**Práctica Final – RDSV**

* **Luis Alberto López Álvarez y Álvaro de Rojas Maraver**

Para la realización de la práctica final de la asignatura, partimos de lo elaborado en la práctica 4, modificando las redes virtualizadas mediante OSM, que ahora contarán con los siguientes requisitos:

* Soporte de QoS implementado mediante SDN con Ryu.
* Añadir servicios adicionales, como el registro de las MACs de la red residencial.

A continuación, se muestra el escenario que se ha implementado en este trabajo final:

Diagrama

Descripción generada automáticamente

**Captura 1. Escenario trabajo final**

Los **requisitos mínimos** desarrollados establecidos en el enunciado, como modificaciones a la práctica 4 son:

* Sustituir el switch de “KNF:access” por un conmutador controlado por OpenFlow.
* Conectividad IPv4 desde la red residencial hacia Internet. Uso de doble NAT en “KNF:cpe” y en isp1 (igual que en la p4).
* Activar la captura de tráfico ARP mediante “arpwatch” en “KNF:cpe“ (escribe logs cuando se entera de otra dirección MAC).
* Gestión de la calidad de servicio en la red de acceso mediante la API REST de Ryu controlando “KNF:access”.
  + Para limitar el ancho de banda de bajada hacia la red residencial.
* Despliegue para dos redes residenciales.
* Todo automatizado mediante OSM y scripts, utilizando como base los comandos de la práctica 4, añadiendo el on-boarding de NS/VNFs y la instanciación de NS mediante el mismo método.

Para realizar el **despliegue** de nuestro proyecto, se pueden seguir los pasos detallados en el **fichero README.txt adjunto** en el proyecto, mientras que en este documento se mostrarán los cambios realizados para lograr las funcionalidades requeridas.

Para ello, las **tareas básicas** que hemos empezado a realizar que permiten cumplir con estos requisitos se enumeran a continuación, una vez se han entendido los detalles internos del escenario de la práctica 4:

* Sustituir los repositorios por nuestros propios repositorios:
  + El contenedor Docker de las KNFs en DockerHub
    - cuenta educaredes 🡪 cuenta luichu
  + El repositorio Helm en GitHub Pages
    - <https://educaredes.github.io/nfv-lab> 🡪 https://Luislopal.github.io/repo-rdsv
* Automatización del despliegue de la práctica anterior mediante OSM y scripts para montar el escenario de las redes.
* Despliegue de lo anterior para dos redes residenciales.
* Comprobación de que existe conectividad IPv4 desde la red residencial hacia Internet mediante un doble NAT: en KNF:cpe y en isp1.
* Añadir la funcionalidad “arpwatch” para capturar tráfico ARP.
  + Activación de la herramienta arpwatch en KNF:cpe
* Gestión de QoS en la frontera de la red de acceso, mediante la API REST de Ryu en KNF:access (y en brgX) para limitar el ancho de banda de bajada hacia la red residencial.
  + Para la red residencial: 12 Mbps de bajada (y 6 Mbps de subida)
  + Para hX1: 8 Mbps mínimo de bajada (y 4 Mbps mínimo de subida)
  + Para hX2: 4 Mbps máximo de bajada (y 2 Mbps máximo de subida)
  + Independiente de la dirección IP asignada por DHCP a hX1 y hX2

Para poder empezar la práctica es necesario montar una carpeta compartida en VirtualBox, con la ruta local en el ordenador anfitrión y punto de montaje /home/upm/shared en las máquinas virtuales (K8S y OSM), como se detalla en el enunciado de la práctica (en nuestro caso, se ha realizado en un único MAC, que se utilizará como máquina anfitriona). Se va a trabajar sobre dos máquinas virtuales: RDSV-K8S (192.168.56.11), que emula las distintas redes y máquinas del escenario y RDSV-OSM (192.168.56.12), que instala el entorno OSM que coordina las funciones de la red virtual.

Dentro de ambas máquinas es necesario realizar unos scripts que realizan la configuración inicial y otorgan los permisos necesarios al escenario, para poder realizar el despliegue posterior correctamente. Además, dentro de la máquina RDSV-K8S, se realizan los despliegues de los escenarios “home” y “server” utilizando los siguientes comandos dentro del directorio donde se ubica la descarga del proyecto.

*> sudo vnx -f vnx/nfv3\_home\_lxc\_ubuntu64.xml -t*

*> sudo vnx -f vnx/nfv3\_server\_lxc\_ubuntu64.xml -t*

Se ha realizado de igual forma un segundo script, que permite apagar el escenario sin tener que introducir ambos comandos de forma recurrente. Podemos ejecutarlo utilizando el siguiente comando:

*> ./0\_OSM\_configuracion\_off.sh*

Ya desde la máquina virtual RDSV-OSM, es necesario registrar el repositorio Helm que se va a utilizar posteriormente, para poder subir tanto los paquetes VNF como los NS a OSM, lo que se conoce como on-boarding, unido a la instanciación de cada una de las dos instancias que se van a desplegar siguiendo el proceso realizado en la práctica anterior. Todo esto se realiza mediante dos scripts, que se adjuntan a continuación y habrá que ejecutar desde la máquina OSM:

Texto

Descripción generada automáticamente

**Captura 2. Script instalar descriptores**

Texto

Descripción generada automáticamente

**Captura 3. Script instanciación renes**

Posteriormente, es necesario ubicar el clúster de k8s existente en el escenario y guardar su identificador y valor de namespace, como se muestra en la siguiente captura, permitiendo gestionar el clúster y desplegar los pods de los servicios de red, que serán necesarios para realizar la configuración de red necesaria a través de los scripts de renes.

**Texto

Descripción generada automáticamente**

**Captura 4. Exportación variable global namespace**

Por ello, una vez definidas cada una de las variables de los pods, tal como se muestra en la captura siguiente, es necesario lanzar los scripts de rennes que configuran las instancias de red, y se adjuntan de igual forma:

**Texto

Descripción generada automáticamente**

**Captura 5. Configuración instancias de red**

Se puede apreciar en la captura siguiente, que la dirección IP que utiliza el NAT del CPE para dar salida al tráfico es la ’10.100.1.1’ correspondiente con la ‘VCPEPUBIP, que representa el enlace con el switch ExtNet1.

Se utilizan dos direcciones, una para cada interfaz dentro de cada una de las redes que hay en cada switch de acceso. En el caso del AccessNet1 la ‘10.255.0.0/24’, siendo la ‘10.255.0.1’ la interfaz que conecta con la instancia de ‘rennes’ y la ‘10.255.0.2’ la que conecta con los equipos h11 y h12. Con ExtNet1 ocurre igual, pero ahora con la red ‘10.100.1.0/24’, utilizando la IP ’10.100.1.1’ para la conexión con ‘rennes’ y la ’10.100.1.254’ para la conexión con Internet.

Texto

Descripción generada automáticamente

**Captura 6. Fichero de configuración osm\_renes1.sh**

Texto

Descripción generada automáticamente

**Captura 7. Fichero de configuración ‘osm\_renes2.sh’**

En este caso, **h21 y h22 han obtenido las direcciones IP ‘10.250.0.23’ y la ‘10.250.0.22’** respectivamente, valores incluidos en el rango de direcciones posibles a adjudicar por DHCP (y justo los dos siguientes a los dos primeros adjudicados previamente a h11 y h12).

En cuanto al uso de la API REST, se añade el bridge a Ryu y se configura la cola de bajada para que tenga una tasa máxima de 12 Mbps, como se puede apreciar en la captura siguiente. La cola correspondiente a hX1 tiene un mínimo de 8 Mbps de bajada y un máximo de 4 Mbps en el caso de hX2, por lo que, una vez está el escenario funcionando por completo, se han llevado a cabo varias pruebas utilizando el comando iperf entre los equipos hX1, hX2 y “VNF:Access” para comprobar que la configuración se ha llevado a cabo correctamente y se adjuntan las pruebas, de igual forma, a continuación.

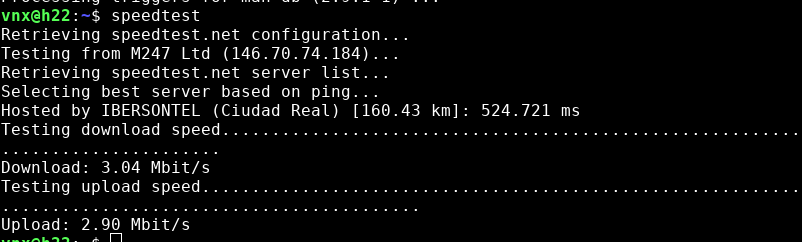
Texto

Descripción generada automáticamente

**Captura 8. Script configuración QoS**

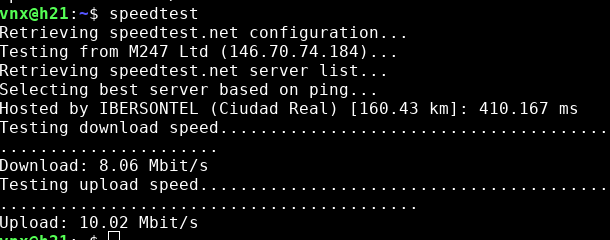
Meter aquí las pruebas de iperf y hablar de ellas un poquito y tal – Al final es speedtest:

Texto

Descripción generada automáticamente

Se puede observar cómo están ambos valores por debajo de los 4 Mbps máximos de bajada determinados mediante la calidad de servicio.

En el segundo caso, se puede ver como la velocidad mínima de bajada para hX1 corresponde con los valores delimitados por la QoS de 8 Mbps como mínimo.





Texto

Descripción generada automáticamente

**Captura 9. Fichero renes\_start.sh (I)**

Texto

Descripción generada automáticamente

**Captura 10. Fichero renes\_start.sh (I)**

De los cambios hechos en renes\_start tb hay que hablar en algún punto de la memoria.