Hypervisor Overlay Networking in a Multi-tenant Environment

Team 6

Aneesh Joshi, Luv Khurana, Pranjal Sharma, Sai Jayesh Bondu

Project Description:

SDN-based deployment of Hypervisor Overlay Networks using VXLAN and GRE tunnels to connect customer VMs in a multi-tenant virtual private cloud environment, where customer has control over his subnets.

Background:

<u>Network Virtualization:</u>
Network Virtualization is the technique that creates logical (virtual) networks that are decoupled from the underlying hardware. This decoupling ensures that the network can support virtual environments such as modern data centers, cloud etc. Network Virtualization came into existence to solve problems in data center networking such as:

<u>MAC-table overflow:</u>
In Data Center and Cloud environments, number of entries in a switch's MAC address table can fill up very quickly due to the presence of a really large number of virtual end systems in addition to physical end systems. This causes subsequent frames to be broadcasted in the LAN, thereby wasting bandwidth.

<u>Solution:</u>
Network Virtualization makes use of tunnels to encapsulate original L2 frames. Therefore, the only mac addresses that appear on the physical network are those of tunnel endpoints.

<u>VLAN exhaustion</u>: 4096 VLANs were enough in the past, but with the advent of cloud computing and large data centers, this number is not enough.

<u>Solution</u>: Most Network Virtualization technologies that exist today allow a minimum of 2^{24} tunnels (~16 Million) to be configured, thereby overcoming the VLAN exhaustion problem at a virtual network level.

Current Technologies:

<u>VXLAN (Virtual eXtensible LAN):</u>
In VXLAN, entire frame is encapsulated in a new packet. VXLAN uses UDP (destination port 8472). VXLAN uses a Network Identifier which is 24 bits. Like other tunneling technologies, there are unicast packets exchanged between the two tunnel endpoints.

<u>NVGRE (Network Virtualization using Generic Routing Encapsulation):</u>
In NVGRE, entire frame is encapsulated in a new packet. NVGRE doesn't use TCP/UDP, it uses GRE tunneling protocol. Like VXLAN, NVGRE uses a Network Identifier of 24 bits. Like other tunneling technologies, there are unicast packets exchanged between the two tunnel endpoints. Unlike, VXLAN, NVGRE header contains an optional FlowID field, which can be used to differentiate between flows.

STT (Stateless Transport Tunneling):
In STT, entire frame is encapsulated in a new packet. STT uses TCP, although as the name suggests, it is stateless. TCP is used to make use of functionality in server NICs. Network Identifier for STT is 64 bits. Even with STT, unicast packets are exchanged between the tunnel endpoints.
-

Related work:

The following 3 technologies have been evaluated.

1. VMWare NSX Data Center

2. Nuage VSP

3. Hyper-V network virtualization

All the above-mentioned products are offering network virtualization facilities to the user through control of their own subnets.

The interface provided by the VMWare's NSX is very intuitive, easy to operate and provides high flexibility. It uses VxLAN tunneling. REST APIs have also been used prominently. Apart from this, features such as Load-balancing and Firewall are also present. Another thing to note is that NSX needs hardware termination, while the others use MPLS over GRE.

Nuage's VSP offers excellent cloud-infrastructure management service. A special feature is that customer does not need proprietary hardware for its operation. It is compatible with customer's existing hardware infrastructure. Scaling up the network is very simple.

Hyper-V network virtualization platform uses NVGRE technology. Feature of live migration is provided.
Another interesting feature is compatibility with existing network infrastructure, which provides plug and play operation.

Our system tries to mimic from the current providers. Here we are providing the control of subnets and end to end connectivity across VPCs via tunnels. We offer both GRE and VXLAN tunneling solutions. We also provide features such as cold VM migration and fail-over case by providing back-up bridges in tenant VPCs.
Above solutions also do provide similar features but there is the feature of live VM migration.

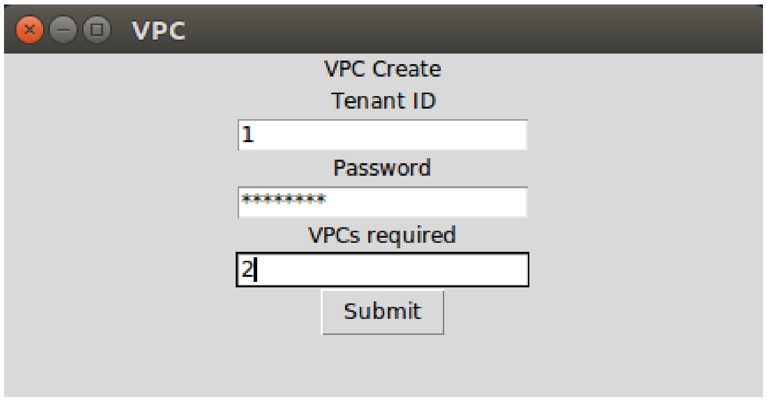
Management Features:

1) Authentication

The tenant is supposed to authenticate himself by providing accurate login details. This ensures that only eligible user gets access to the network database.

2) Configurability

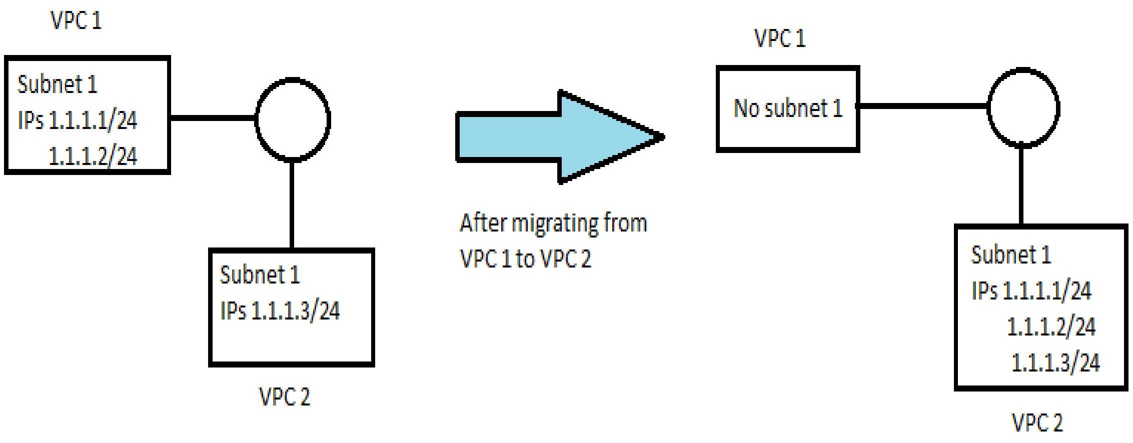
When an overlay network is deployed, the designer must ensure that the overlay can be reconfigured based on the demands of the customer. Reconfiguration can be on the grounds of number of VPCs, number of subnets and number of VMs inside the subnets. Special importance must be given to ease of reconfiguration. APIs can be exposed for the purpose of automation.



Tenant ID: 1
Password: *********
VPCs required: 2

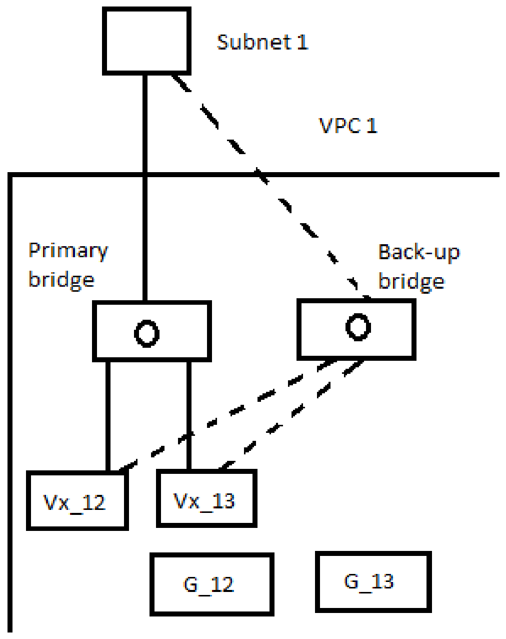
3) Flexibility

To achieve flexibility, we offer on-demand subnet migration capability (cold migration) to the customer. Through this feature the user can control and modify the presence of his subnets in different VPCs. Tunnels are adjusted accordingly.



4) Availability

To provide uninterrupted service, the overlay network should always be available. We intend to provide uninterrupted service through a standby bridge in case the primary bridge goes down. So, we provide back-up bridges and join original tunnel interfaces there.



Progress Status:

Currently, we are achieving the following:

Creation of a router VM to be shared among all tenants.

Upon successful authentication, accepting the Tenant information which includes Tenant ID and Subnet ID and the number of VPCs through a TKinter form.

Creation of a base topology which includes 4 VPCs (Data-centers) connected to 1 managing router namespace. This is done by running<create_vpc> script. Each VPC has a namespace dedicated to it. Veth pairs to attach the namespaces will be created too. GRE devices will be created as part of the base topology. The number of the GRE devices depends on the number of VPCs. Each VPC will have the following number of GRE devices:

[Number of GRE devices = Number of VPCs – 1]
These number of GRE devices must be present in each VPC as these are necessary for connecting private networks over public networks.

Configuration of IP addresses and default routes for the VPCs and router namespace

Subnet creation:
Subnets are created using the <create_subnet> script. Each subnet includes a bridge in the VPC namespace, a different namespace to represent the Tenant's VM, and Veth pairs to connect the customer NS (VM) to its bridge. Per subnet, we will create the following number of VxLAN devices:

[Number of VxLAN devices = Number of VPCs that host the same subnet – 1]

Assignment of IP addresses to hosts, gateways and configuration of default routes.

Addition of routes in the VPC namespaces so that correct GRE data-paths are reached based on the destination.

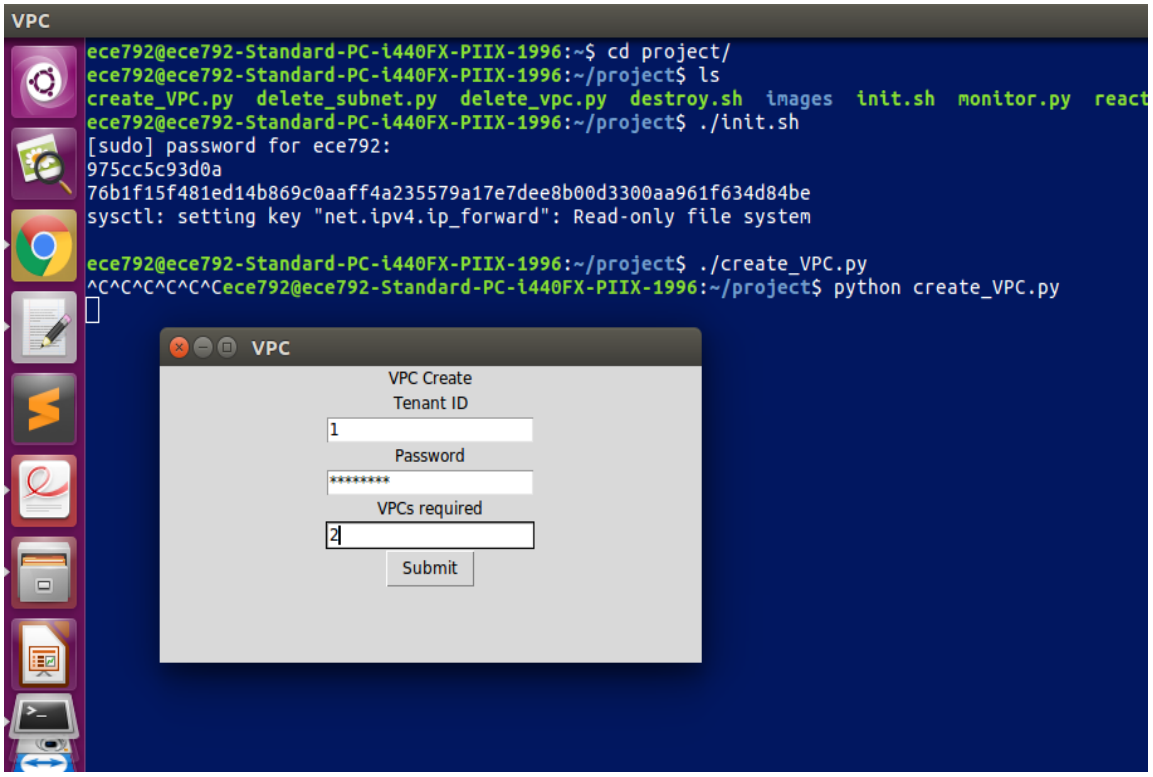
All the information absorbed from the user is maintained in a database (SQLite3). We are maintaining an independent database for each user so that he gets address-space address isolation.

Software Layers:

North-bound interface

A form will be provided to the user. For demo purpose, we have used an interactive form, but user cannot automate through a UI. For that purpose, we intend to expose APIs to the user.
User is prompted with:
- Authentication password
- Number of VPCs
- Subnet IDs

If the user doesn't specify some options, defaults will be used.
Eg. Number of VPCs for that tenant will be chosen as 2 if not specified.



```
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~$ cd project/
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~/project$ ls
create_VPC.py  delete_subnet.py  delete_vpc.py  destroy.sh  images  init.sh  monitor.py  react
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~/project$ ./init.sh
[sudo] password for ece792:
975cc5c93d0a
76b1f15f481ed14b869c0aaff4a235579a17e7dee8b00d3300aa961f634d84be
sysctl: setting key "net.ipv4.ip_forward": Read-only file system

ece792@ece792-Standard-PC-i440FX-PIIX-1996:~/project$ ./create_VPC.py
^C^C^C^C^C^Cece792@ece792-Standard-PC-i440FX-PIIX-1996:~/project$ python create_VPC.py
```

VPC

VPC Create

Tenant ID: 1

Password: ********

VPCs required: 2

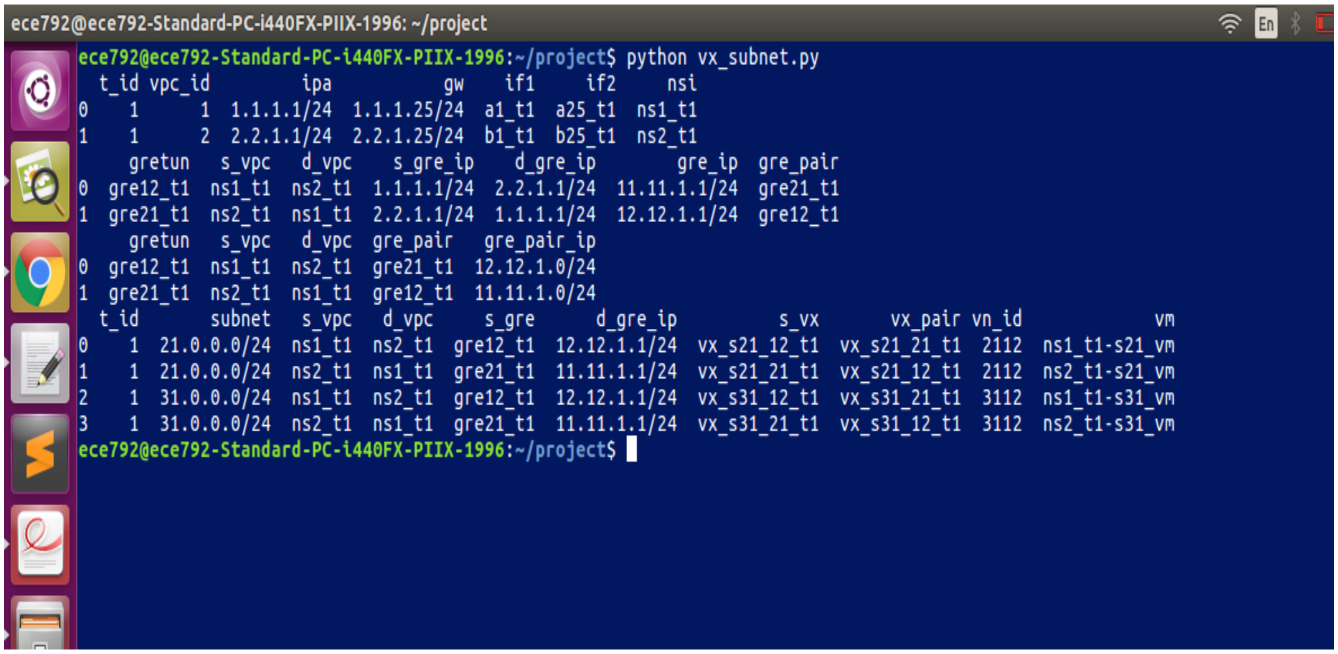
Figure 1: Form displayed to the user

Logic Layer:

When parameters are provided by the user, they are used to generate different parameters associated with their topology. Their database file is maintained as <database_TenantID.db>

The key operation is storing of all the newly-created fields in a database. This is a very important factor as any further operations to be performed on the tenant's subnets or locations, assigned names, etc. will all be accessed through the above maintained table.

The scripts have been developed by employing the CRUD model (Create, Read, Update, Delete). This same model will be employed for operation on VPCs, subnets and tunnels. Default configurations will also be taken care of.



```
ece792@ece792-Standard-PC-i440FX-PIIX-1996: ~/project
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~/project$ python vx_subnet.py
  t_id vpc_id         ipa          gw    if1     if2     nsi
0    1      1  1.1.1.1/24  1.1.1.25/24  a1_t1  a25_t1  ns1_t1
1    1      2  2.2.1.1/24  2.2.1.25/24  b1_t1  b25_t1  ns2_t1
     gretun   s_vpc   d_vpc    s_gre_ip    d_gre_ip        gre_ip  gre_pair
0  gre12_t1  ns1_t1  ns2_t1  1.1.1.1/24  2.2.1.1/24  11.11.1.1/24  gre21_t1
1  gre21_t1  ns2_t1  ns1_t1  2.2.1.1/24  1.1.1.1/24  12.12.1.1/24  gre12_t1
     gretun   s_vpc   d_vpc  gre_pair   gre_pair_ip
0  gre12_t1  ns1_t1  ns2_t1  gre21_t1  12.12.1.0/24
1  gre21_t1  ns2_t1  ns1_t1  gre12_t1  11.11.1.0/24
  t_id       subnet   s_vpc   d_vpc     s_gre       d_gre_ip          s_vx       vx_pair vn_id             vm
0    1  21.0.0.0/24  ns1_t1  ns2_t1  gre12_t1  12.12.1.1/24  vx_s21_12_t1  vx_s21_21_t1  2112  ns1_t1-s21_vm
1    1  21.0.0.0/24  ns2_t1  ns1_t1  gre21_t1  11.11.1.1/24  vx_s21_21_t1  vx_s21_12_t1  2112  ns2_t1-s21_vm
2    1  31.0.0.0/24  ns1_t1  ns2_t1  gre12_t1  12.12.1.1/24  vx_s31_12_t1  vx_s31_21_t1  3112  ns1_t1-s31_vm
3    1  31.0.0.0/24  ns2_t1  ns1_t1  gre21_t1  11.11.1.1/24  vx_s31_21_t1  vx_s31_12_t1  3112  ns2_t1-s31_vm
ece792@ece792-Standard-PC-i440FX-PIIX-1996:~/project$
```

Figure 2: Database table

South-bound interface:

The southbound layer comprises of iproute2 package of linux with python.
The southbound code will help do the infrastructure related tasks such as creation of namespaces, tunnel devices, v-eth pairs and different configurations (Including default setup)

After the setup has been done, and the tunneling interfaces for VXLAN and GRE up, then tunneling functionality can be accessed.

We expect the user to provide parameters regarding the VPCs. But in case, the customer doesn't know, a default topology will be used.

Another thing to note is that, by default, we will provide VxLAN devices for connecting VPCs. If the customer doesn't want to use them, we set the VxLAN devices down. This way of provisioning enables easy toggling of the VxLAN service as per customer's future wishes.

<u>Southbound topology:</u>

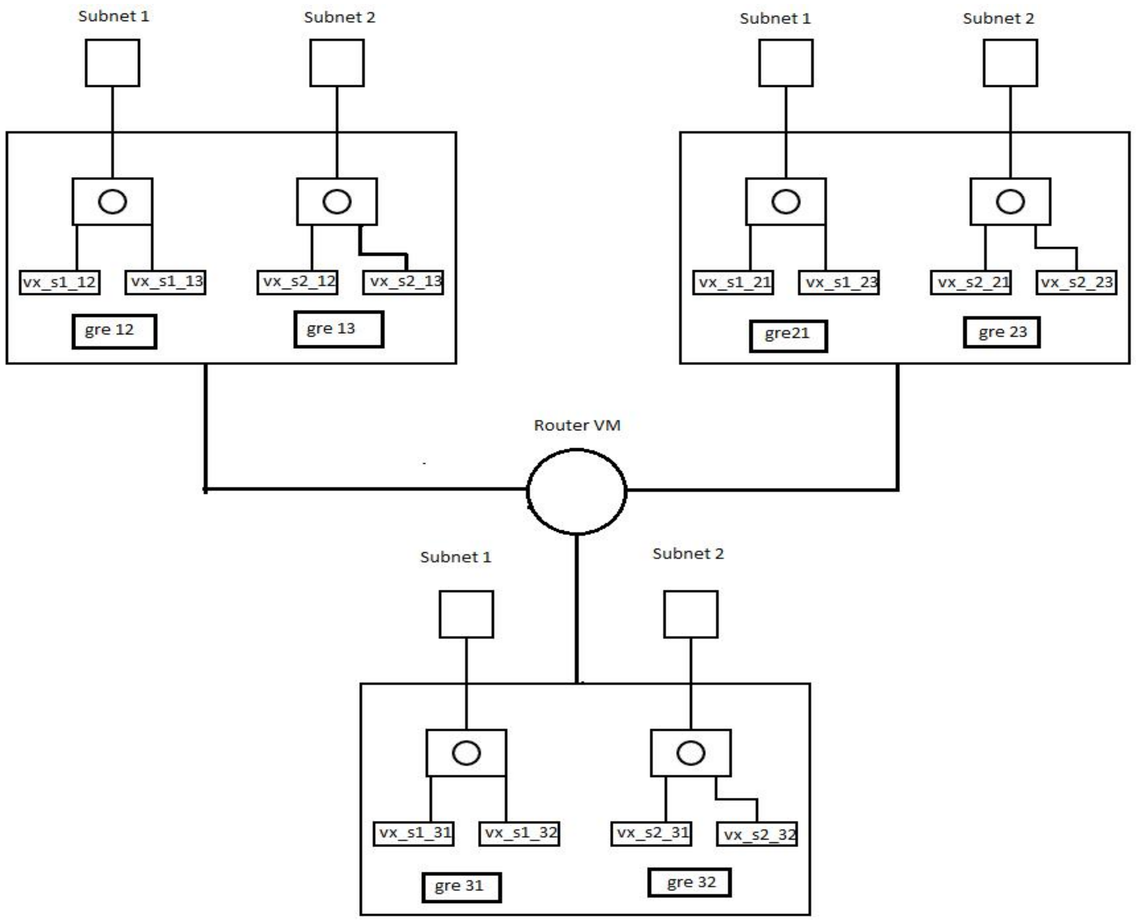


Figure 3: Typical architecture for 1 tenant with multiple subnets in a VPC.

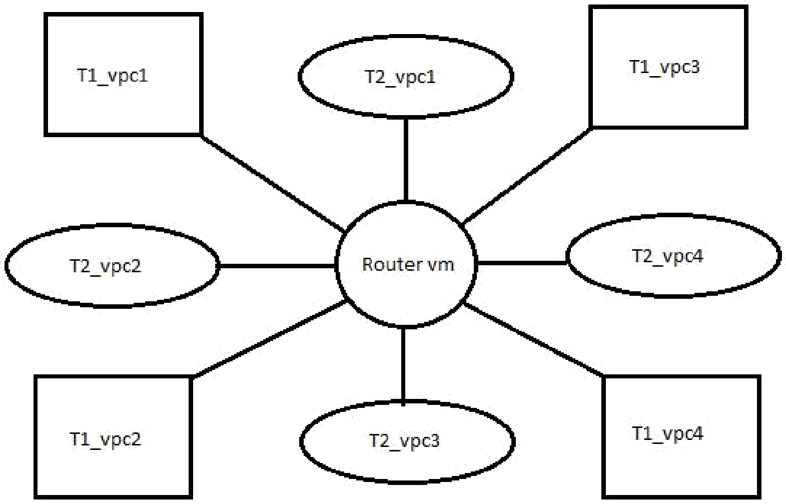


Figure 4: Possible architecture for multiple tenants.