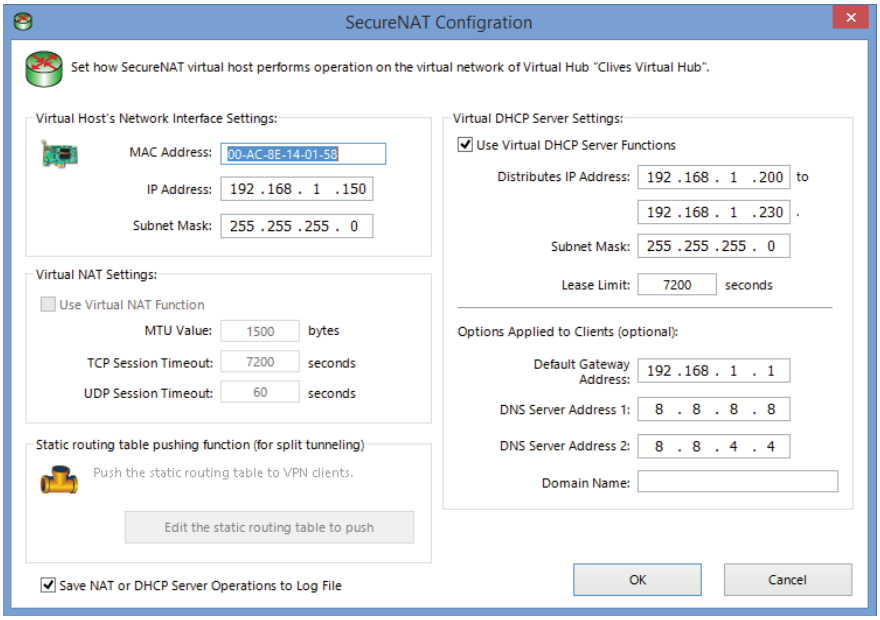

Figure 21: Virtual DHCP Server setup

4. Results

4.1 Virtual DHCP leases

As you can see from the results below all IP clients in the Home environment are now virtualized on the EC2 server located in the cloud. The DHCP server IP address is 192.168.30.1 and the DHCP pool starts at 192.168.1.200.

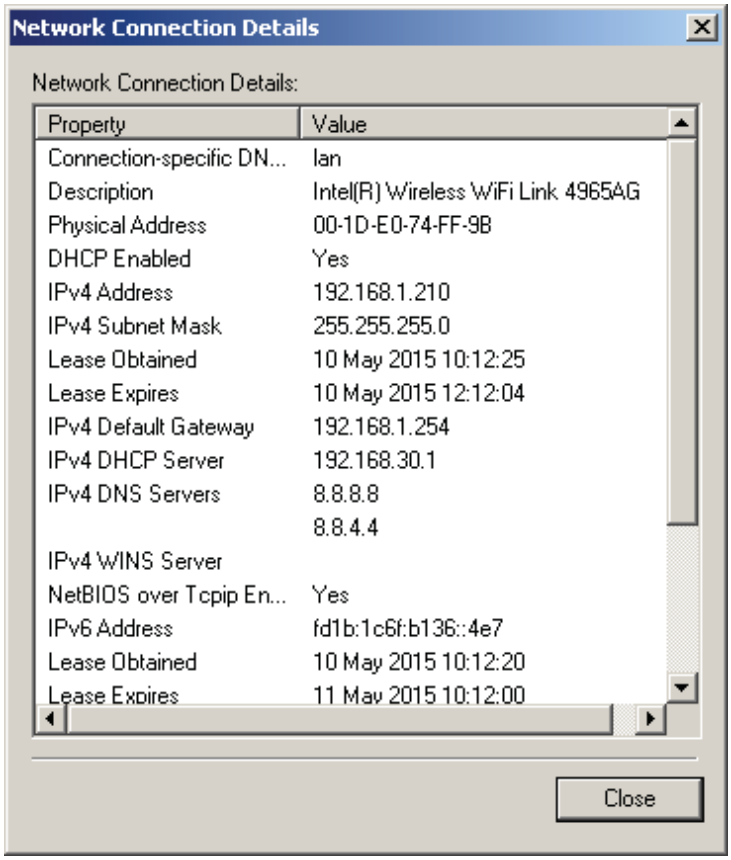

Figure 22: Windows Network Connection Details

The above figure shows the Windows Network Connection details and how it picked up the computer obtained a lease from the Virtual DHCP server. The DHCP address pool was set to 192.168.1.2xx, the netmask set to 255.255.255.0 to comply with the home network and the default gateway, whether it be the main DSL router of another wireless access point within the home. The DHCP server IP address can be set to whatever you like. As we want to reach it easily we set it to 192.168.30.1. DNS servers were set against Googles servers. The lease time was set as default.

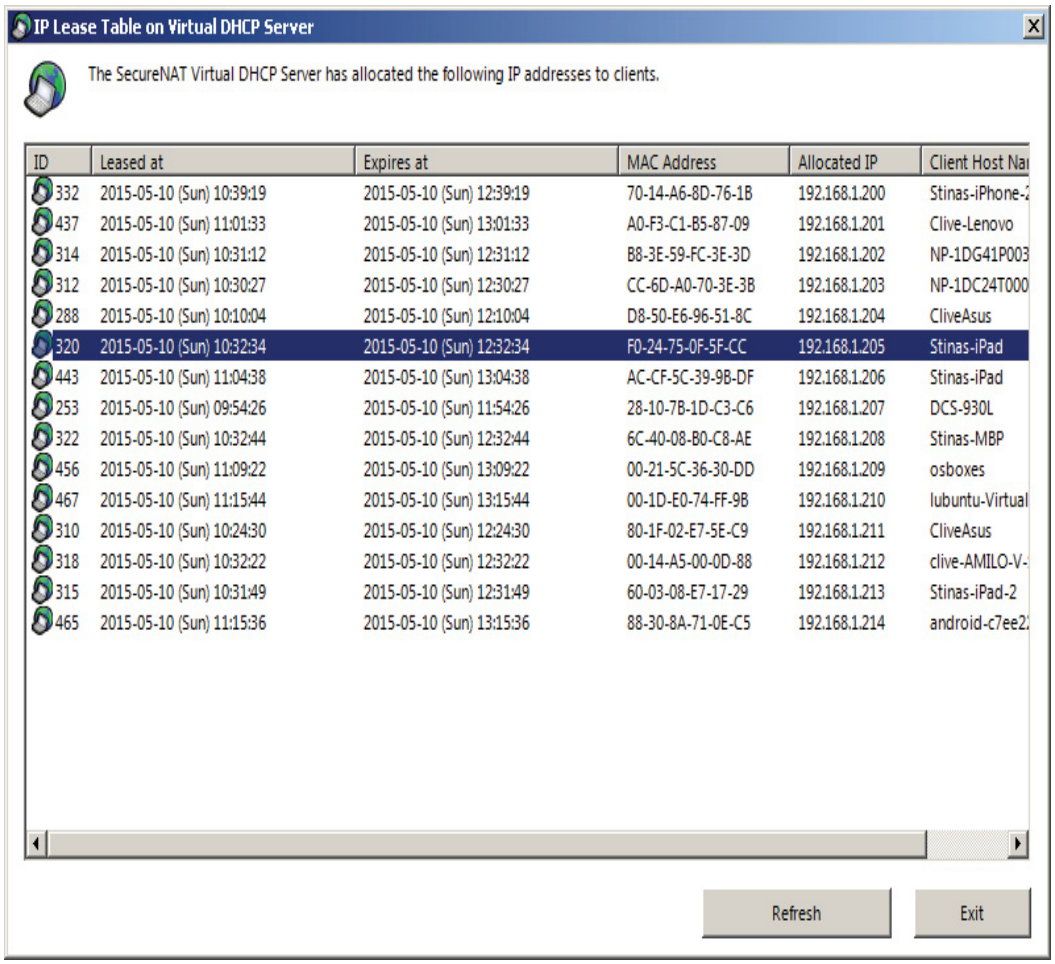

Figure 23: Virtual DHCP IP Lease Table

Since we are now running on the Virtual DHCP server we can connect remotely to check what leases have been disrtubited and what hosts are connected to my network. This is shown in Figure 23. This can be useful to identify any unknown hosts that may have intruded onto your system.

4.2 DHCP Tests

No.	Time	Source	Destination	Protocol	Length	Info
18	2.060985(	192.168.1.25	255.255.255.255	DHCP	306	DHCP Discover - Transaction ID 0xfffaa01
19	2.108150(	192.168.1.25	255.255.255.255	DHCP	306	DHCP Discover - Transaction ID 0xfffaa02
20	2.158172(	192.168.1.25	255.255.255.255	DHCP	306	DHCP Discover - Transaction ID 0xfffaa03
21	2.206066(	192.168.1.25	255.255.255.255	DHCP	306	DHCP Discover - Transaction ID 0xfffaa04
22	2.254140(	192.168.1.25	255.255.255.255	DHCP	306	DHCP Discover - Transaction ID 0xfffaa05
25	2.301480(	192.168.1.25	255.255.255.255	DHCP	306	DHCP Discover - Transaction ID 0xfffaa06
29	2.350068(	192.168.1.25	255.255.255.255	DHCP	306	DHCP Discover - Transaction ID 0xfffaa07
34	2.398022(	192.168.1.25	255.255.255.255	DHCP	306	DHCP Discover - Transaction ID 0xfffaa08
35	2.445441(	192.168.1.25	255.255.255.255	DHCP	306	DHCP Discover - Transaction ID 0xfffaa09

No.	Time	Source	Destination	Protocol	Length	Info
69	6.866783000	192.168.30.151	192.168.1.25	DHCP	344	DHCP Offer - Transaction ID 0xfffaa00
70	6.866789000	192.168.30.151	192.168.1.25	DHCP	344	DHCP Offer - Transaction ID 0xfffaa01
73	6.871936000	192.168.30.151	192.168.1.25	DHCP	344	DHCP Offer - Transaction ID 0xfffaa02
74	6.872143000	192.168.30.151	192.168.1.25	DHCP	344	DHCP Offer - Transaction ID 0xfffaa03
75	6.872147000	192.168.30.151	192.168.1.25	DHCP	344	DHCP Offer - Transaction ID 0xfffaa04
76	6.872148000	192.168.30.151	192.168.1.25	DHCP	344	DHCP Offer - Transaction ID 0xfffaa05
77	6.872149000	192.168.30.151	192.168.1.25	DHCP	344	DHCP Offer - Transaction ID 0xfffaa06
78	6.872151000	192.168.30.151	192.168.1.25	DHCP	344	DHCP Offer - Transaction ID 0xfffaa07
86	6.888639000	192.168.30.151	192.168.1.25	DHCP	344	DHCP Offer - Transaction ID 0xfffaa08
97	7.111233000	192.168.30.151	192.168.1.25	DHCP	344	DHCP Offer - Transaction ID 0xfffaa09

No.	Time	Source	Destination	Protocol	Length	Info
89	6.951798000	192.168.1.25	255.255.255.255	DHCP	328	DHCP Request - Transaction ID 0xfffaa01
90	6.952031000	192.168.1.25	255.255.255.255	DHCP	328	DHCP Request - Transaction ID 0xfffaa02
91	6.952269000	192.168.1.25	255.255.255.255	DHCP	328	DHCP Request - Transaction ID 0xfffaa03
92	6.952651000	192.168.1.25	255.255.255.255	DHCP	328	DHCP Request - Transaction ID 0xfffaa04
93	6.952956000	192.168.1.25	255.255.255.255	DHCP	328	DHCP Request - Transaction ID 0xfffaa05
94	6.953187000	192.168.1.25	255.255.255.255	DHCP	328	DHCP Request - Transaction ID 0xfffaa06
95	6.953490000	192.168.1.25	255.255.255.255	DHCP	328	DHCP Request - Transaction ID 0xfffaa07
96	6.953682000	192.168.1.25	255.255.255.255	DHCP	328	DHCP Request - Transaction ID 0xfffaa08
119	7.155736000	192.168.1.25	255.255.255.255	DHCP	328	DHCP Request - Transaction ID 0xfffaa09

No.	Time	Source	Destination	Protocol	Length	Info
103	7.152807000	192.168.30.151	192.168.1.25	DHCP	344	DHCP ACK - Transaction ID 0xfffaa01
104	7.152811000	192.168.30.151	192.168.1.25	DHCP	344	DHCP ACK - Transaction ID 0xfffaa02
105	7.152813000	192.168.30.151	192.168.1.25	DHCP	344	DHCP ACK - Transaction ID 0xfffaa03
106	7.152814000	192.168.30.151	192.168.1.25	DHCP	344	DHCP ACK - Transaction ID 0xfffaa04
107	7.152815000	192.168.30.151	192.168.1.25	DHCP	344	DHCP ACK - Transaction ID 0xfffaa05
108	7.152816000	192.168.30.151	192.168.1.25	DHCP	344	DHCP ACK - Transaction ID 0xfffaa06
109	7.152817000	192.168.30.151	192.168.1.25	DHCP	344	DHCP ACK - Transaction ID 0xfffaa07
110	7.152818000	192.168.30.151	192.168.1.25	DHCP	344	DHCP ACK - Transaction ID 0xfffaa08
121	7.296371000	192.168.30.151	192.168.1.25	DHCP	344	DHCP ACK - Transaction ID 0xfffaa09

Figure 24: Wireshark packet capture of DHCP Traffic

I used Wireshark on the same machine as the DHCP simulation machine. We captured the wireless interface. As you can see in the figure above we can identify all four stages of the DHCP process. First you have the discover message sent from the user, than the virtual server responds with an offer. In the offer package will we the IP address given to the client and it will also contain other information such as gateway address and what DNS server to use. The third message in the process is a request from the source, the user, and finally the acknowledgement is received from the server. Even though we ran the simulation for 100 requests we only show 10 transactions which is sufficient to give a clear idea of the process.

In the below table we can see the total time taken for the DHCP process from the initial discovery the final acknowledgement. It show consistency across the board for each transaction with mot much fluctuation between each result.

Transaction	Discover	Offer	Request	ACK	Total
0xfffaa01	2.0609	6.8668	6.9517	7.1528	5.0919
0xfffaa02	2.1081	6.8719	6.952	7.1528	5.447
0xfffaa03	2.1581	6.8721	6.9522	7.1528	5.0437
0xfffaa04	2.286	6.8721	6.9526	7.1528	4.8668
0xfffaa05	2.2541	6.8721	6.9529	7.1528	4.8987
0xfffaa06	2.3014	6.8621	6.9531	7.1528	4.8514
0xfffaa07	2.35	6.8721	6.9534	7.1528	4.8028

0xfffaa08	2.398	6.8886	6.9536	7.1528	4.7548
0xfffaa09	2.4454	7.1112	7.1557	7.2963	4.8509

Table 1: DHCP Transaction Measurements

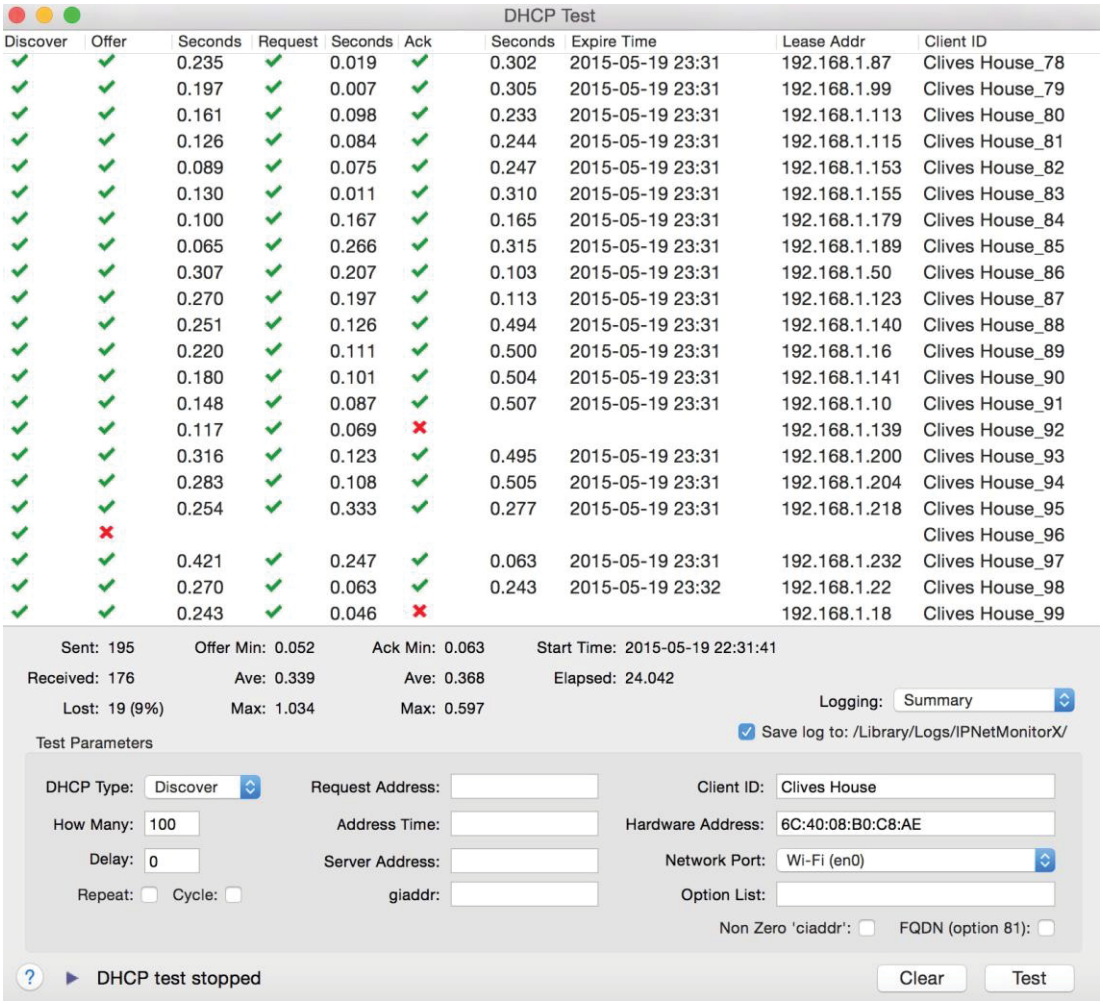

Figure 25: IPNetMonitorX DHCP Test Results for Home Gateway

In Figure 25 we have provided a screenshot of the tool used to simulate DHCP traffic. We tested 100 packets to the home router first. As you can clearly see from the measurements the virtualized server proves to be a better performer. It performs better with the amount of requests lost, it has a lower average time for offer and acknowledgements. It does have a higher time for the minimum offer and acknowledgements compared to the home router. This could be down to numerous factors, one being delay on the line. This is why tests need to be run with a bigger number of transactions so we can evaluate the overall performance. The elapsed time is significantly worse when using the home router. This would be caused by the CPU power of the home router compared to the virtual machine. If a value is input into the delay this difference would be much lower as this has the effect of giving the home router a chance to process the requests before failing. For comparison, the same test screenshot is provided in Figure 26 below. In this case we named the Client ID Clives EC2 Virtual Hub.

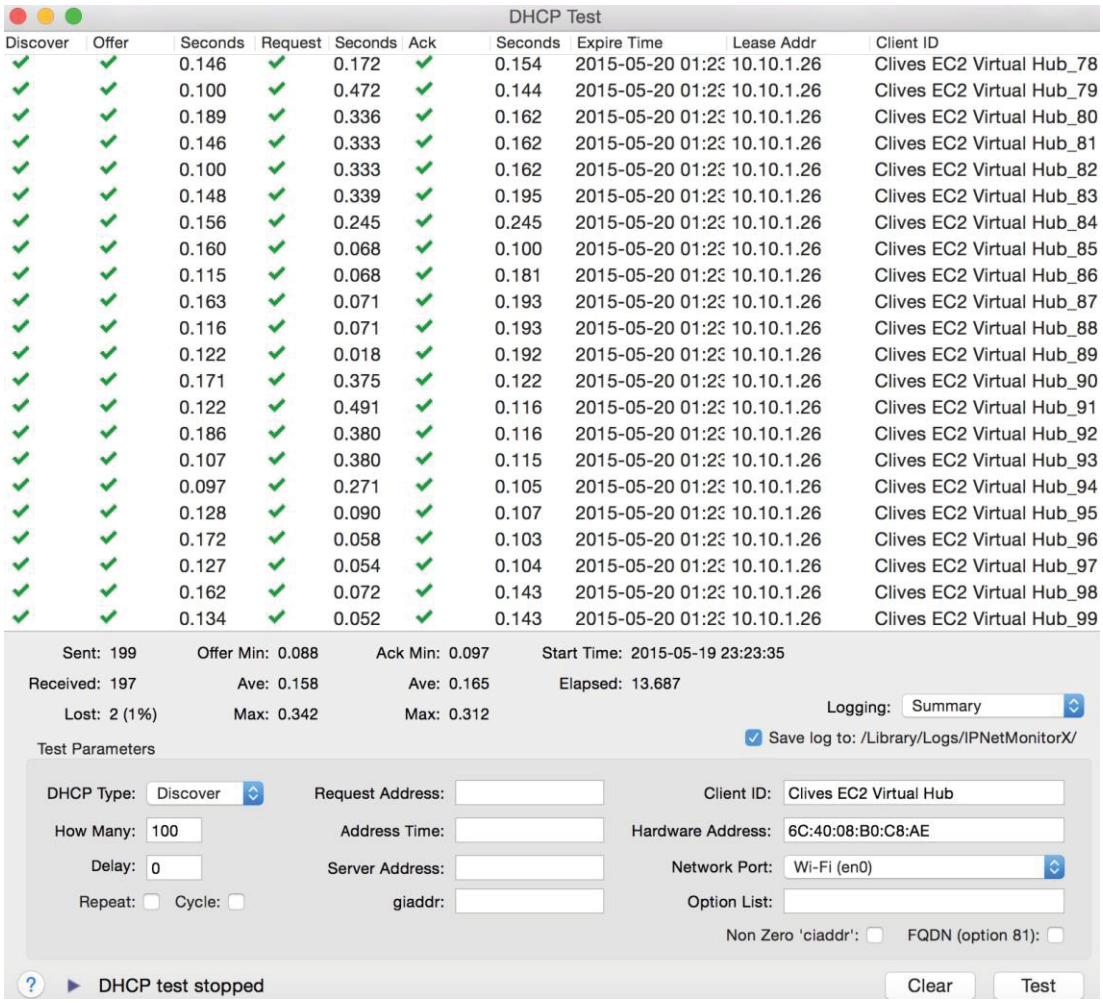

Figure 26: IPNetMonitorX DHCP Test Results for Virtual Router

5. Analysis

5.1 DHCP Traffic Analysis

As you can see from our results the DHCP traffic actually improves by using the virtualized DHCP server. As this cannot be down to the diversity of the link this is a direct result of the processing power of the physical home router**.**

Virtual Diversity Compliance in a virtualized environment exists due to the Service Providers various methods of transporting data. These method elements can include tunneling, QoS, security, policy enforcement, addressing, and operational procedures [16]. As we are using our own Virtual Private Network many of these methods can be eliminated due to the fact that we are running an Ethernet Bridge. Virtual diversity compliance refers to how successfully the NFV environment can successfully achieve the required diversity, minimizing the failure rate that a VNF service is refused due to compatibility issues. A metric to be used in this calculation would be the minimum number of diverse paths to be traversed in order to achieve the required connection. Many different paths which can be taken to reach a NFV service and in the case of our router, we would be connecting it to any available internet connection, whether it is a public wireless access point, tethering through a mobile phone or even through a corporate network. DHCP measurements may vary depending on the type of internet connection you use.

5.2 Virtual Network Operation

Packet delay in a Virtual network can be caused by a few issues. They can be the result of propagation or transmission delay, and also queuing delay at the various Network Virtual Function Components, VNFC, and endpoints [12]. This impacts the quality of service experienced by the user. The goal would be to minimize this latency so any delay would be transparent for the user's perspective. In the case of DHCP this has little effect on the end user as it is not a constant stream of data. If it is only an initial request for an IP address and depending on the lease time there will be no more need for any more communication with the server until the lease time expires or there is a need to renew an IP address.

As the amount of delay can differ from time to time we use the term Packet Delay Variation. It is defined as the variation of the packet latency when the packet traverses through the network [12].

Packet Loss Ratio for NFV, the packet loss to be considered should only consist of the maximum unintentional rate of packet loss. This would include packets lost due to the oversubscription of the service network interconnects. This ensures network stability [6]. This is a cause for concern as it impacts the QoS experienced by any specific user. Lost packets need to be identified and tried to be sent again. The calculated offered load is a key metric in this scenario.

A NFV network outage occurs with the loss of any virtual network connectivity. This causes service latency, decreased quality and availability to the end user. The metric used in this calculation is known as the VNF's Maximum Acceptable Network Transient Time parameter. [17] If a network outage exceeds this parameter then the system will try to recover itself automatically if High Availability, HA, is configured.

6. Conclusion and Future Work

6.1 Conclusion

We set out to achieve a Virtualized DHCP and we succeeded. We also stated that one of the objectives was to match or improve in the virtual environment any function in the physical environment. This was also achieved so much so that the virtualized DHCP performs better than the physical equivalent. This thesis shows viability of the concept of NFV in the Home Environment and how it can be achieved with limited resources. This setup was completed with cheap equipment in my own home and did not require any special hardware. We have proved that this setup can be achieved with high efficiency. This was also designed and implemented in a short amount of time without the need of any specialized lab equipment or software. The only cost to me was the purchase of the hardware. All other firmware, software and cloud server rental was free.

During the setup process no fundamental obstacle blocking what we wanted to achieve was encountered. We achieved NFV without the use of any expensive commercial equipment and proved that it is possible to virtualize for anyone that can follow the procedure we have outlined. In order to take our product to a commercial level the system would need to be hardened to be able to easily and quickly rollout to consumers.

6.2 Discussion and contributions

One advantage would be the realization that you can program DHCP pools for specific users thus enabling the allocation of Quality of Service dynamically to different users and applications. QoS can be applied to different types of traffic rules and can change dynamically, depending on demand. This is all achieved using different priorities for different applications.
Another obvious advantage it that of reliability and availability. Traffic can be rerouted in the case of failure or if maintenance is needed. Once the system has recovered or maintenance finished the original traffic flows can be put back in place. To enable failover a clone of the original Virtual Gateway can be booted up on a different server in a different location. This clone will be identical to the original Gateway. This switchover could also be automated in the case of network or power failure. Once the link is discovered down and the services unreachable, the Residential Gateway can try a secondary Public IP address where the cloned Virtual Gateway, vGW, resides.

Jon Crowcroft, highly respected in the world of Computer Networking and a Professor of Networked System at University Cambridge London in the Computer Science Department, is not a fan of NFV. From the mailing list he is a member of, he vents his disregard towards NFV and why he sees it unnecessary to introduce it. In his opinion, NFV is just another subnet of internet stuff that the developer will have to now consider whilst designing an end to end system. It is a piece of middleware that will possibly complicate development. To quote Crowcroft, he compares the introduction of NFV in the Communications world to a biological example,

"Try as we might, we cannot really see Network Function Virtualization as much more than yet another telco land grab on internet stuff. But somewhat more critically, we view the idea of taking some of our precious middle bodily fluid flow processing functions, and moving them off the box built by a middleware expert, and off the direct path, as actually counter-productive". [27]

I agree and disagree with Crowcroft. There is a reason why some functions should not be virtualized but there is also a good reason and good evidence from this thesis that other functions should actually be virtualized.

Crowcroft goes on to give 3 different examples of Network Functions, load balancer, wan accelerator and firewall, and why it is a disadvantage to move them anywhere other than the customer premises.

In the case of a load balancer, a function that is on the latency critical path before you get to any service, moving this to the cloud will introduce more hops and latency overhead due to virtualization and distance to reach the service. This would be in fact counterproductive.
In the case of a WAN accelerator, this is a service that is localized by definition, they are the middleware put in place to deal with TCP splitting and impedance miss-matches. To have this running in a server in the cloud would make no sense to do the fact that the same job will need to be conducted on the customer premises.

In the case of a firewall, this is a bit easier to understand. When you virtualize a firewall onto the cloud, by doing this is to hand over trust to a third party do deal with your system and protection. Firewalls were made to be running on trusted boundaries.

The device we created has the potential to be commercially deployed quickly and easily and have the benefit of bring your home network with you, wherever you go. It provides a cheap, easily maintained network without the need for a VPN for every device you have. NFV benefits the customer with the quick deployment of newly requested services which can be quickly installed and provisioned. They are also not tied down to what I single box can do, they can access services from a range of servers in the cloud running the services.

6.3 Further work

6.3.1 Increased Reliability

Since we now rely on a virtualized function to maintain our IP addresses in the Home reliability plays an important part in our setup. If you lose connection to the cloud server you will need to point your home gateway to another server to continue receiving IP addresses. In order to achieve this you can setup clusters in your virtualized environment. This can also be related to Distributed NFV.

A cluster in the cloud would normally be setup to offload different services over multiple servers so there is no reliability on a single server to be running all the services. This could also work well to improve our reliability if we could setup multiple servers that are running the same Virtualized Network Function so that if one server goes down, the other will kick in. There would be a need for some software synchronization between the servers to ensure that they all have the same database and are aware of any recent changes in the network. This would be a good topic for some further work to contribute to this thesis.

6.3.2 Improvements in System Performance

For my home network setup the equipment used was sufficient to provide adequate performance for the amount of IP addresses we am running in the home. But what happens in 5 years' time when the amount of devices that need to be connected to the internet is growing rapidly and most require broadband. Take for example the topic of Internet of Things. Every appliance in the home will be connected. We will be all living in smart homes where every day appliances such as utility meters, humidity sensors, temperature, gas, and smoke detectors will need internet connections. Because of all these connections the DHCP

server we used in this thesis may not be able to handle the traffic in the future. So scalability is an important factor in the future of NFV. This could be further investigated by carrying out tests with the same basic routers but pushing them to the limit in order to achieve peak performance measurements. The same principle could also be used to test performance in the case of the Service provider. The same basic principles we have used can also be used in a much bigger enterprise setting, possibly with much more powerful Servers in the cloud and communications with a lot more home gateways simultaneously to get a measurement of capacity.

6.3.3 Improved Fault Management

Virtualization architectures and concepts, like the one considered in this thesis, or the NFV architecture are complex, i.e. they compromise a large number of diverse software components. For example, the considered concept in this thesis relies on a VPN tunnel between the OpenWRT router and the VM.

However, debugging or finding performance bottleneck in architectures that comprise more multiple components is inherently more complicated than in systems with fewer components. Moreover, the multi-level nature of the components, e.g. delays in a VPN are typically caused by the network layer and the response times by a virtual DHCP server are caused on application layer even depending on the load of the server hosting the virtual function. In addition, the behavior of these subcomponents are often known, e.g. the delay of a VPN tunnel might differ essentially when the tunnel is established over a challenged WAN, which reveals long packet delays, in comparison to a less challenged WAN or even MAN/LAN environments with short delays.

Hence, multi-level fault correlation techniques are needed for fault identification, localization and root cause analysis in virtualization and NFV architectures [28] This requirement, for example, can be translated into the question how can one implement a robust multi-level monitoring and fault detection architecture for the considered virtualized DHCP concept? Other questions are, should the monitoring be based on a passive, active or hybrid monitoring concept? At which components should these monitoring point be implemented? These questions should be included in future extension of the considered DHCP virtualization architecture. All-in-all, the answers on these questions can due to the high complexity of the systems, apparently only be answer by further experimental research. That means it is recommended that the next spiral within the experimental research methodology should focus on enabling techniques and protocols for multi-level fault management.

7. References

[1] ETSI GS NFV 001: "Network Functions Virtualisation (NFV); Use Cases
[2] Rao Battula, L. Network Security Function Virtualization (NSFV) towards Cloud computing with NFV Over Openflow Infrastructure: Challenges and Novel approaches. In International Conference on Advances in Computing, Communications and informatics (ICACCI) Digital Networking Freescale, Hyderabad, India. 2014
[3] ETSI ISG on NFV: "Network Functions Virtualisation – An Introduction, Benefits, Enablers, Challenges & Call for Action" White Paper 1; Oct. 2012, Available at http://www.etsi.org/technologies-clusters/technologies/nfv
[4] http://cacm.acm.org/magazines/2011/8/114933-the-robustness-principle-reconsidered/fulltext
[5] *http://www.cyaninc.com/products/blue-planet-sdn-platform/insights/nfv-orchestration*
[6] Yunpeng Han , 2012, Parents Controlled Home Internet Cafes
[7] Daiyuu Nobori, January 16, 2013, Design and Implementation of SoftEther VPN
[8] Sheikh Riaz Ur Rehman , February 2009, Investigation of different VPN Solutions and Comparison of MPLS, IPSec and SSL based VPN Solutions
[9] Sanchez, Fernando, April 2015, Tethered Linux CPE for IP service delivery
[10] OpenWRT Wiki, http://wiki.openwrt.org/
[11] http://www.sustworks.com/site/prod_ipmx_overview.html
[12] ETSI ISG on NFV: "Network Functions Virtualisation – Network Operator Perspectives on Industry Progress" White Paper #3; Oct. 2014, Available at http://portal.etsi.org/NFV/NFV_White_Paper3.pdf
[13] ETSI GS NFV 003: "Network Functions Virtualisation (NFV); Architectural Framework.
[14] ETSI GS MAN 001: "Network Functions Virtualisation (NFV); Management and Orchestration.
[15] http://www.dlink.com.my/products/?idproduct=588&idCategory=271
[16] Zimmerman et al.2014. OpenFlow-enabled SDN and Network Functions Virtualization Open Network Foundation Solution Brief
[17] Whiteaker, J., Schneider, F. and Teixeira, R, 2011 Explaining Packet Delays under Virtualization ACM SIGCOMM Computer Communication Review. 41.
[18] ETSI GS NFV-PER 001: "Network Functions Virtualisation (NFV); Virtualization Requirements
[19] Jo-Philipp Wich, OpenWRT structure and design. http://wiki.confine-project.eu/_media/soft:openwrt-talk-2012-06-01.pdf
[20] ETSI ISG on NFV: "Network Functions Virtualisation – Network Operator Perspectives on Industry Progress" Update White Paper; Oct. 2013, Available at http://portal.etsi.org/NFV/NFV_White_Paper2.pdf
[21]https://www.softether.org/4-docs/1-manual/3._SoftEther_VPN_Server_Manual/3.4_Virtual_Hub_Functions#3.4.11_Cascade_Connection_Functions
[22] http://en.wikipedia.org/wiki/Buildroot
[23] http://svnbook.red-bean.com/en/1.6/svn.serverconfig.svnserve.html
[24] http://www.makelinux.net/books/lkd2/ch02lev1sec3
[25]https://www.softether.org/4-docs/1-manual/7._Installing_SoftEther_VPN_Server/7.1_Before_Install
[26] http://wiki.openwrt.org/doc/recipes/relayclient
[27] http://www.postel.org/pipermail/end2end-interest/2015-April/009292.html
[28] NFV(15)000093rl." Evolution of the PoC Framework with ISG hot topics"

ANNEX

Portforward.txt

```
config rule
        option src              wan
        option dest_port        443
        option target           ACCEPT
        option proto            tcp

config rule
        option src              wan
        option dest_port        992
        option target           ACCEPT
        option proto            tcp

config rule
        option src              wan
        option dest_port        1194
        option target           ACCEPT
        option proto            tcp

config rule
        option src              wan
        option dest_port        5555
        option target           ACCEPT
        option proto            tcp
```

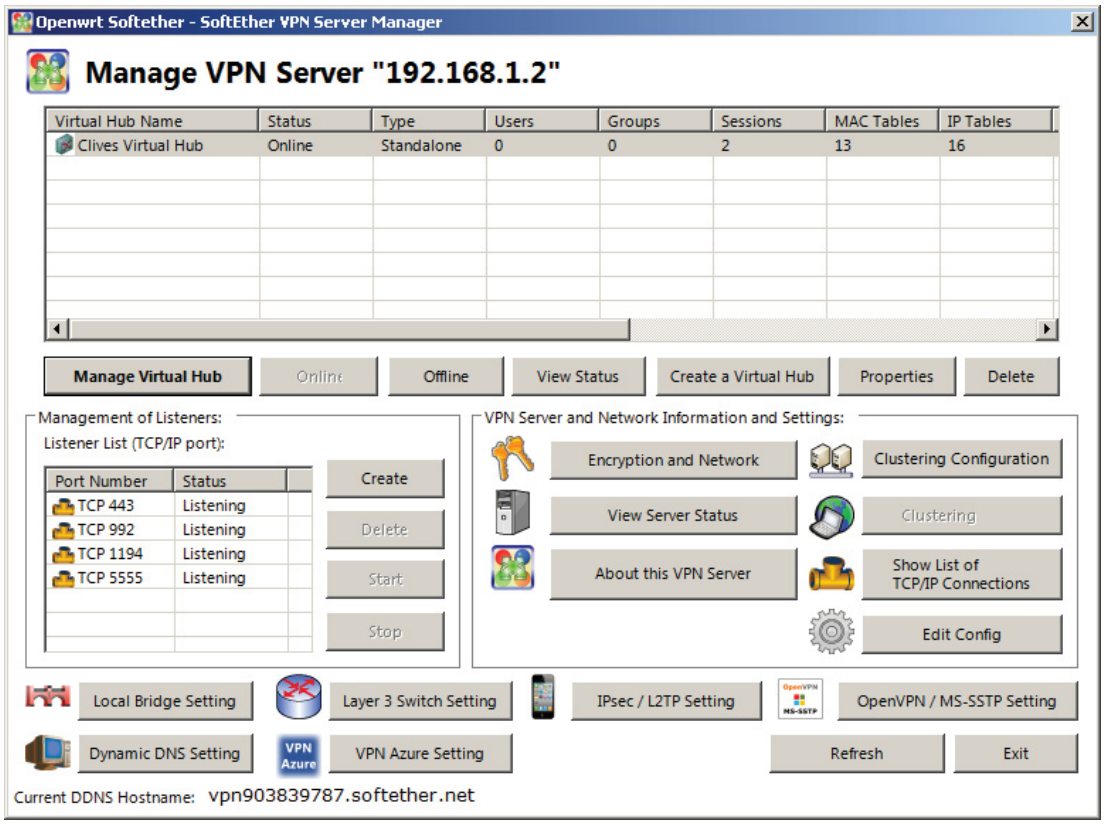

Figure 27: Connection to Softether OpenWRT bridge using Server Manager

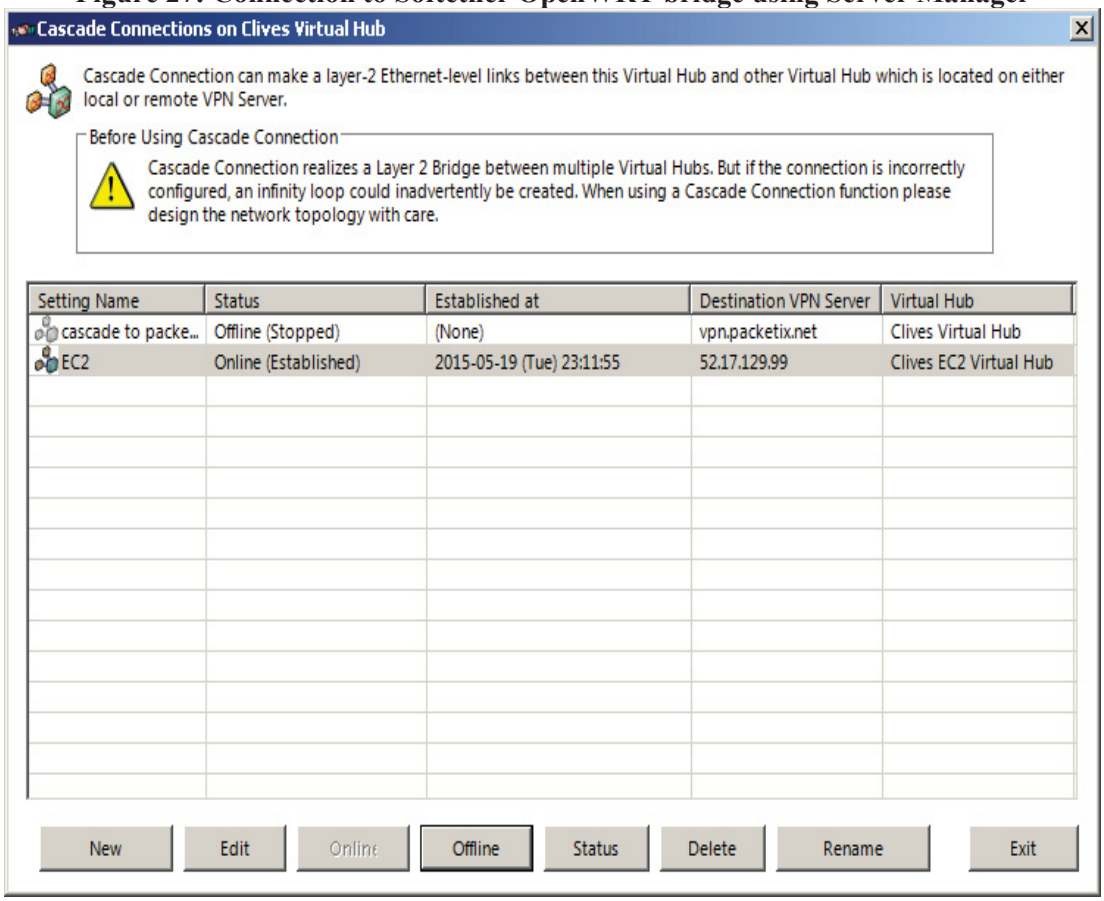

Figure 28: Cascade connections from Residential OpenWRT Router to EC2 SoftEther

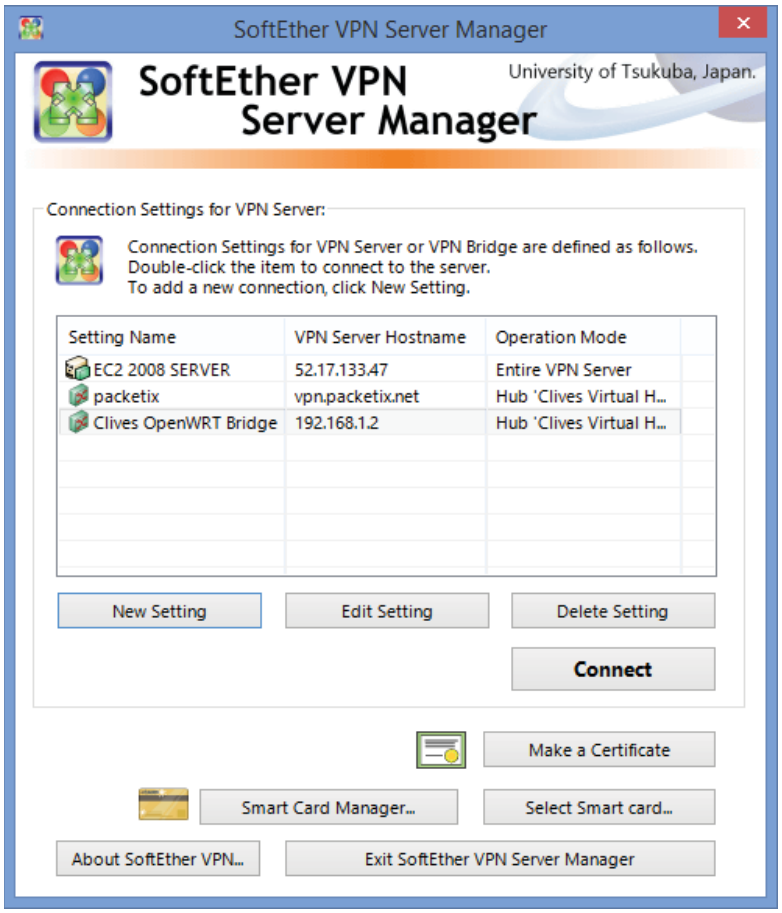

Figure 29: SoftEther VPN Server Manager

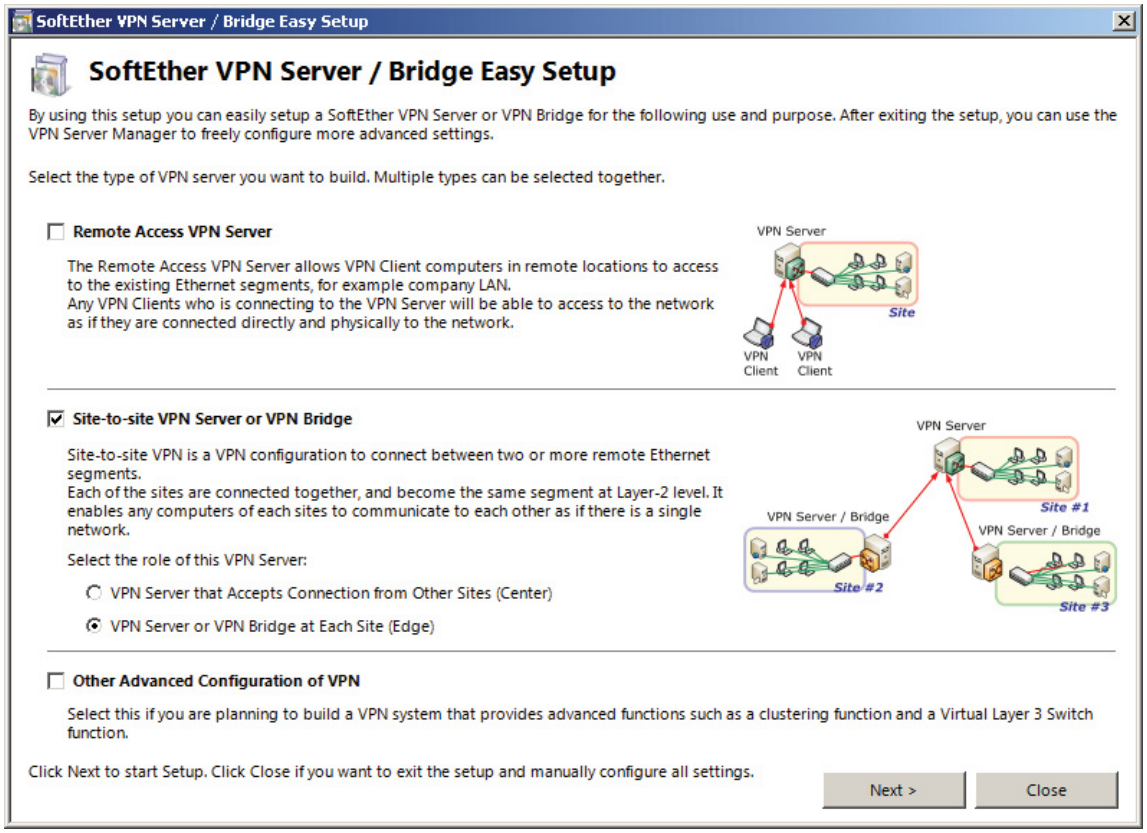

Figure 30: SoftEther Initial Wizard Screen