FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Faculty 2: Computer Science and Engineering

Network Slicing in an Emulated Network Environment: Voice over IP Communication through GENEVE

Subject: Mobile Computing – WS 2020/21

(Prof. Prof. Dr.- Ing. Ulrich Trick)

Author: Group 10

Anik Biswas (1324853)

Manash Chakraborty(1325085)

Dipanjan Saha(1324523)

Date of submission: 8th February, 2021

Abstract

Modern day networks such as the emerging Fifth Generation(5G) network supports vast number of services deployed in the network which requires diverse, complex and optimum network performance requirement for accommodating different types of traffic. Network Slicing is a method of creating logical end-end virtualized networks over a common physical network infrastructure. This logical segmentation of the network through slice is equipped to realize requirements demanded by a particular application or service. For this reason, Network Slicing is considered to be a key component for technologies such as 5G which requires diverse Service Level Requirements(SLR). Network Function Virtualization (NFV) is one of the concepts that is applied for realizing scalable slices of logical networks over physical network. Virtualization helps Network Operators to efficiently create network slices that can support specific applications or users within the physical network. Network slices can span across network domains such as Access, Edge or Transport Network and can be deployed across different network operators.

In this project we implement a network topology consisting of multiple network slices in Containernet Emulator environment. The topology consists of Access, Edge and Transport Network and accommodates Network slices for different tenants or end-users which are located in different access networks. The tenants can access Voice over IP (VoIP) service and communicate to other end-users within specific network slices and separation of service for different slices is ensured. The network slicing in the form of logical and virtual segmentation of network in edge and transport network is implemented through Generic Networking Virtualization Encapsulation (GENEVE) technology. Additionally, a few extensions features such as integrating Mininet WIFI, DHCP, Traffic Prioritization and Service Failover mechanism in the event of Server failure has been implemented in this project.

Keywords: Network Slicing, 5G, NFV, Containernet, VoIP, GENEV

1. Introduction

1.1 *Motivation (Dipanjan Saha)*

Networking Technology in recent years have been evolving to be more oriented towards service-based architecture. Different networks need to handle different types of heterogeneous traffic. With this type of heterogeneity in traffic flow, it is imperative to treat each type of traffic as per its requirement. One of the approaches to cater to this requirement is network slicing which is logical partitioning of the network tailored for different service specific requirement. Multiple virtual networks are sliced on top of a shared network in network slicing. Within the constraints imposed by the underlying physical networks, each slice of the network can have its own logical topology, security rules, and performance characteristics. Different slices can be dedicated to different purposes, such as ensuring priority access to capacity and delivery for a specific application or service or isolating traffic for specific users or device classes. For example, separate slices can used for Video Service, Voice Service, File Service or for different group of users in the same network. Motivation of this project is to implement network slices in fairly complex network infrastructure while addressing the challenges and difficulties in realizing the same.

1.2 Project Overview and Goal (Dipanjan Saha)

The Project objective is to create Network Slicing enabled for tenants located across different access network. One Access Network would connect to other Access Network via Edge and Transport Network. Voice over IP service (SIP server) would be hosted somewhere in the Internet (Transport Network). IP based Network slices would be deployed in the network. The network slices would be accessible by only a specific group of end-end users. The implementation would allow users within the slice to have separate and dedicated logical connectivity over the physical network for accessing VoIP Services. The logical separation of the network would be implemented through GENEVE Network protocol and separation of service would be ensured. Additionally, we attempt to integrate some extensions to this project which are enabling DHCP services for hosts in access networks, providing Wireless Access to tenants in some of the Access Networks and implementing a mechanism to switchover VoIP service to a backup VoIP server in event of failure in the primary server.

2. Technologies Used

This section highlights the important technologies that have been used to implement this project. Basic overview of the technologies and their functionalities in this project is discussed below.

2.1 Network Virtualization (Anik Biswas)

Network Virtualization is the process of combining Hardware and Software Resources of Network Technology and network functionality into a logical virtualized network entity. In this project we have implemented and used virtual network connections to logically segment the entire network infrastructure.

2.2 Ubuntu 18.04 (Anik Biswas)

Ubuntu is a Debian based Linux Distribution OS. Our entire Network Emulation has been implemented in Ubuntu 18.04.5 LTS version OS. Basic Linux System and Network Administration techniques have been used throughout the project.

2.3 Containernet (Anik Biswas)

Containernet is an open-source network emulator. Containernet is a fork of Mininet network emulator and supports Docker Containers as Hosts in emulated network environment. Mininet can create virtual network imitating real-time scenario and can be run on real kernel, switch or application code in a single machine. Our project is emulated and tested in Containernet virtualization platform. Containernet provides access of powerful Python API. With Python API it would be possible to customize the network as per design requirement such as topology, bandwidth, routing and switching, security features etc. The power of Mininet resides within some useful features of the Linux kernel such as process groups, CPU bandwidth isolation and network namespaces. These features allow Mininet to produce a lightweight emulation of a small-to-medium size network within a single Linux kernel. Mininet’s ability to use real Linux Kernel and CPU, will allow us to produce lightweight emulation of real-time small scale networks.

```
containernet@containernet-VirtualBox:~$ sudo mn
[sudo] password for containernet:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
containernet> nodes
available nodes are:
c0 h1 h2 s1
containernet> dump
<Host h1: h1-eth0:10.0.0.1 pid=3279>
<Host h2: h2-eth0:10.0.0.2 pid=3284>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=3289>
<Controller c0: 127.0.0.1:6653 pid=3272>
containernet>
```

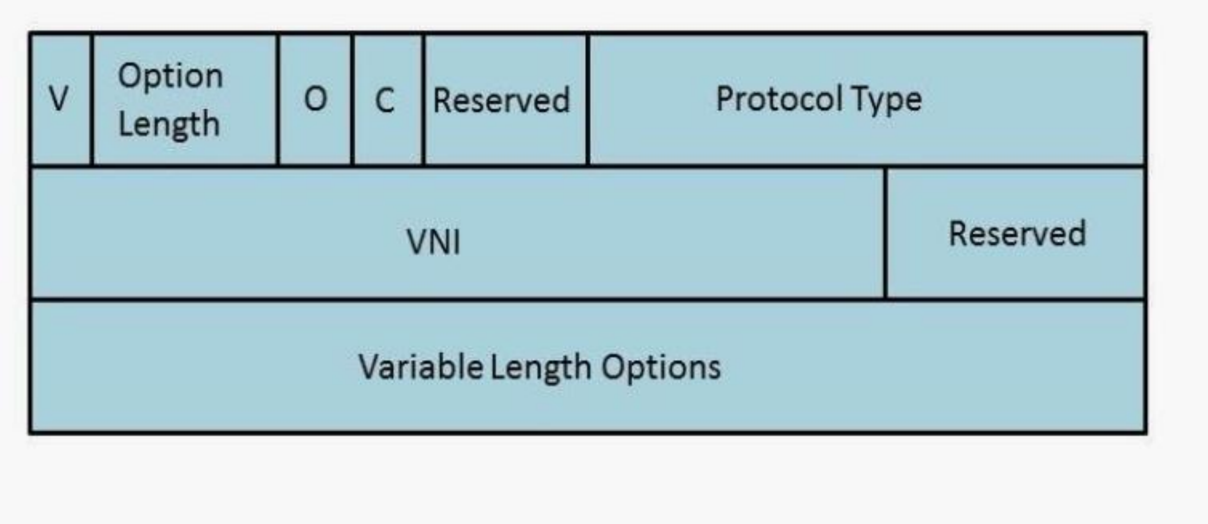
Default Network Topology 1

2.4 Generic Networking Virtualization Encapsulation (GENEVE) (Anik Biswas)

Generic Networking Virtualization Encapsulation (Geneve) is an encapsulation network protocol developed by the Internet Engineering Task Force (IETF) to bring together the efforts of other initiatives such as VXLAN and NVGRE in order to stop the proliferation of encapsulation protocols. GENEVE encapsulated packets flow through standard backplanes, switches and routers. It involves unicast or multicast addressing when the packet is flowing through one tunnel end-point to another. The receiving tunnel endpoint decapsulate the GENEVE header, checks for any included options and moves the packet to the end user point within the virtual network indicated by the tunnel identifier.

The GENEVE data format can be considered as a combined upgradation of VXLAN, NVGRE and STT. GENEVE does not mention any Control Plane, it is expected to work with any other encapsulation protocols.

In our implementation we used GENEVE as the tunnelling protocol to create logical segmentation in the Network to Virtualized Network.



GENEVE Header

2.5 Voice Over IP(VoIP) (Manash Chakraborty)

Voice over Internet Protocol (VoIP), also known as IP telephony, is a method and set of technologies for delivering voice communications and multimedia sessions over IP networks like the Internet. The terms Internet telephony, broadband telephony, and broadband phone service all refer to the delivery of communications services (voice, fax, SMS, and voice-messaging) over the Internet rather than the public switched telephone network (PSTN), also known as regular phone service (POTS). In our implementation we use VoIP services for the users in access network. SIP Server is hosted in Transport Network. Kamailio module is used to implement the SIP services.
On the client side Linphone is used as SIP client.

2.6 Mininet-Wifi (Manash Chakraborty)

Mininet-WiFi is a wireless scenario emulation tool that allows experiments to replicate real-world networking environments with high-fidelity. Mininet-WiFi adds virtual wireless stations and access points to the well-known Mininet emulator while maintaining the original SDN functionality and the light weight of the software for virtualization. In one of the Access Network, we have used Mininet-WiFi for some of the radio stations to access the network services.

2.7 Wireshark (Manash Chakraborty)

Wireshark is a tool used as a network or protocol (also known as network sniffer) analyzer to analyze network structure for different protocols. The analyzer operates on the operating systems of Unix, Linux and Microsoft Windows, using the GTK+ toolkit and pcap widgets for packet capturing. Under the GNU General Public License, Wireshark has many tcp dump features. The difference is that it supports a GUI and has filtering functions for information. Wireshark also allows the user to see all traffic through the network. In our Network implementation and testing we have used Wireshark extensively for Network performance analysis.

2.8 Docker (Manash Chakraborty)

Docker is an open platform for applications to develop, transport and run. Docker enables us to separate applications from our infrastructure so that software can be delivered quickly. We can manage our infrastructure with Docker in the same way our applications are managed. In the containernet platform we have used Docker images in hosts to realize specific service functionality such as VoIP, DHCP etc.

3 Environment Setup

The following sequences of component integration and installation was performed to build the necessary environment to carry our project.

3.1 *Virtual Box and Ubuntu Installation (Dipanjan Saha)*

Since we did not have any Linux based system available, we used Oracle Virtual Box to install necessary Linux distro in it. We created a Virtual Machine of 4 GB RAM and 30 GB Hard disk space. We installed Ubuntu 18.04.05 LTS distro in the same and enabled NAT connection to access Internet.

3.2 *Containernet Installation (Dipanjan Saha)*

Since we needed Docker support as well as Docker integration, we installed Containernet with Mininet Wifi. The module is available in GitHub and necessary steps can be referred from there. We performed Bare metal installation to successfully integrate the package in our Ubuntu Platform.

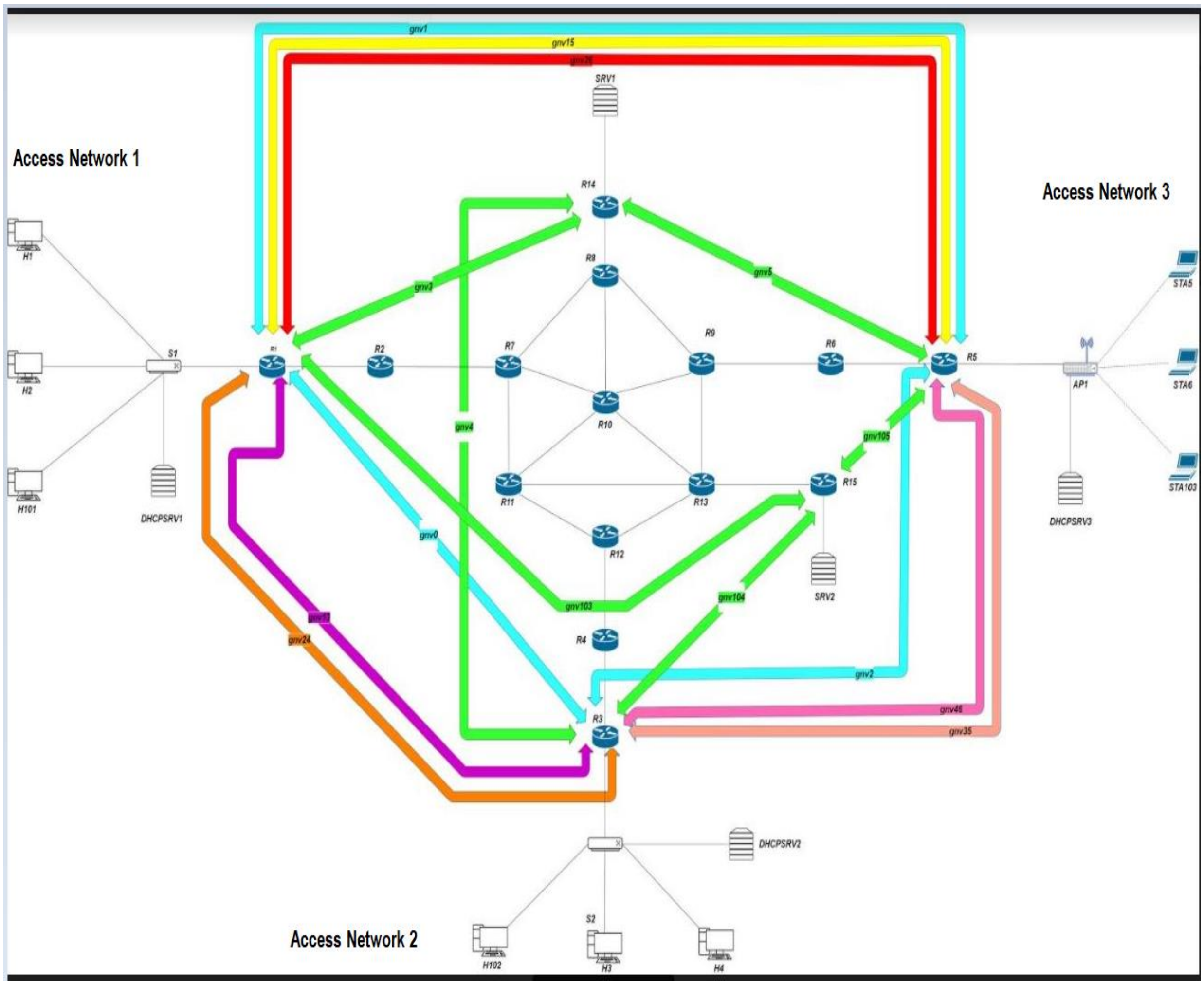
3.3 *Additional packages (Dipanjan Saha)*

The following packages were installed in the Ubuntu platform.

- **Wirehsrak** : This package is installed from Linux apt for network performance analysis.
- **Quagga** : This package is installed to implement Dynamic Routing like OSPF in Ubuntu platform.

4 Implementation

4.1 Network Topology (Anik Biswas)



Network Topology

The topology consists of three Access Networks named Access Network 1,2 and 3. Access Network consist of tenants connected to an OVS switch. Each access Network is connected to the Transport network via Edge Routers. One Primary and one backup VoIP server is hosted in the Transport Network. The Primary VoIP server is connected to Router R14 and the back server to Router R15. As an extension to this project one DHCP sever has been integrated in each Access Network. The server is connected to the OVS switch. As another extension to this project Mininet Wifi has been implemented in Access Network 3. The radio Stations are wirelessly connected to the Access Point in that network. The access point also connects the DHCP Server in the Network.

2.1 *Routing (Anik Biswas)*

Open Shortest Path First (OSPF) routing topology is configured across the Transport and Edge network. The configuration is done in the Linux platform with the help of Quagga Suite. Default Routing is configured in Access Network endpoint Routers to advertise internal Routes.

2.2 Network Slicing (Manash Chakraborty)

Dedicated GENEVE end-end virtual connection is built between Each Access Networks, between Access Networks and VoIP Servers (both Primary and backup), between individual hosts. Network Slices between individual tenants are realized through a combination of end-to-end GENEVE Tunnels and IP Table filtering in terminal routers. 6 Network Slices have been implemented with the above combination. The slices are between the following end points: Host **H1 to H3**, **H2 to H4**, **H1 to H5**, **H2 to H6**, **H3 to H5** and **H4 to H6.** Service and network access separation between each slice is ensured which means H2 cannot access the slice between H1 and H3 and other slices.

2.3 VoIP Service Realization with Docker (Anik Biswas)

As part of one of the core objectives of the project we had to implement VoIP services in Docker and use the same in our network emulation. One of the standalones in built docker image in Containernet has been configured as VoIP server by using Kamailio package. Specific users are created with credentials. Kamailio Database is created in the server module. The configured Docker images is called in our python script. Also, for the tenants to access VoIP service linphone package is installed in one of the Docker images. The same is called in the python script on the tenants and used for end-user VoIP services.

In the service provisioning no specialized configuration has been implemented in the end-users.

3 Extension to Increase Project Functionality

In addition to the existing implementation the following extensions to the original Project is implemented.

3.1 *Mininet WiFi (Manash Chakraborty)*

In Access Network 3 Mininet Wifi has been implemented. Access Point AP1 is configured to accommodate Wifi Functionality for the radio stations located in the access Network.

3.2 DHCP service (Manash Chakraborty)

Local DHCP server is attached in the OVS switch and configured to provide DHCP to tenants in the local network. The DHCP configuration is done by configuring dnsmasq package (editing dnsmasq.conf file) in a standalone docker images and configured with desired dhcp range.

3.3 Enabling Mechanism for Service failover (Manash Chakraborty)

Two Servers are hosted in the transport network one Primary and another backup. We have implemented a mechanism in which user service will automatically failover to backup server or vice versa in case of failure in Primary server. The same is realized with the help of Shell script called in our Mininet python script. The script which is running on the end point routers containing tunnels for VOIP service, monitors for active status of the servers by continuous ping and when one of the server fails the concerning tunnel is forced down and traffic switchovers to backup tunnel.

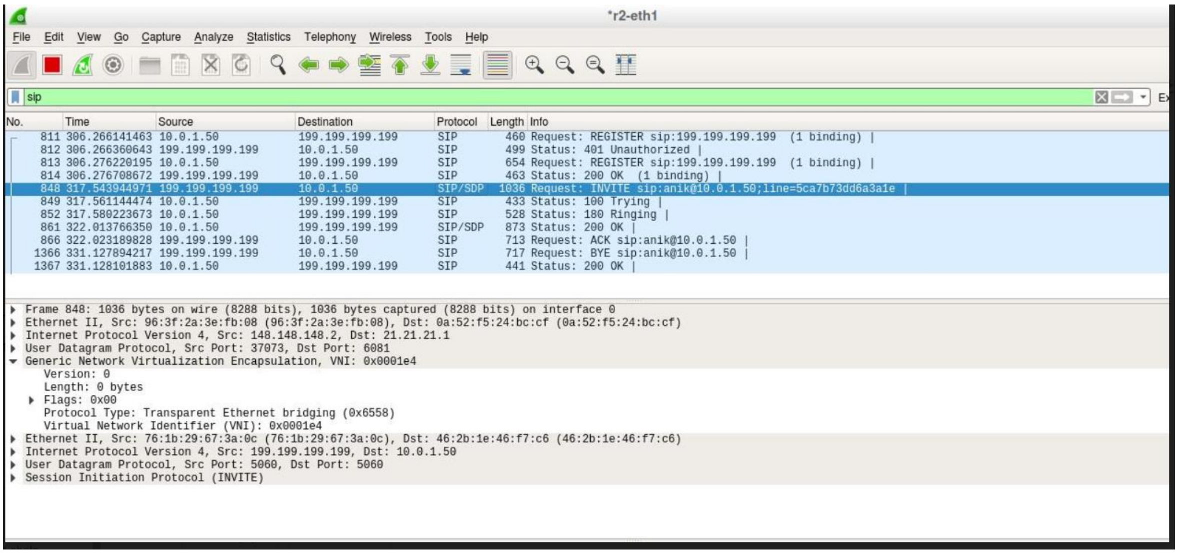
2.1 Traffic Prioritization (Dipanjan Saha)

Traffic is implemented in edge network Routers with the help of iproute2 package which enables traffic control feature in the environment. For traffic prioritization queuing control mechanism has been implemented. Classful qdisc – traffic control mechanism has been implanted which uses Hierarchical Token Bucket algorithm. With this the exit interface of the edge router is configured with classful HTB qdisc. Different classes with different Minimum and Maximum bandwidth capacity is created. Then tc filter is applied to specific source IPs of the tenants (H1,H2 etc) in combination with the created classes. This mechanism allows to allocate priorities to individual src IPs and network slices. If the bandwidth is throttled then it allocates traffic preference with better priority slices.

3 Testing and Results

We performed testing in various components of the network infrastructure. Below is a summary of the same:

Between Tenants and Call Server Networks (Manash Chakraborty)

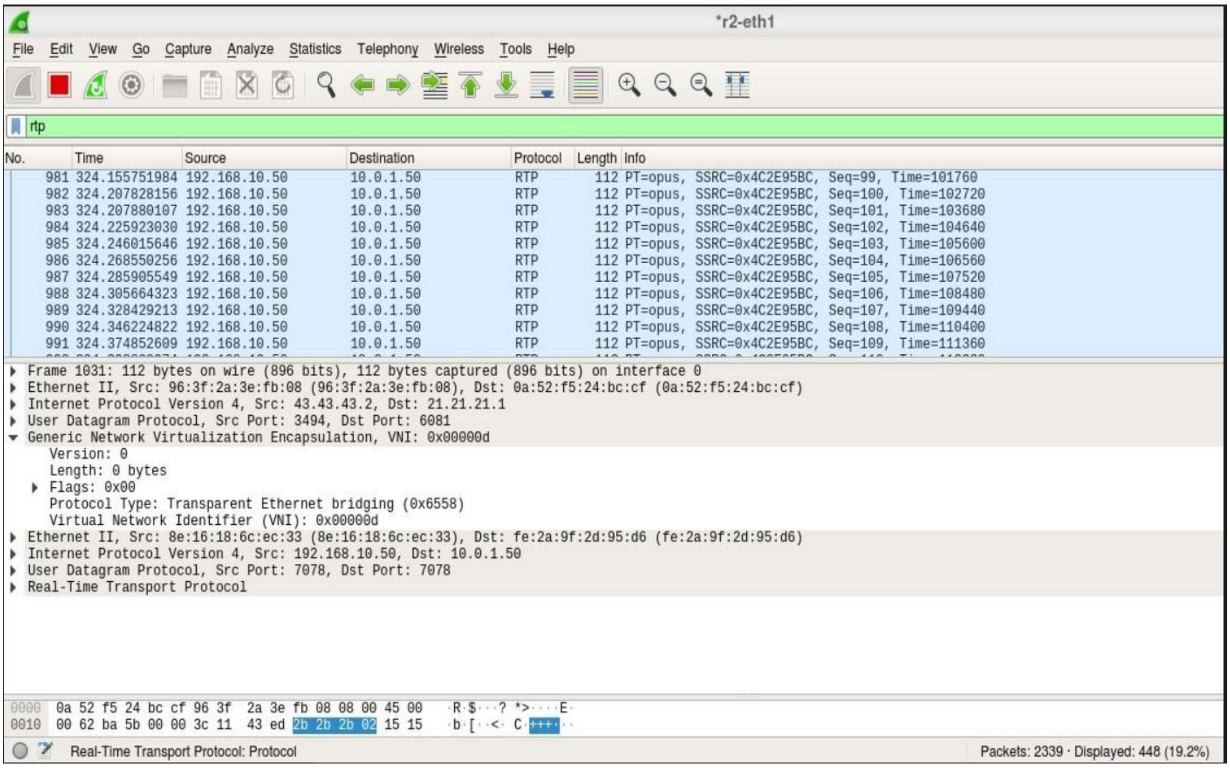


SIP Registration

As per the above diagram Tenant can register to the SIP call server through GENEVE tunnel.

Between Tenants in Access Networks (Anik Biswas)

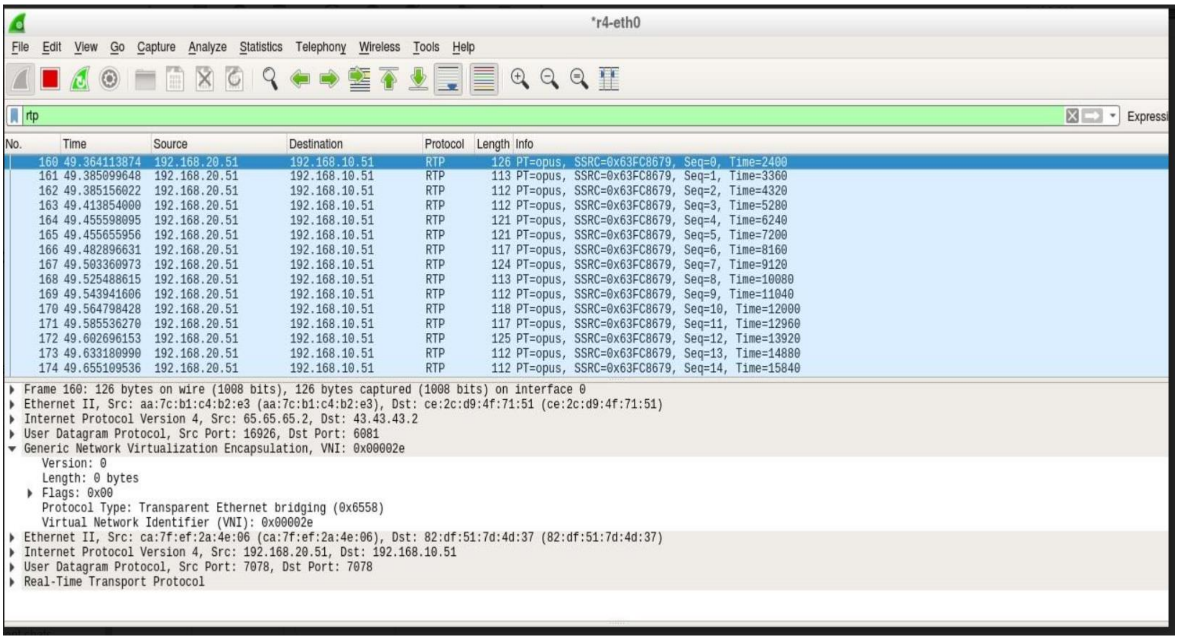
As per the description of network slices before, tenants within the implemented network slices are able to communicate to each other but which are not included in the network are not able to achieve the same.



Between H1 and H2

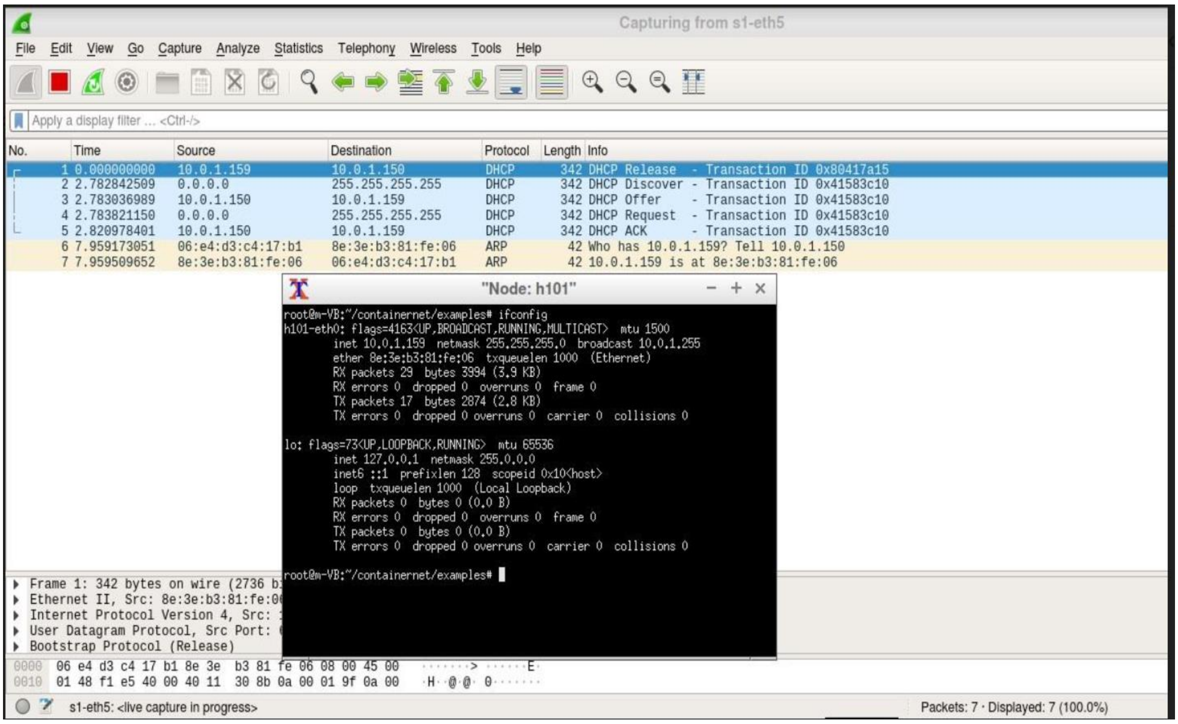
As per the pic above H1 can communicate and call to H3 through GENEVE Tunnel gnv0.

Between Tenants in Access Networks to tenants in WiFi Network (Manash Chakraborty)



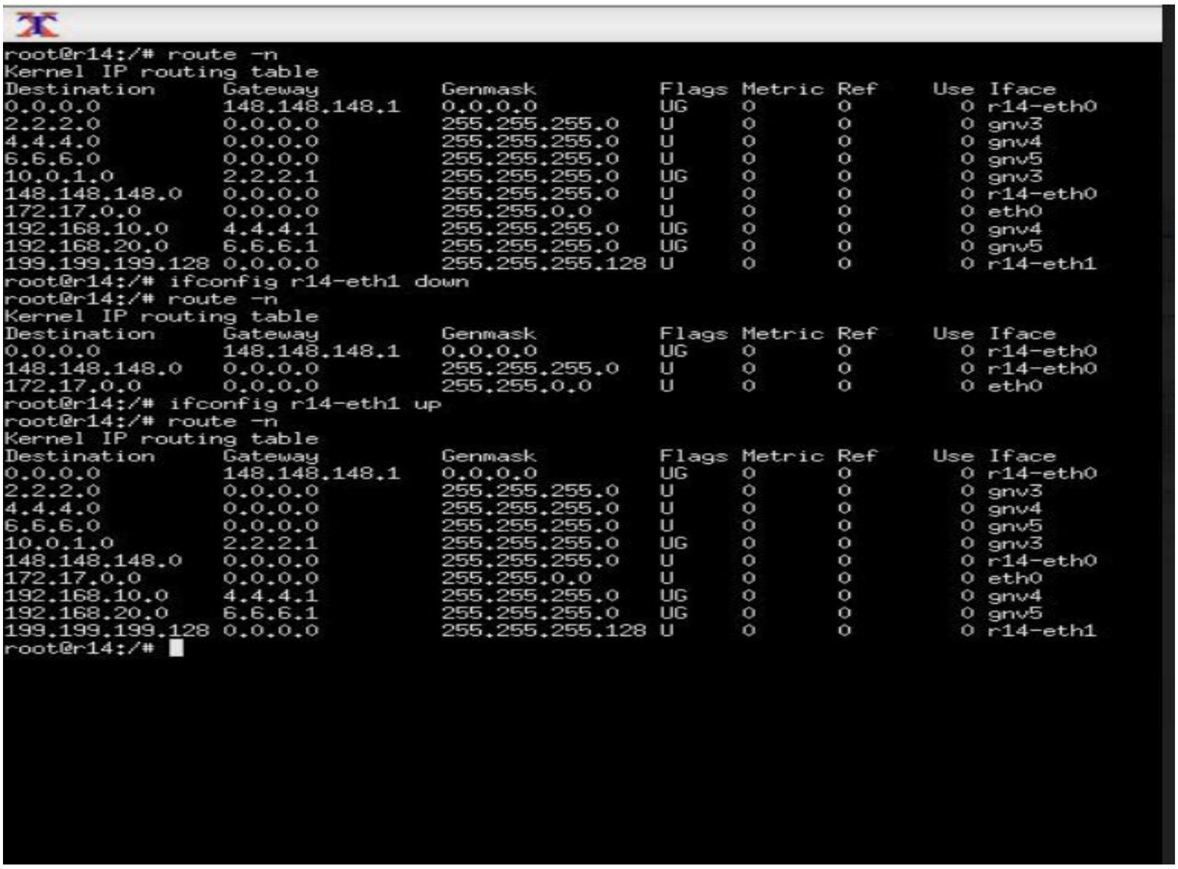
Radiostation Connectivity

DHCP Service validation (Dipanjan Saha)



DHCP service validation

Service Failover (Manash Chakraborty)



Switchover to Backup Server

The above diagram demonstrates the failover from primary to backup server in the event of primary server failure.

4 Challenges

DHCP (Anik Biswas)

- DHCP Server Incompatibility with Docker:
  When we realized DHCP service, any host with Docker image was not able to receive the same.
- If one of the Router has been provided with any docker image then it was not able to forward DHCP request to other hosts in its connected network. For this reason we had to implement DHCP in local network.

Other challenges (Dipanjan Saha)

Our network design had dependency on 1 kamailio server for call establishment. Which created a situation where if the server fails call establishment between all the users will be affected.

Solution: we added another server with same IP address but putting it on to a different less specific subnet mask (Server1 with /25 and server2 with /24). We created separate tunnels from every access network to the secondary server with a higher metric value. On edge routers for the server/datacentre we are running a script which is monitoring server connectivity using ICMP, if any of the server goes down then script will disable the geneve tunnel interfaces on the routers.
A similar script is running on the edge router for the access network which is monitoring the server side geneve tunnel end point using ICMP. When it loses reachability to the tunnel end point it removes the route to the server assigned to that geneve tunnel forcing the traffic on to the backup server link.