An Investigation into Network Emulation as a Testing Platform

Robert Game

Abstract

In recent years technology for large enterprise has advanced significantly due to the introduction of virtualisation of infrastructure. This has allowed even the smallest of enterprise to adopt low-cost, virtual copies of their Production servers as a test bed and facilitates the segregation of Production and Development by holding a copy of the Production server for testing in Pre-Production. This means that when planning changes to a server it can first be tested in order to ensure that no unanticipated issues occur. This practise has been well adopted on the server infrastructure, however, when looking at network infrastructure there has not been significant uptake in the use of virtualisation to facilitate testing in this way. Many Network Operations teams rely on the use of a physical lab for the testing of changes, however these labs take considerable time to set up both from a cabling perspective and from configuring each device. This project aims to investigate more efficient methods of testing network changes through the use of router virtualisation. In order to do this I will investigate the Cisco's new product VIRL and how it can be used to create a tool which will provide a testing platform for network changes. In addition to simulating the network infrastructure this project will also investigate meaningful methods of creating testing evidence for use in large infrastructure. When implementing network changes in large infrastructure this is particularly useful evidence to have as Change Management teams continuously seek to avoid risk.

Contents

Chapter 1

Introduction

In recent years the use of a second, entirely separate infrastructure has been used to develop tools, test changes and simulate outages of IT Systems. In a large scale infrastructure these environments are usually created for testing changes on Server or Application environments, however the need for real traffic and the cost of dedicated hardware makes this difficult in the Network Infrastructure. The use of a physical lab takes time to set up and teams usually do not use them unless they are planning for large scale infrastructure changes.

This project will aim to create a tool intended for use in a Network Operations team in large scale enterprise. This tool will aim to pull together different aspects of Network Management tools and use information available to create a network model inside of the Cisco VIRL emulated environment. Using this model the Network Operations team can test environment changes on a model of their own network. Using data gathered from this simulation and the changes observed during any configuration edits the tool will then aim to create a piece of testing evidence which can be used to support any Change Management required on the infrastructure.

Large organisations, which are the primary target of the proposed tool, will often have Configuration Repositories and Configuration Management Databases (CMDB) that store relevant configurations and server locations. Alternatively, a networks team may want to generate a emulated version of an unknown network in order to understand its topology and configuration. The proposed tool will combine network emulation technology with these data sources to allow a Network Administrator to mimic changes to a production network and observe the impact that they have on data flow and connectivity. In particular the tool will generate testing evidence as proof of any impact (or lack of) when raising a network change.

The proposed tool provides value to a Network Engineer as it allows them to be more flexible in their network testing. With the proposed solution an Engineer will no longer need to create a physical clone of the network in a lab environment, nor will the need rely on outdated network documentation. A emulated copy of the network can be produced from a node-edge representation of the topology along with the configurations for devices. This will save the Engineer time in the planning of changes as well as allowing them to implement the changes in a non-production environment, in addition to the time savings an Engineer will also be able to produce testing evidence which can be used for Change Management purposes.

1.1 Scope

This solution is targeted at medium to large enterprise which have a considerable routing set-up For example a global or regional backbone which connects multiple branch offices to a Headquarters. My tool will support both Interior and Exterior Routing protocols. In particular I am using my experience in the finance industry where Network Operations teams also work in an ISP role connecting through EBGP as well as traditional interior Distance Vector or Link State routing. This will create a tool that is scalable to the users needs, allowing a user to create a emulated copy of their local network through dynamic device discovery or a larger scale Engineer to produce an emulated version of a section of their network using Configuration Repositories and Network Diagrams.

1.2 Objectives

- Create a tool which can take information from existing resources and create a working model of the Network Infrastructure within Cisco VIRL.
- Extend the tool so that it can dynamically discover new network devices.
- Facilitate the creation of a simulation within five minutes of starting the tool.
- Produce a visual representation of the network for the user to view
- Allow the user to interact with the emulation through traditional methods such as SSH or telnet
- Investigate new methods for an Engineer to interact with a network
- The production of detailed testing evidence during the simulation, such as neighbour tables, routing entries or link statuses.

Chapter 2

Current Solutions and Cisco VIRL

2.1 Current Solutions

Full network simulation is a relatively new concept which has been embraced recently due to the uptake of hypervisors and virtualisation of Operating Systems. With regards to Network Simulation there has been a slowed uptake due to the majority market share that Cisco has and their limited offerings in the virtualisation market.

Other network device vendors such as HP and Huawei have offered simulated versions of their network equipment. These simulations tend to focus on creating a learning environment for their tools and as such do not perform adequately as simulated real world devices. These devices tend to work as a simulator in that they mimic the behaviour of a network, and not the devices. For example, the software will decide to build an OSPF relation should the required fields match on the devices. In contrast, a simulator (and real device) will build the relationship through a series of negotiations and packet transfers. As a result these simulators cannot be used reliably as network testing tool as the behaviour is predefined and may not be a true reflection of real hardware.

The Open Source community has played a role in this field, a project named GNS3 was started in 2006 which aimed to create unified tool to pull together hardware simulation and topology design into one UI. From this they created GNS3, a front end which allowed users to create a topology in a drag and drop style and then start equipment as required. GNS3 is intended to be a multi-vendor tool but is particularly used for Cisco simulations. Using a hypervisor called Dynamips the tool can create a VM of a Cisco router which can be interconnected to create a real topology. In addition to this, GNS3 can be adapted as a network emulator, which enables a user to connect their simulated topology to their physical network.

GNS3 works well for Cisco router simulation, however it falls short when looking at Layer 2 simulation. Features such as Spanning Tree Protocol rely heavily on the fast processing speed of Application Specific Integrated Circuitry (ASICs) which are specific chips within the switches that can quickly process frames. Cisco IOS combined with Dynamips cannot process frames fast enough in order to successfully implement these protocols. Therefore when testing redundancy scenarios such as outages which force a Layer 2 topology change cannot be tested. In addition to the issues outlined above there is no support for network discovery and the building of topologies for network testing.

From the review of current solutions there has not been a tool which can generate simulations of existing hardware. The only solutions at current are bare-bones building tool for network designs. The proposed tool is different in that its purpose is to be used in existing infrastructure to test disaster recovery or planned configuration changes.

Note: This is going to be expanded to include some papers on both simulation and network management techniques. Ideally taking form as a literature review

2.2 Cisco Virtual Internet Routing Lab (VIRL)

The tool will be implemented using Cisco VIRL as a simulation engine, VIRL is pre-release software which forms the back end for a future Cisco product called Cisco Modelling Labs. This tool provides simulation using a new version of Ciscos IOS and NexusOS, tailor made for virtualisation and fast deployment.

VIRL is a relatively new offering from Cisco, and is their first offering of software specifically created to support virtual copies of their routers. It was first released to the public in November 2014 as a stand-alone client installed on a local machine. However for the purposes of this Project I have been fortunate enough to have access to the University's own server-based version of VIRL. This provides support for up to 200 instances in comparison to the Personal Edition which is capped at 15 instances. This has enabled me to take advantage of extra features such as Servers which can be attached to a network topology, providing added features such as network analysis and traffic creation.

Connection to the VIRL Server is provided though an Eclipse-based front-end called VM Maestro. This front-end provides an sandbox to build network topologies using the provided images of Cisco hardware within VIRL. For this project I will be concentrating on the use of Cisco's IOSv image, an image of the traditional IOS but adapted for use with a hypervisor. This allows the image to be run on a server without any dedicated ASICs. VIRL can then interconnect these devices provide connectivity and traffic. Using these connections the Routers can use routing protocols such as OSPF, EIGRP or BGP in order to create routing tables - this enables the user to simulate the Layer 3 functionality of their topology.

VIRL has also been designed with flexibility in mind for it's users, one specific feature of note is the support of a tool called Autonetkit. This integration allows VIRL to create a fully functioning topology in a matter of seconds as configurations such as ip addressing and routing protocols can be passed off to Autonetkit which will auto-generate a working topology.

In addition to the features provided for functionality within VM Maestro, there are also a variety of configurations available to connect VIRL to the outside world. The first available connection is FLAT Networking - this provides a layer 2 interface which can bridge VIRL with physical hardware. This is particularly useful as it allows a user to integrate Layer 2 functionality with their simulation, an area in which VIRL is currently lacking. The second method of connectivity is through a SNAT connection on the VIRL simulation, this allows the simulation to assign external addresses to simulated objects such as Routers and Servers.

Server support within VIRL is provided through pre-defined image varying in system requirements. The default images are Ubuntu but other images can be imported if required. These servers can be connected to the internet in order to customise them for to the users needs, an example would be installing an SNMP Server in order to poll the routers for information.

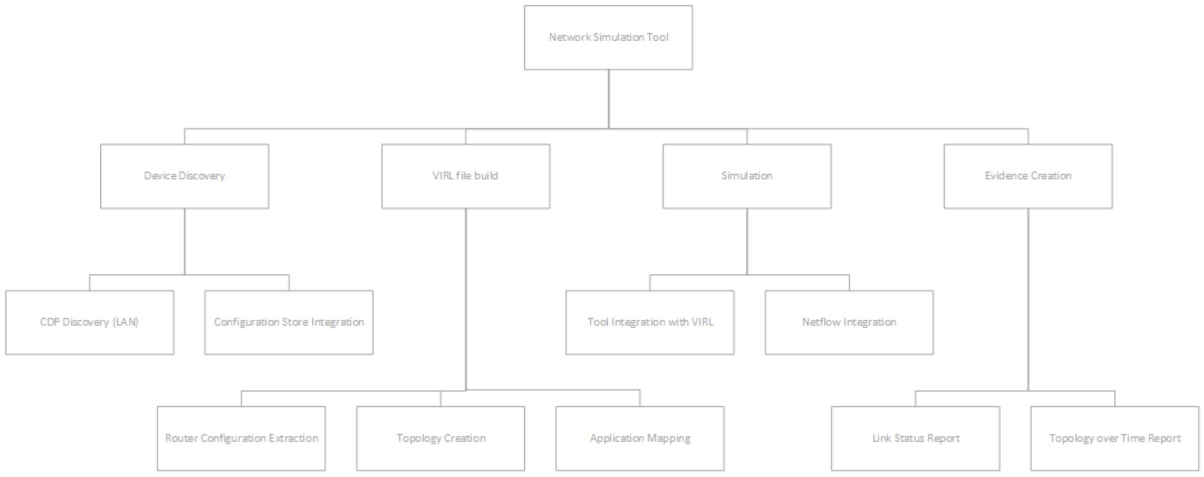
Chapter 3

Proposed Tool - Network Emulator

3.1 High-Level Design

At the highest level the proposed tool will provide an Engineer with an emulated version of their physical network for use as a test bed for changes or disaster recovery simulations. At a more granular level the tool is split into four distinct sections: Network Discovery, Network Representation, Simulation and Network Interaction. This chapter will focus on these sections from a design perspective, investigating how they can be implemented and which technologies can support their creation. Figure # below shows these sections further broken down into separate sub-tasks.

Todo: Create a flowchart for the system



3.2 Network Creation

An emulated network can be created using two distinct inputs, the first is a copy of the configuration files for all devices in the network which will determine how the components interact with

one another. For example, a configuration file can provide the settings for a routing protocol such as OSPF, specifying information required to exchange routes with other devices and choosing the networks they wish to advertise to their peers. The second input is a form of detailing the physical layout of the network, representing the devices as nodes and their inter-device links as edges. Representing this data in a format that is easily readable by a program allows the proposed tool to create emulation with all the interfaces and connections of the physical network, an accurate reproduction of these components ensures that the emulation will behave in the same way as it's physical counterpart.
Todo: Extracts of Config files and visio/json
The proposed tool will provide two options for creating an emulated network. First allowing a user to create a version of a well-known network by allowing the input of already existing configuration backups and topology definitions from tools such as Rancid or Opsware. Alternatively, the user will be able to create an emulated version of their local network by using the tool to discover, map and backup their local devices before passing the information on to the simulation element.

3.2.1 Explicit Input:

A well-maintained network will have two components of management, the first is a detailed collection of network diagrams detailing the positions of routers and switches, the interfaces used for these links and any IP addressing these connections have. Tools used for the creation of these documents are programs such as Microsoft Visio or yED, these tools enable a Network Engineer to draw their diagrams and save them in a user-friendly visual format. A further extension of these tools allows the saving of a topology in a programmer-friendly format such as json or xml, the proposed tool will investigate using this data format for representing the emulated network. Once the topology of the network has been created it has no defined behaviour.
The second management process is a backup tool of network device configurations, large enterprise keep a backup of all router configurations as a precaution in case of unintended changes or hardware failure. If given access to these backups the proposed tool will be able to create an emulated network whose behaviour is defined by these files.

3.2.2 Network Discovery

For networks that do not have the management techniques defined above, the tool will also be able to discover devices and create an emulated version dynamically. As before, the network will require both the topology definition and the device configurations to function. In order to gather this information it will utilise some existing network technologies and standards in order to gather the information required in the most efficient method possible.
In order to map the network topology dynamically, the proposed tool will use a combination of SNMP and CDP/LLDP in order to discover the devices in the local network. These technologies, when combined will allow for the tool to create a representation of the network in an edge-and-node format which can be parsed in the emulation software. CDP is a protocol created by Cisco which details a devices local neighbours giving information such as the neighbour's hostname, device type, IP Address, the connections local port and destination port. This information can be gathered from a device by using a standard called SNMP (Simple Network Management

Protocol)
Network configuration files can be extracted from running devices in multiple ways, the most common is to log on to a device and copy it's configuration to a network store such as a tftp (Trivial File Transfer Protocol) server where they are stored for later use. An alternative method is through further use of SNMP to poll a device for it's configuration which is returned over a terminal. The tool will use one of these methods to connect to and store a devices configuration. Once both the topology and the configuration of a network is obtained, it can then be passed to Emulation.

3.3 Network Representation

Although the VIRL system provides a topology view for running simulations, I am not intending on using this as part of the proposed tool. Instead, I will investigate methods of displaying a network using the information gathered in the Network Discovery section. I aim to investigate tools such as yED which allows the use of extensions to interact with network drawings. For example, when a topology has been drawn and displayed, it may carry additional attributes such as it's IP Address, using this information the graphing software can be extended to implement features such as a terminal emulator for direct access to a device, or an API for sending defined commands to an emulated network. In addition to the interaction with the emulation, I will also investigate methods of representing feedback from devices. This information can be displayed directly on the visual topology with labels such as the status of a network connection and it's current throughput. This idea can be extended and the use of colour on a network link can show the forwarding path for a destination IP address from it's source. Any changes made to the routing configuration during this time will be reflected by the colour of the links and any impact of the forwarding tables for a destination can be seen.

In addition to the feedback required from a visual representation of a network, there also needs to be a level of interactivity with the network view. A topology can be viewed as it's physical layout but it may have other configurations underneath this layer such as Autonomous Systems, Routing Areas or Redundancy Pairs. This information should be viewable by changing a filter on the network representation as it gives a more granular view on how the network components interact with one another. Adding these extra filters will enable a Network Engineer to view the impact of planned changes on additional aspects of the network which may not have been considered previously.

VIRL File Generation From the data gained from above, my tool will create a .virl file which defines the topology and configurations in Ciscos VIRL Simulations. This can define topology designs, interface connections and router configurations. Once this file has been generated it can be passed to Cisco VIRL to build the simulation.

Simulation in VIRL VIRL offers the flexibility to make configuration changes on demand. In addition to this link downs and hardware failures can be simulated within the tool. From VIRL the user will be able make any changes they wish.

Testing Evidence collection The proposed tool will utilise features such as Netflow, SNMP and Syslog in order to gather information from the simulation. From these flows the tool will create an evidence report which can be used in order to demonstrate any impact (or lack of) on the network infrastructure.

3.4 Requirements

Todo: Need to formally document these

3.5 Risks

- Constraints on access to University infrastructure
  - External to the VIRL Server there may be a requirement to have an external server which is able to access the VIRL network - as this is in the domain of IT Services, this may not be possible
- VIRL may not provide the functionality this project requires
  - As the current implementation of the Cisco VIRL service is pre-release, there is some aspect of missing functionality or suffers from pre-release bugs
- Added complexity through lack of documentation
  - As Cisco VIRL is still pre-release and limited availability there is a lack of documentation and community support for the product
- Requirement of Python in Cisco Development
  - Cisco products (In particular onePK) require a knowledge of Python. This is a new language that I have not developed in. Therefore there may be added difficulty in understanding its function.
- Risk of hardware failure
  - Project may be delayed in the event of hardware failure, particularly the VIRL Server in place which requires a significant time investment to install

3.6 Equipment Used

One of the goals of this project is the functionality for a user to import a custom topology into the system which can then be represented in VIRL. To achieve this I have designed a library of Topologies which could represent a scenario for which a Network Engineer would need to simulate in order to test changes.

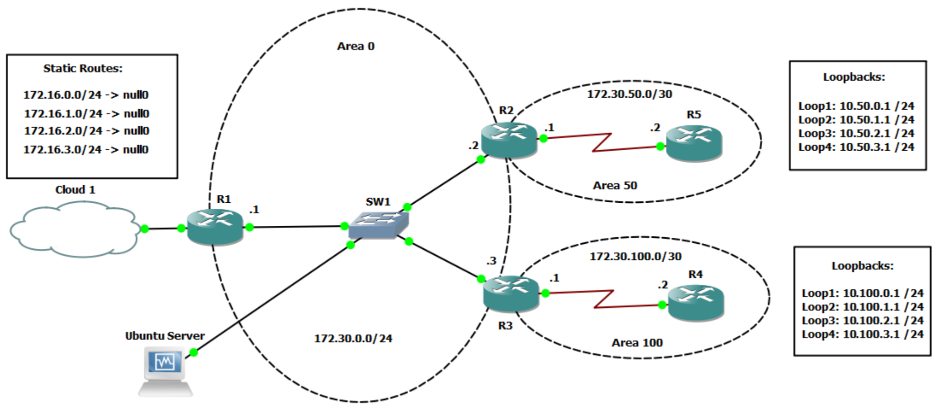
The creation of these systems needs to be done in a real-world environment, however due to the constraints I have on physical equipment for this project I have opted to create this topology in GNS3 (See Existing Solutions). GNS3 provides full simulation of IOS images and therefore is the closest representation of physical equipment. These simulations were created and run on a Windows PC using the GNS3 client. In addition to router simulation, GNS3 also has support for Virtualbox integration within the tool, this allows Virtual Servers to be connected to a topology and function correctly on a network.

To do: Further information on SNMP agent used on the 'Real' equipment

Chapter 4

Implementation

4.1 Router Discovery:



A key feature of the a simulation tool is the ability to replicate a 'Real-world' network in emulated space. In order to gather the information required for this feature an extra tool set for Network Discovery has been created. This allows a user to discover devices on the network and extract interface information and router configurations in order to build an emulated instance of the network.

For the development of this Router Discovery tool the above topology was created and used. As explained earlier this is implemented within GNS3 to utilise it's simulation of real IOS images. The above shows a Topology I have created as a use-case for simulation testing, however in order to add the Network Discovery tool I have included an Ubuntu 14.04 server and a connection to the Internet (Cloud 1).

The Ubuntu server is implemented using VirtualBox integration within GNS3 and allows direct connection to the 172.30.0.0/24 network. This Ubuntu server will hold the tool used for Router Discovery and Configuration Extraction. As this server is sitting on the same private network as all of the routers this allows me to find all the routers using defined ip addresses or through using CDP.

The connection to the Internet is provided through the cloud in the diagram above. This is inbuilt functionality within GNS3 which allows the topology to interface with the local hosts connection. This is implemented by adding a loopback to the local PC and bridging the internet connection to this loopback. GNS3 then connects to this loopback address and provides an interface to the Internet in the form of the Cloud object. Once this connectivity is provided the topology has a defined router which is used for internet connections, in this scenario R1 provides the connection. This router is then configured to provide dhcp to any servers on the local subnet, providing a default gateway and DNS Servers for the connected hosts. This router is also configured to provide NAT services in order to allow these private addresses to communicate to the outside world. Once internet connectivity is established the system can deploy.

Data from these devices will be extracted using an SNMP agent. Cisco Discovery Protocol (CDP) is a protocol created to collate important information about a Cisco device and broadcast it out to the network. For example the `show cdp neighbours` command on R1 above yields the following information:

```
R1# show cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID        Local Intrfce    Holdtme    Capability Platform Port ID
R2               Fas 0/0           157         R S I    3725     Fas 0/0
R3               Fas 0/0           156         R S I    3725     Fas 0/0
```

From the above I can see that R1 is directly connected to two routers, R2 and R3. I can also see which interface each side of the connection is on. As both are currently sitting on `Fas 0/0` this indicates that they are both connected to a switch. This is valuable information when understanding the topology as it defines which cables are connected where.

This information is useful as it allows a user to view what devices are directly connected without having access to an overall topology. However, the need for physically connecting to the device and running commands on the CLI is not optimal for an automated system as is the one being created. In addition to the process of logging on to a router, there is also the credentials required to facilitate this access. This may be an extra method of access that a Network Administrator would not be comfortable giving to an automated system. As a resolution to these issues the tool will utilise SNMP as a way of polling devices, in particular a Cisco MIB called CISCO-CDP-MIB allows this information to be gathered and accessed using SNMP. From this MIB I can gather the hostname and ip address of each neighbour to a device and from this I can then poll the neighbour to find details of his neighbour. This can be done programmatically until no further neighbours are found.

Once all the required routers are discovered the tool will then begin to implement extraction of each router's configurations. It will do this by using SSH to connect to each Router that has been identified and back-up it's starting configuration to the server. The server will then have knowledge of the network topology from the SNMP data and a copy of each Router configuration. From here a valid .virl file can be created of the topology.

4.2 VIRL File Creation

The VIRL engine expects a defined file type which is used to define the topology layout and the configurations used on first run of the simulation. In order to have the real-world network elements correctly represented in the simulation the topology and network configurations need to be correctly reflected in the VIRL file.

In order to convert the extracted data into the VIRL file I have created a python program on the local server. This program will parse each router in the topology and it's accompanying configuration file and create a correct virl file. Once this parsing is complete the topology is ready for simulation.

4.3 Simulation

Once a correct .virl file has been created the tool is ready to begin the simulation. This can be done in one of two ways, either by opening the VM Maestro client and loading the simulation manually...

Note: The available simulation methods supported within VIRL are currently not understood. It is clear that the simulation can be started using the GUI with a correct virl file, however this would not work well with the tool design as it requires the use of another program. The desired method would be to use a command line interface to to create a simulation. Details of that simulation can then be gathered using the REST API on the VIRL server.

Simulations within VIRL will be created using a setting on the simulation engine known as a 'Private Simulation Network'. This option creates a management network on the simulation which allows access to the management ports of the Routers on the topology. In addition to this it also provides a Linux container (LXC) on the network on this management network. One issue that arises from keeping this tool separate from VM Maestro is that the ip addressing schema produced by VIRL for these hosts is only viewable from the GUI. I have identified that this information is available be sending an authenticated REST request to the API running on the VIRL server, the response to this will provide information regarding the simulation details. A particular detail of note is information relating to this LXC host is it's ip address (Which is defined by VIRL and is allocated per simulation) or a unique port number to connect on via the VIRL server ip. This allows access from outside of the VIRL simulation. The below output from an API call details this information:

```
"rob.My_Topologies@topology-gymofc.virl.~lxc-flat.External Address": {
    "Annotation": "131.231.97.151",
    "SimulationHost": "158.125.102.75",
    "managementIP": "131.231.97.151",
    "simExpires": null,
    "simID": "My_Topologies@topology-gymofc",
    "simLaunch": "2015-01-06T16:06:59.534563",
    "simStatus": "ACTIVE"
  },
  "rob.My_Topologies@topology-gymofc.virl.~lxc-flat.External Port": {
    "Annotation": "59313",
    "SimulationHost": "158.125.102.75",
    "managementIP": "59313",
    "simExpires": null,
    "simID": "My_Topologies@topology-gymofc",
    "simLaunch": "2015-01-06T16:06:59.534563",
    "simStatus": "ACTIVE"
  },
```

The above can provide flexibility to the way the system connects to VIRL. There are two methods of connection using the LXC host, the first is to use the ip address assigned to the host, this information is provided in the ”External Address” section. However this address can be dynamically assigned by the VIRL server and could be any address on the 131.231.97.128/25 network. A second method is to use the ”External Port” section above, instead of using the ip address assigned to the LXC host this instead uses the primary ip address of the VIRL server (158.125.102.75). The creation of the simulation informs the server to begin listening on this port, any connections to the server on this port will automatically be passed through to the LXC host. Due to this, I have opted to add a call to the API in order to check which ip address has been assigned. If this were not the case then only one simulation could be made at any given time, this would be achieved by defining which port was used for access through the VIRL Server ip.

4.4 Network Monitoring

To do: Investigate whether current Open Source solutions such as Cacti or MRTG will provide the output require or whether a new solution would be added as part of this project

4.4.1 SNMP Monitoring

IOSv instances, the focus of this tool support SNMP just as a physical router would. Using this access I can set up an SNMP Monitor on a server within the simulation which could provide detailed statistics such as interface counters. Using this information the system can determine at a high-level where traffic is flowing.

4.4.2 Netflow

Cisco Flexible Netflow is supported by Cisco IOSv and gives a more detailed view of traffic across router interfaces. Configuring each IOSv instance as a Flow Exporter and adding an collector such as Cacti will provide a detailed breakdown of traffic based on it’s Layer 3 and 4 information. This adds further granularity to the testing evidence that is created.

4.4.3 Packet Capture

All interfaces within VIRL have a defined ’tap’ interface on the VIRL Server, this can be used for packet captures. Using a packet capture in addition to the Netflow and SNMP will allow the tool to inspect the traffic at a much more granular level. One use case of this would be the testing of a routing change on an established session and how an application can deal with this.

Note: Access to these tap interfaces may require access to the VIRL server in the admin role

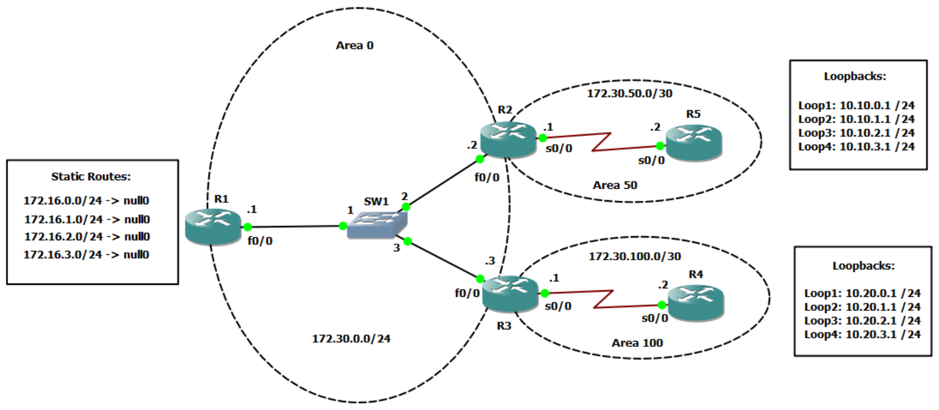
4.4.4 SDN Sources using OnePK

Software Defined Networking (SDN) is a relatively new concept in the network space and although has not fully been adopted in real-world standards it can be used to extract information from routers with a relatively small footprint in production. Where traditionally a routing table of a Router would be extracted by using a tool such as a screen-scraper or SNMP, the routing table from a OnePK enable device can be requested using an API-like interface. This approach scales more efficiently than using CLI-scraping as the data can be requested through an constant connection rather than opening a telnet/SSH session. Data returned from a OnePK call can either return data in a structured format or as a copy of the text displayed on the CLI. In the case of the structured data the appropriate information can be accessed directly, in the case of screen data it can be extracted using regular expressions.

4.5 Testing Scenarios

To support the development of this project I have created a set of network topologies with complex configurations. These are a representation of a network which a user may wish to simulate and can demonstrate added complexities which may be encountered in a real-world networks. In addition to this, they provide an already established routing platform which can be used for testing network changes. As described in the previous section "Equipment Used" these topologies have been created inside of GNS3.

4.5.1 Topology 1: Multi-area OSPF



I have designed this Topology to represent a simple network with a complex OSPF routing set-up, which may be a possible scenario that a network engineer may wish to experiment with before implementing changes in Production.

The topology represents the OSPF backbone of a large enterprise network, it includes three OSPF areas; area 0 (Backbone), area 50 and area 100. In this scenario R2 and R3 are taking the role of Area Border Routers (ABR) and providing route summarisation between the areas. For both area 50 and area 100 there are also addition Loopbacks created on the end routers which provide additional routes for this summary.

In addition to the routes being introduced via the loopbacks on R4 and R5 the topology also injects external routes into OSPF on R1. These are static routes defined on R1 which represent routes from either the Internet or routes from another routing protocol such as EIGRP, IS-IS or BGP. In this form the routes are sent to null 0 (blackhole) as there is no need for connectivity to them.

4.5.2 Topology 2: EIGRP Based Routing

4.5.3 Topology 3: Service Provider BGP Topology

Implementation Notes

Simulation Tools

Simulation Start

A simulation can be started on the VIRL host itself by specifying the .virl file to be processed and the user workspace for it to be used in. This process is shown below:

```
virl_std_client simengine-launch --virl-file /path/to/starter.virl
Session ID is starter-Ewn0Mn
export STD_SESSION=starter-Ewn0Mn
```

REST API

Details of the current running simulations for an account can be determined via the following URL: http://virl.lboro.ac.uk:19399/roster/rest/

The output of this called is shown in section 4.3

By analysing the output of this data the public ip address of the LXC Host can be determined and access to the simulation can be granted.

LXC Host

The LXC host is created when the simulation is first started, by selecting the Management Network type as a 'Private Simulation Network'. Upon the start of a simulation an additional Linux Container (LXC) will be created. This container will be available on a public ip address and can be accessed externally (within the Loughborough network)through SSH. Once connected to the container all other hosts on the network are directly connected and can be accessed using their management ip address.

Connectivity Tools

Creation of SNAT Connection

As described in the LXC Host section, all hosts can be accessed externally by using the LXC host as a jumphost into the simulation. However, this does not provide routing to the outside world. A use-case for the SNAT object would be a single point of exit to the outside world, as would be used in a standard network infrastructure. Connecting a SNAT object to a designated 'external' router can provide access to the entire simulation if required.

Default Routes

For any devices that require a SNAT object, a route needs to be added on the router for connection to the SNAT object. This can take the form of a default route (below)

```
ip route 0.0.0.0 0.0.0.0 10.254.0.1
```

Server IP Addressing

When connected to a multipoint connection a server is assigned an ip address in the subnet range, this is usually done sequentially. Alternatively this can be defined in the .virl file.

Server DNS

DNS Servers can be added to a host by editing the /etc/resolv.conf file. Alternatively they can be provided to a server if DHCP is enabled on the router.

RANCID Installation & Use

Look to set up SSH rather than telnet (For security)
Change Protocol type on router side (2vs1)
NOTE: Hostnames in the .cloginrc file need to be lowercase

Initial Discovery

Use cdpr on the discovery server in order to discover the default gateway address + port

sudo apt-get install cdpr

sudo cdpr

Python Path

export PYTHONPATH=/opt/cisco/onep/lib/python2.7/site-packages

SNMP to get CDP Neighbors

snmpwalk -v 2c -c public 172.30.0.1 .1.3.6.1.4.1.9.9.23.1.2.1.1.6.1.2

.1.3.6.1.4.1.9.9.23.1.2.1.1 - Gives information of sh cdp neigh

1.3.6.1.4.1.9.9.23.1.2.1.1.6 - Gives the hostnames of CDP neighbours

.1.3.1.4.1.9.9.23.1.2.1.1.4 - Gets the ips for the above

.1.3.6.1.4.1.9.9.23.1.2.1.1.7 - Gives local interfaces

.1.3.6.1.2.1.4.1 - Gives ip forwarding (1 is router, 0 is a switch)

.1.3.6.1.4.1.9.9.23.1.2.1.1.7 - Gives interfaces on neighbour