UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DISEÑO Y TRIAL TEST DE UN SISTEMA DE MONITOREO SOBRE EL EVOLVED PACKET CORE VIRTUALIZADO

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

PABLO IGNACIO LOZA VALENZUELA

PROFESOR GUÍA:
CARLOS MARCHANT LIBERONA

MIEMBROS DE LA COMISIÓN:
ALFONSO EHIJO BENBOW
ALBERTO CASTRO ROJAS

SANTIAGO DE CHILE
2019

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL
ELÉCTRICO POR: PABLO IGNACIO LOZA
VALENZUELA FECHA: MARZO 2019 PROF.
GUÍA: CARLOS MARCHANT LIBERONA

DISEÑO Y TRIAL TEST DE UN SISTEMA DE MONITOREO SOBRE EL EVOLVED PACKET CORE VIRTUALIZADO

En la era de la Información, las Tecnologías de Información y Comunicaciones avanzan vertiginosamente. Cada vez los servicios son más personalizados y variados, además con mayores exigencia de calidad, y por ello, las compañías proveedoras de servicios de comunicaciones deben adaptarse de manera rápida y a costos razonables. Es por esto que la tecnología NFV destaca para entregar dinamismo a la Red Móvil, aplicando eficiencia en sus recursos.

En los nuevos estándares en los que trabaja la 3GPP para el 5G, se plantea pasar del paradigma de Funciones de Red ligadas al Hardware por las Funcionciones virtualizadas que están construidas en Software, y así tener sistemas de conexión que puedan escalar de una forma más flexible y ágil, cambiando el enfoque general de cómo se implementan los servicios.

Considerando lo anterior, este trabajo consiste en una guía para la implementación de un Virtual
Evolved Packet Core en un ambiente NFV, utilizando solo herramientas Open Source, como OpenStack para la administración del ambiente NFV, y OpenAirInterface para la implementación de un vEPC y vRAN, además se agregará un Dashboard con Grafana para monitorear las principales KPI del vEPC.

Como resultado queda una guía didáctica para interiorizarse en el mundo de la Virtualización, aplicado a las Redes Móviles. Queda desarrollado como poder hacer pruebas sobre el ambiente NFV para así probar características como la agilidad y escalabilidad del Software. Se concluye en este trabajo, la importancia de las nuevas habilidades que deben adquirir los Proveedores de Comunicaciones en el ámbito de la Virtualización, aunque siendo compatibles sus conocimientos sobre Redes Móviles tradicionales, deben agregar este nuevo paradigma de cambio del Hardware hacia el Software.

Por tanto queda esta guía práctica para introducirse en el mundo de las NFV, desde el punto de vista de la Telefonía móvil, con los pasos a seguir para la construcción y monitoreo de un Virtual Evolved Packet Core.

SUMMARY OF THE MEMORY TO OPT THE
TITLE OF ELECTRICAL CIVIL ENGINEER
BY: PABLO IGNACIO LOZA VALENZUELA
DATE: MARCH 2019 GUIDE: CARLOS
MARCHANT LIBERONA

Design and Trial Test for a Monitoring System of a virtualized Evolved Packet Core

In the Information Age, Information and Communication Technologies are advancing vertiginously. Each time the services are more personalized and varied, in addition to higher quality requirements, and therefore, the companies providing communications services must adapt quickly and at reasonable costs. That is why NFV technology stands out to deliver dynamism to the Mobile Network, applying efficiency in its resources.

In the new standards in which 3GPP works for the 5G, it is proposed to move from the paradigm of Network Functions linked to the Hardware by the Virtualized Functions that are built in Software, and thus have connection systems that can scale in a more flexible way and agile, changing the general focus of how services are implemented.

Considering the above, this work consists of a guide for the implementation of a Virtual Evolved Packet Core in an NFV environment, using only Open Source tools, such as OpenStack for the administration of the NFV environment, and OpenAirInterface for the implementation of a vEPC and vRAN, In addition, a Dashboard with Grafana will be added to monitor the main KPIs of the vEPC.

As a result there is a didactic guide to internalize in the world of Virtualization, applied to Mobile Networks. It is developed as being able to test the NFV environment in order to test characteristics such as the agility and scalability of the Software. We conclude in this work the importance of the new skills that Communications Providers must acquire in the field of Virtualization, although their knowledge of traditional Mobile Networks is compatible, they must add this new paradigm of hardware change to the Software.

Therefore this practical guide is left to enter the world of NFV, from the point of view of mobile telephony, with the steps to follow for the construction and monitoring of a Virtual Evolved Packet Core.

Dedicado a mi familia, gracias por creer en mi educación, los amo.

Agradecimientos

Quiero partir agradeciendo a mi familia, quienes incondicionalmente han estado a mi lado, apoyándome y creyendo más en mi que yo mismo. A mi Mamita Querida Santa Madre, que día a día nos ama y cuida con sus formas especiales; sin tu infinito cariño no lo habría logrado. A mi Viejo, que te admiro profundamente, apostaste todo en lo que creías, como fue nosotros y nuestra educación. A mi Hermano grande, creo que eres una persona única y gigante a su modo, constantemente me sorprendes, sé que a pesar todo siempre seremos tú y yo contra el mundo. Gracias familia por ayudarme a ser mejor persona y más humilde.

También agradezco a mi Profesor Guía Carlos Marchant, por su dedicación y apoyo, estando atento ante cualquier problema que surgiera. Y al sabio Ángel Sessarego, quien estuvo preocupado en que el trabajo avanzara, superando todo tipo de trabas, y obvio… nunca faltó tu consejo de vida. A Claro Chile por darme la oportunidad de trabajar con ustedes.

A la natación que me ha dado tanto, entrar al agua es y será un disfrute que me desconecta de todo, algo muy necesario en lugares como Beauchef. A mis amigxs de la rama y Cal y Canto, que me hicieron sentir que pasaba por una Universidad diversa y rica diversas personas, algo poco visto en las divididas facultades. Gracias por todos los buenos momentos.

Al DK y Los Tres Negros, mis mejores anécdotas universitarias están con ustedes. Atesoro las risas a llorar, tallas nuestras, y un humor negro hasta en los momentos más oscuros, santo remedio. Simio no mata a simio, pero rata mata a rata.

Gracias por tanto, perdón por tan poco.

Tabla de contenido

Lista de Abreviaturas

2G	Second Generation	NFVI	NFV Infraestructure
3GPP	Third Generation Partership Project	NIC	Network Interface Card
4G	Fourth Generation	NSD	Network Service Descriptor
5G	Fifth Generation	OAI	Open Air Interface
AMPQ	Advanced Message Queue Protocol	OPEX	Operating Expense
API	Application Programming Interface	OS	Operating System
APN	Access Point Name	OVS	Open Virtual Switch
ARP	Allocation and Retention Priority	OS-MA-NFVO	NFV Orchestrator - Management and Orchestration
AS	Access Nostrum	P GW	PDN Gateway
AUC	Authentication Center	PCEF	Police Control Enforcement Function
CAPEX	Capital Expediture	PCRF	Police Charging Rules Function
CN	Core Network	PDCP	Packet Data Coverage Protocol
CoS	Class of Service	PDN	Packet Data Network
COTS	Commercial off the Shelf	PLMN	Public Land Mobile Network
CPU	Central Process Unit	PNF	Physical Network Function
DAS	Direct Attached Storage	POD	Plain Old Data
DL	Down Link	PS	Packet Switching

DNS	Domain Name System	QCI	QoS Class Identifier
DPI	Deep Packet Inspection	QoS	Qualty of Service
E-UTRAN	Evolved UTRAN	RAN	Radio Access Network
E2E	end to end	RAT	Radio Access Technology
EMS	Element Management System	RLC	Radio Link Control
eNB	Evolved Node B	RoI	Return of Investment
EPC	Evolved Packet Core	RRM	Radio Resources Management
EPS	Evolved Packet System	S GW	Signaling Gateway
ETSI	European Telecommunication Standar Institute	SAE	System Architecture Evolution
FTP	File Transfer Protocol	SAN	Storage Area Network
GBR	Guaranteed Bits Rate	SCTP	Stream Control Transmission Protocol
GPRS	General Packet Radio Service	SDF	Service Data Flow
GTP	GPRS Tunnel Protocol	SDN	Software Defined Network
GUI	Grapfic User Interface	SFC	Service Function Chaining
HPLMN	Home PLMN	SGSN	Serving GPRS Support Node
HSS	Home Suscriber Server	SLA	Service Level Agreement
I/O	Input Output	SSH	Secure Shell
IDS	Inspection Data System	TA	Tracking Area

IEEE	Institute of Electrical and Electronics Engineers	TCP	Transmission Control Protocol
IP	Internet Protocol	TFT	Traffic Flow Template
IPv4	IP version 4	TO	Telecommunication Operator
IPv6	IP version 6	UDP	User Datagram Protocol
IT	Information Technology	UE	User Equipment
KPI	Key Performance Indicator	UL	Up Link
LTE	Long Term Evolution	UMTS	Universal Mobile Telecommunication System
M2M	Machine to Machine	UTRAN	UMTS Terrestrial Radio Access Network
MAC	Medium Access Control	VIM	Virtualized Infraestructure Manager
MANO	NFV Management and Orchestration	VM	Virtual Machine
MME	Mobility Managment Entity	VNC	Virtual Network Computing
MS	Mobile Station	VNF	Virtual Network Function
N-PoP	Network Point of Presence	VNFM	VNF Manager
NAS	Non Access Stratum	VoIP	Voice over IP
NAT	Network Address Translator	VPC	Virtual Private Cloud
NFV	Network Function Virtualization	VPLMN	Visited PLMN

Capítulo I: Introducción

Motivación

La nueva generación de telecomunicaciones, además de tener objetivos de mejorar la experiencia de servicio del usuario y la más alta disponibilidad posible, si no que también se busca mantener una sustentabilidad multidimensional, en el ámbito económico, ambiental y operativo. Esta tarea no es trivial dada la cantidad de demanda creciente en usuarios (incluyendo el emergente IoT que trae todas las conexiones de las personas con las máquinas, y entre las mismas máquinas, como es M2M) y bit rate para los próximos años[1]. Además, se debe tener en consideración el cambio dinámico que existe con respecto a la carga que debe llevar la red, por ejemplo, al momento de emisión de un partido de fútbol, en donde una gran cantidad de usuarios accede a medios de video de manera súbita, fuera de los usos habituales. Otro ejemplo sería a miras del 5G con implementaciones como la Telemedicina, la cual requiere calidades y confianzas altamente exigentes. Estos problemas actualmente no se ajustan tener una visión de red estática, que al no poder adaptarse dinámicamente se termina aumentando los costos y tiempos de despliegue de manera significativa. Por ende, constantemente se ha buscado una red mucho más flexible que la actual, con recursos dinámicos cuando estos sean requeridos.

Por lo que, en este trabajo se busca una implementación real de unos de los elementos claves que existen en la virtualización para poseer una red más flexible y sostenible en el tiempo. Además, considerando el fuerte empuje que el último tiempo ha tenido el software Open Source, se diseñó un trabajo en base a herramientas que fueran abiertas al público, las cuales han estado muy activas en la validación de prototipos, pruebas de conceptos y, finalmente en la punta de la línea de avance de las tecnologías de telecomunicaciones móviles.

[1] Grijpink, F., Ménard, A., Sigurdsson, H. and Vucevic, N. (2018). *The road to 5G: The inevitable growth of infrastructure cost*. [online] McKinsey & Company. Available at: https://www.mckinsey.com/industries/telecommunications/our-insights/the-road-to-5g-the-inevitable-growth-of-infrastructure-cost .

Objetivo General

El objetivo de este trabajo es diseñar y probar una interfaz gráfica para monitoreo, que entregue el estado del funcionamiento con los índices claves sobre un vEPC a implementar en un ambiente NFV.

Objetivos Específicos

1. Estudiar el Estado del Arte de la virtualización del EPC en un ambiente NFV, con enfoque en la función del EPC.
2. Diseñar e implementar una nube privada en OpenStack capaz de generar un ambiente virtualizado para una prueba E2E de LTE.
3. Implementar un vEPC con OAI en un ambiente NFV.
4. Realizar una prueba de funcionamiento de LTE, para revisar el desempeño del vEPC.
5. Diseñar la interfaz gráfica OS-MA-NFVO para monitorear el vEPC.

Alcances

1. Realizar un estudio general de EPC y su contexto en LTE, considerando cada una de las funciones de red que desempeña cada nodo lógico que lo compone.
2. Implementar un vEPC, que sea suficiente para realizar pruebas de paquetes de control del estándar que se utiliza para 4G-LTE.
3. Diseñar una interfaz gráfica adaptada para el operador que entregue la descripción de una función del vEPC, con las características de KPIs en una base de datos histórica.
4. Realizar un test de la interfaz de monitoreo de la NFV con la emulación de pruebas para el vEPC.

Capítulo II: Antecedentes

4G-Long Term Evolution

La RAT o tecnología de acceso por radio 4G-LTE está diseñada con el objetivo de solucionar las creciente demanda de los usuarios de datos en telefonía celular por mayor velocidad de conexión y una mejor calidad de servicio, también por el lado de los operadores de telecomunicaciones, bajar gastos operacionales y disminuir la inversión, buscando en lo posible que la red tenga una menor complejidad de operación y reúna la banda de frecuencia operativa para la comunicación por radio, evitando la fragmentación en distintas tecnologías.
Para lograr esto, 4G incluye el sistema de acceso de las redes móviles a la Red mediante el sistema Evolved Packet System (EPS), que se alejó del modelo de conmutación circuitos para emigrar a un sistema totalmente basado en conmutación de paquetes (Packet-switched PS), esto con el sentido de llegar sin interrupciones entre el equipo del usuario (User Equipment UE) y la Red de Paquetes de Datos PDN durante la movilidad.
Dentro de EPS podemos diferenciar dos ramas, la primera es Long Term Evolution (LTE) que incluye mejoramiento y re-estructuración del Universal Mobile Telecommunication System (UMTS) (acceso por radio de los celulares a las antenas de radio) al Evolved UTRAN (E-UTRAN). Mientras la segunda rama incluye todo el resto de los aspectos que no tengan que ver en la conexión por radio, que se engloban en el System Architecture Evolution (SAE). Veremos que el corazón del SAE consiste en el Evolved Packet Core (EPC) el cual tiene los componentes para el control y flujo de esta comunicación hasta la Red del Operador.

Arquitectura General

La red PLMN (Public Land Mobile Network) en esta generación soporta conexión IP (acceso a la Red del operador y servicios de Internet) hasta el UE, siempre y cuando este abonado para este servicio, además esto lo hace en distintos niveles de calidad de servicio (QoS) ligados a diferentes servicios, por ejemplo es distinta la conexión necesaria para una llamada que para una descarga FTP. Se desarrolló una arquitectura general para lograr lo anteriormente mencionado, en la Ilustración 1 se muestran los distintos elementos de la EPS que desempeñan diferentes funciones para que el UE pueda conectarse a la Red IP, además en los conectores se explicita cuales son los protocolos o interfaces que utilizan para comunicarse entre ellos.

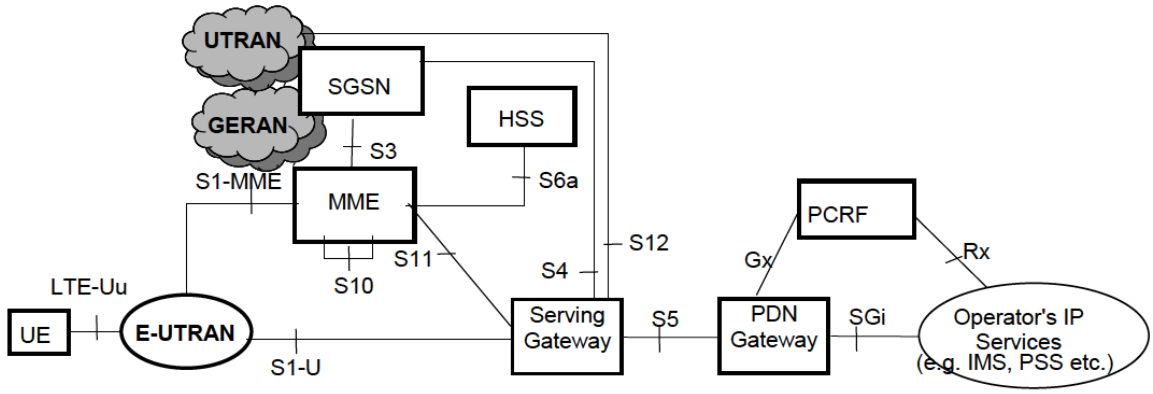


Ilustración 1: Elementos de la EPS[2]

Evolved Packet Core

El EPC llamado así en la System Arquitecture Evolution (SAE) proyecto que definió este elemento de la red, es el principal responsable de controlar a los UE y establecer a los portadores que mantienen la conectividad de datos hasta el usuario móvil. Se definió por la 3GPP en su Release 8 como una Arquitectura Plana con el objetivo de manejar los paquetes de manera eficiente en su performance y mantener controlados los costos asociados a la operación. También se separaron por plano los datos, por una parte está la señalización, la cual tiene relación con todo lo necesario para poder transmitir la información, pero que no es información. Por otro lado está el plano de usuario, la cual es toda la información que se quiere transmitir. Esto se realiza para el mejor uso del operario al dimensionar e implementar de manera mejor guiada.

Nodos Lógicos del EPC

Los nodos lógicos fundamentales para el buen funcionamiento consisten en el MME, P-Gateway y S–Gateway, aparte de estos hay otros nodos entre los más relevantes los que cumplen la funcionalidad de mantener un banco de datos centralizado de los usuarios y un inspector de los portadores que según la aplicación da los permisos al usuario requiere.

Police Charging Rules Function (PCRF)

El PCRF es el "inspector" encargado de velar que se cumplan las normas de control del tráfico de datos, supervisando que correspondan las Calidades de Servicio (QoS) a la categoría de datos requerida, priorizando los portadores que requieran mayores estándares de tráfico, inclusive en emergencias sólo permitiendo sesiones de esta índole, rechazando otros requerimientos.
Asimismo, este nodo lógico debe controlar la función, que reside en P-Gateway, asociado al cargo monetario del usuario basado en el flujo de datos requeridos por el mismo, conocido como Police Control Enforcement Function (PCEF).

[2] Ilustraciones tomadas de 3GPP TS 23.401, p.19-20 (2018).

Home Suscriber Server (HSS)

La base de datos HSS es la que contiene la información correspondiente a todos los suscriptores de un operador de red, tales como la Calidad de Servicio QoS inscrita para el EPS del usuario y sus restricciones de Roaming. También indica a los P-Gateway cuales usuarios UE tienen permiso de acceso, en otras palabras, desde que lugares físicos tiene acceso a servicios de internet un UE específico, esto lo hace en forma de etiqueta de acuerdo a la convención del DNS llamado Access Point Name (APN), o mediante indicación de cual dirección IP del P-Gateway se le asignará.
Lo anterior consiste en información estable que se espera no cambie siempre, pero el HSS también contiene información dinámica como es el MME al cual el usuario fue relacionado al momento de iniciar una sesión, y además sostiene los datos de los vectores de autentificación y la llave de seguridad que se generan en el Authentication Center (AUC).

Packet Data Network Gateway (P-Gateway)

El PDN Gateway o P-Gateway es un elemento crítico para el funcionamiento del EPC porque es la puerta de comunicación entre el EPC y la Packet Data Network (PDN) del operador (Internet, o servicios de operador como el IMS). Un mismo UE puede acceder a uno o varios PDN-GW. Las funciones que opera este nodo están suscritas a la información que le entrega el HSS, entre las que se encuentra la asignación de la dirección IP del UE, igualmente de aplicar las normas de QoS que le exige y los cargos asociados al flujo de datos. También se encarga de tomar los paquetes del Downlink del usuario (desde la PDN al EPC) e irlos clasificando en su portador correspondiente, dependiendo de los diferentes QoS que el paquete pide, esto se hace basado en plantillas Traffic Flow Templates (TFT). P-GW realiza la aplicación del QoS para garantizar el ancho de banda del portador mínimo o guaranteed bits rate (GBR) bearers. Este nodo también sirve como anclaje en para la interconexión entre redes 3GPP y non-3GPP (e.g. WiMax, CDMA2000, etc...).

Mobility Managment Entity (MME)

El MME es el nodo de control que procesa la señalización entre el UE y la Core Network (CN) en el protocolo Non Access Stratum (NAS). Entre las varias funciones (ver TS 24.401) está el establecimiento y mantenimiento de los portadores, además de establecimiento de la conexión y su seguridad entre la PDN y el UE, todo esto es manejado en la capa de administración de movilidad del protocolo NAS.

Serving Gateway (S-Gateway)

El S-Gateway es la puerta de enlace que termina en la interfaz que une el EPC y E-UTRAN, todos los paquetes IP que llegan al UE son entregados desde el EPC por el S-GW pasando por el comunicador de radio E-UTRAN, a cada UE se le asigna un único S-GW mientras se le asocie al EPS. Este es utilizado como anclaje de referencia durante la movilidad, esto es mientras el UE puede estar moviéndose entre estaciones bases (eNodeB). Además, el S-GW es usado para mantener la información de los portadores asignados a ese UE, en los intertantos que este se

encuentra en estado IDLE[3], en misma línea, también deja esperando en buffer los datos de bajada (Downlink) mientras el MME ubica la celda del UE (paging) para restablecer los portadores.
Además, la función S-GW en redes visitadas puede contar con funciones administrativas como controlar el volumen de datos para su posterior cobro, pudiendo hacer además intercepciones legales. Entre otras varias funciones (ver TS 23.401) sirve como anclaje de movilidad para la interconexión con otras redes tipo 3GPP del 3G o 2G.

[3] El IDLE mode o modo inactivo, el móvil no tiene conexión activa a la red, y cualquier transmisión de datos requerirá un establecimiento (o restablecimiento) de una conexión de control, para comenzar a transmitir datos. En el caso específico de 4G, mantiene actividades de soporte como Recepción Discontinua (DRX), System Information para acceso, re-selección de celda, e información de búsqueda.

Network Function Virtualization

Introducción

En la actualidad, las redes están sobrepobladas de middle-boxes, que consisten en aplicaciones propietarias sobre hardware propietario, como puede ser un Firewall o NAT. Por tanto, para lanzar un nuevo servicio a ser puesto en producción en la red, es necesario desplegar una gran variedad de estas aplicaciones y acomodarlas según las capacidades del hardware que indica el valor en particular. Esto a su vez, manteniendo esta dinámica, cada día se vuelve trabajo cada vez más difícil y con plazos de implementación no acorde a los requerimientos de hoy. Un ejemplo de esta rigidez en la forma de hacer las cosas, es la evolución del protocolo IPv4 al IPv6, el cual ha demorado más de 10 años en su desarrollo, aún cuando una enorme cantidad de equipos en la actualidad sigue trabajando en IPv4 dado la dificultad de poder actualizar este protocolo al funcionar sobre hardware de propietario. Adicionalmente, debido al incremento del portafolio de servicios que entrega la red, ésta se debe estar escalando en su capacidad de infraestructura periódicamente, lo cual también implica lentitud y además gastos de CaPex y OpEx.

Por tanto, el foco en los últimos años de todas las entidades bajo el esfuerzo de OpenStack y la ETSI ha sido el siguiente:

- En primer lugar, separar el software propietario del hardware propietario, éste último reemplazarlo por equipamiento de red Commercial-Off-The-Shelf (COTS), que puede satisfacer las necesidades de uso general en lugar de fines personalizados, proporcionando también muchas más capacidades con un costo menor que los equipos de red especializados (por ejemplo, hardware basado en x86)
- En segundo lugar tomar las funciones de red (software), es decir, tomar este software y montarlo sobre un hardware de uso común (no-propietario) con el fin de comprtir estos recursos de hardware entre las distintas funciones. De esta forma, se consigue Virtualizar las funciones de red.

De esta forma, la mayoría de los Operadores de Telecomunicaciones (TO) esperan separar las funciones de red de los dispositivos diseñados especialmente e implementarlos como software que podría implementarse en hardware estándar COTS. En esta línea, varias compañías grandes de telecomunicaciones del mundo y ETSI están estandarizando el concepto de Network Function Virtualization (NFV).
NFV transforma la forma en que los Operadores de Telecomunicaciones planifican la red utilizando tecnologías de virtualización de IT (Information Technology) estándar, es decir, consolidando varios tipos de equipos de red propietarios en equipos de gran volumen basados en COTS (ver Ilustración 8). Basándose en el desarrollo actual de las tecnologías de virtualización, la aparición de NFV posibilita que la mayoría de los operadores logren una gran flexibilidad de red y un nuevo y

rápido ciclo de implementación de servicios. De esta forma, los Operadores de Telecomunicaciones puede satisfacer los requisitos de los clientes en continuo crecimiento y reducir al mismo tiempo la operación de la red y el costo de mantenimiento. Sin embargo, esta nuevas oportunidades estarán acompañados de ciertos desafíos relevantes de resolver, por ejemplo, la flexibilidad requerida en la red se logra introduciendo el plano de virtualización, que pudiera generar problemas de seguridad y escalabilidad al no estar la tecnología lo suficientemente madura, y agregando una capa más a los sistemas que antes no contemplaba, por tanto dando un cierto grado mayor de dificultad desde su diseño hasta la operación.

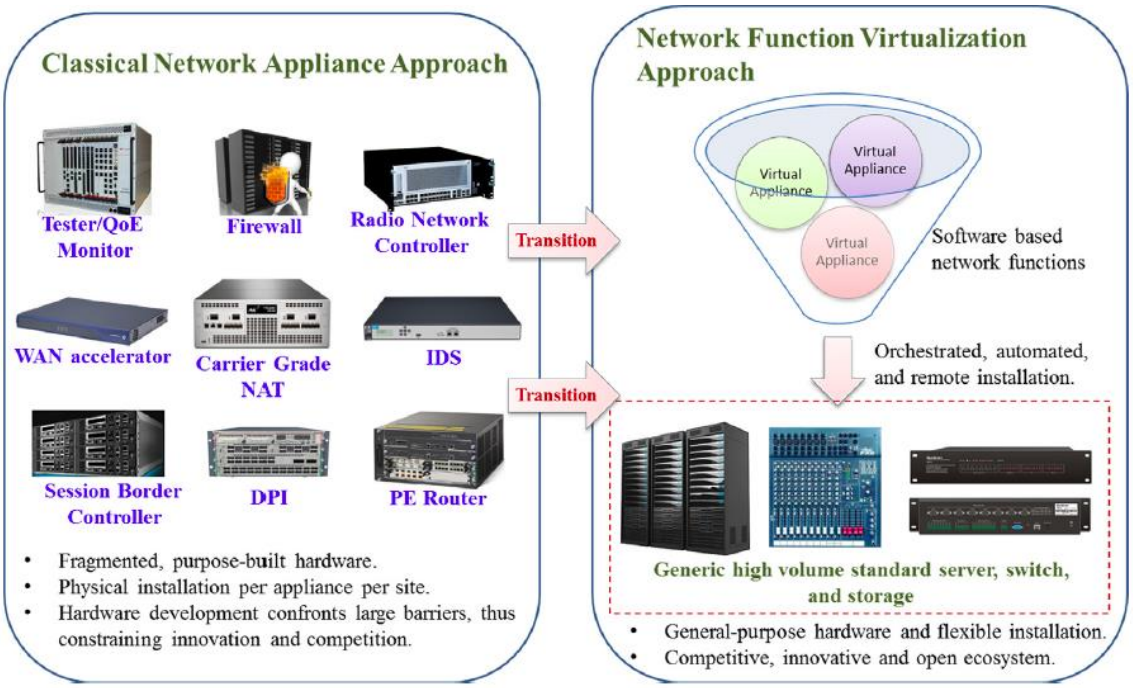


Ilustración 8: Visión de la aplicación clásica de una red a una virtualizada[4]

Modelo NFV ETSI

A continuación, se indican una serie de conceptos basado en el modelo ETSI para el ambiente NFV.

1. Función de red física (PNF): una PNF es la implementación de un bloque de funciones de red especializadas, con comportamiento e interfaces externas bien definidas. Hoy, un PNF se refiere a un nodo de red o un dispositivo físico, ya que está estrechamente relacionado con el cumplimiento de un fin específico. Esta funcionalidad se mantiene constante en el tiempo, debido a que está ligado estrechamente al Hardware estático, donde en la mayoría de los casos sus cambios consiste en aumentar sus capacidades.

[4] Ilustración tomada de Yi, B. *"A comprehensive survey of Network Function Virtualization" p.* 223-224 (2018)

2. Infraestructura de virtualización de funciones de red (NFVI): NFVI proporciona un entorno de red compuesto por componentes de hardware y software, en el que se pueden implementar, gestionar y ejecutar VNF. Un NFVI puede atravesar múltiples lugares geográficos, mientras que las conexiones entre estos diferentes lugares geográficos también se consideran parte de este NFVI.
3. Sistema de gestión de elementos (EMS): un EMS es un conjunto de Gestores de elementos (EM) individuales que se encargan de gestionar las instancias de VNF en términos de creación de instancias, ejecución y despliegue durante sus ciclos de vida.
4. Gestión y orquestación (MANO): NFV introduce algunas capacidades nuevas en la red de comunicación, mientras que MANO es el elemento utilizado para gestionar y acomodar estas nuevas capacidades. En particular, MANO se puede dividir en tres entidades, es decir, Virtualized Infrastructure Manager (VIM), VNF Manager (VNFM) y NFV Orchestrator (NFVO), que son responsables de la gestión de NFVI, la asignación de recursos, la virtualización de funciones, etc...
5. Función de red virtual (VNF): Es la implementación de software de PNF, que debe proporcionar los mismos comportamientos funcionales e interfaces de operación externa que una función física (PNF). Una VNF puede estar compuesta por uno o más elementos. Por un lado, si se implementa una VNF en una sola Máquina Virtual (VM), se compone de un solo componente. Por otro lado, si se implementa un VNF en varias VM, se compone de múltiples componentes, donde cada VM aloja un componente. Tomando el EMS como ejemplo, en realidad es una VNF que consta de muchos componentes individuales (es decir, EM) que se distribuyen en diferentes máquinas virtuales.
6. Punto de presencia de red (N-PoP): N-PoP indica la ubicación donde se implementan la PNF y la VNF. Se puede acceder a los recursos correspondientes, como memoria y almacenamiento desde N-PoPs.

A continuación en la Ilustración 9, y para una mejor visualización de la relación entre estos conceptos, se muestran en azul y diferenciados en una estructura de capas, donde SFC significa Service Function Chaining Working Group.

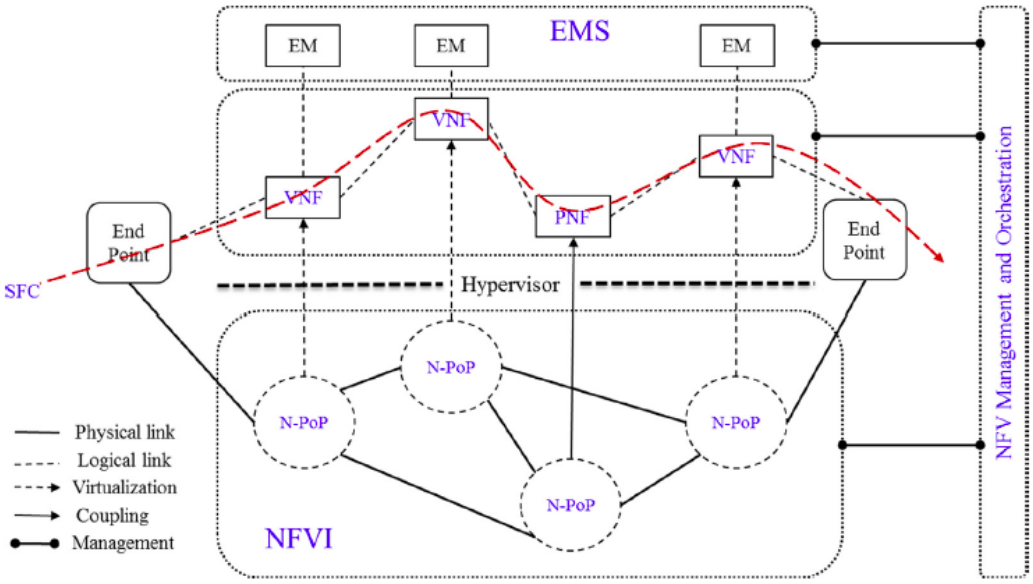


Ilustración 9: Relación en la terminología NFV con una visión de servicio end to end[5]

Arquitectura General

La arquitectura general de referencia entregada por la ETSI, se puede apreciar en la Ilustración 10. En particular, la NFVI (Hardware/Infraestructura) corresponde al plano de datos, que reenvía datos y proporciona recursos para ejecutar servicios de red. MANO (Orquestación y Administración) corresponde al plano de control, que es responsable de construir las conexiones entre varias VNF y orquestar recursos en NFVI. La capa VNF corresponde al plano de la aplicación (Software), que alberga varios tipos de VNF que pueden considerarse aplicaciones.

[5] Ilustración tomada de Yi, B. *"A comprehensive survey of Network Function Virtualization"*, p.223-224 (2018).

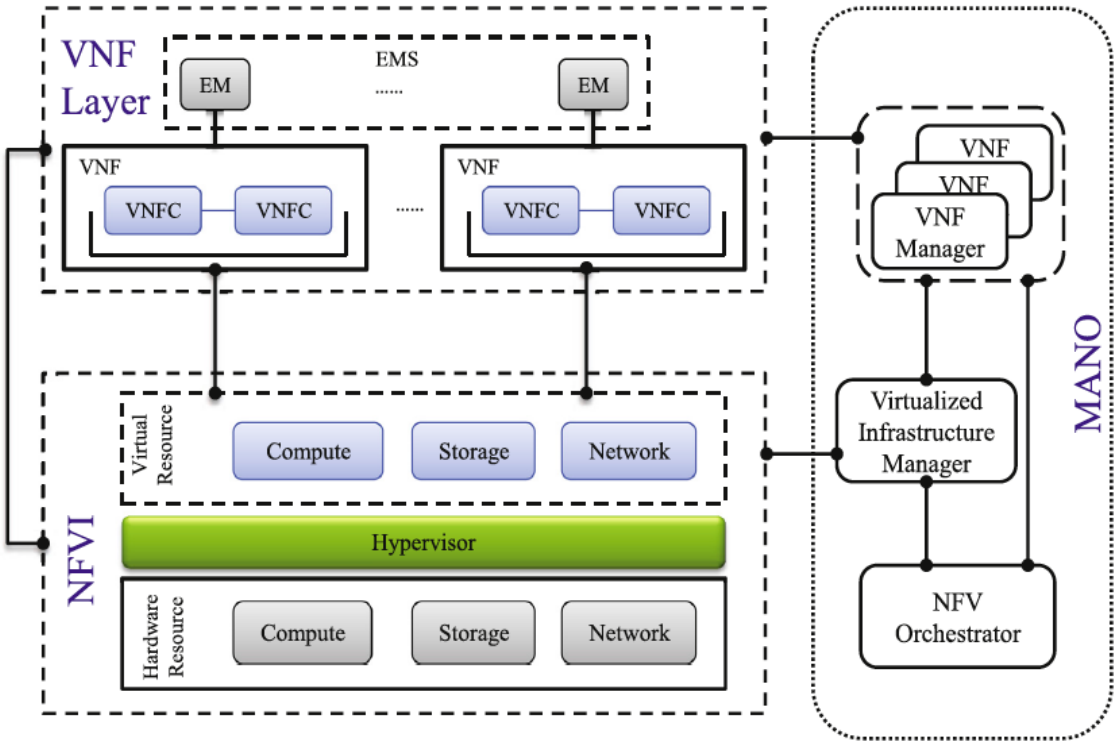


Ilustración 10: Arquitectura general de NFV[6]

Capa NFVI

La NFVI su principal función es, estando debajo de la capa de virtualización y sobre el Hardware, toma los recursos físicos como el cómputo, almacenamiento y tarjeta de red, y provee con estos los recursos virtualizados de los mismos al resto del ambiente NFV. Al implementar un conjunto de dispositivos de red de propósito general en ubicaciones distribuidas, la NFVI puede satisfacer varios requisitos de servicio, como latencia y localidad, y reducir el costo de la red en CaPex y OpEx, esto dado que se desliga del Hardware. Basado en el hardware de uso general, la NFVI también proporciona un entorno de virtualización para la implementación y ejecución de las VNF. Aunque las arquitecturas de las NFVI actuales son generalmente las mismas entre ellas, sus implementaciones reales pueden diferir mucho. De acuerdo con la parte inferior de la Ilustración 10, la arquitectura de referencia de NFVI se divide en tres capas, es decir, infraestructura física, capa de virtualización e infraestructura virtual. Cada uno de ellos se presenta con sus funciones y características en las siguientes subsecciones.

Capa Física

La infraestructura física de NFVI es básicamente servidores de propósito general, los cuales proveen de cómputo y storage. En particular, los servidores que entregan computo, se les llama Nodo de Cómputo, como los dedicados a la memoria, se les

[6] Ilustración tomada de Yi, B. *"A comprehensive survey of Network Function Virtualization"*, p.223-224 (2018).

llama Nodo de Almacenamiento. Entre ellos se comunican entre si a través de tarjetas de red con interfaces físicas.
El rendimiento de procesamiento de NFVI se puede mejorar utilizando los mecanismos de mejora de CPU, Storage y Networking, como por ejemplo, el primero se puede cumplir conectando los aceleradores de hardware en los servidores estándar COTS para acelerar la velocidad de procesamiento de paquetes o admitiendo una gran página para reducir el tiempo de búsqueda. En las tajetas de red se puede optimizar utilizando Network Interface Card (NIC) inteligentes para promediar la carga, o agregando otros coprocesadores (por ejemplo, FPGA) para acelerar el procesamiento de datos. También existen mecanismo de aceleración del procesamiento con computos en cola, o mecanismo de Hardware Offload.

CPU Hardware

En el contexto NFV, cada nodo de cálculo se puede realizar en forma de un procesador de mono-núcleo o multi-núcleo (Single-Core o Multi-Core). Actualmente, existe una gran diversidad de servidores que se pueden usar como nodos de cómputo de propósito general. De acuerdo a sus características se pueden dividir en cuatro tipos:

- Tower Server: se refiere a una computadora independiente que está construida en un gabinete vertical conocido como la torre. Generalmente, los servidores de la torre se construyen con un cierto grado de robustez considerado para reducir el tiempo de inactividad del servicio y evitar posibles daños. Sin embargo, debido al gran volumen y peso del servidor de la torre, el espacio del piso puede ser una gran limitación para la expansión de NFVI. Además, dado que los servidores de torre son realmente independientes entre sí, cada uno de ellos necesitará un sistema auxiliar completo que incluya un sistema de enfriamiento individual, un monitor, dispositivos de I/O (por ejemplo, un teclado), etc.
- Rack Server: En comparación con el servidor de la torre, una torre solo contiene un servidor, un rack puede contener múltiples servidores apilados uno sobre el otro, lo que no solo reduce el espacio requerido, sino que también consolida los recursos de la red.
- Blade Server: Los servidores blade normalmente se colocan dentro de un receptáculo de cuchillas para formar un sistema de cuchillas que cumple con los estándares IEEE[7] de unidades de bastidor. En comparación con los servidores de torre y rack, el servidor blade permite más potencia de procesamiento en menos espacio de rack, ya que comparte ciertos elementos de hardware entre los servidores blade dentro del mismo gabinete.
- Hyper-converged Solution: Consolida los recursos informáticos, de almacenamiento y de red en un solo cuadro, logrando así una gran escalabilidad simplemente agregando o eliminando dinámicamente dichos cuadros. Aunque este mecanismo ofrece muchos beneficios, como disponibilidad, seguridad y respaldo, reduce la flexibilidad del despliegue, la configuración, la escala y la mejora de la red debido a las características de alta convergencia, es decir se limíta el dinamísmo a lo que los cuadros

[7] Electronic industries association, Define: EIA-310, [Online] https://www.server-racks.com/eia-310.html.

puedan formar en sus distintas combinaciones, que a su vez pueden causar la ineficiente utilización del hardware afectando rendimiento.

Storage Hardware

Se considera Hardware de Storage, a los dispositivos capaces de guardar, de manera temporal y/o permanente, información. Muchos elementos de red funcionan en torno al Hardware de Almacenamiento, por ejemplo el Video Streaming[8], donde la red debe ser capaz de mantener un caché (almacenamiento temporal) del video. En particular, estos dispositivos de almacenamiento se usan generalmente en los siguientes tres aspectos:

- Direct Attached Storage (DAS): Indica el almacenamiento conectado de los servidores a través de una ruta de comunicación directa y el servidor directamente conectado solo puede acceder a dicho almacenamiento.
- Network Attached Storage (NAS): Indica un dispositivo de almacenamiento que proporciona acceso a archivos a computadoras heterogéneas en la red, es decir, el archivo se comparte entre estas computadoras.
- Storage Area Network (SAN): Similar a NAS, SAN también proporciona acceso a almacenamiento de datos compartidos. La diferencia es que SAN comparte los datos en la unidad de bloque en oposición a la unidad de archivo de NAS.

Networking Hardware

El formato de estos equipos es usualmente de los dispositivos de capa dos y tres del Modelo OSI (Capa de Enlace y Red), estos son switches o routers, , aunque gradualmente están siendo remplazados por equipos que soporten solamente protocolos de ruteo estándar, o solo el protocolo OpenFlow[9], incluso ambos.

Capa Virtualizada

Consiste en una capa de Software que administra al Hypervisor, el cual distribuye los recursos físicos de la capa inferior para asignarlos a unidades aisladas (por ejemplo, VM o contenedores), aunque estas unidades virtuales compartan la infraestructura, cada una tiene asignado los periféricos tanto físicos, como virtuales necesarios para sostenerse de manera independiente.

Durante el ciclo de vida de las funciones de red virtualizadas, hay varias rutinas que deben ser cubiertas por la capa virtualizada, como la activación de una VM, la eliminación, la migración en línea y el escalamiento dinámico de las mismas. Para esto, el Hypervisor puede ajustar de forma dinámica la asignación entre los recursos físicos y los recursos virtuales asignados a las máquinas virtuales, de modo que se pueda lograr una portabilidad de los recursos a alto nivel entre las distintas VMs.

---

[8] Servicio de red que entrega material multimedia de video de manera online, como por ejemplo Netflix.

[9] Según el documento “OpenFlow: Enabling Innovation in Campus Networks” de 2008 en la Universidad de Stanford. Se define como un protocolo emergente y abierto de comunicaciones que permite a un servidor de software determinar el camino de reenvío de paquetes que debería seguir en una red de switches. Con el protocolo OpenFlow, una red puede ser gestionada como un todo, no como un número de dispositivos que gestionar individualmente, es el propio servidor el que dice a los switches dónde deben enviar los paquetes. Con esta tecnología, las decisiones que impliquen el movimiento de paquetes están centralizadas, por lo que la red puede ser programada independientemente de los switches.

En la actualidad los hipervisores utilizados en SDN, son similares a los utilizados en NFV, como es el caso de FlowVisor. Pero hay que tener en cuenta que el hipervisor en SDN reside entre el plano de datos y el plano de red, mientras que en NFV, reside entre la infraestructura física y el acceso virtual infraestructura.
Los Hipervisores de dividen en dos tipos según en sobre que se ejecuta su software. Los Hipervisores de Tipo 1, también conocidos como *nativos* o de *baremetal*, son los que ejecutan su código directamente sobre el hardware en el que son instalados. Por contraparte, los de Tipo 2, o también conocidos como *hosted*, son los que entre su código de Software y el Hardware se instala un Sistema Operativo (llamado Host OS), por tanto el Hipervisor funciona sobre un Sistema Operativo, lo cual vendría siendo una capa extra de Software.
Actualmente los Hypervisor mayormente utilizados en NFV, son Linux KVM, Citrix Xen, Microsoft Hyper-V y VMware ESXi, los cuales son detallados a continuación:

- KVM: Es un software open source, muy utilizado en sistemas operativos basados en Linux. Utilioza sistemas operativos que se ejecutan dentro la VM, llamados Guest OS. KVM es considerado un Hypervisor de tipo 2, ya que trabaja sobre un OS. Se utiliza en aplicaciones sensibles al tiempo.
- Xen: También consiste en código de fuente abierta, el cual permite ejecutar multiples GuestOS en un mismo equipo físico. A diferencia de KVM, que se ejecuta sobre un HostOS, Xen funciona directamente sobre la capa física, por tanto, es clasificado como un Hypervisor tipo1. Xen soporta para-virtualización (una forma liviana de virtualizar, que requiere un kernel y drives habilitados para esto) y full-virtualización.
- Hyper-V: Software comercial que está integrado en los servers Windows de Microsoft que ofrece virtualización *carrier.grade* (nivel de operadores de red) para empresas con data center o clouds. Además, vSwitches basados en Hyper-V tienen la capacidad de contenedores aislados (contenedores dentro de una VM, se hablará más adelante de estos) y traffic shaping.
- ESXi: Hypervisor tipo 1 comercial, el cual ofrece virtualización para nivel empresarial para que consoliden aplicaciones en menos hardware. Aunque ofrece alta disponibilidad y tolerancia a fallas, es mucho más costoso que los anteriores[10], dependiendo del caso. Las características anteriores de ESXi se logran maximizando el I/O throughput con la menor cantidad de ciclos de computo (uso de CPU), aunque esto se puede variar si es utilizado en aplicaciones especiales que sean sensibles a la latencia y jitter.

Además de la tecnología del Hypervisor, está la opción de utilizar Contenedores. Las diferencias se pueden apreciar en la ilustración 11, donde se puede ver que el contenedor no requiere separar el sistema operativo en GuestOS, esto puede ahorrar overhead de cómputo en comparación con un Hypervisor, ya que las aplicaciones corren directamente sobre el HostOS. Esta misma ventaja lleva implícito varios riesgos de seguridad, debido a la dependencia del HostOS.

[10] Análisis de costos por casos en: VMware, vsphere, [online]. Disponible en: https://www.vmware.com/products/vsphere .

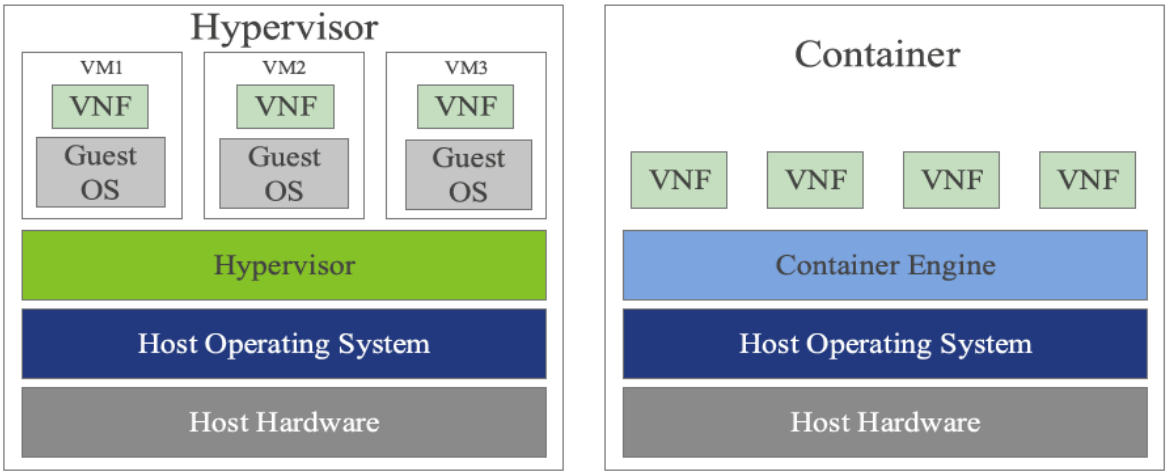


Ilustración 11: Comparación entre Hypervisor y Contenedor[11]

Capa de Infraestructura Virtual

Como se puede apreciar en la Ilustración 10, la capa de infraestructura virtual abastecida de los recursos del Hardware, los convierte en elementos virtualizados (Virtual CPU, Storage y Networking) a través de Software como un Hypervisor, este nuevo tipo de recurso se explica a continuación.

Virtual CPU

El computo virtual, o vCPU proviene de la virtualización de elemento de cómputo en el hardware, como el CPU. Ésta es usualmente generada por el Hypervisor a través de APIs, que entrega una cierta capacidad que puede ser utilizado en el ambiente de las VNFs. También se puede dar como en SDC (Software Defined Compute), el cual pasa la capacidad de cómputo en una cloud, que puede ser compartido on-demand por una interfaz central.

Virtual Storage

La virtualización del almacenamiento separa su administración del hardware. Se constituye en forma de DAS, SAN y NAS (explicado anteriormente en el capítulo Storage Hardware). Esto permite la creación de depósitos de memoria compartibles, flexibles y escalables. También esto trae características como el snapshot y backup[12]. Software Defined Storage (SDS) es otra forma de virtualizar almacenamiento, la cual crea memorias virtuales, y luego las conecta, de tal manera que parezca una sola unidad de storage virtualizado.

Virtual Networking

Las redes virtuales son similares a las redes tradicionales, pero estas permiten la interconexión de VMs, servidores virtuales, entre otro elementos, manteniéndose en un mismo ambiente virtualizado. Aunque se sigan los principios de redes físicas, las funcionalidades son controladas desde el software, por ejemplo, con

[11] Ilustración tomada de Yi, B. *"A comprehensive survey of Network Function Virtualization"*, p.223-224 (2018).

[12] Mientras un Backup consiste en un respaldo de una base de datos, el Snapshot es un set de referencias respaldadas de una base de datos en ciertos periodos de tiempo. Ver más en: https://searchdatabackup.techtarget.com/definition/storage-snapshot .

adaptadores Ethernet virtuales o Switches virtuales. Otro aspecto importante a tener en cuenta, es que con elementos de red virtualizados, se puede tener una red de máquinas virtuales sin la necesidad de Hardware adicional dedicado a las redes, usando los mismos protocolos.

Capa de Funciones de Red Virtualizadas VNF

Las Virtual Network Function (VNF), son la abstracción final de las Physical Network Function (PNF), realizando la misma función, pero como software sobre un hardware compartido. Como se puede apreciar en la ilustración 10, esta capa puede estar compuesta por varias VNF aisladas, cada una con sus respectivas componentes controladoras (VNF Controller VNFC). Los Element Manager (EM) son las entidades administradores de la VNF, los cuales arman el Element Management System (EMS).
La gran diversidad de VNFs hace que se puedan encontrar en toda la arquitectura de la NFV, como por ejemplo los vRouter puede estar en la capa de infraestructura virtual (NFVI), otra VNF como OpenDaylight que es un software encargado de controlar, puede ser considerado dentro de la capa de Administración y Orquestación (MANO).

Vendors	Product/Solution	Description
Infoblox	virtual secure Domain Name System (DNS)	An expansion solution of the original DNS, which reduces the business and operation risk during the network transition to NFV and SDN.
NEC	NFV C-RAN	A cloud based RAN with automate L2/L3 functions adding and removing for NFV to meet the traffic demand in NFV.
NFWare	virtual carrier grade NAT	Centralized network addresses translator.
Oracle	IMS Session Delivery	An agile IMS solution for delivering consumer VoIP and VoLTE services.
Red hat	Linux-Atomic host	A lightweight platform support running applications in Linux containers.
6wind	Virtual Accelerator	Accelerating packet processing for virtual network infrastructure in NFV.
	Turbo IPsec	A software Virtual Private Network (VPN) appliance deployed on COTS servers in bare mental environment with the same functionality of the legacy IPsec VPN gateways.
Cisco	Virtual Port Channel	Allowing physical links between two devices to appear as a single port channel to a third device.
	Virtual Managed Services	A cloud native solution for delivering new software defined WAN services to business customers.
Ericsson	Virtual Router	A carrier-grade software system which offers network operators the ability to deploy services with high agility and performance.
	vEPC	Virtualized Evolved Packet Core networks.
Juniper	vMX series edge router	Revolutionary carrier-grade virtual routing for enterprises and service provider networks.
	vSRX integrated virtual firewall	A virtual firewall designed for enterprises and service providers to achieve capabilities of firewall, security and automation in a virtual machine.
Nominum	N2	A virtual communication platform depending on browsers.
	Vantio CacheServe	A DNS solution which integrates the N2 platform.
F5	Policy Enforcement Manager	Optimizing and monetizing networks with context-aware policy enforcement.
	Local Traffic Manager	Application delivery with programmable infrastructure in a reliable, secure and optimized way.
	Virtual Edition	Deploying software-defined application service in hybrid, virtual and cloud environments.
Metaswitch	Perimeta Session Border Controller	Carrier-grade virtualized network function which supports large scale communication security.
	Clearwater	IP Multimedia Subsystem in cloud computing.
	Voice over LTE	A VoLTE solution built from the ground up using cloud native service methodologies.
Brocade	virtual Mobile Analytics	Brocade network visibility platform for end-to-end mobile networks.
	SteelApp Traffic Manager	A leading virtual application delivery platform for virtual environments or cloud.
	Vyatta 5600 vRouter	Networking industry leader in virtual router with high performance and scalability.
	virtual Application Delivery Controller	A software based VNF solution for fast, reliable application delivery across the virtual and cloud platforms at massive scale.

Tabla 2: Ejemplos de VNFs[13]

En la Tabla 2 se aprecia un conjunto representativo de distintos tipos de VNF. Actualmente las VNF se implementan de dos maneras distintas, como ambiente de

[13] Tabla tomada de Yi, B. *"A comprehensive survey of Network Function Virtualization"* , p.223-224 (2018).

Virtual Machine (VM) o como Contenedor (Container). Mientras el primero ofrece un ambiente aislado de la completitud de una computadora real, el contenedor se limita ejecutar los procesos esenciales, y tener las capas de computación necesarias para la VNF. Bajo estas dos visiones, se puede visualizar las siguientes posibilidades de implementación en la ilustración 12. Existen combinaciones de ambas, donde por ejemplo un container se instala sobre una maquina virtual, esta combinación logra ejecutar con más simpleza que una VM al tener las características de un Container, a la vez también adquiere el aislamiento de una VM, aunque no logra la misma ligereza de un Container a secas propiamente tal. También hay intentos de Clear Linux por crear el "Clear Container", la cual es una idea similar a un Container dentro de una VM, pero con la diferencia clave de utilizar una VM con una reducción drástica en su ligereza, solo conservando las características esenciales, esto para tener un mejor rendimiento. Por último Unikernel consiste en un tipo de container que solo se relacione con las librerías exclusivamente necesarias del Host OS (Bin/Libs) para esa VNF.

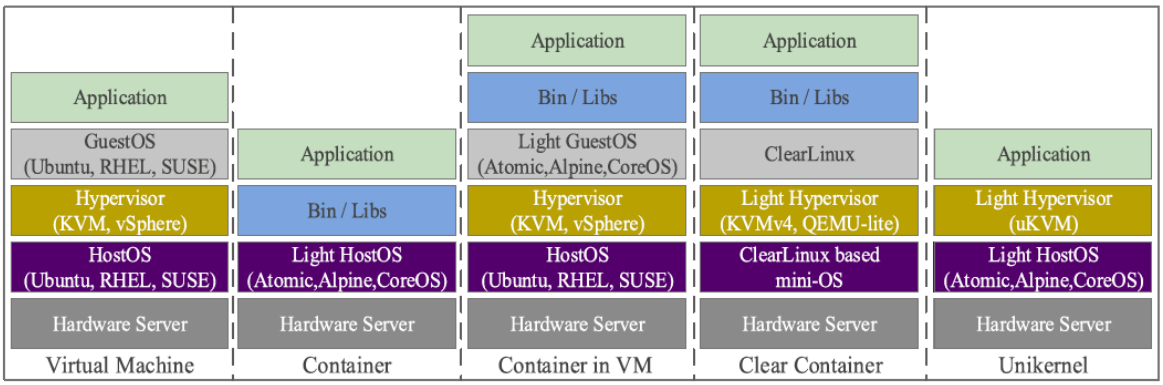


Ilustración 12: Ambientes de Virtualización para VNF[14]

Administración y Orquestación

La principal responsabilidad del Management and Orchestration (MANO), es administrar el contexto virtualizado de la NFV. Esta responsabilidad se divide en tres áreas, como se aprecia en la ilustración 13, y derivado del modelo ETSI se tiene particularmente el VIM, la NFVO y el VNFM.

En relación al despliegue de un servicio de red (Network Service NS), el NFVO administra globalmente los recursos, valida y autoriza los requerimientos de recursos de la NFVI, gestionando su ciclo de vida, administrando las políticas para las instancias de servicios de red y integrando las distintas VNFs para poder implementar el servicio final. Luego, el VNFM administra el ciclo de vida de las distintas VNFs instanciadas, además se establece que una VNF es administrada por una sola VNFM, pero una VNFM puede estar relacionadas a varias VNFs. El VIM administra y controla los recursos NFVI (cómputo, almacenamiento y red), aun cuando éste tambien puede ser configurado para manejar un tipo específico de recurso, por ejemplo solamente cómputo.

[14] Ilustración tomada de Yi, B. *"A comprehensive survey of Network Function Virtualization"*, p.223-224 (2018).

A pesar de las distinción en l os roles mencionado anteriormente, es muy común ver que estas tres áreas se traslapen o fusionen, incluso siendo usualmente implementadas como una sola entidad.

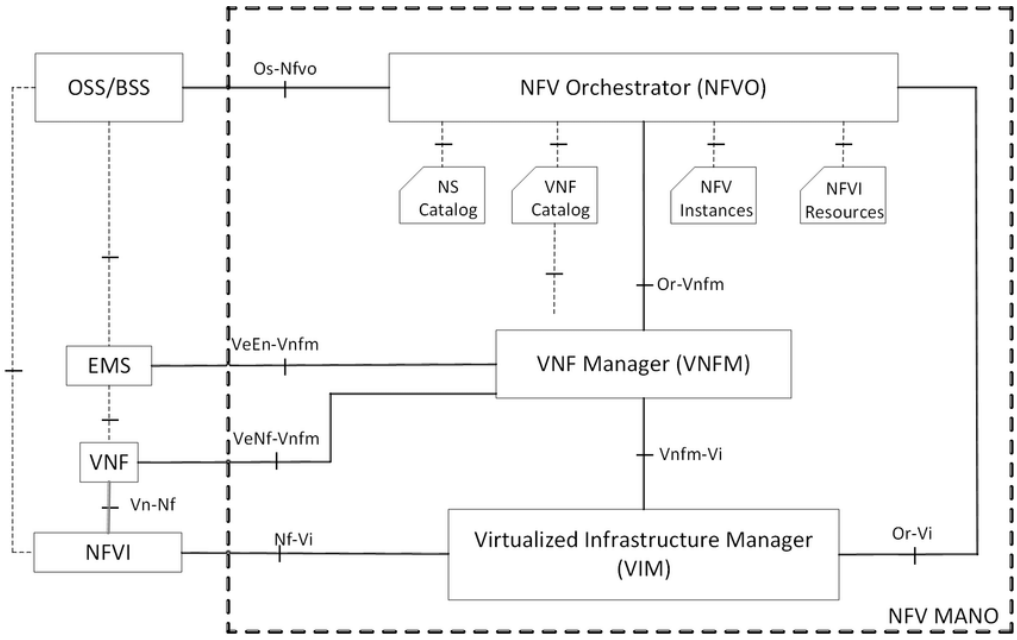


Ilustración 13: Componentes del MANO[15]

Co existencia con Sistemas Legacy

Un sistema de soporte de operaciones (OSS) es un componente de software que permite a un proveedor de servicios monitorear, controlar, analizar y administrar los servicios en su red. Estos tipos de aplicaciones de software, junto con un sistema de soporte de negocios (BSS), soportan la mayoría de las actividades de atención al cliente, incluidas las solicitudes, facturación y asistencia.
La coexistencia entre NFV y sistemas Legacy (como BSS y OSS) es inevitable, y desde el punto de vista de la operación, se da en muchos casos que no es directamente compatible. Por ejemplo, la administración BSS/OSS debe ser separado en la red estática, la cual es compuesta por los sistemas legacy, y la administración de una red que va cambiando dinámicamente a tiempo real, en redes flexibles con controlador SDN. Por tanto, aunque el ambiente NFV permita capacidades a la red como la flexibilidad de la misma, estas siguen siendo diseñadas para convivir con las PNFs existentes, las cuales realizan sus tareas sobre Hardware de Propietario con las limitaciones conocidas.

[15] Ilustración tomada de: Santos, Jose & Wauters, Tim & Volckaert, Bruno & de Turck, Filip. (2017). Fog Computing: Enabling the Management and Orchestration of Smart City Applications in 5G Networks.

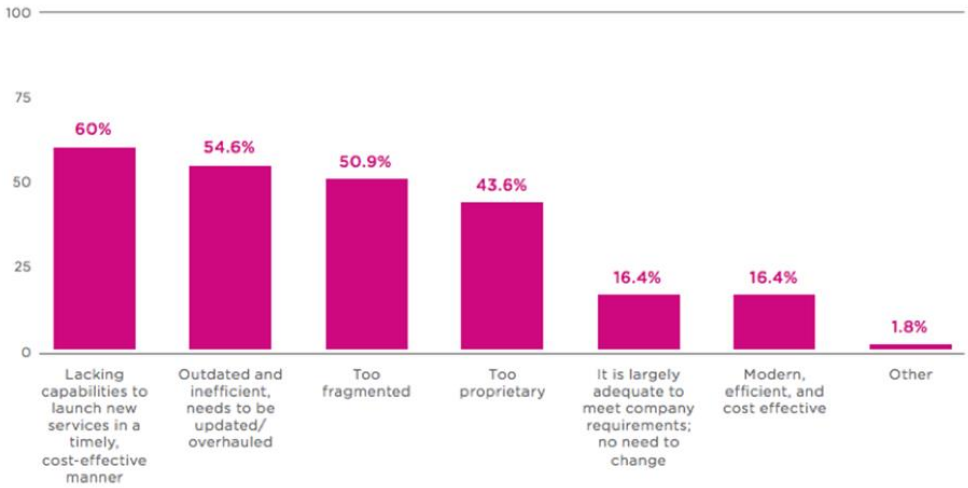


Ilustración 14: Distintos tipos de problemas de utilización de BSS/OSS con NFV[16]

Otro problema conocido de los sistemas legacy son la escasez de direcciones IPv4, pero que son inamovibles a estos equipos. Es por tanto fundamental que en ambientes NFV considere la característica de mantener habilitado IPv6 y también compatibilidad con IPv4. Otra consideración en este ámbito es el IP muticast con sus topologías asociadas. Es por esto que las VNF deban desplegar topologías adecuadas para poder habilitarlo, el cual es un problema NP-hard (problemas complejos de programación).

Y así, se pueden encontrar varios problemas de compatibilidad que deterioren el rendimiento propio de las VNF a pasar a producción. Por esto, esta transición que de por sí es muy sensible requiere de plazos bastante largos y costos altos, donde finalmente el Orquestador pueda realizar la abstracción necesaria de los sistemas físicos y virtuales.

16 Fuente de Información: https:a//www.sdxcentral.com/sdn/definitions/operations-support-system-oss-definition/ .

5G y NFV

ETSI[17] plantea que la red 5G se caracterizará por redes móviles convergentes, ágiles y robustas, basadas en tecnologías NFV y SDN, capaces de soportar funciones y aplicaciones de red que abarcan varias
redes y dominios de servicios sobre una misma infraestructura. El crecimiento de casos de uso y entornos de 5G previstos implica mejorar la escalabilidad, latencia y capacidad de admitir un número masivo de sesiones simultáneas, así como una fiabilidad y seguridad altas. Estas características proponen desafíos a quienes operen estas nuevas redes, la cuales tendrán que soportar entre otras cosas la segmentación de redes (Network Slicing), los principios de diseño nativo de la nube, la administración de servicios de extremo a extremo, la computación perimetral, la cloudificación RAN, los servicios multisitio de dominio, la administración de licencias NFV (cabe destacar sobre este punto, que es un gran cuestionamiento sobre la verdadera capacidad de disminuir costos de NFV, también hace llamado al auge de los códigos Open Source), la seguridad, la confiabilidad y la escalabilidad.

Network Slicing

Se anticipa que la Quinta Generación móvil abrirá la oportunidad de innovación a nuevas integraciones verticales. Sin embargo antes de poder concretarlo, se requieren innumerables servicios de red, todos estos con requisitos divergentes que las futuras redes deberán soportar de manera eficiente para que esto sea válido. Network Slicing aparece como una solución natural para acomodar simultáneamente, en una infraestructura de red común, la amplia gama de servicios que demandarán los casos de uso específicos de la nuevos paradigmas en la industria.
A través de tecnologías como SDN y NFV, la virtualización de la red proporciona la capacidad de programación, flexibilidad y modularidad que se requieren para crear múltiples redes lógicas (virtuales), cada una adaptada para un caso de uso dado, esto como parte de capas superiores (ver Ilustración 15) sobre una misma red física común. Estas redes lógicas se denominan Network Slicing. El concepto de redes virtuales independientes y desplegadas sobre una misma infraestructura (por ejemplo, VPN), aunque hay características específicas que hacen que los segmentos de red sean un concepto novedoso, ya que Network Slicing aunque no necesariamente diferente a una VPN, esta integra muchas más tecnologías emergentes, relacionadas a NFV, SDN, EPC, RAN, entre otras, para así formar redes virtuales independientes para fines más específicos y más capaces, como propone los nuevos usos del 5G. Definimos los network slice como redes lógicas de extremo a extremo (E2E) que se ejecutan en una red subyacente común (física o virtual), mutuamente aisladas, con control y gestión independientes, y que se

[17] Tetsuya, N. (2017). *Network Operator Perspectives on NFV priorities for 5G*. [online] Etsi.org. Available at: https://www.etsi.org/blog-subscription-information/entry/network-operator-perspectives-on-nfv-priorities-for-5g .

pueden crear a pedido. Estas redes autocontenidas deben ser lo suficientemente flexibles para acomodar simultáneamente diversos casos de uso impulsados por el negocio de múltiples clientes en una infraestructura de red.

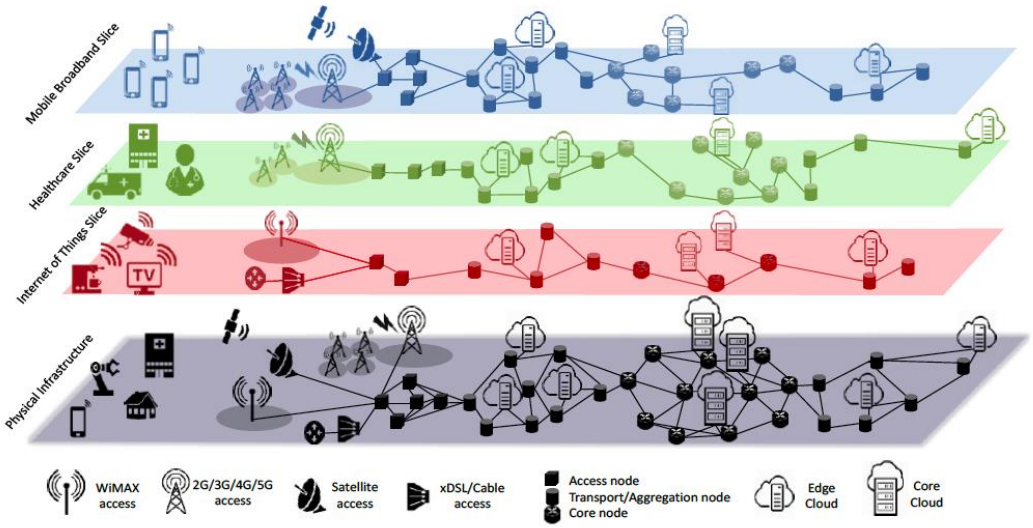


Ilustración 15: 5G Network Slicing se ejecutan en una red de múltiples proveedores y de acceso múltiple subyacente común. Cada sector se gestiona de forma independiente y aborda un caso de uso particular.

Network Service Chaning

NFV entrega la virtualización de funciones de red que históricamente han sido proporcionadas por legacy middleboxes y gateways, las cuales entregan servicios de red tales como firewalls, filtros de contenido, sistemas de detección de intrusos (IDS), Inspección de paquetes (DPI), traducción de direcciones de red (NAT), serving gateway GPRS support node (SGSN / GGSN), etc. Estas funciones entregarlas como VNFs, en el marco de exigencias de las NFV.
Este nuevo paradigma de elementos no atados al Hardware, si no que dependan de la programabilidad del software, promete ahorro de gastos operativos. Esto principalmente a través de una mejor gestión de las funciones de red. Además, NFV promete ganancias de capital al ejecutarse en hardware de servidor genérico y aprovechando las tecnologías en la nube para compartir recursos. Por fin, NFV proporcionará mayor agilidad para introducir nuevas funciones de red, lo que resultará en más rápido time to market de nuevos servicios. El Service Chaining consiste en el reenvío de tráfico de datos a través de un conjunto deseado de funciones de red (es decir, servicios o middleboxes.
Por razones de costo y eficiencia, los operadores intentan evitar enviar los distintos tráficos por servicios, cuales requieren una selección distinta de middle-boxes, a través de todas las middle-boxes posibles (ver en la Ilustración 16, que está el tráfico por servicio verde y rojo, cada uno debe pasar por ciertas cajas intermedias, y lo que se desea evitar es que no se vean obligadas a tener que pasar por todas, por lo anteriormente dicho). Según el tipo, el acuerdo de nivel de servicio (SLA) y otros factores, una política de aprovisionamiento dicta un conjunto ordenado de

servicios limitado para cada tráfico flujo para atravesar. Además, para fines de equilibrio de carga, el tráfico se envía desde una de las muchas instancias del mismo servicio. Los operadores a menudo han lidiado con este problema ya que la mayoría de estos servicios tenían un comportamiento de reenvío diferente que va desde ser servicios directos (es decir, no se puede acceder a IP), a reescritura encabezados de paquetes para funcionalidades internas (haciendo imposible el uso de tecnologías de etiquetado), y por tanto se vuelve complejo el problema de guiarlos por una cierta cadena de servicios intermedios.
Este problema es considerado NP-hard (de compleja programación), ya que para solucionarlo se debe además de implementar las VNFs de manera dinámica en ubicaciones determinadas, se debe dirigir el tráfico por las VNFs. Se han propuesto métodos heurísticos y determinísticos, donde el segundo ha sido demostrado casi imposible de implementar[18], mientras los heurísticos han demostrado ser mucho más eficientes en el tiempo, como por ejemplo el caso de Pham et al[19].

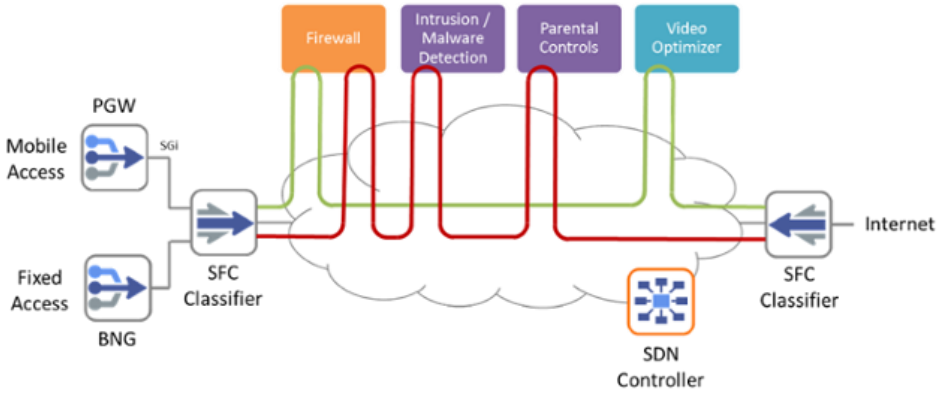


Ilustración 16: Idea de Network Chaning de dos distintos servicios que encadenan middleboxes.[20]

Métricas para Monitorear Despliegues de Redes Virtualizadas

Hay muchos proveedores potenciales de componentes de Infraestructura NFV y una flexibilidad significativa en la configuración de estos componentes para un mejor rendimiento. También hay muchos proveedores potenciales de funciones de red virtual (VNF), que se suman a las combinaciones posibles en este entorno. Con esta flexibilidad adicional, los proveedores de servicio deben trabajar para garantizar que su infraestructura esté correctamente dimensionada y que los

[18] M.F. Bari, et al., Orchestrating virtualized network functions, IEEE Trans. Netw. Serv. Manag. PP (99) (2016) 1–14, doi: 10.1109/TNSM.2016.2569020 .

[19] C. Pham, N.H. Tran, S. Ren, W. Saad, C.S. Hong, Traffic-aware and energy ef- ficient vNF placement for service chaining: joint sampling and matching ap- proach, IEEE Trans. Serv. Comput. PP (99) (2017) 1–14, doi: 10.1109/TSC.2017. 2671867 .

[20] Cloudify. (2016). *Cloudify Orchestrates Service Function Chaining (SFC) at MEF | Cloudify*. [online] Disponible en: https://cloudify.co/2016/04/28/cloudify-orchestrates-service-function-chaining-at-mef-openstack-summit-nfv-tosca-orchestration-network-automation.html .

recursos den abasto a los servicios, para que estos cumplan los requisitos de rendimiento.
En Testing de la ETSI NFV-TST 008, se encuentra la especificación de métricas para el NFVI que incluye:

- La caracterización de los diferentes tipos de servicios en términos de sus KPI críticos y los parámetros que describen sus requisitos de infraestructura.
- Un análisis de cómo medir los KPI para evaluar la utilización de recursos y el espacio en la infraestructura operativa de NFV.

Los requisitos computacionales y de red a través de VNF pueden variar significativamente dependiendo del tipo de función de red. Por ejemplo, la latencia y la tasa de paquetes por segundo son factores clave para los VNF que requieren el rendimiento de reenvío de paquetes (es decir, firewalls y enrutadores). En la Tabla 3 a continuación se muestra los requirimientos claves para una VNF en general.

Type of VNF	Example Usage	Example KPIs and Metrics Networking Virtualized Resources	Example KPIs and Metrics for Application Services	Example KPIs and Metrics for Virtualized Storage
General applicability (not specific to a particular VNF or network plugin type or host)	NVFI	VIF egress.drops VIF egress errors VIF bps In/Out VIF PPS In/Out VIF total bytes VIF total packets VIF Flows/sec VNF egress.drops VNF egress errors VNF bps In/Out VNF PPS In/Out VNF total bytes VNF total packets VNF Flows/sec VNF Number of Active Flows VNF Throughput BPS & PPS Host egress.drops Host egress errors Host bps In/Out Host PPS In/Out Host total bytes Host total packets Host Flows/sec Host Number of Active Flows Host Throughput BPS & PPS Host ipv4tables.rule_count Host ipv6tables.rule_count	Number of VCPUs (IO, Processing Cores) VNF VCPU Utilization VNF VCPU Pinning to Physical Cores Hyper-Threading NUMA VCPU Poll/Event Driven Memory size & Hugepages/ Default L1/L2/L3 Cache utilization & Cache Misses Per VF VLAN range VF Anti-Spoofing Count VF Multicast/Broadcast Host.cpu.io_wait Host.cpu.ipc Host.cpu.l3_cache.miss Host.cpu.l3_cache.usage Host.cpu.mem_bw.local Host.cpu.mem_bw.remote Host.cpu.mem_bw.total Host.cpu.usage	host.disk.io.read host.disk.io.write host.disk.response_time host.disk.read_response_time host.disk.write_response_time host.disk.smart.hdd.command_timeout host.disk.smart.hdd.current_pending_ sector_count host.disk.smart.hdd.offline_ uncorrectable host.disk.smart.hdd.reallocated_sector_ count host.disk.smart.hdd.reported_ uncorrectable_errors host.disk.smart.ssd.available_reserved_ space host.disk.smart.ssd.media_wearout_i ndicator host.disk.smart.ssd.reallocated_sector_ count host.disk.smart.ssd.wear_leveling_ count host.disk.usage.bytes host.disk.usage.percent host.memory.usage host.memory.swap.usage host.memory.dirty.rate host.memory.page_fault.rate host.memory.page_in_out.rate

Tabla 3: Ejemplos de métricas para una VNF general[21]

[21] Tabla tomada de Yi, B. *"A comprehensive survey of Network Function Virtualization"* , p.223-224 (2018).

Capítulo III: Metodología

A continuación, se procede a describir la metodología que se empleará para implementar un laboratorio de pruebas capaz de emular un vEPC, recolectar datos de alguna VNF del vEPC, y diseñar una GUI que muestre los datos.

Plan de Trabajo

Se estructura el trabajo mediante la división de bloques funcionales que forman un flujo de trabajo como se puede ver en la ilustración 17.

Ilustración 17: Flujo bloques de trabajo

Se selecciona el vEPC más adecuado, se considerará varias plataformas de código abierto que ofrecen NFV con distintas capacidades y requerimientos, luego utilizar herramientas para el orden y recopilación de bases de datos del estado de una función del virtual EPC, según las características de cualquier NFV. Una vez obtenido los datos, se filtran los índices más relevantes para el funcionamiento, para un monitoreo efectivo para el operario. Por último, se diseñará una interfaz gráfica para el acceso de esta información.

VNF Implementadas

Para la construcción de un entorno de trabajo adecuado, se seleccionará un vEPC de varias plataformas de código abierto que ofrecen NFV con distintas capacidades y requerimientos. Se escoge OAI que integra funciones RAN y EPC desagregadas y virtualizadas de la red móvil celular (4G con miras a 5G), así como también aplicaciones de borde móvil de forma programable en una arquitectura escalable de microservicios.

Para el montaje de este laboratorio se usa la siguiente disposición:

- Cloudlab: es un servicio de infraestructura dedicada para la construcción de nubes (Cloud Computing) pensada en investigadores. OpenStack ofrece usar prestado un proyecto en Cloudlab donde está montado en una Virtual Private Cloud (VPC). Esta opción evade la necesidad de tener a disposición

el hardware y el tiempo dedicado a montar OpenStack, pero el uso de este proyecto con sus recursos es por tiempo limitado, y se debe volver a pedir el equipo (hasta 16 horas con extensión bajo aceptación del portal).

Recolección y Representación de Datos

Las VNF para su correcta operación deben monitorearse como las Funciones de red físicas de hoy, utilizando las clásicas KPI de los operadores de red (por ejemplo, tráfico de entrada máximo, número de interfaces 10G compatibles, número máximo de sesiones por segundo, número máximo de flujos, etc.), pero a esto se debe agregar la perspectiva de virtualización (una vistazo general de estas está en la Tabla 3).
Para esto el módulo Celiometer está disponible para recopilar la información las VMs, Container, y otros elementos de un ambiente virtualizado. Este proyecto tiene la capacidad de normalizar y transformar la data (alertas, métricas, entre otras) de una nube que utilice módulos de OpenStack o afines. Desde aquí obtendremos los datos necesarios para las KPI que nos permitirá monitorear las VNFs.
Una vez obtenido los datos, estos se deben filtrar para un monitoreo efectivo de estos, para así obtener la información más relevante, además desde lo visual de manera eficiente para el operario. Se representa entonces, en KPI el estado del sistema: VNFs implementadas, aplicaciones en contenedores, con sus respectivas cargas de trabajo, recursos de red y utilización de disco. Estas información quedará dispuesta en gráficos en el dashboard Grafana.

Antecedentes de la Emulación

CloudLab: Construye tu propia Nube

Esta plataforma de desarrollo fue constituida para el desarrollo de nubes a nivel científico en Estados Unidos. Utiliza tecnologías de Emulab y parte de GENI[22] con un Hardware con más de 15.000 núcleos, entre otras cosas. Este equipamiento está distribuido en tres Universidades (Utah, Wisconsin y Carolina del Norte). Y es de acceso restringido a quienes requieran una cuenta y justifiquen la investigación de nube a realizar.
Uno de los objetivos clave de CloudLab es permitir repetir experimentos de manera sencilla, entregando los recursos orientados al mismo, y además dando paso a innovaciones: su objetivo es facilitar que los investigadores obtengan el mismo entorno de software y hardware para que puedan repetir o desarrollar el trabajo de los demás.
Brinda visibilidad completa de todos los aspectos de la instalación y está diseñado para minimizar el impacto que los slicings tienen entre sí.

[22] Información disponible en: https://www.cloudlab.us/technology.php .

Si bien CloudLab tiene la capacidad de aprovisionar una Virtual Private Cloud (VPC) ejecutando el hipervisor Xen, se espera que los usuarios hagan uso responsable de estos, debido al consumo energético que requiere mantener el Hardware, agregado que todo recurso utilizado podría estar siendo usado por otro investigador. Los usuarios pueden construir sus propias máquinas virtuales en esta VPC utilizando cualquier hipervisor que deseen, desarrollando dentro de este ambiente aislado el experimento de nube para su proyecto.

Conceptos Básicos

Un Profile encapsula todo lo necesario para ejecutar un experimento. Consta de dos partes principales: una descripción de los recursos (hardware, almacenamiento, red, etc.) necesarios para ejecutar el experiment y los artefactos de software que se ejecutan en esos recursos.

La especificación de recursos está en el formato RSpec[23]. La RSpec describe una topología completa: esto incluye los llamados "nodes" en los que se ejecutará el software, el almacenamiento al que están conectados y la red que los conecta. Los nodos pueden ser máquinas virtuales o servidores físicos. La RSpec puede especificar las propiedades de estos nodos, como la cantidad de RAM que deben tener, la cantidad de núcleos, etc., o puede hacer referencia directamente a una clase específica de hardware disponible en uno de los clústeres de CloudLab. La topología de la red puede incluir enlaces punto a punto, LAN, etc. y puede ser construida desde Ethernet o Infiniband.

OpenAirInterface5G: Software 5G para la Inovación Inalámbrica

La aplicación inalámbrica OpenAirInterface (OAI) se presenta como una plataforma adecuada y flexible hacia un ecosistema open LTE. La plataforma ofrece una implementación basada en software de código abierto del sistema LTE que abarca toda la pila de protocolos del estándar 3GPP tanto en E-UTRAN como en EPC. Se puede usar para construir y personalizar una estación base LTE y una red Core sobre un servidor y conectar un UE comercial para probar diferentes configuraciones de red, monitoreo, y el dispositivo móvil en tiempo real. OAI se basa en una arquitectura de frontend de radio de software alojada en PC, en la cual se pueden realizar distintas combinaciones entre simulación, emulación, y hardware dedicado de propietario:

- OAI UE: OAI eNB + OAI EPC
- OAI UE: OAI eNB + Commercial EPC
- OAI UE: Commercial eNB + OAI EPC
- OAI UE: Commercial eNB + Commercial EPC
- Commercial UE: Commercial eNB + OAI EPC
- Commercial UE: OAI eNB + Commercial EPC
- Commercial UE: OAI eNB + OAI EPC

[23] Más información sobre este formato en la guía de Emulab, en el capítulo 10.2, disponible en: http://docs.emulab.net/advanced-topics.html#%28part._rspecs%29 .

OAI Componentes

Los dos componentes básicos del Softwar de OAI son el acceso de radio eNodeB, y el Core o EPC. La comunicación radial tiene dos modalidades, puede ser montada sobre hardware dedicado para comunicaciones (OpenHardware utilizado en SDR o dedicado de vendor), como también cuenta con la capacidad de emular sobre COTS los equipos. Los componentes del core son OpenEPC, estos son MME, HSS y S/P GW.

Plataforma Software

Los repositorios de software se dividen entre openairCN que corresponde a la Core Network, y el repositorio openair5G, el cual corresponde a la red de acceso. El core puede ser conectado con otros sistemas PDN (IMS, OpendayLight, Internet, etc.), además de ser montado en Cloud Computing genérico en un ambiente OpenStack. La combinación de estos dos conjuntos de paquetes de software incluye actualmente una implementación compatible estándar de un subconjunto de la Versión 10 LTE para UE, eNB, MME, HSS, SGW y PGW en equipos de computación estándar basados en Linux (ver Ilustración 18).

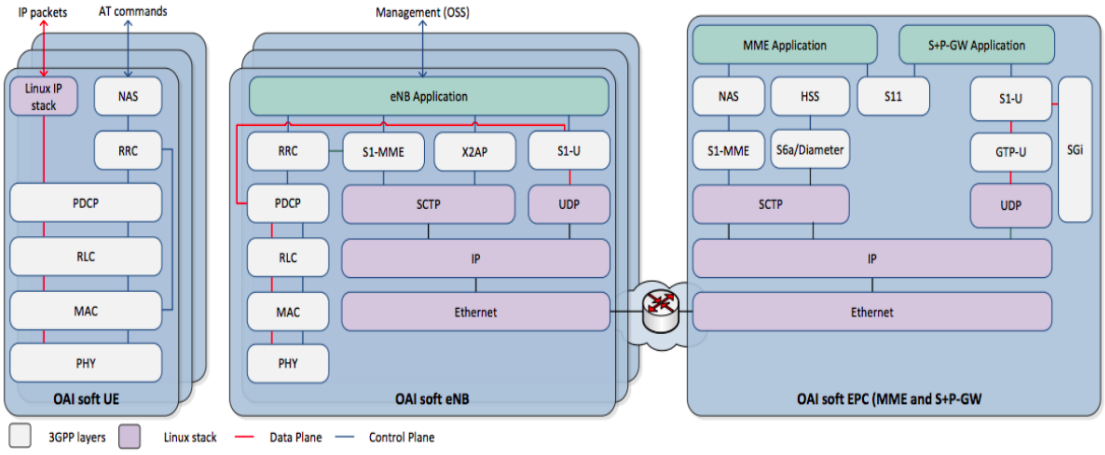


Ilustración 18: Bloques de Software que componen OAI[24]

Workflow

Se utiliza un flujo de trabajo de experimento secuencial. Se definen cinco pasos consecutivos: descripción del escenario, configuración, ejecución, monitoreo, análisis, donde cada paso se divide en varios sub-pasos como se explica en la Ilustración 19:

[24] Ilustración tomada de: Nikaein, N., Knopp, R., Kaltenberger, F., Gauthier, L., Bonnet, C., Nussbaum, D., & Ghaddab, R. (2014, September). OpenAirInterface: an open LTE network in a PC. In *Proceedings of the 20th annual international conference on Mobile computing and networking* (pp. 305-308). ACM.

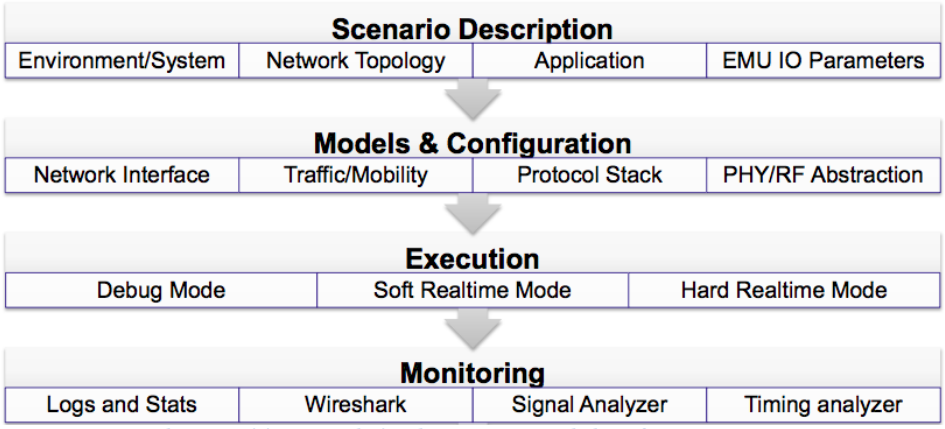


Ilustración 19: Flujo de Procesos del Software OAI[25]

- Descripción del Escenario: escrito en formato xml, los cuatro elementos principales: (1)System/enviroment: donde se definen los parámetros como ancho de banda, frecuencia, modelo de canal, entre otros. (2)Network Topology: donde se establece el tipo de red, cantidad y distribución inicial, tipo de movilidad. (3)Application: se define la aplicación y/o patrón de tráfico emulado en términos de tamaño de paquetes y tiempos de partidas. (4)EMU IO Parameters: es el set-up de los parámetros de emulación y métodos de análisis.
- Configuración: define la secuencia de componentes a inicializar basado en la descripción del escenario. (1)Network Interface: se configura la OAI IP. (2)Traffic and Mobility: se establece el patrón de tráfico y la movilidad. (3)Protocol Stack: se configuran las capas de protocolos según el tipo de escenario.
- Ejecución: El ambiente de ejecución por el emulador para sincronizar a los nodos y correr el experimento. (1)Debug Mode: la emulación se está corriendo en el User Space sin conectividad IP Linux. (2)Soft RealTime Mode: La emulación monta conexión IP y este se ajusta a la capa 2. (3)Hard RealTime Mode: La emulación se ejecuta en real time/low latency kernel con conectividad IP Linux.
- Monitoreo: se elige entre monitoreo activo o pasivo. (1)Logs and Stats: se coleccionan huellas y logs files. (2)Packet Traces: se hace una captura de los paquetes de control y señalización.

OpenStack para la creación Nubes Públicas y Privadas

Introducción

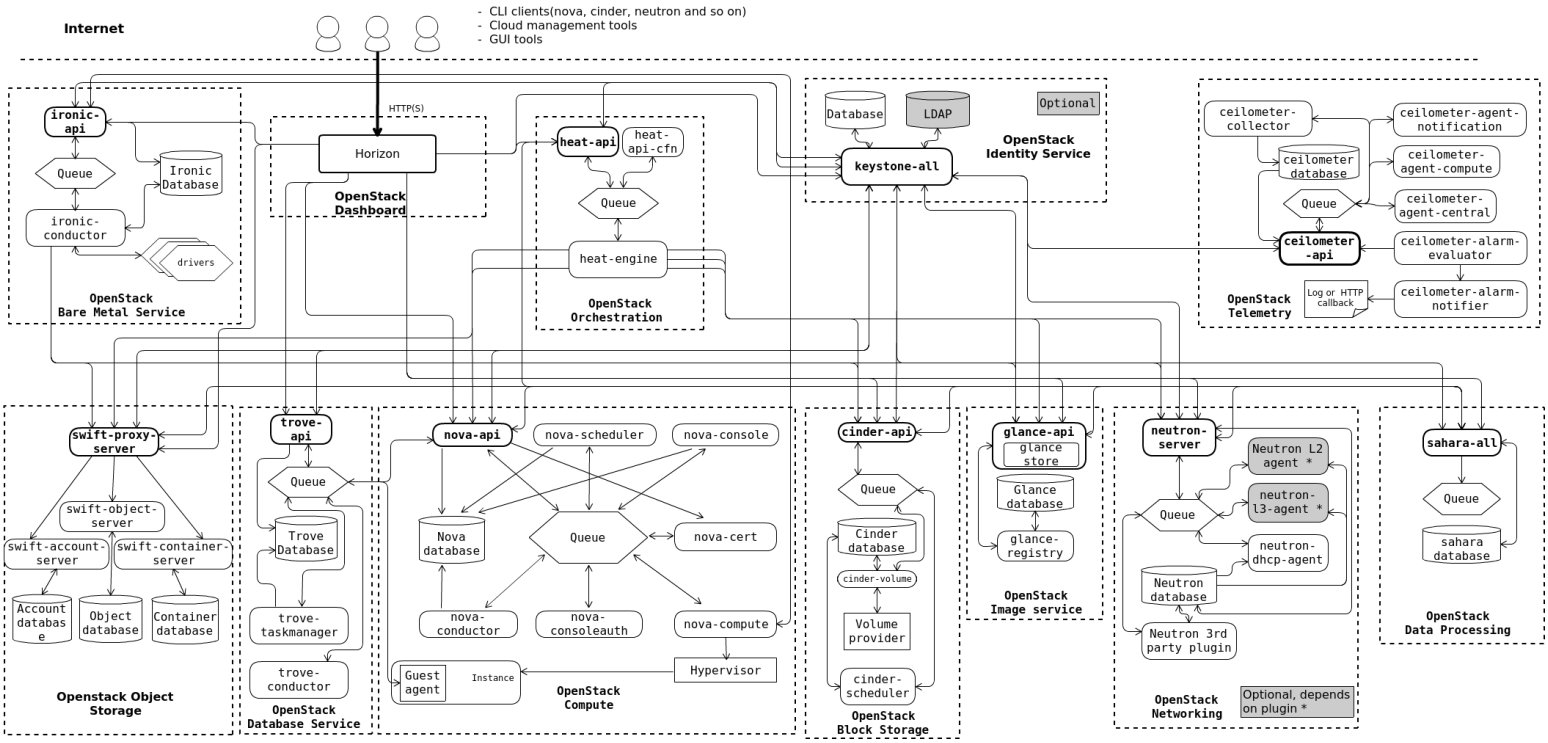
Openstack Project, consiste en diferentes proyectos de desarrollo para sistemas independientes separados, los cuales conforman la operación de una nube. El

[25] Ilustración tomada de: Nikaein, N. (2015). OpenAirInterface Simulator/Emulator. *2015-07-01)[2016-08-06]. http://www. openairinterface. org/docs/oai-oaisim-desc. pdf.*

montaje de este Software para la construcción de nodos se pueden componer de una parte o todos estos modulos (Neutron, Nova, y otros explicados más adelante). Algunas características de estos proyectos, como facilidad para escalar, automatización, rápido desarrollo debido a su característica Open Source, y gran cantidad de documentación, son lo que lo hace tan atrayente para la industria.
El código completo de Openstack es abierto, consiste en un ambiente IaaS el cual busca proveer una plataforma de computación ubicua para servicio de nubes públicas o privadas. El desarrollo de esta herramienta está dirigido por la OpenStack Foundation[26].
Su construcción modular y flexible entrega la solución mediante un set de servicios independientes que trabajan entre sí, en la cual cada una provista de una API para su fácil integración.
Es escalable en varios sentidos, Openstack fue diseñado para ser escalado horizontalmente como sea requerido, pensando en negocios crecientes.

Componentes

La arquitectura más básica de OpenStack consiste en cuatro nodos independientes, de cómputo, red, almacenamiento y un nodo controlador. Cada nodo trabaja independiente del otro, se mantienen conectados mediante las API o RabbitMQ. La comunicación entre estos nodos es mediante el protocolo Advanced Message Queue Protocol (AMQP), vía RabbitMQ utilizando proveedores de publicadores y consumidores.



[26] Los objetivos sobre el desarrollo de OpenStack por la OpenStack Foundation se puede ver en: https://www.openstack.org/foundation/

Ilustración 20: Arquitectura General de OpenStack[27]

En la Ilustración 20, se puede apreciar un diagrama relacional de cómo trabajan los distintos proyectos de Openstack, a continuación, se detallan algunos de los proyectos utilizados en este trabajo.

- Compute (NOVA): es el administrador de la infraestructura cloud computing, el cual lleva acabo todas las actividades para los ciclos de vida de las instancias que controla. También es capaz de manejar todos los recursos computacionales de la nube. No es un virtualizador, sino un administrador del Hypervisor, para realizar las tareas de virtualización utiliza libvirts APIs que soportan gran variedad de tecnologías de virtualización (LxC, KVM, XEN y VMware).
- Image Service (Glance): reenvía imágenes de Sistemas Operativos. Se encarga de manejar un repositorio de imágenes, aunque no es responsable de su almacenaje.
- Object Storage (Switft): se encarga del almacenamiento en la nube como un repositorio de objetos (ya sean videos, audios, imágenes para Glance, documentos, etc.), para que aplicaciones puedan almacenar y recolectar información mediante una API. Swift se asegura que la información sea replicada en un Cluster Server.
- Dashboard (Horizon): es una GUI para los servicios de Openstack basado en la web, esto lo hace mediante el módulo django-openstack.
- Identify Service (Keystone): maneja la base de datos de los usuarios, haciéndose repsonsable de provisionar los Tokens de AAA, y sus tipos de autentificación. Integra servicios de directorio back-end como LDAP, autorizaciones de alto nivel y Microsoft AuthZ API, entre otros servicios de proxy para autentificación.
- Networking (Neutron o Quantum): entrega Networking as a Service para otros componentes de Openstack, esto lo logra mediante la creación y atar puertos deVirtual Switches a la vNIC de las distintas VMs. Esto permite interoperabilidad y orquestación de servicios de red para IaaS, verticalizando las capas 2 y 3 (Modelo OSI) para la creación de entornos virtuales de red.
- Block Storage (Cinder): implementa servicios y librerías on-demand para entregar recursos de block storage, esto es proveer de bloques de almacenamiento a las unidades virtuales que se creen.
- Orchestration (Heat): maneja múltiples VMs de una mediante plantillas, las cuales pueden, entre otras cosas, automatizar. Heat Engine provee las principales funciones de orquestación y soporta la creación de modelos cloud Apps desarrollados en scripts.

[27] Ilustración tomada de OpenStack Documentation, Guía de Instalación, disponible en: https://docs.openstack.org/install-guide/get-started-logical-architecture.html .

- Telemetry (Celiometer): está subdividido en varios proyectos que se encargan de coleccionar y publicar información de la nube de varios objetivos como logs, alarmas, métricas, entre otros.
- Database (Trove): es Database as a Service para Openstack. Trove entrega la base para manejar y automatizar la provisión relacional o no-relacional de la base de datos.
- Bare Metal Provisioning (Ironic): entrega capacidad a servidores de Hardware dedicados en vez de VMs. También soporta plug-ins de máquinas con estándar de vendor específico.

Gnocchi: Metric as a Service

Consiste en una herramienta para almacenar e indexar datos y recursos en series de tiempo a gran escala, como los que se pueden ver en la Tabla 4. Este software resulta muy útil en ambiente de Cloud, además de ser dinámico y potencialmente multi-tenant. Ha sido diseñado para manejar grandes cantidades de tuplas (valor proveniente de varias mediciones asociados a un timestamp), siendo a la vez eficiente, escalable, y tolerante a errores, esto con el objetivo de no tener dependencia en sistemas complejos de almacenamiento.
El enfoque que toma Gnocchi consiste en vez de almacenar datos antes de procesarlos, hace una agregación de ellos previamente para luego almacenarlos. Esto resulta útil en combinación Celiometer, ya que Gnocchi es capaz de conectarse a su API y recolectar toda la información que este tiene de manera ordenada en series de tiempo, y a su vez entregarlas mediante http REST a un dashboard configurado con Grafana, para así poder visualizar la información.
A continuación, se presenta un listado con las características principales de Gnocchi:

- Interfaz HTTP REST.
- Escalable horizontalmente.
- Agregación de métricas.
- Procesa métricas por lotes.
- Archiva reglas de filtrado.
- Buscador por valor en las métricas.
- Recursos estructurados.
- Historial de recursos.
- Multi-tenant
- Soporta Grafana, Prometheus, Nagios/Icinga, protocolo Statsd, Collectd plugin, InfluxDB.

Arquitectura General

Gnocchi consta de varias características: utiliza API REST HTTP, contiene un daemon compatible con statsd opcional y un daemon[28] de procesamiento asíncrono (llamado gnocchi-metricd). Los datos son autentificados y entendidos mediante la API REST HTTP, mientras el daemon statsd se queda escuchando por

[28] Un daemon (nomenclatura usada en sistemas POSIX), servicio (nomenclatura usada en Windows) o programa residente (nomenclatura usada en MS-DOS) es un tipo especial de proceso informático no interactivo, es decir, que se ejecuta en segundo plano en vez de ser controlado directamente por el usuario.

estadísticos mediante puertos TCP o UDP. Mientras tanto el daemon Gnocchi-metricd realiza operaciones (computación estadística, limpieza de métricas, etc.) en los datos recibidos en segundo plano.

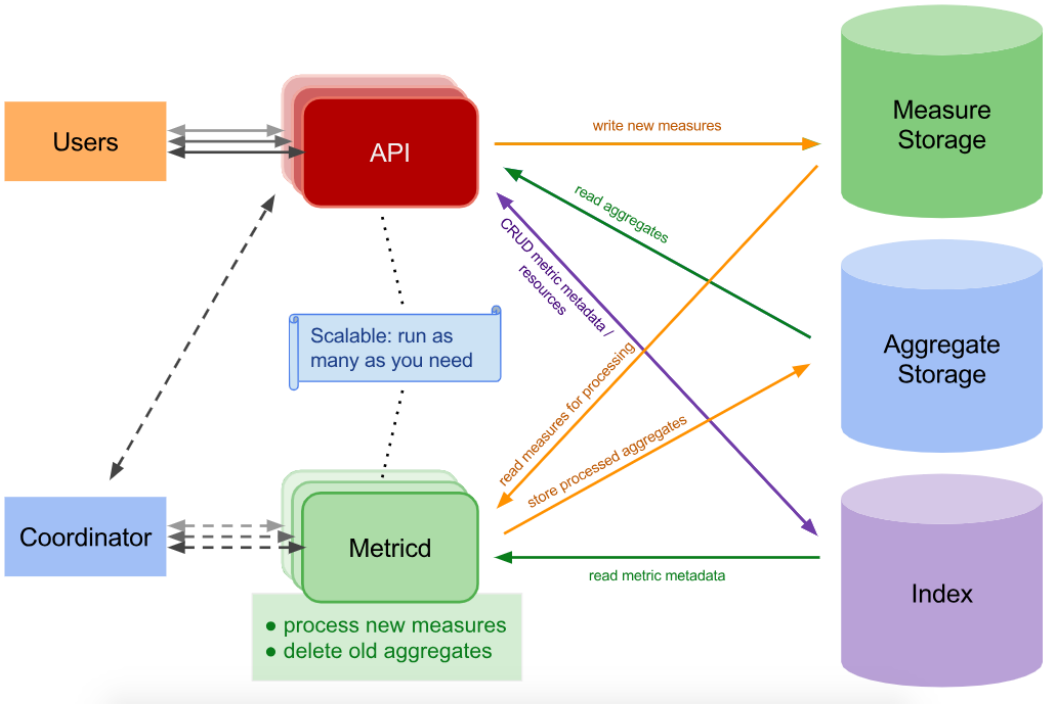


Ilustración 21: Arquitectura General de Gnocchi[29]

Como se puede ver en la Ilustración 21 hay tres componentes externos que Gnocchi necesita para funcionar correctamente:

- Un almacenamiento de medida entrante
- Un almacenamiento de métricas agregadas.
- Un índice

Grafana: Dashboard de Análisis y Monitoreo

Esta herramienta de fuente abierta consiste en un sistema de análisis y visualización de métricas, es utilizado frencuentemente para ver de una manera más elaborada series de datos. Las características que posee son las siguientes:

- Visualización: Gráficos rápidos y flexibles a pedido del cliente. Panel de PlugIn para tener distintas maneras para visualizar métricas y logs.
- Alertas: Define reglas para las alertas de manera visual, Grafana las evaluará continuamente y te enviará notificaciones.
- Notificaciones: recibe una notificación cada vez que se genere una alerta, ya sea por correo o por otros medios.
- Dashboards Dinámicos: crea, modifica o reusa distintos dashboard con plantillas.

[29] Ilustración tomada de https://opensource.com/article/17/11/getting-started-gnocchi .

- Combina Fuentes de Datos en un mismo gráfico.
- Notas: deja comentarios de eventos específicos en los mismos gráficos.
- Filtros: puedes agregar filtros o valores claves durante la marcha, los cuales son aplicados también a los datos que van ingresando.

Celiometer

El proyecto Ceilometer es un servicio de recopilación de datos que brinda la capacidad de normalizar y transformar datos en todos los componentes principales de OpenStack actuales con trabajo en curso para dar soporte a futuros componentes de OpenStack.

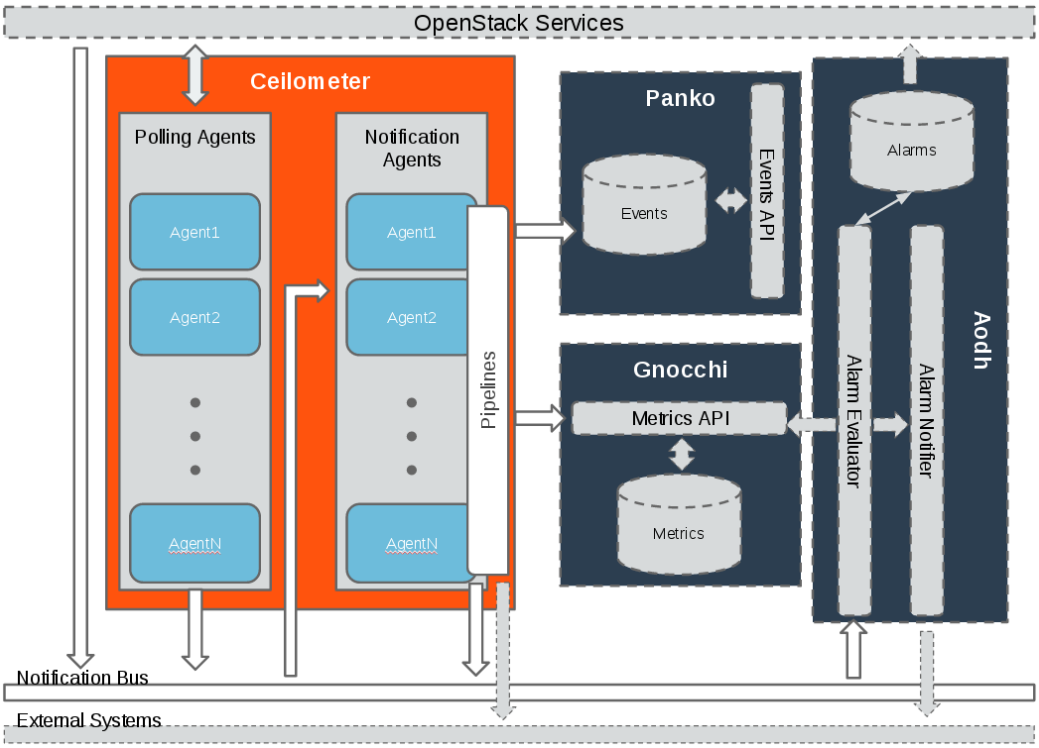


Ilustración 22: Aquitectura General de Celiometer[30]

Cada uno de los servicios de Ceilometer está diseñado para escalar horizontalmente. Se pueden agregar trabajadores y nodos adicionales según la carga esperada. Ceilometer ofrece dos servicios principales:

- Polling Agent: daemon diseñado para sondear los servicios de OpenStack y crear medidores.
- Notification Agent: daemon diseñado para escuchar notificaciones en la cola de mensajes, convertirlas en Eventos y Muestras, y aplicar acciones de canalización.

[30] Ilustración tomada de: Docs.openstack.org. (2019). *OpenStack Docs: System Architecture*. [online] Disponible en: https://docs.openstack.org/ceilometer/pike/contributor/architecture.html .

Los datos normalizados y recopilados por Ceilometer se pueden enviar a varios objetivos. Gnocchi fue desarrollado para capturar datos de medición en un formato de serie de tiempo para optimizar el almacenamiento y la consulta. Gnocchi está destinado a reemplazar la interfaz de base de datos de medición existente. Además, Aodh es el servicio alarmante que puede enviar alertas cuando se rompen las reglas definidas por el usuario. Por último, Panko es el proyecto de almacenamiento de eventos diseñado para capturar datos orientados a documentos, como registros y acciones de eventos del sistema.

Tipos de Datos

El ceilómetro está diseñado para recopilar mediciones de los servicios de OpenStack y de otros componentes externos. Si desea agregar nuevos medidores a los existentes, debe seguir las pautas que se dan en esta sección[31].

- Tipos de Datos

Tipo	Descripción
Acumulativo	Incrementando en el tiempo (horas por instancia).
Delta	Cambiante en el tiempo
Gauge	Objetos discretos (Por ejemplo: IP flotantes, subida de imágenes) y valores fluctuantes (Disk I/O)

Tabla 4: Tipos de Datos que entrega Celiometer

- Unidades

Nombre	Tipo	Unidad	Recurso	Origen	Soporte	Nota
Compute.node.cpu.percent	**Gauge**	**%**	**Host ID**	**Notificación**		**Utilización de la CPU.**
Memory.usage	**Gauge**	**Mb**	**Instance ID**	**Pollster**	**Libvirt, Hyper-V, vSphere, XenAPI**	**Volumen de RAM utilizado por alguna instancia.**
Disk.device.read.bytes	**Acumulativo**	**B**	**Disk ID**	**Pollster**	**Libvirt, Hyper-V**	**Volumen de lecturas.**
Disk.device.write.bytes	**Acumulativo**	**B**	**Disk ID**	**Pollster**	**Libvirt, Hyper-V**	**Volumen de escrituras.**
Network.incoming.bytes	**Acumulativo**	**B**	**Interface ID**	**Pollster**	**Libvirt, Hyper-V**	**Número de bytes de**

[31] Más información se va actualizando en el sitio de OpenStack oficial: https://docs.openstack.org/ceilometer/latest/contributor/measurements.html .

						entrada.
Network.outgoing.bytes	**acumulativo**	**B**	**Interface ID**	**Pollster**	**Libvirt, Hyper-V**	**Número de bytes de salida.**

Tabla 5: KPIs principales entregadas por Celiometer

Montaje del Ambiente de Laboratorio

OpenStack: Nube en Ambiente NFV/SDN

Openstack consiste en un constructor de nubes privadas o públicas, para este experimento se eligió utilizar una nube privada, debido a que los requerimientos del proyecto era hacer pruebas de concepto, las cuales requerían tener los resultados de manera privada a un solo cliente, y no existía interés porque otros usuarios pudieran acceder a la nube.
Para la instalación de la nube se utilizó la plataforma CloudLab, cual entrega Infraestructura como un servicio (IaaS)[32] a investigadores que estén realizado desarrollos sobre Cloud Computing. Mediante un RSpec (ver Anexo B) se facilita una VPC con las siguientes características:

1. Cómputo: 20 virtual CPU (Eight 64-bit ARMv8 (Atlas/A57) cores a 2.4 GHz)
2. Memoria RAM: 62,7 Gbs ECC RAM (8x 8 GB DDR3-1600 SO-DIMMs)
3. Almacenamiento: 313 GBs de Flash (SATA3 / M.2, Micron M500, hardware AES-256 encryption)
4. NIC: puertos de 1Gbps

Estos recursos eran destinados de manera virtualizada para trabajar de manera vertical, sin acceso al Backhaul, esto tiene como beneficio que cualquier error crítico que suceda sobre el experimeto queda confinado al VPC, sin afectar otras instancias que puedan estar funcionando en ese momento en los servidores de CloudLab. Por el lado negativo, no se tiene acceso directo a los recursos físicos por el cual ocurre el experimento, además que el acceso solo puede ser remoto, estos problemas pueden ser mitigados con conexiones al VNC sobre SSH, así se puede realizar comandos directamente por CLI, o utilizar la GUI (si existe) con TeamViewer[33].
La topología se puede ver en la Ilustración 23, esta consiste en dos VMs pensadas para la escalabilidad de una nube, en donde la primera llamada “ctl” es el controlador del cluster de la nube, y todas las otras instancias que tenga, sean provistas de la capacidad agregada. En el caso de este experimento solo se tiene

[32] La información del Hardware disponible de esta plataforma se puede encontrar en https://www.cloudlab.us/hardware.php .

[33] Aplicación para control remoto de licencia gratuita, disponible para servidores que utilicen Cloud Ubuntu LTS, ver Anexo C para una guía paso a paso de instalación.

otra VM llamada “cp-1”. Estas se conectan mediante una red virtual correspondiente a dos enlaces y un OVS, con capacidad E2E de un 1Gbps, y salida a Internet.

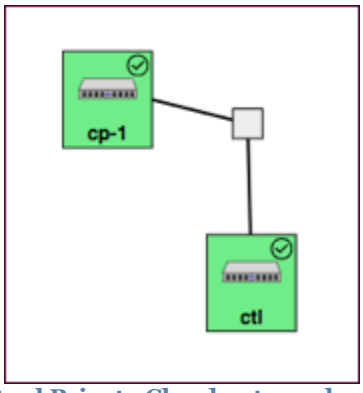


Ilustración 23: Topología Virtual Private Cloud entregada por CloudLab para OpenStack

En la instalación de OpenStack se cuenta con que la VPC está operando con Ubuntu 16.04, las cuales se les equipo con el conjunto de proyectos de OpenStack llamado Pike, los cuales son los siguientes:

- Neutron
- Cinder
- Manila
- Trove
- Heat
- Designate DNS
- Glance
- Sahara
- Celiometer
- Keystone
- Swift
- Nova

Para la instalación de Openstack mínimo existe un pack llamado DevStack, en el Anexo D hay un tutorial de como instalarlo, cabe destacar que se debe elegir la versión que soporte la versión Linux que se esté usando, y que todos los proyectos instalados son independientes y potencialmente compatibles con otros software que cumplan funcionalidades en la nube.

Montaje: OAI sobre OpenStack

Una vez instalados sobre la infraestructura con las funcionalidades que nos entrega el modelo Pike de OpenStack, es posible armar el ambiente para un vEPC emulado trabajando en condiciones SDN/NFV. Para levantar el ambiente existen varios caminos, el más automatizado es mediante una Orquestación, la cual consiste en entregarle una plantilla (ver Anexo L), con toda la información a Heat, el cual, mediante MANO, puede ir configurando para que los hypervisores generen las VNF

necesarias, con los recursos que requieren. El despliegue también se puede realizar de manera manual, primero se debe agregar al repositorio las imágenes OS de las VMs a Glance (ver Anexo E). Luego desde la interfaz gráfica Horizon, se realizó el proceso de instalación directa de cada componente del sistema virtualizado, ver Anexo J.
La implementación de los componentes virtualizados de la infraestructura móvil OpenAirInterface5G se explica en términos de implementación en la parte superior de la nube de OpenStack. Para la integración total de la nube, OpenAirInterface5G necesita tener su funcionalidad traducida para que OpenStack pueda entender los mecanismos de comunicación. Dado que OpenAirInterface5G utiliza protocolos de tunneling, así como la encapsulación de varios protocolos en TCP / SCTP y UDP, la comunicación puede experimentar una degradación cuando el EPC se implementa en la parte superior de la nube, debido a que estos encapsulamientos se pueden perder una vez salidos de las máquinas virtuales, los cuales son esenciales para su direccionamiento. Es por esto que usamos una configuración VPN simple para transportar el tráfico S1 entre instancias de eNBs y MME / SPGW, parecido a un tunel pre establecido. Esto tiene el efecto secundario positivo de que eNB se puede colocar en cualquier parte de Internet, incluso detrás de los NAT, y no necesita una segunda red externa en OpenStack, por lo que en principio cualquier nube pública de OpenStack se puede usar para implementar el núcleo, aunque queda sujeta a las versiones que va sacando OAI. Además, el tráfico S1 entre eNB y MME / SPGW está cifrado y los eNB están autentificados.

Recolección de Datos

Dado que la finalidad de este experimeto es recolectar información clave del comportamiento del vEPC, se utilizan las herramientas de OpenStack para el mantenimiento de los Ciclos de Vida de una VNF, para esto se ocupa el proyecto Celiometer, el cual da servicio de recopilación de datos que brinda la capacidad de normalizar y transformar datos en todos los componentes principales de OpenStack actuales con trabajo en curso para dar soporte a futuros componentes de OpenStack.

Extracción desde Celiometer

El proyecto Ceilometer creó 2 métodos para recopilar datos:

1. Notification Agent que toma los mensajes generados en el bus de notificación y los transforma en muestras o eventos de Ceilometer.
2. Polling Agent, sondeará alguna API u otra herramienta para recopilar información en un intervalo regular. El enfoque de sondeo puede imponer una gran importancia en los servicios API, por lo que solo debe usarse en puntos finales optimizados.

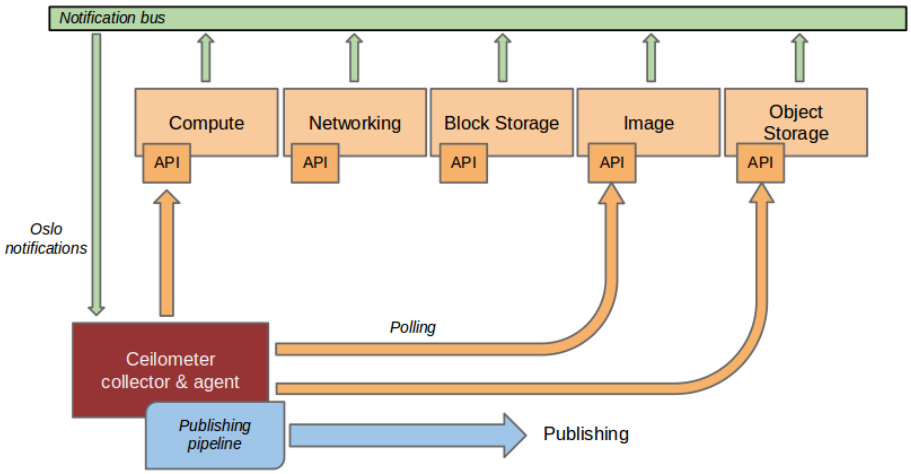


Ilustración 24: Extracción de Datos de la VPC mediante Celiometer[34]

Los datos pasan a ser procesados, Ceilometer ofrece la posibilidad de tomar los datos recopilados por los agentes, manipularlos y publicarlos en varias combinaciones a través de múltiples canales. Esta funcionalidad es manejada por los agentes de notificación. Ver Ilustración 23.

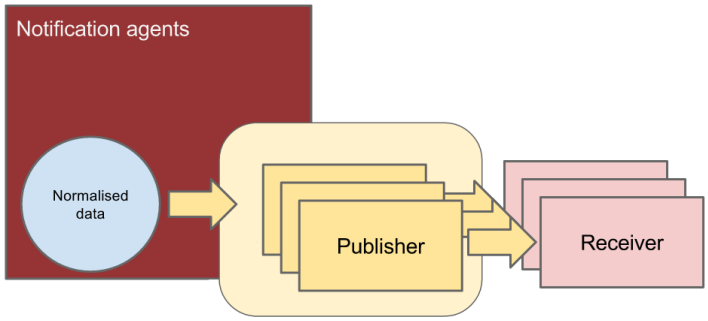


Ilustración 25: Publicación de los Datos

Luego estos paquetes pueden ser publicados por varias maneras, el método de transporte que se utiliza para este experimento es gnocchi, que publica muestras/eventos a Gnocchi API Ver Ilustración 25. Para este experimento se utilizó la API Metric cuyo End Point de servicio del la VM "ctl" se ubicaba en el puerto 8041 (Este se puede revisar desde Horizon, en la barra lateral Proyecto>Acceso a la API >Metric).

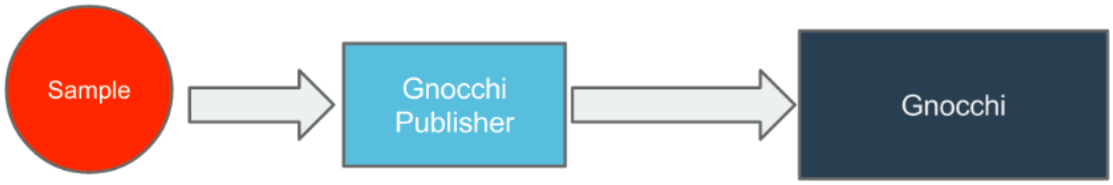


Ilustración 26: Toma de datos por la Database Gnocchi

Finalmente las métricas son almacenadas, agregadas e indexadas en la base de datos de Gnocchi como se puede ver en la Ilustración 26. Estos datos quedan

---

34 Ilustraciones 24-26 tomadas de: Docs.openstack.org. (2019). *OpenStack Docs: System Architecture*. [online] Disponible en: https://docs.openstack.org/ceilometer/pike/contributor/architecture.html .

disponibles como series de tiempo, las cuales después en la API de Grafana son posible de diferenciar en las distintas KPIs vistas en el capítulo de Métricas de Monitoreo para Redes Virtualizadas, separadas por las distintas instancias que has sido creadas. La configuración de la API de Gnocchi también puede ser configurada desde Grafana, utilizando la HTTP que entrega Metric de Celiometer, en modo Servidor, y un token que brinda seguridad a la comunicación que se puede encontrar en la configuración de Gnocchi.

Públicación de las métricas en el Dashboard

Las series de tiempo indexadas en la base de datos de Gnocchi son accesibles por Grafana mediante consultas HTTP, esta viene en forma de plugin la cual puede ser desplegada mediante script[35]. Una vez enlazada la base de datos, las series de tiempo se pueden graficar de múltiples maneras como ofrece Grafana, también como servidor HTTP, se pueden crear distintos usuarios, con distintos permisos para revisar y/o modificar los dashboards.

35 Guía para agregar la base de datos Gnocchi en Grafana: https://gnocchi.xyz/grafana.html

Capítulo IV: Resultados

A continuación, se presentarán los resultados más relevantes del trabajo realizado para la instalación de una plataforma de Private Cloud, la emulación del vEPC y el dashboard de monitoreo.

Plataforma Private Cloud

Los gráficos presentados a continuación fueron extraídos del portal de CloudLab directamente, se puede tener una apreciación cualitativa de los resultados representados, pero no la tabla misma de estos, debido a que es información protegida por la misma plataforma. El periodo de tiempo expuesto corresponde a un día, durante el cual se realizó el experimento de montaje en OAI, el cual se detallará más adelante.

Unix Load Avarage

El primer gráfico representa la medición del cómputo (Unix Load) que realiza el sistema. Una computadora completamente inactiva tiene un promedio de carga 0. Cada proceso en ejecución que usa o espera los recursos de la CPU agrega 1 al promedio de carga. Por ejemplo, si su sistema tiene una carga de 5, cinco procesos están usando o esperando la CPU.

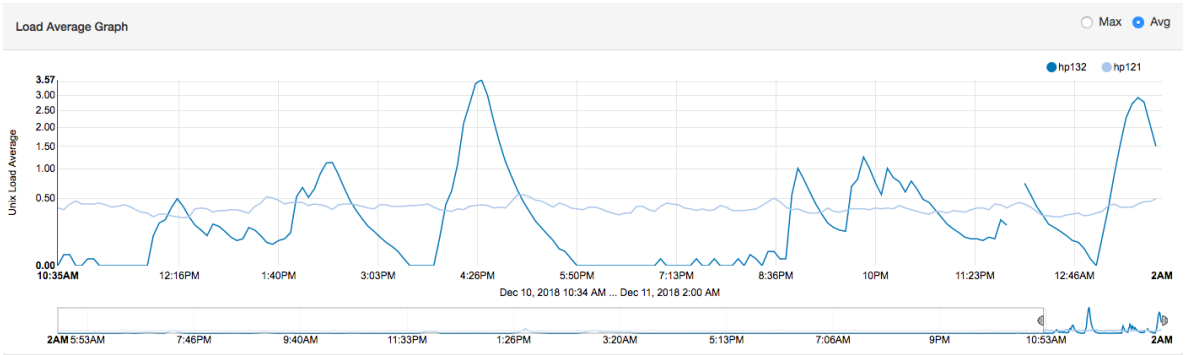


Ilustración 27: Gráfico de Unix Load del VPC

Packets Traffic

Los siguientes dos gráficos muestran los tráficos de datos que transimite y recepciona la VPC sumados. La medición a cuantos paquetes son traspasados por segundo (Packets Per Second), y el periodo de tiempo muestreado es el mismo del gráfico anterior, consiste en el día en que se realizó la prueba de Montaje en OAI.

- El primero corresponde a solamente el tráfico de control, el cual no corresponde a paquetes que tengan que ver directamente con el

experimento, sino a revisar todo lo relacionado al canal de comunicación hacia el exterior del experimento (por ejemplo Internet), pero que no sea la comunicación misma.

- El segundo corresponde al tráfico de datos, que es la información que llega o envía al VPC para todos los procesos que requieren internet.

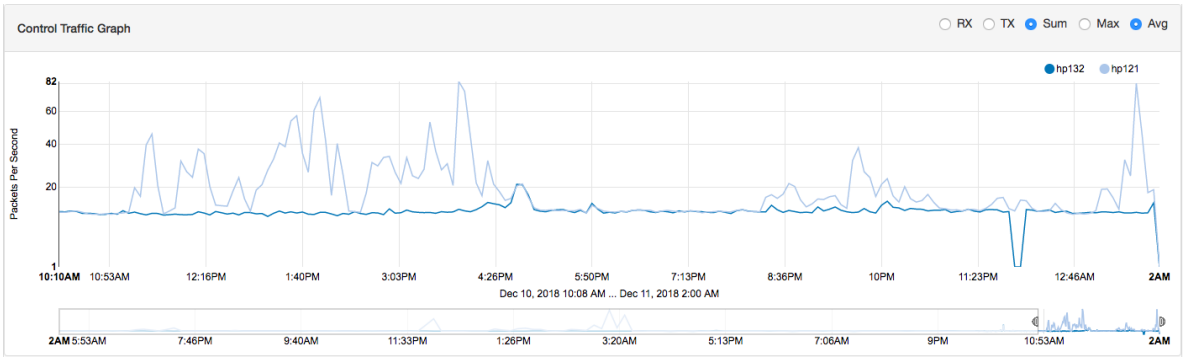


Ilustración 28: Gráfico Tráfico de Control del VPC

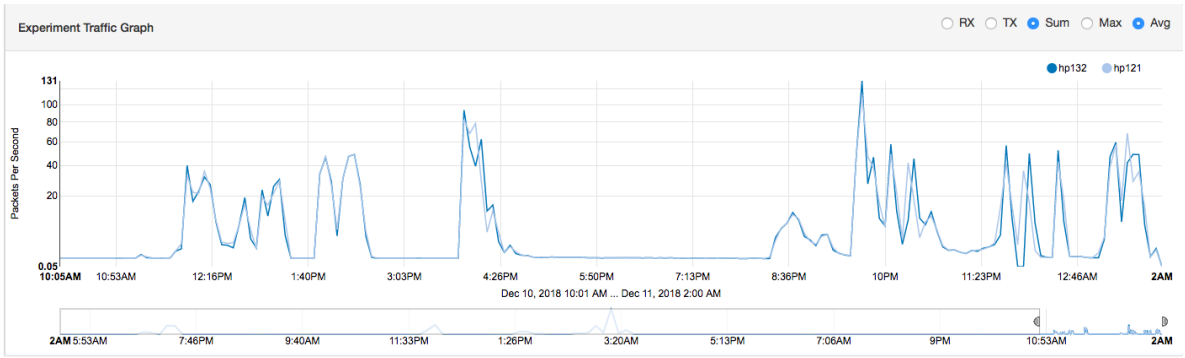


Ilustración 29: Gráfico Tráfico de Datos del VPC

Virtualización de Red y VM

En la Ilustración 30, se puede apreciar la topología de red que se creó para el montaje, esta es tomada desde Horizon, muestra los elementos ejecutados para equipar el vEPC en su ambiente emulado.

La red que conecta la entera VPC con el internet es ext-net, con dirección IP pública, la cual no se puede utilizar para las VMs directamente debido a que es utilizada por los operadores de CloudLab, pero igualmente se pueden usar direcciones IP flotantes sobre la misma. Para todas las redes que se creen, deben utilizar esta red externa como GW de salida.

Se crearon dos redes virtuales, la primera flat-lan-net-1 es utlizada para enviar comando a distancia sobre las VM, para esto, primero se establece una conexión SSH entre el computador del usuario y ctl de la VPC mediante la dirección pública, luego ctl como se conectó a la red flat-lan-net-1, puede reconocer la dirección de las VMs conectadas a esta red y ser también conectadas mediante SSH, así se puede controlar desde cualquier computador las VMs mientras se tengan las llaves. La

segunda red creada es OAI, esta entregará las conexiones entre SAE y RAN, estos servirán de canales para luego establecer los portadores del 4G.
Para establecer los GW y conectar las redes, también se utilizan routers los cuales aportan interfaces ethernet para hacer las conexiones, para esto se les asignó la subred.
Se crearon dos VMs, con el gestor que entrega Horizon para utilizar las herramientas de OpenStack, esto se aprecia en la Ilustración 30.

Ilustración 30: Topología de las VMs, Network y Routers montadas sobre OpenStack

Emulación del vEPC

Para comprobar el funcionamiento del vEPC se realizaron test con la emulación de una RAN la cual le intenta enlazar UEs. Para poder ir viendo el funcionamiento de las pruebas se utilizaron dos métodos, uno de tiempo real el cual consistía en analizar los paquetes que se transferidos, y otro con Mscgen la cual utiliza un archivo log guardado con los mensajes intercambiados de manera gráfica por protocolo.

- Sniffer: Se intervino la interfaz S1 de los nodos, la cual corresponde a la interfaz entre el MME (por parte del SAE) y eNB (por parte de la RAN), también se filtraron los mensajes que correspondieran al protocolo S1, los mensajes intercambiados para un enlazamiento exitoso se pueden ver en la Ilustración 30.

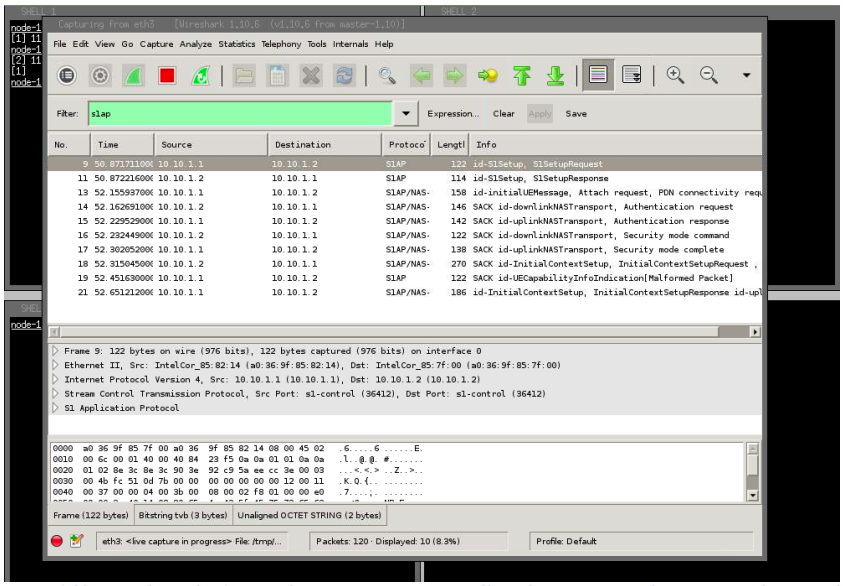


Ilustración 31: Sniffer sobre la interfaz entre SAE y flat-lan-1-net durante el attachment del UE

- Message Sequence Chart: durante el intercambio de mensajes para enlazar el UE con el MME, hay una serie de comunicaciones intermedias entre los distintos nodos del LTE, estos fueron dejando un rastro de lo que se hizo en un log, luego con el programa Mscgen[36] produce una PNG la cual representa la comunicación de distintos protocolos entre los distintos nodos conforme va avanzando el tiempo, se puede ver en la ilustración 32 la salida del programa Mscgen luego de realizar un enlazamiento exitoso de un UE con el MME.

36 Programa incluido en el script de OAI, el cual permite ilustrar mediante códigos y logs el intercambio de protocolos. Para más información http://www.mcternan.me.uk/mscgen/

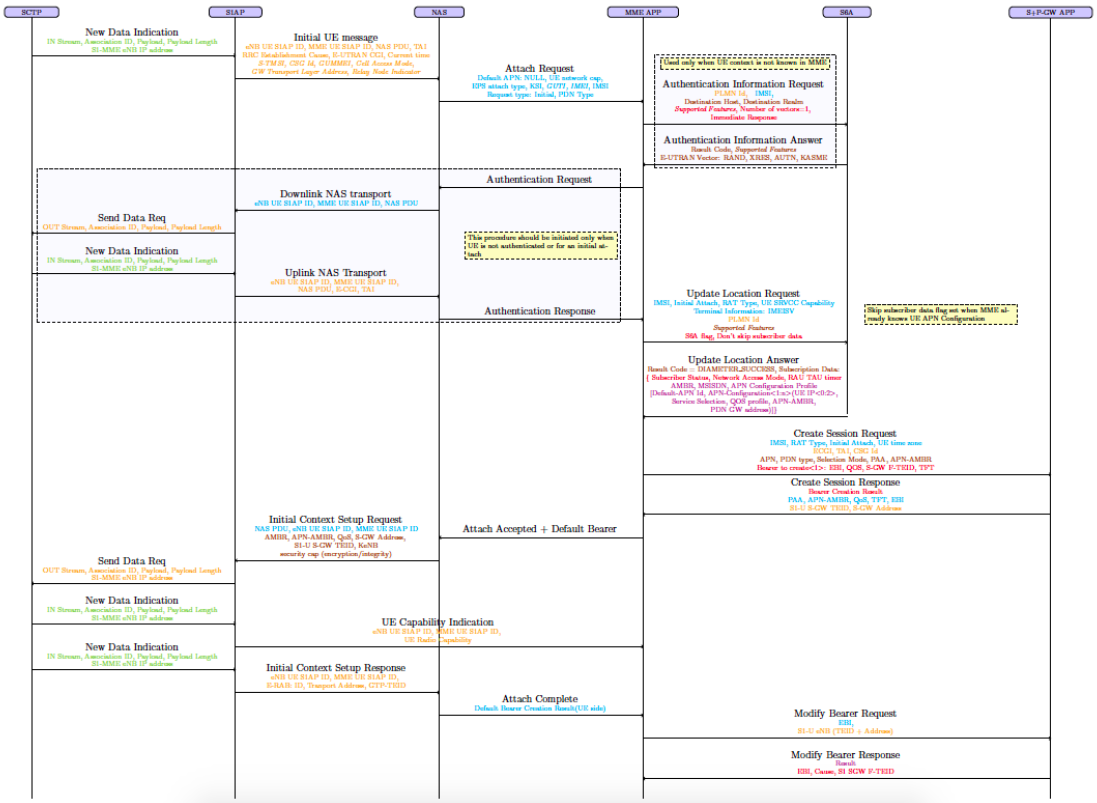


Ilustración 32: Secuencia de mensajes generado por Mscgen para la creación de un portador en el SAE

MME	MME application or default MME parameter
S1AP	S1AP provided parameter
NAS	NAS provided parameter
S6A	HSS provided parameter
S11	S+P-GW provided parameter
SCTP	SCTP provided parameter
Optional	Optional parameter

A continuación, se muestran tablas que contienen variables descriptivas de los gráficos que se muestran en los Dashboards. Los parámetros elegidos considera los siguientes valores: Promedio, Desviación estandar, Máximo y Mínimo.

Los intervalos de tiempo (granuralidad llamado en Gnocchi) escogidos para recopilar datos es cada 5 minutos, donde se entrega una dupla por nodo, correspondientes al promedio y valor máximo durante este intervalo de tiempo.

Para el análisis del experimento se escogió mostrar el estado histórico, el cual incluye todas las medidas que pudieron ser tomadas a lo largo del montaje y puesta en marcha. También se muestra la ejecución efectiva de OAI durante una prueba de enlazamiento entre un UE y el MME. En Resultados 1, el número "Valor Experimento" refiere a la recolección de datos mientrar el Software OAI realiza el enlace en la interfaz S1, esto sucede en la fecha 10 de diciembre del 2018, entre las 15:45 y las 16:35. En "Valor Histórico", hace referencia al monitoreo durante toda la semana en que se trabajó para montar el laboratorio sobre OpenStack, lo cual dura una semana aproximadamente.

CPU

Porcentaje de Utilización del CPU por la instancia de la cantidad de vCPU asignada.

Parámetro	**Valor Experimento %**	**Valor Histórico %**
Promedio	17,36	6,43
Desviación	11,02	1,83
Máximo	24,74	24,74
Mínimo	7,44	1,00

Resultados 1: Datos sobre el uso de CPU

RAM

Volumen de RAM utilizado por la instancia de la cantidad de memoria asignada.

Parámetro	**Valor Experimento Mb**	**Valor Histórico Mb**
Promedio	894,77	943,98
Desviación	626,27	768,82
Máximo	2046	2099,75
Mínimo	120	34,5

Resultados 2: Datos sobre el uso de RAM

Almacenamiento

Escritura de Datos

Volumen de escritura de datos en Bytes.

Parámetro	**Valor Experimento B**	**Valor Histórico B**
Promedio	4.009.424	92928,81
Desviación	4677872,34	241849,75
Máximo	10.198.907	2770671,34
Mínimo	1.488	0

Resultados 3: Datos sobre la escritura de datos en Almacenamiento

Lectura de Datos
Volumen de lectura de datos en Bytes.

Parámetro	**Valor Experimento B**	**Valor Histórico B**
Promedio	128193,30	11.483
Desviación	238502,93	62531,9406
Máximo	728547,70	901.116
Mínimo	0	0

Resultados 4: Datos sobre la lectura de datos en Almacenamiento

Networking

Paquetes Entrantes
Número entante de datos de red en Bytes.

Parámetro	**Valor Experimento B**	**Valor Histórico B**
Promedio	126429,89	7009,31
Desviación	234658,36	33671,94
Máximo	806917	570317
Mínimo	0,25	0

Resultados 5: Datos sobre la entrada de datos en la Red

Paquetes Saliendo
Número saliente de datos de red en Bytes.

Parámetro	**Valor Experimento B**	**Valor Histórico B**
Promedio	159684,46	19129,87
Desviación	271953,89	85923,37
Máximo	890332	994227
Mínimo	0,22	0

Resultados 6: Datos sobre la salida de datos en la Red

Dashboard de Monitoreo

Como última etapa de este trabajo se entrega un dashboard con distintos gráficos que monitorean las principales KPI (basado en la discutido en el Marco Teórico), esto se realiza mediante el programa Opensource Grafana.
La información es entregada por Gnocchi la cual la dispone como varias series de tiempo.
A continuación, se muestran dos resultados, el primero (ver Ilustración 33) luego de operar el vEPC para el enlazamiento exitoso de un UE, bajo el mismo contexto mostrado anteriormente.
Se pueden notar peaks que corresponden a los mensajes de enlazamiento, los cuales son alrededor de las 14:00 hrs, los peaks de alrededor muestran las tareas necesarias para mantener funcionando el vEPC, y tareas durante IDLE state.
El segundo (ver Ilustración 34) muestra la recopilación histórica durante una semana, durante la cual se realizó todo el montaje y la puesta en marcha del vEPC y la RAN. En el proceso histórico se puede ver una utilización de la memoria RAM constante, debido a que se dejó funcionando el sistema sin parar durante los tres primeros días con el caché sin limpiar.
Los gráficos que se decidieron tomar son las principales KPIs de la tabla de Métricas para Monitorear Redes Virtualizadas, en particular para VNF de aplicabilidad general, que consisten en Utilización de la CPU, Uso agregado de la memoria, Lecturas/Escrituras en el HDD y Entrada/Salida de bytes hacia la Red.

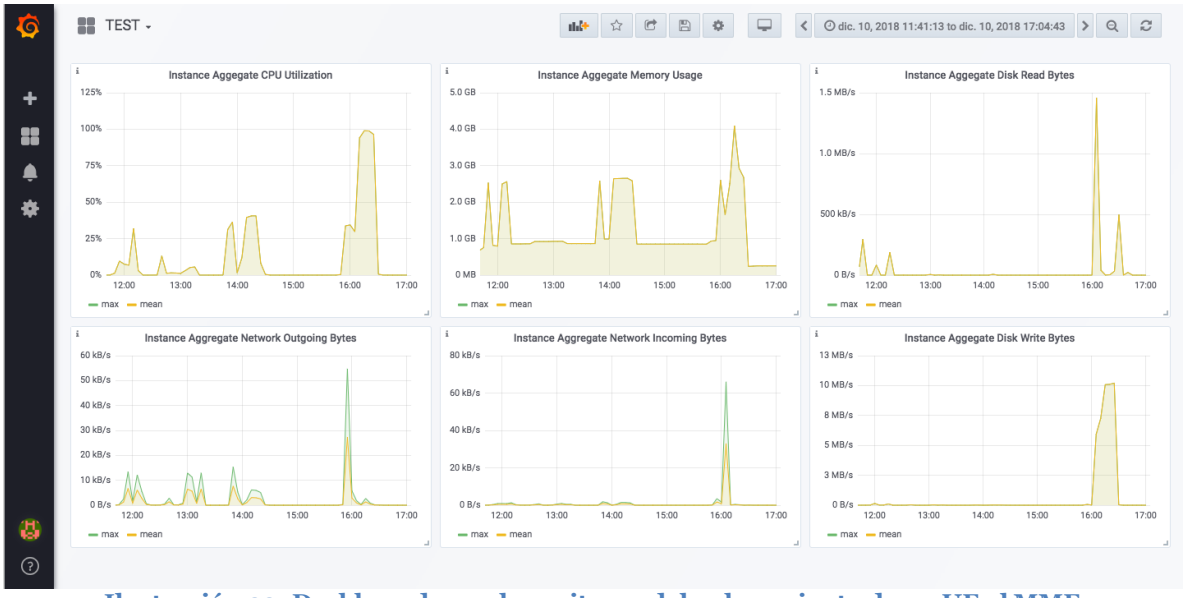


Ilustración 33: Dashboard con el monitoreo del enlazamiento de un UE al MME

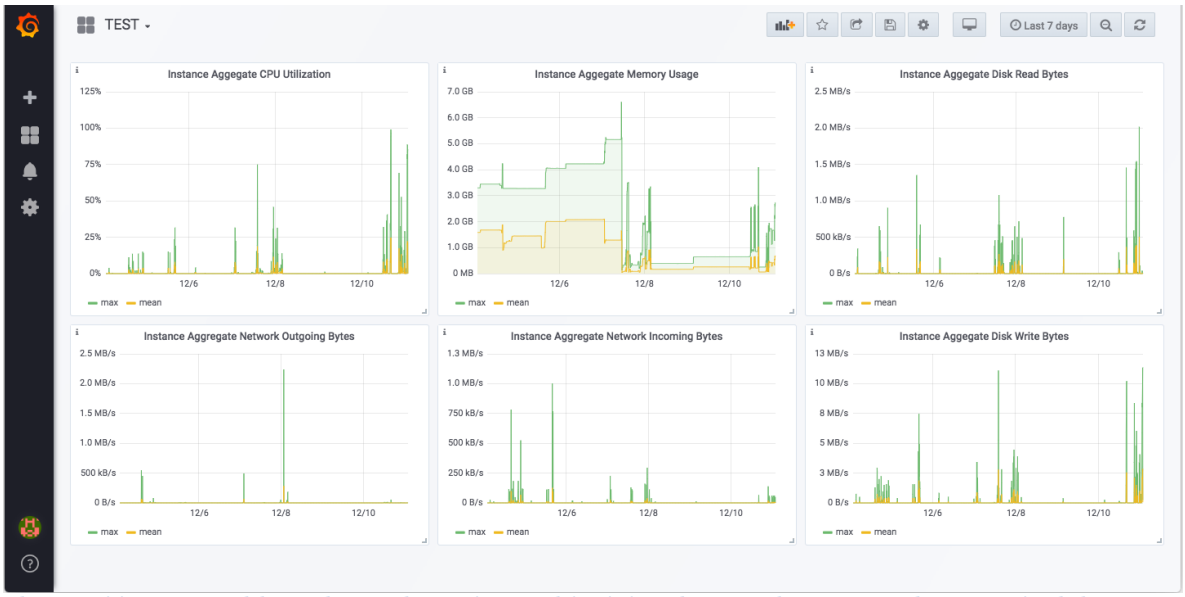


Ilustración 34: Dashboard con el monitoreo histórico durante la semana de montaje del vEPC y la RAN

Capítulo V: Conclusiones

Estudio de la Virtualización

En la red móvil, el EPC juega un papel crucial para cerrar la brecha entre los usuarios móviles y dispositivos que son gestionados por los operadores de redes móviles, y el paquete red de datos. El EPC se encuentra actualmente bajo la presión de acomodar el número cada vez mayor de dispositivos móviles, nuevos tipos de servicios y aplicaciones, y cumplir con los nuevos requisitos introducidos en la era 5G.
Debido a los desafíos que debe enfrentar la red a medida que crece y diversifica, el EPC a mutado en términos de control y acoplamiento de datos, teniendo que sustentar funciones de red basadas en cuantiosas mejoras hardware unidas a software estático, y se ha visto complejo sostener estos nuevos dispositivos y servicios, y a la vez cumplir los requisitos de 5G sin cambiar radicalmente la arquitectura.
En este caso, los operadores de red y los proveedores de servicios pueden ejecutar funciones basadas en software (es decir, VNF). Estos son montados sobre nuevos servidores basados en COTS en lugar de hardware especialmente diseñado, lo cual se traduce en reducir CapEx y OpEx.
Además, considerando la relación complementaria entre SDN y NFV, cada vez más investigaciones se centran en la aplicación de esta arquitectura integrada en otros escenarios, como 5G y IoT. Por lo tanto, es prometedor que la arquitectura integrada NFV y SDN lideraría la tendencia de red (es decir, la softwareización) en un futuro próximo.

Laboratorio de Pruebas

En este trabajo, se muestra la factibilidad de implementar un Laboratorio para Pruebas de LTE en un ambiente completamente emulado, con varias características que se explican en el marco de la virtualización. Se utilizó un ambiente SDN/NFV, sobre el cual se puede armar la topología requerida a base de distintas VMs y elementos de red virtualizado. Utilizando los medios que OpenStack y sus distintos proyectos realizan para la creación de, entre otras cosas, una VPC. Esto entregó la ventajas como la seguridad de estar aislado de los procesos con los cuales puede

estar compartiendo en la misma infraestructura física, sin tener que afectarlos en casos de fallas.
Es presentada también la plataforma OpenAirInterface, la cual a demostrado ser relevante para las pruebas de concepto y prototipos en sistemas 4G e incluso a miras al 5G. De manera concreta en este estudio el diseño e implementación de un vEPC. Se estudió su arquitectura y procedimientos generales, en la cual se utilizaron la modalidad de OAISIM, la cual simula E2E la comunicación de UEs. Se consiguió hacer funcionar OAI en las instancias entregadas por OpenStack, lo cual permitió ver su performance en un ambiente NFV/SDN.
También cabe destacar que utilizando herramientas de código abierto y los servidores de propósito general (a manera más específica de este trabajo sobre servidores de CloudLab) se logró desarrollar una instancia de aprendizaje académico, con un acercamiento práctico de las herramientas utilizadas en la industria de las Telecomunicaciones, además de demostrar capacidades competentes en las necesidades de la industria.
Por último, Grafana se muestra como una solución visual versátil, inteligente y llena de agregables. Dadas las capacidades de nubes como OpenStack y bases de datos como Gnocchi, las cuales entregan las alarmas y métricas, Grafana permitió mostrar de mucho mejor manera las KPIs de los procesos para monitorear su estado.

Trabajos Posteriores

Se dejan a dispocisión guías de instalación y puesta en marcha de las distintas herramientas virtuales utilizadas, esto con el fin de poder realizar futuros trabajos o innovaciones del experimento. Se espera poder realizar distintos proyectos sobre el Laboratorio de Pruebas presentado.
Se plantea la posibilidad de realizar pruebas de brechas de seguridad, para dar un punto de acceso 4G engañoso. Para esto remplazar la parte simulada correspondiente al RAN, con hardware dedicado para esta función (por ejemplo, con eNodeB tipo Fentomcell, que son en cirta forma portables), con un diseño enfocado a engañar UEs en corto alcance, y poder realizar pruebas de hackeo. Este escenario se vuelve de interés debido a que el acceso a un vEPC bastaría ser soportado en la nube, como quedo demostrado en este trabajo, sin la necesidad de portar el servidor por el atacante, y cualquiera puede tener acceso a los códigos de estas instancias.
También es posible modificar la plantilla presentada en el Anexo J, esto con miras a diseñar un sistema más autonomo para el manejo del vEPC, aplicando las conocidas técnicas de Cloud Computing como Autoscaling y Autohealing.
Por último, se propone el diseño de mejorar el Laboratorio de Pruebas pensando en la capacidad de poder verificar distintos elementos del 4G y futuro 5G que sean propiedad del vendor, y su compatibilidad con el estándar, poder tener interfaces que conecten el vEPC con estos equipos protegidos por propietario, y así asegurar la capacidad multivendor de estos.

Capítulo VI: Bibliografía

[1] Alcatel Lucent (2009). The LTE network Architecture—A comprehensive tutorial. *Strategic Whitepaper*.

[2] 3GPP TS 23.401. (2018). General Packet Radio Service (GPRS) Enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) Access.

[3] 3rd Generation Partnership Project. (2018). Network Architecture (Release 15). *3GPP TS 23.002*, *6*.

[4] 3GPP TS 36.300. (2017). General Packet Radio Service (GPRS); Overall Description; Stage 2.

[5] 3GPP Technical Specification 24.301, Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3 (Release 8), www.3gpp.org.

[6] Yi, B., Wang, X., Li, K., & Huang, M. (2018). A comprehensive survey of network function virtualization. *Computer Networks*.

[7] Rotsos, C., King, D., Farshad, A., Bird, J., Fawcett, L., Georgalas, N., ... & Race, N. (2017). service orchestration standardization: A technology survey. *Computer Standards & Interfaces*, *54*, 203-215.

[8] ETSI GS NFV-IFA 013: "Network Functions Virtualisation (NFV); Management and Orchestration; Os-Ma-Nfvo reference point - Interface and Information Model Specification".

[9] ETSI, N. F. V. *Management and Orchestration Network Service Templates Specification*. DGS/NFV-IFA014, Work In Progress.

[10] Portal.etsi.org. (2018). Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Report on management of NFV-MANO and automated deployment of EM and other OSS functions.

[11] Corici, M., Gouveia, F., Magedanz, T., & Vingarzan, D. (2010, May). Openepc: A technical infrastructure for early prototyping of ngmn testbeds. In *International Conference on Testbeds and Research Infrastructures* (pp. 166-175). Springer, Berlin, Heidelberg.

[13] Brown, G. (2017). Virtual Probes for NFVI Monitoring. Intel & Qosmos a division of ENEA.

[14] Nikaein, N. (2015). OpenAirInterface Simulator/Emulator. *2015-07-01) http://www. openairinterface. org/docs/oai-oaisim-desc*

[15] Ricci, R., & Eide, E. (2014). The CloudLab Team. *Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications. USENIX*, *39*(6).

[16] Markelov, A. (2016). *Certified OpenStack Administrator Study Guide*. Apress.

[17] GitLab. (2018). *oai / openairinterface5G*. [online] Available at: https://gitlab.eurecom.fr/oai/openairinterface5g.

[18] Lima, S., Rocha, Á., & Roque, L. (2017). An overview of OpenStack architecture: a message queuing services node. *Cluster Computing*, 1-12.

[19] Ordonez-Lucena, J., Ameigeiras, P., Lopez, D., Ramos-Munoz, J. J., Lorca, J., & Folgueira, J. (2017). Network slicing for 5g with SDN/NFV: concepts, architectures and challenges. *arXiv preprint arXiv:1703.04676*.

[20] ETSI GS NFV-TST 008 V2.4.1, “Network Functions Virtualisation (NFV) Release 2; Testing; NFVI Compute and Network Metrics Specification”.

Capítulo VII: Anexos

Anexo A. License Apache 2.0

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses

granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work

by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

Anexo B. RSpec para CloudLab

```
<rspec xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:jacks="http://www.protogeni.net/resources/rspec/ext/jac
ks/1"
xmlns:client="http://www.protogeni.net/resources/rspec/ext/cl
ient/1"
xmlns:emulab="http://www.protogeni.net/resources/rspec/ext/em
ulab/1" xmlns="http://www.geni.net/resources/rspec/3"
xsi:schemaLocation="http://www.geni.net/resources/rspec/3
http://www.geni.net/resources/rspec/3/request.xsd"
type="request">
  <rspec_tour
xmlns="http://www.protogeni.net/resources/rspec/ext/apt-
tour/1">
  </rspec_tour>
  <node client_id="ctl" exclusive="true">
    <sliver_type name="raw">
      <disk_image
name="urn:publicid:IDN+utah.cloudlab.us+image+emulab-
ops//UBUNTU18-64-OSCN-Q"/>
    </sliver_type>
```

```
    <interface client_id="ctl:if0">
      <ip address="10.11.10.1" netmask="255.255.0.0"
type="ipv4"/>
    </interface>
    <services>
      <install
url="http://www.emulab.net/downloads/openstack-setup-
v33.tar.gz" install_path="/tmp"/>
      <execute shell="sh" command="sudo mkdir -p /root/setup
&amp;&amp; (if [ -d /local/repository ]; then sudo -H
/local/repository/setup-driver.sh 2&gt;&amp;1 | sudo tee
/root/setup/setup-driver.log; else sudo -H /tmp/setup/setup-
driver.sh 2&gt;&amp;1 | sudo tee /root/setup/setup-
driver.log; fi)"/>
    </services>
    <jacks:site id="1"/>
    <emulab:rootkey private="false" public="false"/>
  </node>
  <node client_id="cp-1" exclusive="true">
    <sliver_type name="raw">
      <disk_image
name="urn:publicid:IDN+utah.cloudlab.us+image+emulab-
ops//UBUNTU18-64-OSCP-Q"/>
    </sliver_type>
    <interface client_id="cp-1:if0">
      <ip address="10.11.10.2" netmask="255.255.0.0"
type="ipv4"/>
    </interface>
    <services>
      <install
url="http://www.emulab.net/downloads/openstack-setup-
v33.tar.gz" install_path="/tmp"/>
      <execute shell="sh" command="sudo mkdir -p /root/setup
&amp;&amp; (if [ -d /local/repository ]; then sudo -H
/local/repository/setup-driver.sh 2&gt;&amp;1 | sudo tee
/root/setup/setup-driver.log; else sudo -H /tmp/setup/setup-
driver.sh 2&gt;&amp;1 | sudo tee /root/setup/setup-
driver.log; fi)"/>
    </services>
    <jacks:site id="1"/>
    <emulab:rootkey private="false" public="false"/>
  </node>
  <link client_id="flat-lan-1">
    <interface_ref client_id="ctl:if0"/>
    <interface_ref client_id="cp-1:if0"/>
    <link_type name="lan"/>
```

```
  </link>
  <emulab:routable_pool client_id="ctl" count="4" type="any">
    <jacks:site id="1"/>
  </emulab:routable_pool>
  <ns0:profile_parameters
xmlns:ns0="http://www.protogeni.net/resources/rspec/ext/johns
ond/1">
    <ns0:parameter>CONTROLLER="ctl"</ns0:parameter>
    <ns0:parameter>NETWORKMANAGER="ctl"</ns0:parameter>
    <ns0:parameter>COMPUTENODES="cp-1 "</ns0:parameter>
    <ns0:parameter>DATALANS="flat-lan-1"</ns0:parameter>
    <ns0:parameter>DATAFLATLANS="flat-lan-1"</ns0:parameter>
    <ns0:parameter>DATAVLANS=""</ns0:parameter>
    <ns0:parameter>DATAVXLANS="0"</ns0:parameter>
    <ns0:parameter>DATATUNNELS=1</ns0:parameter>
    <ns0:parameter>MGMTLAN=""</ns0:parameter>
    <ns0:parameter>DO_APT_INSTALL=1</ns0:parameter>
    <ns0:parameter>DO_APT_UPGRADE=0</ns0:parameter>
    <ns0:parameter>DO_APT_DIST_UPGRADE=0</ns0:parameter>

<ns0:parameter>DO_UBUNTU_CLOUDARCHIVE_STAGING=0</ns0:paramete
r>
    <ns0:parameter>DO_APT_UPDATE=1</ns0:parameter>
    <ns0:parameter>ADMIN_PASS_HASH=''</ns0:parameter>
    <ns0:parameter>ENABLE_HOST_PASSTHROUGH=1</ns0:parameter>

<ns0:parameter>ENABLE_NEW_SERIAL_SUPPORT=0</ns0:parameter>
    <ns0:parameter>DISABLE_SECURITY_GROUPS=0</ns0:parameter>

<ns0:parameter>DEFAULT_SECGROUP_ENABLE_SSH_ICMP=1</ns0:parame
ter>
    <ns0:parameter>USE_NEUTRON_LBAAS=1</ns0:parameter>
    <ns0:parameter>CEILOMETER_USE_MONGODB=0</ns0:parameter>
    <ns0:parameter>VERBOSE_LOGGING="False"</ns0:parameter>
    <ns0:parameter>DEBUG_LOGGING="False"</ns0:parameter>
    <ns0:parameter>TOKENTIMEOUT=14400</ns0:parameter>
    <ns0:parameter>SESSIONTIMEOUT=14400</ns0:parameter>
    <ns0:parameter>KEYSTONEUSEMEMCACHE=0</ns0:parameter>
    <ns0:parameter>QUOTASOFF=1</ns0:parameter>
    <ns0:parameter>ML2PLUGIN=openvswitch</ns0:parameter>

<ns0:parameter>USE_DESIGNATE_AS_RESOLVER=1</ns0:parameter>
    <ns0:parameter>EXTRAIMAGEURLS=''</ns0:parameter>
    <ns0:parameter>OSRELEASE='queens'</ns0:parameter>
  </ns0:profile_parameters>
  <emulab:password name="adminPass"/>
```

```
  <data_set
xmlns="http://www.protogeni.net/resources/rspec/ext/profile-
parameters/1">
    <data_item
name="emulab.net.parameter.greDataLanCount">1</data_item>
    <data_item
name="emulab.net.parameter.ml2plugin">openvswitch</data_item>
    <data_item
name="emulab.net.parameter.keystoneVersion">0</data_item>
    <data_item
name="emulab.net.parameter.firewall">False</data_item>
    <data_item
name="emulab.net.parameter.enableNewSerialSupport">False</dat
a_item>
    <data_item
name="emulab.net.parameter.tempBlockstoreMountNodes"></data_i
tem>
    <data_item
name="emulab.net.parameter.quotasOff">True</data_item>
    <data_item
name="emulab.net.parameter.computeHostBaseName">cp</data_item
>
    <data_item
name="emulab.net.parameter.vlanDataLanCount">0</data_item>
    <data_item
name="emulab.net.parameter.controllerDiskImage"></data_item>
    <data_item
name="emulab.net.parameter.multiplexFlatLans">False</data_ite
m>
    <data_item
name="emulab.net.parameter.doCloudArchiveStaging">False</data
_item>
    <data_item
name="emulab.net.parameter.computeNodeCount">1</data_item>
    <data_item
name="emulab.net.parameter.enableHostPassthrough">True</data_
item>
    <data_item
name="emulab.net.parameter.networkManagerDiskImage"></data_it
em>
    <data_item
name="emulab.net.parameter.tempBlockstoreMountPoint"></data_i
tem>
    <data_item
name="emulab.net.parameter.tokenTimeout">14400</data_item>
    <data_item
```

```
name="emulab.net.parameter.networkManagerHost">ctl</data_item
>
    <data_item
name="emulab.net.parameter.doAptDistUpgrade">False</data_item
>
    <data_item
name="emulab.net.parameter.osLinkSpeed">0</data_item>
    <data_item
name="emulab.net.parameter.localBlockstoreMountPoint">/image-
dataset</data_item>
    <data_item
name="emulab.net.parameter.blockstoreURN"></data_item>
    <data_item
name="emulab.net.parameter.sessionTimeout">14400</data_item>
    <data_item
name="emulab.net.parameter.enableInboundSshAndIcmp">True</dat
a_item>
    <data_item
name="emulab.net.parameter.localBlockstoreSize">0</data_item>
    <data_item
name="emulab.net.parameter.ubuntuMirrorPath"></data_item>
    <data_item
name="emulab.net.parameter.computeDiskImage"></data_item>
    <data_item
name="emulab.net.parameter.publicIPCount">4</data_item>
    <data_item
name="emulab.net.parameter.managementLanType">vpn</data_item>
    <data_item
name="emulab.net.parameter.computeNodeCountSite2">0</data_ite
m>
    <data_item
name="emulab.net.parameter.fromScratch">False</data_item>
    <data_item
name="emulab.net.parameter.ipAllocationStrategy">script</data
_item>
    <data_item
name="emulab.net.parameter.vxlanDataLanCount">0</data_item>
    <data_item
name="emulab.net.parameter.blockstoreMountNode">ctl</data_ite
m>
    <data_item
name="emulab.net.parameter.flatDataLanCount">1</data_item>
    <data_item
name="emulab.net.parameter.controllerHost">ctl</data_item>
    <data_item
name="emulab.net.parameter.blockstoreReadOnly">True</data_ite
```

```
m>
    <data_item
name="emulab.net.parameter.enableDebugLogging">False</data_it
em>
    <data_item
name="emulab.net.parameter.keystoneUseMemcache">False</data_i
tem>
    <data_item
name="emulab.net.parameter.tempBlockstoreSize">0</data_item>
    <data_item
name="emulab.net.parameter.localBlockstoreReadOnly">True</dat
a_item>
    <data_item
name="emulab.net.parameter.doAptUpdate">True</data_item>
    <data_item
name="emulab.net.parameter.enableVerboseLogging">False</data_
item>
    <data_item
name="emulab.net.parameter.firewallStyle">none</data_item>
    <data_item
name="emulab.net.parameter.ceilometerUseMongoDB">False</data_
item>
    <data_item
name="emulab.net.parameter.osNodeType"></data_item>
    <data_item
name="emulab.net.parameter.doAptInstall">True</data_item>
    <data_item
name="emulab.net.parameter.localBlockstoreURN"></data_item>
    <data_item
name="emulab.net.parameter.disableSecurityGroups">False</data
_item>
    <data_item
name="emulab.net.parameter.enableNeutronLoadBalancing">True</
data_item>
    <data_item
name="emulab.net.parameter.extraImageURLs"></data_item>
    <data_item
name="emulab.net.parameter.doAptUpgrade">False</data_item>
    <data_item
name="emulab.net.parameter.useDesignateAsResolver">True</data
_item>
    <data_item
name="emulab.net.parameter.ubuntuMirrorHost"></data_item>
    <data_item
name="emulab.net.parameter.release">queens</data_item>
    <data_item
```

```
name="emulab.net.parameter.keystoneUseWSGI">2</data_item>
    <data_item
name="emulab.net.parameter.localBlockstoreMountNode">ctl</dat
a_item>
    <data_item
name="emulab.net.parameter.blockstoreMountPoint">/dataset</da
ta_item>
  </data_set>
</rspec>
```

Anexo C. Instalación de Teamviewer remota

Instala el OS Ubuntu Cloud que más te acomode, los repositorios se encuentran en:
https://cloud-images.ubuntu.com/
Luego para tener un light Desktop (GUI) en el servidor corre el siguiente comando sobre SSH:

```
$ sudo apt-get install --no-install-recommends ubuntu-desktop gnome-
panel gnome-settings-daemon metacity nautilus gnome-terminal
```

Instala el servidor VNC para controlar tener acceso a la GUI remotamente:

```
$ sudo apt-get install vnc4server
```

Descarga el package que contiene Teamviewer con el siguiente comando:

```
$ wget https://download.teamviewer.com/download/teamviewer_i386.deb
```

Por último instala Teamviewer:

```
$ sudo dpkg -i teamviewer_i386.deb
```

Observación: El puerto default en que funciona VNC es el 5900

Anexo D. Instalación de OpenStack con DevStack

Se asume que está instalado OS Linux al menos de las últimas dos versiones de Ubuntu, la última versión de Fedora, CentOS 7,Debian o OpenSUSE. Se crea una usuario que no sea root, pero con capacidad de sudo:

```
$ sudo useradd -s /bin/bash -d /opt/stack -m stack
```

Se le agregan los privilegios:

```
$ echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
$ sudo su - stack
```

Descargamos el repositorio de DevStack, el cual contiene scripts de instalación y plantillas para una configuración:

```
$ git clone https://git.openstack.org/openstack-dev/devstack
$ cd devstack
```

Determina tu dirección IP:

```
$ sudo ifconfig
```

Crea un archivo de configuración local.conf, con 4 claves mínimas para su funcionamiento, además de la dirección IP:

```
$cat
>local.conf<<EOF
[[local|localrc]]
ADMIN_PASSWORD=secret
DATABASE_PASSWORD=\$ADMIN_PASSWORD
RABBIT_PASSWORD=\$ADMIN_PASSWORD
SERVICE_PASSWORD=\$ADMIN_PASSWORD
HOST_IP=x.x.x.x
RECLONE=yes
EOF
```

Inicia la instalación corriendo el siguiente script:

```
$ ./stack.sh
```

Una vez terminada la instalación tendrás instalados keystone, glance, nova, cinder, neutron, y horizon. Para instalar otros proyectos de OpenStack hay guías disponibles en https://docs.openstack.org/rocky/install/.

Anexo E. Agregar Imágenes de OS al repositorio de Glance

Primero debemos agregar las imagenes de los OS que utilizaremos en las VMs que luego serán los nodos del LTE, partimos descargando los OS:

```
$wget -P /tmp https://cloud-images.ubuntu.com/xenial/current/xenial-server-cloudimg-amd64-disk1.img
$ wget -P /tmp https://cloud-images.ubuntu.com/trusty/current/trusty-server-cloudimg-amd64-disk1.img
```

Luego estas son agregadas al repositorio de Glance:

```
$ openstack image create --file /tmp/xenial-server-cloudimg-amd64-disk1.img --disk-format qcow2 --container-format bare --public xenial-server-cloudimg-amd64-disk1
$ openstack image create --file /tmp/trusty-server-cloudimg-amd64-disk1.img--disk-format qcow2 --container-format bare --public trusty-server-cloudimg-amd64-disk1
```

Anexo F. Creación de un par de llaves para la comunicación SSH

Elige una carpeta segura donde guardar tus llaves, generalmente se utiliza:

```
$ cd /Users/"tu username"/.ssh
```

Generas la llave con el siguiente comando:

```
$ ssh-keygen -t rsa -b 4096 -C "tu_email@example.com" -f cloud.key
```

Se te pedirá ingresar una clave como passphrase:

```
Enter passphrase (empty for no passphrase): [Type a passphrase]
Enter same passphrase again: [Type passphrase again]
```

Agregamos la llave al SSH-Agent, partiendo primero ssh-agent:

```
$ eval "$(ssh-agent -s)"
#Agent pid 59566
```

Por último agregamos la llave recien creada:

```
$ ssh-add -K ~/.ssh/id_cloud
```

Luego para ingresar a la VM mediante SSH se utiliza:

```
$ ssh -i cloud.key <username>@<instance_ip>
```

Anexo G. Instalación del software OAI para el EPC

Tutorial tomado del GitLab de OpenAirInterface project, buscar en https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/home

Example Setup

IP of eNB : 192.168.12.117
IP of EPC : 192.168.12.81

OAISIM sim card information

IMSI : 208930100001111
Ki Value : 8baf473f2f8fd09487cccbd7097c6862
Operator key : 1006020f0a478bf6b699f15c062e42b3

EPC Setup

1. FQDN Configuration

/etc/hosts must contain the fqdn of the MME and HSS (same since both run on the same host):
(assuming the realm is openair4G.eur)

```
127.0.0.1 localhost
127.0.1.1 <hostname>.openair4G.eur <hostname>
```

2. Configuring and Building HSS

2.1 Modified the oai_db in HSS database (Accessing HSS database through http://127.0.0.1/phpmyadmin is recommended)

- In table mmeidentity, enter the record corresponding to your MME:

- In table pdn, enter the record allowing the IMSI mentioned above(208930100001111) to connect to an APN
- In table users, make sure the record with imsi=208930100001111 having correct key and mmeidentity_idmmeidentity. key should be 8baf473f2f8fd09487cccbd7097c6862 and mmeidentity_idmmeidentity should match the ID you put in mmeidentity table

2.2 Also, you have to change the configuration file /usr/local/etc/oai/hss.conf like this:

```
HSS :
{
## MySQL mandatory options
MYSQL_server = "127.0.0.1";     # HSS S6a bind address
MYSQL_user   = "MYSQL_USERNAME";  # Database server login
MYSQL_pass   = "MYSQL_PASSWORD";  # Database server password
MYSQL_db     = "oai_db";        # Your database name

## HSS options
OPERATOR_key = "1006020f0a478bf6b699f15c062e42b3"; # OP key for OAISIM

RANDOM = "true";                                    # True random or
only pseudo random (for subscriber vector generation)
## Freediameter options
FD_conf = "/usr/local/etc/oai/freeDiameter/hss_fd.conf";
};
```

2.3 Run HSS

```
$ ~/openair-cn/SCRIPTS$ ./run_hss
```

3. Configuring and Building MME

3.1 Check the following fields in your /usr/local/etc/oai/mme.conf

```
 GUMMEI_LIST = (
     {MCC="208" ; MNC="93"; MME_GID="4" ; MME_CODE="1"; }           #
YOUR GUMMEI CONFIG HERE
    );
TAI_LIST = (
     {MCC="208" ; MNC="93";  TAC = "1"; }                           #
YOUR TAI CONFIG HERE
    );
 NETWORK_INTERFACES :
 {
      # MME binded interface for S1-C or S1-MME  communication (S1AP),
can be ethernet interface, virtual ethernet interface, we don't advise
wireless interfaces
      MME_INTERFACE_NAME_FOR_S1_MME     = "eth1";                   #
YOUR NETWORK CONFIG HERE
      MME_IPV4_ADDRESS_FOR_S1_MME       = "192.168.12.81/24";       #
YOUR NETWORK CONFIG HERE

      # MME binded interface for S11 communication (GTPV2-C)
      MME_INTERFACE_NAME_FOR_S11_MME    = "eth1:11";                #
YOUR NETWORK CONFIG HERE
      MME_IPV4_ADDRESS_FOR_S11_MME      = "192.11.12.81/24";        #
YOUR NETWORK CONFIG HERE
      MME_PORT_FOR_S11_MME              = 2123;                     #
YOUR NETWORK CONFIG HERE
 };
S-GW :
{
    # S-GW binded interface for S11 communication (GTPV2-C), if none
selected the ITTI message interface is used
    SGW_IPV4_ADDRESS_FOR_S11              = "192.21.12.81/24";       #
YOUR NETWORK CONFIG HERE
};
```

3.2 Run MME

```
$ ~/openair-cn/SCRIPTS$ ./run_mme -i
```

3. Configuring and Building SPGW

3.1 Check the following fields in your /usr/local/etc/oai/spgw.conf

```
S-GW :
{
    NETWORK_INTERFACES :
    {
        # S-GW binded interface for S11 communication (GTPV2-C), if
none selected the ITTI message interface is used
        SGW_INTERFACE_NAME_FOR_S11              = "eth1:21";
# STRING, interface name, YOUR NETWORK CONFIG HERE
        SGW_IPV4_ADDRESS_FOR_S11                = "192.21.12.81/24";
# STRING, CIDR, YOUR NETWORK CONFIG HERE

        # S-GW binded interface for S1-U communication (GTPV1-U) can be
```

```
ethernet interface, virtual ethernet interface, we don't advise
wireless interfaces
        SGW_INTERFACE_NAME_FOR_S1U_S12_S4_UP    = "eth1";
# STRING, interface name, YOUR NETWORK CONFIG HERE, USE "lo" if S-GW
run on eNB host
        SGW_IPV4_ADDRESS_FOR_S1U_S12_S4_UP      = "192.168.12.81/24";
# STRING, CIDR, YOUR NETWORK CONFIG HERE
        SGW_IPV4_PORT_FOR_S1U_S12_S4_UP         = 2152;
# INTEGER, port number, PREFER NOT CHANGE UNLESS YOU KNOW WHAT YOU ARE
DOING

        # S-GW binded interface for S5 or S8 communication, not
implemented, so leave it to none
        SGW_INTERFACE_NAME_FOR_S5_S8_UP         = "none";
# STRING, interface name, DO NOT CHANGE (NOT IMPLEMENTED YET)
        SGW_IPV4_ADDRESS_FOR_S5_S8_UP           = "0.0.0.0/24";
# STRING, CIDR, DO NOT CHANGE (NOT IMPLEMENTED YET)
    };
}
P-GW =
{
    NETWORK_INTERFACES :
    {
        # P-GW binded interface for S5 or S8 communication, not
implemented, so leave it to none
        PGW_INTERFACE_NAME_FOR_S5_S8          = "none";
# STRING, interface name, DO NOT CHANGE (NOT IMPLEMENTED YET)

        # P-GW binded interface for SGI (egress/ingress internet
traffic)
        PGW_INTERFACE_NAME_FOR_SGI            = "eth1";
# STRING, YOUR NETWORK CONFIG HERE
        PGW_MASQUERADE_SGI                    = "yes";
# STRING, {"yes", "no"}. YOUR NETWORK CONFIG HERE, will do NAT for you
if you put "yes".
        UE_TCP_MSS_CLAMPING                   = "no";
# STRING, {"yes", "no"}.
    };

    # Pool of UE assigned IP addresses
    # Do not make IP pools overlap
    # first IPv4 address X.Y.Z.1 is reserved for GTP network device on
SPGW
    # Normally no more than 16 pools allowed, but since recent GTP
kernel module use, only one pool allowed (TODO).
    IP_ADDRESS_POOL :
    {
        IPV4_LIST = (
                      "172.16.0.0/12"
# STRING, CIDR, YOUR NETWORK CONFIG HERE.
                    );
    };

    # DNS address communicated to UEs
    DEFAULT_DNS_IPV4_ADDRESS     = "192.168.12.100";
```

```
# YOUR NETWORK CONFIG HERE
    DEFAULT_DNS_SEC_IPV4_ADDRESS = "8.8.4.4";
# YOUR NETWORK CONFIG HERE

    # Non standard feature, normally should be set to "no", but you may
need to set to yes for UE that do not explicitly request a PDN address
through NAS signalling
    FORCE_PUSH_PROTOCOL_CONFIGURATION_OPTIONS = "no";
# STRING, {"yes", "no"}.
    UE_MTU                                  = 1500
# INTEGER
};
```

3.2 Run SPGW

```
$ ~/openair-cn/SCRIPTS$ ./run_spgw -i
```

Anexo H. Instalación de OAI para la RAN

Tutorial tomado del GitLab de OpenAirInterface project, buscar en https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/home

OAISIM Setup

1. Configuring OAISIM

Modify ~/OPENAIR_DIR/target/PROJECTS/GENERIC-LTE-EPC/CONF/enb.band7.generic.oaisim.local_mme.conf and make sure the following fields are matching what you need.

```
//It should be like this for oaisim sim card
 tracking_area_code  =  "1";
 mobile_country_code =  "208";
 mobile_network_code =  "93";

 mme_ip_address      = ( { ipv4       = "192.168.12.81"; //IP of MME
                           ipv6       = "192:168:30::17";
                           active     = "yes";
                           preference = "ipv4";
                         }
                       );

 NETWORK_INTERFACES :
 {
     ENB_INTERFACE_NAME_FOR_S1_MME            = "eth0";
     //The local interface goes to MME
     ENB_IPV4_ADDRESS_FOR_S1_MME              = "192.168.12.117/24";
     ENB_INTERFACE_NAME_FOR_S1U               = "eth0";
     //The local interface goes to SGW
     ENB_IPV4_ADDRESS_FOR_S1U                 = "192.168.12.117/24";
     ENB_PORT_FOR_S1U                         = 2152; # Spec 2152
```

```
};
```

2. Compiling and Running OAISIM

```
cd ~/OPENAIR_DIR/cmake_targets
./build_oai -x -c --oaisim --UE
cd ~/OPENAIR_DIR/cmake_targets/tools/
sudo -E ./run_enb_ue_virt_s1
```

3. Setting Routing
In order to get the packets route back to UE, you also need to set the routing (for ubuntu edit /etc/network/interfaces)

```
ip rule add from 172.16.0.2/32 table 200
ip rule add to 172.16.0.2/32 table 200
```

4. Test
You can try to ping through oip1 interface, for example:

```
$ ping google.com -m 1 -I oip1
```

Anexo I. Instalación requerimientos previos OAI

Tutorial tomado del GitLab de OpenAirInterface project, buscar en https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/home

Ubuntu 14.04.3 LTS/Linux Kernel version 3.19 for OAISIM

Install a standard 64-bit 14.04.3 system. Note: the latest version from Feb. 2016, 14.04.04 comes with kernel 4.2 by default which will not work with OAI (master branch).

Low-latency kernel installation
To install the low-latency kernel 3.19 on top of Ubuntu 14.04, 64-bit architecture type
Example kernel tested on our side for openairinterface5g installation is linux-image-3.19.0-61-lowlatency

```
$ sudo apt-get install linux-image-3.19.0-61-lowlatency linux-headers-
3.19.0-61-lowlatency
```

After a reboot, `$ uname -a` should give the following output:
`Linux [NAME] 3.19-lowlatency #201408132253 SMP PREEMPT Thu Aug 14 03:01:44 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux`

Kernel Requirements for Core Network
Notes for installing 4.7.x kernel from source with GTP module (TESTED)

```
$ sudo apt-get install xz-utils build-essential wget libncurses5-dev
libssl-dev
$ sudo apt-get build-dep linux-image-$(uname -r) ncurses-bin
$ wget https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.7.1.tar.xz
$ unxz linux-4.7.1.tar.xz
$ tar -xovf linux-4.7.1.tar
$ cd linux-4.7.1
# See Documentation/Changes for list of required minimum software to
compile #the kernel
$ make menuonfig #search for GTP option and enable it as a kernel
module and #save the kernel config file
# look for CONFIG_GTP in .config and make sure it is enabled as a
kernel module
$ CONFIG_GTP=m
$ make -j`nproc`
$ sudo make modules_install
$ sudo make install
```

The OpenAirInterface repository

The OpenAirInterface software can be obtained from our gitLab server. You will need a git client to get the sources. The repository is currently used for main developments.

Prerequisite

You need to install the subversion/git using the following commands:

```
$ sudo apt-get update
$ sudo apt-get install subversion git
```

Using EURECOM Gitlab

The openairinterface5g repository holds the source code for (eNB RAN + UE RAN).
For legal issues (licenses), the core network (EPC) source code is now moved away from the above openairinterface5g git repository. This EPC code is now splitted into two git projects (openair-cn with apache license and xtables-addons-oai with GPL license).
Configure git with your name/email address (only important if you are developer and want to checkin code to Git):

```
$ git config --global user.name "Your Name"
$ git config --global user.email "Your email address"
```

Add a certificate from gitlab.eurecom.fr to your Ubuntu 14.04 installation:

```
echo -n | openssl s_client -showcerts -connect gitlab.eurecom.fr:443
2>/dev/null | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' |
sudo tee -a /etc/ssl/certs/ca-certificates.crt
```

Disable certificate check completely if you do not have root access to /etc/ssl directory

```
git config --global http.sslverify false
```

In order to checkout the Git repository (for OAI Users without login to gitlab server) Checkout RAN repository (eNB RAN + UE RAN):

```
$ git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git
$ git clone https://gitlab.eurecom.fr/oai/openair-cn.git
$ git clone https://gitlab.eurecom.fr/oai/xtables-addons-oai.git
```

Anexo J. Pasos para instalación manual de OAI en Openstack

1. En la barra lateral Proyecto>Computo>Instancias se deben lanzar dos instancias (una para la RAN que incluirá eNB+UE y otra que será el EPC que incluirá MME+S/P-GW+HSS) con las siguientes características.
    a. Origen de Arranque > Elegir Imagen > Seleccionar Xenial para el EPC y Trusty para la RAN.
    b. Sabor Medium como mínimo.
    c. Red conectada a flat-lan-1-net (mínimo deben estar en al menos una subred que contenga la VM "ctl" de Openstack para enviar los comandos, además entre EPC y RAN).
    d. Asociarlo a un Par de Keys (ver Anexo F para una guía de como crear las llaves).
2. Instalar los requerimientos previos en ambas VMs detallados en el Anexo I.
3. Seguir los tutoriales de instalación de OAI del Anexo G e I para el EPC y RAN respectivamente.

Caso Roaming

En la Ilustración 2, se representa el caso que un UE pide acceso en una PLMN de otra compañía, por tanto la red visitada (VPLMN) puede hacer las consultas a la compañía original del usuario (HPLMN) con otros protocolos mostrados en las conexiones entre S-Gateway y P-Gateway, también entre el HSS y MME de los que hablaremos más adelante. También, pueden darse los casos también en que solo que la VPLMN haga uso de su propio P-Gateway y red IP.

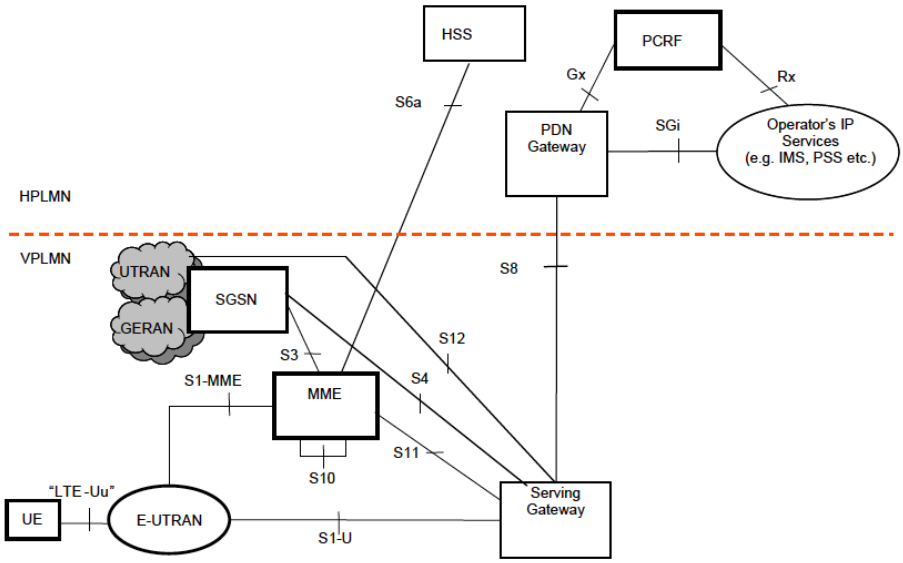


Ilustración 2: Arquitectura en caso de Roaming

Non Access Stratum (NAS)

El NAS es un conjunto de protocolos cuya función principal es soportar la movilidad del UE, y sostener los procedimientos que administran sesiones para establecer y mantener la conectividad IP entre UE y el PDN-Gateway. También existen los protocolos NAS de Seguridad que agrega funciones adicionales al NAS como tal, para integrar la protección y el cifrado de la propia señalización NAS.
En general Non Access Stratum es muy similar a UMTS, siendo el principal cambio la libertad del EPS para concatenar procedimientos, para establecer conexiones y portadoras más rápidas.
Algunas de sus características principales del protocolo NAS son:

- Cuando un UE se activa y acceda a una PLMN, el MME crea un listado contextual (lista de portadores y capacidades del terminal, entre otros), el cual es descargado desde la base de clientes del HSS, logrando mayor rapidez de ejecución.

- Para reducir el overhead en E-UTRAN en periodos de larga inactividad (ECM-IDLE), primero elimina la información relacionada al UE y esta es retenida en el MME. Durante el ECM-IDLE el UE avisa su locación cuando se mueve fuera del Tracking Area (TA), el encargado de mantener la ubicación del UE, es el MME.
- Cuando hay paquetes activos en la ruta Downlink, el MME busca el UE objetivo en todos los eNodeB del Área de Busqueda (Track Area TA). Posteriormente, al conectarse el UE este inicia Service Request que resulta en el estado "ECM-CONNECTED", y de este modo genera la información relacionada al UE y sus portadores (con su restablecimiento) en el E-UTRAN. Para mejorar la velocidad de este procedimiento, se permitió la concatenación del NAS al Access Stratum (AS) para activación de portadores (algunos procedimientos fueron a propósito hechos para funcionar en simultáneo por razones de velocidad).
- La autentificación y seguridad de los datos de usuario y señalización, esto se realiza entre UE y MME/HSS con el uso de llaves de seguridad y encriptación de los portadores.

Red de Acceso E-UTRAN

La RAN o Red de Acceso por Radio de LTE es E-UTRAN, la cual consiste en una red de varios eNodeB como se aprecia en la ilustración 3. Su arquitectura es plana, ya que no tiene un controlador central y todos los nodos se conectan entre sí mediante la interface X2. La conexión hacia el EPC, se realiza mediante la interfaz S1-MME, donde van todos los mensajes de control del MME. Mientras que por la interface S1-U, pasaron los datos de usuario (o información a transmitir) hasta el S-GW, quien los direcciona para salir hacia su destino. El interfaz hacia el UE, es la LTE-Uu con protocolo de comunicación AS.

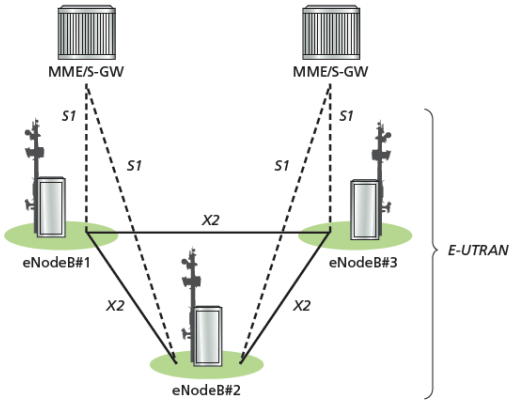


Ilustración 3: Arquitectura E-UTRAN[1]

Funciones de Radio

- La Radio Resources Management (RRM), cubre todas las funciones relacionadas a las portadoras de radio, siendo algunas encargadas del control de las portadoras de radio, del control de su admisión, del control de movilidad y programación, y localización dinámica de los recursos para el UE, tanto para Uplink como para Downlink.

- Compresión de las cabeceras de paquetes IP en la interfaz LTE-Uu para su uso eficiente, sobre todo para intercambio de aplicaciones de voz (VoIP) que corresponden a una gran cantidad de paquetes con pocos datos de usuario, comparado con los datos de control en las cabeceras.
- Seguridad y encriptación de todos los datos que van por radio.
- Conectividad al EPC, consiste en la señalización hacia el MME y los portadores al Serving Gateway.

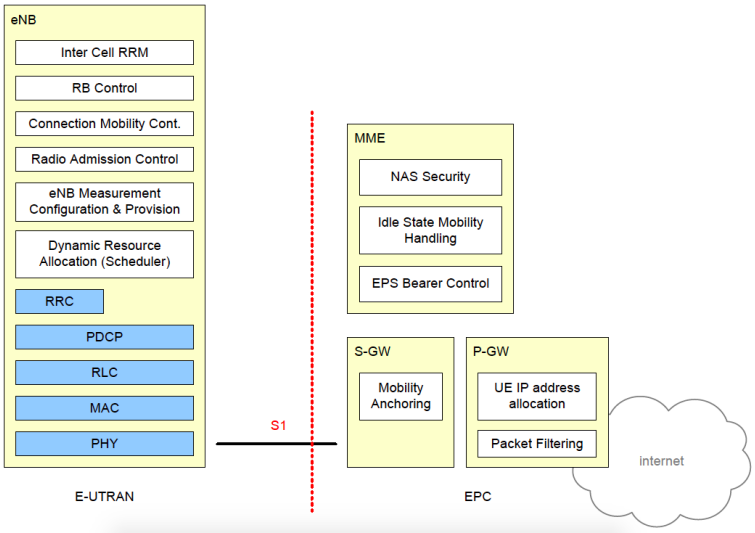


Ilustración 4: División de Funciones entre E-UTRAN y EPC

Se puede resumir las funciones en la ilustración 4, donde los cuadros amarillos dividen las unidades lógicas descritas anteriormente, los cuadros blancos representan funciones, y los azules protocolos de radio en estilo de capas del modelo OSI[1].

Protocolos

Plano de Usuario

Para poder hacer llegar los paquetes IP de datos al usuario estos se encapsulan en un protocolo GPRS Tunneling Protocol (GTP), donde son enrutados sobre este protocolo desde el PDN Gateway hasta el eNodeB, para que este último se lo transmita al UE. Las interfaces por las cuales avanza este protocolo, son las CN, y la EPC, específicamente las S1 y S5/S8.

En la Ilustración 5 se puede apreciar la pila de protocolos usados en el plano de usuario, en los cuadros azules corresponden al E-UTRAN, los cuales son Packet Data Coverage Protocol (PDCP), Radio Link Control (RLC), Medium Access Control (MAC), y la capa física elegida por el eNodeB dependiendo del sitio.

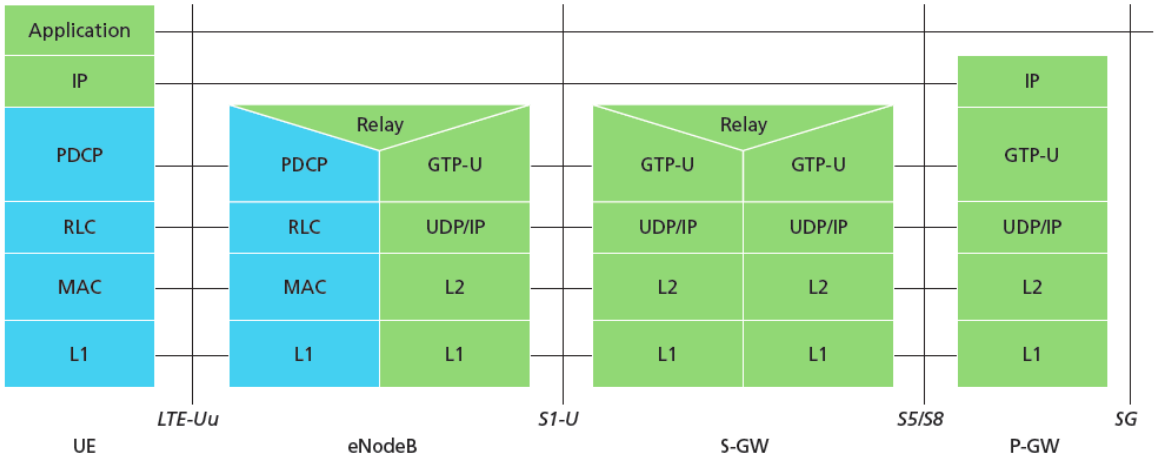


Ilustración 5: Pila de Protocolos para el Plano de Usuario

Para el manejo del Plano de Usuario durante casos como un handover (Se denomina handover o traspaso al sistema utilizado en comunicaciones móviles celulares con el objetivo de transferir el servicio de una estación base a otra cuando la calidad del enlace es insuficiente en una de las estaciones. Este mecanismo garantiza la realización del servicio cuando un móvil se traslada a lo largo de su zona de cobertura.)o cuando no se dispone de una unidad centralizada en E-UTRAN que realice el control, los nodos eNodeB deben encargarse de esta tarea por sí mismos. Esto lo realizan en la capa asociada al PDCP, en tanto la RLC y la MAC se renuevan al pasar el UE a una nueva celda celular.

Plano de Control

En la Ilustración 6, se puede apreciar la pila de protocolos para el plano de control, y en color azul para el E-UTRAN que utiliza el protocolo AS. Las capas bajas, son las mismas del plano de usuario, solo que en el caso de control no hay compresión de los encabezado los paquetes de datos IP.

Radio Resource Control (RRC), es la función principal en el protocolo AS, siendo responsable de establecer los portadores de radio y configurar las capas bajo esa, utilizando señalización RRC entre el eNodeB y UE.

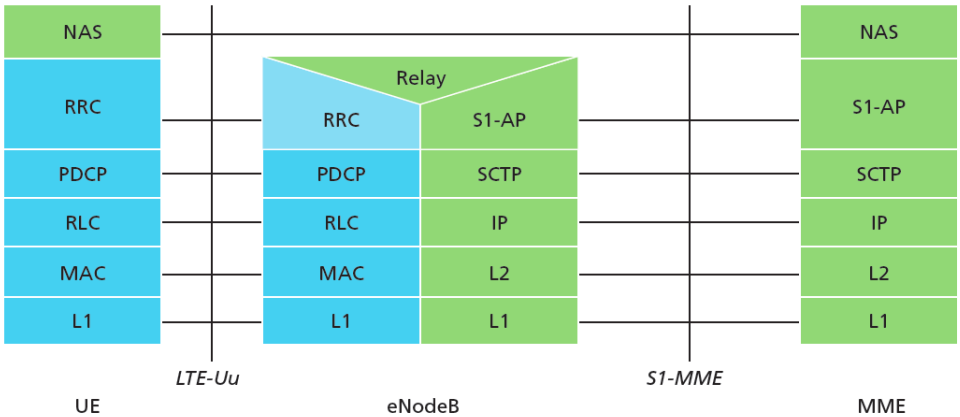


Ilustración 6: Pila de Protocolos del Plano de Control

QoS y Portadores EPS

El sistema EPS se encarga de la interconexión de los UE con las redes IP externas (Packet Data Network), soportando los flujos de tráfico agregado, los cual consisten en uno o más Service Data Flow[1] (SDF). Estos servicios tienen distintos requerimientos de QoS, dependiendo de su naturaleza. Por ejemplo, VoIP por ser un servicio de transmisión de voz, éste debe enviar una gran cantidad de paquetes de pocos Bytes cada uno en comparación con los grandes archivos (Mb o Gb) descargados a través de los protocolos FTP desde los buscadores Web, que son paquetes grandes que requieren fidelidad en la transmisión. En ambos casos se asignan distintas clases de Calidad de Servicio, o distintas Colas (Queue)segun el tipo de tráfico requerido. El primer caso (VoIP), requiere valores de Delay y Jitter bajos para obtener una transmisión de voz efectiva. En el segundo caso (transferencia de un archivo via FTP), y dado que es un tráfico solo de datos, se requiere sólo de una baja tasa de perdida de paquetes.
En la parte de acceso (RAN), los eNodeB puede tener conectividad a una PDN, circulando por el EPC sobre un portador EPS en el que se configura un camino o "túnel" con el protocolo GPRS Tunneling Protocol (GTP). Incluso si este resulta ser un paquete de datos IP, este quedará incluido sobre el protocolo GTP.
En líneas generales, se puede dividir los portadores en dos grandes categorías, basado en la naturaleza de su aplicación y la QoS o cola que requieren:

- Portadores GBR. Son los cuales tienen asignado un GBR, a los cuales se les designa recursos de transmisión de manera permanente al establecerse o modificarse. Se les puede dedicar mayores tasas de transmisión si estas están disponibles, en cuyos casos se puede designar un parámetro de Maximum Bit Rate (MBR) para poner una cota máxima a la tasa de transmisión del portador GBR.
- Portadores Non-GBR. Son los que no tienen garantizado un piso mínimo de tasa de transmisión. Estos son utilizados en aplicación con QoS o Colas que no requieren mayormente de estas características, como descargas FTP. Para estos portadores no se designa un ancho de banda permanente.

La encargada de entregar los requisitos de la clase QoS, o la Clase de Servicios, es el eNodeB. Cada portador tiene asociado un QoS Class Identifier (QCI) y un Allocation and Retention Priority (ARP).
Como se puede apreciar en la Tabla 1, cada QCI se puede caracterizar por prioridad, delay presupuestado y tasa de pérdida tolerada. Este determinará como será tratado el paquete de datos IP en el eNodeB. Se han estandarizado una docena de estos, para que los distintos vendors de equipamiento puedan tener el mismo entendimiento de las características del servicio que los soportarán para sus aplicaciones, y así así sea consecuente su tratamiento, administración, condiciones y estrategia de cobro.

QCI	RESOURCE TYPE	PRIORITY	PACKET DELAY BUDGET (MS)	PACKET ERROR LOSS RATE	EXAMPLE SERVICES
1	GBR	2	100	10^{-2}	Conversational voice
2	GBR	4	150	10^{-3}	Conversational video (live streaming)
3	GBR	5	300	10^{-6}	Non-conversational video (buffered streaming)
4	GBR	3	50	10^{-3}	Real-time gaming
5	Non-GBR	1	100	10^{-6}	IMS signaling
6	Non-GBR	7	100	10^{-3}	Voice, video (live streaming), interactive gamin
7	Non-GBR	6	300	10^{-6}	Video (buffered streaming)
8	Non-GBR	8	300	10^{-6}	TCP-based (for example, WWW, e-mail), chat, F p2p file sharing, progressive video and others
9	Non-GBR	9	300	10^{-6}	

Tabla 1: Estándar de QCI para LTE

El QCI del portador, definirá el modo de configuración de la RLC y como el programador de la interface MAC manejará los paquetes que son enviados por este portador (como por ejemplo la organización de procesamiento, manejo de colas y organización de la tasa de traffic shaping[1]).
El protocolo ARP es utilizado para el control de llamada de admisión, como por ejemplo decidir cuándo se requiere establecer un portador frente a una congestión de tráfico. También, gobierna la priorización por ejemplo, si un portador establecido es desechado por el requerimiento de establecer uno nuevo. Una vez establecido un portador, el ARP no tiene influencia en cómo se maneja el portador.

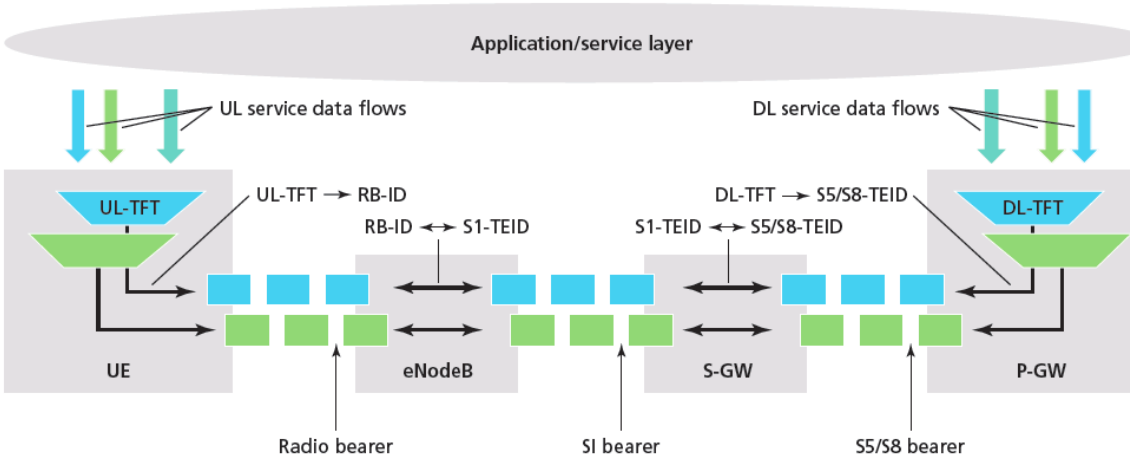


Ilustración 7: Portadores EPS a través de distintas Interfaces

Los portadores de la EPS, deben circular por varias interfaces entre los nodos de su sistema, cada una con identidad propia. Por tanto, se requiere que cada nodo pueda llevar un seguimiento de cada portador entre sus puertas de entrada y salida, esto lo hace mediante un mapeo del Id de los portadores en los túneles GTP, como se aprecia en la ilustración 7.

Los paquetes de DL y UL, son filtrados en sus distintos portadores según plantillas TFT que utilizan información de las cabeceras del paquete TCP/IP, para determinar la naturaleza del tráfico de origen destino y tipo de aplicación, entre otros.

Como parte de procesamiento del inicio de anclaje de un UE a la PDN, a éste se le asigna una dirección IP por el P-GW y al menos un portador Non-GBR. Ésta durará por todo el tiempo que el UE esté anclado a la red. Adicionalmente, las características propias de ésta, son tomadas en base a la información asociada del UE en el HSS.

Anexo L. Template para desplegar OAI sobre OpenStack

```
description: >
  Heat Orchestration Template (HOT) to deploy OpenAir Core Network vEPC.
  Networks, security groups, volumes, ports and instances get created by
heat,
  lastest openair core network code is compiled on the build instance and
  deployed to the SPGW/MME/HSS instances. Once the stack is created, it
  provides the SSH commandline to access the vEPC.

### STACK INPUTS
parameters:
  key:
    type: string
    label: SSH Keypair
    description: Name of the SSH keypair for logging in into instances
    constraints:
    - custom_constraint: nova.keypair
    default: "OAI-Admin"
  image:
    type: string
    label: Ubuntu Image
    description: Name of the Ubuntu image (needs os-*-config installed)
    constraints:
    - custom_constraint: glance.image
    default: "ubuntu-trusty-os-config"
  extnet:
    type: string
    label: External Network
    description: Name of the external network containing 2 free floating
IPs
    default: "Lab3-Internet"
  base_url:
    type: string
    label: Base URL
    description: Base URL to fetch kernel config and database dump from
    default: "https://inostack.vptt.ch"
  run_flavor:
    type: string
```

```
    label: Run Flavor
    description: Flavor to use for normal instances
    constraints:
    - custom_constraint: nova.flavor
    default: "m1.tiny"
  build_flavor:
    type: string
    label: Build Flavor
    description: Flavor to use for the building instance
    constraints:
    - custom_constraint: nova.flavor
    default: "m1.medium"
  run_vol_size:
    type: number
    label: Run Volume Size
    description: Volume size in GB to use for normal instances
    default: 5
    constraints:
    - range: { min: 5, max: 10 }
  build_vol_size:
    type: number
    label: Build Volume Size
    description: Volume size in GB to use for the building instance
    default: 20
    constraints:
    - range: { min: 20, max: 50 }
  hss_name:
    type: string
    label: HSS Hostname
    description: Hostname of the HSS instance
    constraints:
    - length: { min: 3, max: 10 }
      description: Hostnameg name must be between 3 and 10 characters
    - allowed_pattern: "[a-z0-9]*"
      description: Hostname contains only lowercase characters and numbers
    default: hss
  mme_name:
    type: string
    label: MME Hostname
    description: Hostname of the MME instance
    constraints:
    - length: { min: 3, max: 10 }
      description: Hostname name must be between 3 and 10 characters
    - allowed_pattern: "[a-z0-9]*"
      description: Hostname contains only lowercase characters and numbers
    default: mme
  spgw_name:
    type: string
    label: SPGW Hostname
    description: Hostname of the SPGW instance
    constraints:
    - length: { min: 3, max: 10 }
      description: Hostname name must be between 3 and 10 characters
    - allowed_pattern: "[a-z0-9]*"
      description: Hostname contains only lowercase characters and numbers
    default: spgw
  build_name:
```

```
    type: string
    label: Build Hostname
    description: Hostname of the build instance
    constraints:
    - length: { min: 3, max: 10 }
      description: Hostname name must be between 3 and 10 characters
    - allowed_pattern: "[a-z0-9]*"
      description: Hostname contains only lowercase characters and numbers
    default: build
  dns1:
    type: string
    label: Internal Network DNS 1
    description: Upstream DNS server 1
    default: "8.8.8.8"
  dns2:
    type: string
    label: Internal Network DNS 2
    description: Upstream DNS server 2
    default: "8.8.4.4"
  int_cidr:
    type: string
    label: Internal Network CIDR
    description: Internal network IPv4 Adressing in CIDR notation
    default: 172.16.0.0/24
  enb_cidr:
    type: string
    label: eNB (VPN) CIDR
    description: eNB (VPN) network IPv4 Adressing in CIDR notation
    default: 172.31.0.0/24
  realm:
    type: string
    label: Realm
    description: Realm (depends on database and hostnames used)
    default: inostack
  enb_count:
    type: number
    label: eNB Count
    description: Number of eNBs to support (power of 2)
    default: 4
    constraints:
    - allowed_values:
      - 2
      - 4
      - 8
      - 16
  enb_start:
    type: number
    label: eNB Start Address
    description: Start address of eNBs in eNB VPN network
    default: 10
    constraints:
      - range: { min: 10, max: 230 }
  enb_hostname_prefix:
    type: string
    label: Hostname Prefix for eNBs
    description: Hostname prefix for eNBs
    constraints:
```

```
    - length: { min: 3, max: 9 }
      description: Hostname prefix must be between 3 and 9 characters
    - allowed_pattern: "[a-z]*"
      description: Hostname prefix contains only lowercase characters
    default: enb
  spgw_kernel_version:
    type: string
    label: SPWG Kernel Version
    description: Kernel version to use for SPGW instance (>=4.7)
    default: 4.7.7
  ue_dns:
    type: string
    label: UE DNS
    description: DNS pushed to UEs
    default: 8.8.8.8
  ue_cidr:
    type: string
    label: UE CIDR
    description: UE address pool in CIDR notation
    default: 10.10.10.0/24
  ue_mtu:
    type: number
    label: UE MTU
    description: MTU pushed to UEs
    default: 1500
    constraints:
    - range: { min: 1000, max: 8000 }
  db_file:
    type: string
    label: HSS DB File
    description: SQL dump file for HSS database to download from base_url
    default: oai_db_swisscom_ino.sql
  db_pass:
    type: string
    label: HSS DB Password
    description: Password for the HSS database (user hssadmin)
    hidden: true
    default: change_me
  operator_key:
    type: string
    label: Operator Key
    description: Operator Key to use (depends on DB)
    default: "11111111111111111111111111111111"
    constraints:
    - length: { min: 32, max: 32 }
      description: Operator key is 32 digits
    - allowed_pattern: "[0-9]*"
      description: Operator key is 32 digits
  mcc:
    type: number
    label: MCC
    description: MCC to use (depends on DB)
    default: 228
  mnc:
    type: number
    label: MNC
    description: MNC to use (depends on DB)
```

```
    default: 88
  mme_gid:
    type: number
    label: MME_GID
    description: MME_GID to use (depends on DB)
    default: 32768
  mme_code:
    type: number
    label: MME_CODE
    description: MME_CODE to use (depends on DB)
    default: 1
  tac:
    type: number
    label: TAC
    description: TAC to use (depends on DB)
    default: 1
  ca_country:
    type: string
    label: CA Country
    description: Country for Certificates
    default: CH
  ca_state:
    type: string
    label: CA State
    description: State for Certificates
    default: Bern
  ca_city:
    type: string
    label: CA City
    description: City for Certificates
    default: Bern
  ca_company:
    type: string
    label: CA Company
    description: Company for Certificates
    default: Swisscom
  ca_unit:
    type: string
    label: CA Unit
    description: Organisational unit for Certificates
    default: INO
  ca_email:
    type: string
    label: CA Email
    description: Email Address for Certificates
    default: Daniel.Balsiger@swisscom.com
  ssh_pub:
    type: string
    label: SSH Public Key
    description: SSH public key for instance interconnect
    default:                                                    "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQC6WwxDHQKgvVTqzC2S4+apYFmVTTI3N1+kTnOJZ5+K/P
o9D3Uq89LY2hdRUNcBvOIDW1lGJHLoodyR/W6UaLvT1eTG1le72yCpcliq/nuDc6eUex2Mqz3C
MBqWhg3L+21yme0vlT3w0S+uhkiq8s95OMCYMc60bSTHN/RqLB1o8dsLAqix6W5lDxcwVpl8V7
viBTvZBUSqTHKA0AXJ4uQDdBqHZY9iUcyyKVzcEaOnd4RdAJPAD+au3dKlpfEdKQjxHTV41QM+
VkuImQUMe5MQ7gU4DbHj7YSA3fJhki3jwFHMzFNbYulUgwBdNCFVTcGdQEUib6dXW8LOH59FmU
ut"
```

```
  ssh_priv:
    type: string
    label: SSH Private Key
    description: SSH private key for instance interconnect
    hidden: true
    default: >
      -----BEGIN RSA PRIVATE KEY-----
      MIIEowIBAAKCAQEAulsMQx0CoL1U6swtkuPmqWBZlU0yNzdfpE5ziWefivz6PQ91
      KvPS2NoXUVDXAbziA1tZRiRy6KHckf1ulGi709XkxtZXu9sgqXJYqv57g3OnlHsd
      jKs9wjAaloYNy/ttcpntL5U98NEvroZIqvLPeTjAmDHOtG0kxzf0aiwdaPHbCwKo
      seluZQ8XMFaZfFe74gU72QVEqkxygNAFyeLkA3Qah2WPYlHMsilc3BGjp3eEXQCT
      wA/mrt3SpaXxHSkI8R01eNUDPlZLiJkFDHuTEO4FOA2x4+2EgN3yYZIt48BRzMxT
      W2LpVIMAXTQhVU3BnUBFIm+nV1vCzh+fRZlLrQIDAQABAoIBACnu8OxtK7k1wVTw
      StUB2VaFqsLQ0xrfp/LWAGOL4LeqwzhYMRpQMULAmHygvzDR6t2sgYMqEn1MZtCn
      AWn9wz4gpFElzComXcwjQdaAWxSyQqRDq9uKcOQwZNs4IQSkd/VQs7GAWKbGu37/
      Enz9FDiHz7avhn7NDHiTm7kEYj3JxBCw9QnpuljFu7BZlg1S4eK/rGz7in1hLo62
      gDfGcDgUgmlJ1aG1rYP2bETsXe0necHDOvQe6rlmdTsX89y+q4QxqYGEGdjCoFZZ
      F0mEPbS9//9GS+rER5tuSi3bD3k73zg1+e8lHAjlAR3hNmT3JKOEHB0kmTAJyrpL
      +Gl9WIECgYEA4ekwnl3wT2p2eqmhqC+MoMcUpOtAaoA2ShG3CHcp/U6VcWIMBe7o
      KMaJcfLp3zerAeyzRfR7t1E83alIQb9cRRd1pbhLjM60d+n6HwT0//4SBiShfQ/m
      K7Ch1zOaB4h8O8qWDT9Uz6b0AMJA9S6BZ2We3iOT5ysS6MPOwgQAbCECgYEA0y0n
      4zgq1YFE+56xXAt8/BKqv9nbtNfKganXTXMah+oz546CVUJDPL3z99Zp1w3rUvlJ
      JH2wbnqvcPCuCvFdCPXpowkN5EWUTUrjeYohDXDZPaJ1pYZXeMKFyVgzZFeP+UKK
      82gxY5O+kC1I0b2J2u0gxm74huSgBIRx54ZqDg0CgYEAhQsT+vBPyjVkuTCVZ7s5
      Eqar3cQ+F3qSfmSYan/jVq6lDDU153ifeQQThewNF8xtBaEkoxoskfVh5xj+2Nmd
      uYLrYkF7HN3PIp/FEeeVcf1rF/sSr9hhMXHAnkBhgfY7U+snG34ksHYeVSQRpVNS
      GlaajTBetlGDvVkztscsiIECgYBTd0KPtVCAyLIqPaPePJAu1XX1lDcZeD0LGMUH
      UJpI5BGV0SbEagdHR9DYwT9eB5teVTdKm/8S+5zCJ+6yVomuE/w/O0HpWnLuRc44
      6JZ9yH+ks8SKItoJ2eClHx5Y5575Jwrif+kdcXTdaXihpaeKBzVwGMZUEqMIhgy7
      NM5QNQKBgAT80nwpu/KJasbPd/5vlnWKkk9wW1pJJJyGvyrHvQkHYSJmrGNhci+X
      geZreHUVqcR4VeRP14D71+zIT04r4jcio9deNBRYXdWuAhKVfvEh5iOXfuaBF24e
      m4y0viefFNyys5/BQI0kpCtJirvAtlsQg8ig+2ALnCk5rLNbgui4
      -----END RSA PRIVATE KEY-----

### ORDER OF INPUTS
parameter_groups:
- label: "Passwords and Keys"
  parameters:
  - db_pass
  - ssh_pub
  - ssh_priv
  - key
- label: "Cloud Networks, Flavors, Images and Volumes"
  parameters:
  - extnet
  - int_cidr
  - dns1
  - dns2
  - image
  - run_flavor
  - run_vol_size
  - build_flavor
  - build_vol_size
- label: "FQDNs, Realm, SPGW kernel, HSS Database and EPC Configuration"
  parameters:
  - base_url
  - spgw_kernel_version
```

```
  - realm
  - spgw_name
  - mme_name
  - hss_name
  - build_name
  - db_file
  - operator_key
  - mcc
  - mnc
  - mme_gid
  - mme_code
  - tac
- label: "eNB and UE Settings"
  parameters:
  - ue_cidr
  - ue_dns
  - ue_mtu
  - enb_cidr
  - enb_count
  - enb_start
  - enb_hostname_prefix
- label: "Certificate Settings"
  parameters:
  - ca_country
  - ca_state
  - ca_city
  - ca_company
  - ca_unit
  - ca_email

### STACK RESOURCES
resources:
### NETWORKS & ROUTERS
  internal_net:
    type: OS::Neutron::Net
    properties:
      name: OAI-InternalNet

  internal_subnet:
    type: OS::Neutron::Subnet
    depends_on: internal_net
    properties:
      name: OAI-InternalSubnet
      ip_version: 4
      network_id: { get_resource: internal_net }
      dns_nameservers:
        - { get_param: dns1 }
        - { get_param: dns2 }
      cidr: { get_param: int_cidr }

  router:
    type: OS::Neutron::Router
    properties:
      name: OAI-Router

  router_interface:
    type: OS::Neutron::RouterInterface
```

```
    depends_on: [ internal_subnet, internal_net, router ]
    properties:
      subnet: { get_resource: internal_subnet }
      router: { get_resource: router }

  router_gateway:
    type: OS::Neutron::RouterGateway
    depends_on: router
    properties:
      network: { get_param: extnet }
      router_id: { get_resource: router }

  floating_ip:
    type: OS::Neutron::FloatingIP
    depends_on: [ internal_subnet, spgw_port, router_interface ]
    properties:
      floating_network: { get_param: extnet }
      port_id: { get_resource: spgw_port }

### VOLUMES
  build_vol:
    type: OS::Cinder::Volume
    properties:
      name: { get_param: build_name }
      size: { get_param: build_vol_size }
      image: { get_param: image }

  hss_vol:
    type: OS::Cinder::Volume
    properties:
      name: { get_param: hss_name }
      size: { get_param: run_vol_size }
      image: { get_param: image }

  mme_vol:
    type: OS::Cinder::Volume
    properties:
      name: { get_param: mme_name }
      size: { get_param: run_vol_size }
      image: { get_param: image }

  spgw_vol:
    type: OS::Cinder::Volume
    properties:
      name: { get_param: spgw_name }
      size: { get_param: run_vol_size }
      image: { get_param: image }

### PORTS
  build_port:
    type: OS::Neutron::Port
    depends_on: [ build_secgroup, internal_subnet ]
    properties:
      network_id: { get_resource: internal_net }
      fixed_ips:
        - subnet_id: { get_resource: internal_subnet }
      security_groups: [{ get_resource: build_secgroup }]
```

```
  spgw_port:
    type: OS::Neutron::Port
    depends_on: [ spgw_secgroup, internal_subnet ]
    properties:
      network_id: { get_resource: internal_net }
      fixed_ips:
        - subnet_id: { get_resource: internal_subnet }
      security_groups: [{ get_resource: spgw_secgroup }]

  hss_port:
    type: OS::Neutron::Port
    depends_on: [ hss_secgroup, internal_subnet ]
    properties:
      network_id: { get_resource: internal_net }
      fixed_ips:
        - subnet_id: { get_resource: internal_subnet }
      security_groups: [{ get_resource: hss_secgroup }]

  mme_port:
    type: OS::Neutron::Port
    depends_on: [ mme_secgroup, internal_subnet ]
    properties:
      network_id: { get_resource: internal_net }
      fixed_ips:
        - subnet_id: { get_resource: internal_subnet }
      security_groups: [{ get_resource: mme_secgroup }]

### SECURITY GROUPS
  spgw_secgroup:
    type: OS::Neutron::SecurityGroup
    properties:
      description: Add security group rules for SPGW instance
      name: { get_param: spgw_name }
      rules:
        - remote_ip_prefix: 0.0.0.0/0

  build_secgroup:
    type: OS::Neutron::SecurityGroup
    properties:
      description: Add security group rules for build instance
      name: { get_param: build_name }
      rules:
        - remote_ip_prefix: 0.0.0.0/0
          protocol: tcp
          port_range_min: 22
          port_range_max: 22
        - remote_ip_prefix: 0.0.0.0/0
          protocol: icmp

  hss_secgroup:
    type: OS::Neutron::SecurityGroup
    properties:
      description: Add security group rules for HSS instance
      name: { get_param: hss_name }
      rules:
        - remote_ip_prefix: 0.0.0.0/0
```

```
          protocol: tcp
          port_range_min: 22
          port_range_max: 22
        - remote_ip_prefix: 0.0.0.0/0
          protocol: icmp
        - remote_ip_prefix: 0.0.0.0/0
          protocol: 132

  mme_secgroup:
    type: OS::Neutron::SecurityGroup
    properties:
      description: Add security group rules for MME instance
      name: { get_param: mme_name }
      rules:
        - remote_ip_prefix: 0.0.0.0/0
          protocol: tcp
          port_range_min: 22
          port_range_max: 22
        - remote_ip_prefix: 0.0.0.0/0
          protocol: icmp
        - remote_ip_prefix: 0.0.0.0/0
          protocol: 132

### INSTANCES

  build_vm:
    type: OS::Nova::Server
    depends_on: [ build_port, build_vol, router_interface ]
    properties:
      name: { get_param: build_name }
      flavor: { get_param: build_flavor }
      key_name: { get_param: key }
      block_device_mapping:   [{   device_name:   "vda",   volume_id   :   {
get_resource : build_vol }, delete_on_termination : "true" }]
      networks:
        - port: { get_resource: build_port }
      user_data_format: SOFTWARE_CONFIG
      user_data: |
        #!/bin/bash
        echo "195.176.209.235 opnfv.vptt.ch opnfv" >> /etc/hosts

  spgw_vm:
    type: OS::Nova::Server
    depends_on: [ spgw_port, spgw_vol, router_interface ]
    properties:
      name: { get_param: spgw_name }
      flavor: { get_param: run_flavor }
      key_name: { get_param: key }
      block_device_mapping:   [{   device_name:   "vda",   volume_id   :   {
get_resource : spgw_vol }, delete_on_termination : "true" }]
      networks:
        - port: { get_resource: spgw_port }
      user_data_format: SOFTWARE_CONFIG
      user_data: |
        #!/bin/bash
        echo "195.176.209.235 opnfv.vptt.ch opnfv" >> /etc/hosts
```

```
  hss_vm:
    type: OS::Nova::Server
    depends_on: [ hss_port, hss_vol, router_interface ]
    properties:
      name: { get_param: hss_name }
      flavor: { get_param: run_flavor }
      key_name: { get_param: key }
      block_device_mapping:   [{   device_name:   "vda",   volume_id   :   {
get_resource : hss_vol }, delete_on_termination : "true" }]
      networks:
      - port: { get_resource: hss_port }
      user_data_format: SOFTWARE_CONFIG
      user_data: |
        #!/bin/bash
        echo "195.176.209.235 opnfv.vptt.ch opnfv" >> /etc/hosts

  mme_vm:
    type: OS::Nova::Server
    depends_on: [ mme_port, mme_vol, router_interface ]
    properties:
      name: { get_param: mme_name }
      flavor: { get_param: run_flavor }
      key_name: { get_param: key }
      block_device_mapping:   [{   device_name:   "vda",   volume_id   :   {
get_resource : mme_vol }, delete_on_termination : "true" }]
      networks:
        - port: { get_resource: mme_port }
      user_data_format: SOFTWARE_CONFIG
      user_data: |
        #!/bin/bash
        echo "195.176.209.235 opnfv.vptt.ch opnfv" >> /etc/hosts

### SOFTWARE CONFIGURATIONS
  etc_hosts:
    type: OS::Heat::SoftwareConfig
    depends_on: [ hss_vm, mme_vm, spgw_vm, build_vm ]
    properties:
      group: script
      inputs:
      - name: realm
        default: { get_param: realm }
      - name: hss_name
        default: { get_param: hss_name }
      - name: mme_name
        default: { get_param: mme_name }
      - name: spgw_name
        default: { get_param: spgw_name }
      - name: build_name
        default: { get_param: build_name }
      - name: hss_ip
        default: { get_attr: [ hss_vm, first_address ] }
      - name: mme_ip
        default: { get_attr: [ mme_vm, first_address ] }
      - name: spgw_ip
        default: { get_attr: [ spgw_vm, first_address ] }
      - name: build_ip
        default: { get_attr: [ build_vm, first_address ] }
```

```
      - name: enb_cidr
        default: { get_param: enb_cidr }
      - name: enb_count
        default: { get_param: enb_count }
      - name: enb_start
        default: { get_param: enb_start }
      - name: enb_hostname_prefix
        default: { get_param: enb_hostname_prefix }
      config: |
        #!/bin/bash
        if [ ! -f /etc/hosts.orig ] ; then cp /etc/hosts /etc/hosts.orig ;
fi
        logger "$0: Creating /etc/hosts..."
        cat > /etc/hosts << __EOF
        # this file is generated by heat with os-*-config
        127.0.0.1 localhost

        $spgw_ip $spgw_name.$realm $spgw_name
        $mme_ip $mme_name.$realm $mme_name
        $hss_ip $hss_name.$realm $hss_name
        $spgw_ip $spgw_name.$realm $spgw_name
        $build_ip $build_name.$realm $build_name

        # this is since we have no DNS for opnfv.vptt.ch
        195.176.209.235 opnfv.vptt.ch opnfv

        __EOF
        i=0
        while [ $i -lt $(expr $enb_count) ] ; do
          host_part=$(expr $enb_start + $i)
          echo   ${enb_cidr%.*}.$host_part   $enb_hostname_prefix$i.$realm
$enb_hostname_prefix$i >> /etc/hosts
          i=$(expr $i + 1)
        done
        cat >> /etc/hosts << __EOF

        __EOF

  ssh_keys:
    type: OS::Heat::SoftwareConfig
    depends_on: [ hss_vm, mme_vm, spgw_vm, build_vm ]
    properties:
      inputs:
      - name: ssh_priv
        default: { get_param: ssh_priv }
      - name: ssh_pub
        default: { get_param: ssh_pub }
      group: script
      config: |
        #!/bin/bash
        logger "$0: Creating SSH keys..."
        mkdir -p /root/.ssh
        echo $ssh_priv > /root/.ssh/id_rsa
        sed     -e     's|-----BEGIN     RSA     PRIVATE     KEY-----|-----
BEGIN_RSA_PRIVATE_KEY-----|' \
            -e 's|-----END RSA PRIVATE KEY-----|-----END_RSA_PRIVATE_KEY--
---|' \
```

```
            -e 's| |\n|g' \
            -e  's|-----BEGIN_RSA_PRIVATE_KEY-----|-----BEGIN  RSA  PRIVATE
KEY-----|' \
            -e 's|-----END_RSA_PRIVATE_KEY-----|-----END RSA PRIVATE KEY--
---|' \
            -i /root/.ssh/id_rsa
        echo $ssh_pub > /root/.ssh/id_rsa.pub
        echo $ssh_pub > /root/.ssh/authorized_keys
        chmod 0644 /root/.ssh/id_rsa.pub
        chmod 0600 /root/.ssh/id_rsa
        chmod 0644 /root/.ssh/authorized_keys
        cat > /root/.ssh/config << __EOF
        StrictHostKeyChecking no
        UserKnownHostsFile /dev/null
        __EOF

  vpn_server:
    type: OS::Heat::SoftwareConfig
    depends_on: spgw_vm
    properties:
      group: script
      inputs:
      - name: mme_name
        default: { get_param: mme_name }
      - name: spgw_name
        default: { get_param: spgw_name }
      - name: enb_cidr
        default: { get_param: enb_cidr }
      - name: enb_cidr
        default: { get_param: enb_cidr }
      - name: enb_count
        default: { get_param: enb_count }
      - name: enb_start
        default: { get_param: enb_start }
      - name: enb_hostname_prefix
        default: { get_param: enb_hostname_prefix }
      - name: realm
        default: { get_param: realm }
      - name: ca_country
        default: { get_param: ca_country }
      - name: ca_state
        default: { get_param: ca_state }
      - name: ca_city
        default: { get_param: ca_city }
      - name: ca_company
        default: { get_param: ca_company }
      - name: ca_unit
        default: { get_param: ca_unit }
      - name: ca_email
        default: { get_param: ca_email }
      config: |
        #!/bin/bash
        logger "$0: Creating /etc/openvpn/clients.txt"
        mkdir -p /etc/openvpn
        touch /etc/openvpn/clients.txt
        echo $mme_name,${enb_cidr%.*}.2 > /etc/openvpn/clients.txt
        i=0
```

```
        net=$enb_cidr
        while [ $i -lt $(expr $enb_count) ] ; do
          host_part=$(expr $enb_start + $i)
          echo        $enb_hostname_prefix$i,${net%.*}.$host_part        >>
/etc/openvpn/clients.txt
          i=$(expr $i + 1)
        done
        logger "$0: Installing openvpn and creating VPN keys..."
        DEBIAN_FRONTEND=noninteractive apt-get install -q -y openvpn easy-
rsa
        mkdir -p /etc/openvpn/easy-rsa
        cp -R /usr/share/easy-rsa/* /etc/openvpn/easy-rsa
        mkdir -p /etc/openvpn/easy-rsa/keys
        cd /etc/openvpn/easy-rsa
        export EASY_RSA="${EASY_RSA:-.}"
        source ./vars
        export KEY_COUNTRY="$ca_country"
        export KEY_PROVINCE="$ca_state"
        export KEY_CITY="$ca_city"
        export KEY_ORG="$ca_company"
        export KEY_EMAIL="$ca_email"
        export KEY_OU="$ca_unit"
        export KEY_NAME="VPN_CA"
        ./clean-all
        "$EASY_RSA/pkitool" --initca
        export KEY_NAME=$spgw_name
        "$EASY_RSA/pkitool" --server $spgw_name
        ./build-dh
        openvpn --genkey --secret keys/hmac.key
        for client in $(cat /etc/openvpn/clients.txt) ; do
          export KEY_NAME=${client%,*}
          "$EASY_RSA/pkitool" ${client%,*}
        done
        cd -
        logger "$0: Creating /etc/openvpn/server.conf..."
        cat > /etc/openvpn/server.conf << __EOF
        daemon
        proto udp
        port 1194
        dev tun
        ca /etc/openvpn/easy-rsa/keys/ca.crt
        cert /etc/openvpn/easy-rsa/keys/$spgw_name.crt
        key /etc/openvpn/easy-rsa/keys/$spgw_name.key
        dh /etc/openvpn/easy-rsa/keys/dh2048.pem
        tls-auth /etc/openvpn/easy-rsa/keys/hmac.key 0
        server ${net%/*} 255.255.255.0
        topology subnet
        client-to-client
        ifconfig-pool-persist /etc/openvpn/clients.txt
        keepalive 10 120
        cipher AES-128-CBC
        comp-lzo
        user nobody
        group nogroup
        persist-key
        persist-tun
        verb 3
```

```
        __EOF
        logger "$0: Starting VPN server..."
        systemctl enable openvpn@server.service
        systemctl restart openvpn@server.service
        service openvpn restart

  vpn_client:
    type: OS::Heat::SoftwareConfig
    depends_on: mme_vm
    properties:
      group: script
      inputs:
      - name: spgw_name
        default: { get_param: spgw_name }
      - name: client_name
        default: { get_param: mme_name }
      - name: server_ip
        default: { get_attr: [ spgw_vm, first_address ] }
      config: |
        #!/bin/bash
        logger "$0: Installing VPN client..."
        DEBIAN_FRONTEND=noninteractive apt-get -y -q install openvpn
        cat > /etc/openvpn/client.conf << __EOF
        daemon
        client
        proto udp
        dev tun
        nobind
        remote $server_ip 1194
        ca /etc/openvpn/keys/ca.crt
        cert /etc/openvpn/keys/$client_name.crt
        key /etc/openvpn/keys/$client_name.key
        tls-auth /etc/openvpn/keys/hmac.key 1
        ns-cert-type server
        cipher AES-128-CBC
        comp-lzo
        user nobody
        group nogroup
        persist-key
        persist-tun
        verb 3
        mute 20
        __EOF
        cd /tmp
        ssh   $spgw_name   "tar   -c   /etc/openvpn/easy-rsa/keys"   |   dd
of=keys.tar
        tar -xvf keys.tar
        mv etc/openvpn/easy-rsa/keys .
        install -v -d -m 0700 -o root -g root /etc/openvpn/keys
        install   -v   -m   0400      -o   root   -g   root   keys/ca.crt
/etc/openvpn/keys/ca.crt
        install   -v   -m   0400      -o   root   -g   root   keys/hmac.key
/etc/openvpn/keys/hmac.key
        install  -v  -m  0400    -o  root  -g  root  keys/$client_name.crt
/etc/openvpn/keys/$client_name.crt
        install  -v  -m  0400    -o  root  -g  root  keys/$client_name.key
/etc/openvpn/keys/$client_name.key
```

```
        rm etc keys keys.tar -rf
        systemctl enable openvpn@client.service
        systemctl restart openvpn@client.service
        service openvpn restart

  update_system:
    type: OS::Heat::SoftwareConfig
    depends_on: [ hss_vm, mme_vm, spgw_vm, build_vm ]
    properties:
      group: script
      config: |
        #!/bin/bash
        logger "$0: Updating system..."
        DEBIAN_FRONTEND=noninteractive apt-get -q -y update
        DEBIAN_FRONTEND=noninteractive apt-get -q -y dist-upgrade
        DEBIAN_FRONTEND=noninteractive apt-get -q -y autoremove
        DEBIAN_FRONTEND=noninteractive apt-get -q -y autoclean
        DEBIAN_FRONTEND=noninteractive apt-get -q -y clean
        logger "$0: Installing git screen and curl..."
        DEBIAN_FRONTEND=noninteractive apt-get -q -y install curl screen
git

  compile_kernel:
    type: OS::Heat::SoftwareConfig
    depends_on: build_vm
    properties:
      group: script
      inputs:
      - name: spgw_kernel_version
        default: { get_param: spgw_kernel_version }
      - name: base_url
        default: { get_param: base_url }
      config: |
        #!/bin/bash
        cd /usr/src
        DEBIAN_FRONTEND=noninteractive apt-get -q -y install xz-utils
build-essential wget libncurses5-dev libssl-dev bc
        DEBIAN_FRONTEND=noninteractive apt-get -q -y build-dep linux-
image-$(uname -r)
        wget           https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-
$spgw_kernel_version.tar.xz
        tar -xf linux-$spgw_kernel_version.tar.xz
        curl -s -O $base_url/config-$spgw_kernel_version-gtp
        cp             config-$spgw_kernel_version-gtp             linux-
$spgw_kernel_version/.config
        cd linux-$spgw_kernel_version
        make oldconfig
        make -j`nproc`
        make INSTALL_MOD_STRIP=1 modules_install
        make install
        cd /root
        # should reboot to use new kernel. Hangs on ubuntu14
        #shutdown -r +1
        #sleep 55

  create_freediameter_certs:
    type: OS::Heat::SoftwareConfig
```

```
    depends_on: build_vm
    properties:
      group: script
      inputs:
      - name: realm
        default: { get_param: realm }
      - name: mme_name
        default: { get_param: mme_name }
      - name: hss_name
        default: { get_param: hss_name }
      - name: ca_country
        default: { get_param: ca_country }
      - name: ca_state
        default: { get_param: ca_state }
      - name: ca_city
        default: { get_param: ca_city }
      - name: ca_company
        default: { get_param: ca_company }
      - name: ca_unit
        default: { get_param: ca_unit }
      config: |
        #!/bin/bash
        DEBIAN_FRONTEND=noninteractive apt-get -y -q install openssl
        certs_dir=/root/certs
        make_one_cert() {
          name=$1
          openssl genrsa -out $name.key.pem 1024
          openssl req -new -batch -out $name.csr.pem -key $name.key.pem -
subj
/CN=$name.$realm/C=$ca_country/ST=$ca_state/L=$ca_city/O=$ca_company/OU=$c
a_unit
          openssl ca -cert cacert.pem -keyfile cakey.pem -in $name.csr.pem
-out $name.cert.pem -outdir . -batch
        }
        mkdir -p $certs_dir
        cd $certs_dir
        mkdir -p $certs_dir/demoCA/
        touch $certs_dir/demoCA/index.txt
        echo 01 > $certs_dir/demoCA/serial
        openssl req -new -batch -x509 -days 3650 -nodes -newkey rsa:1024 -
out          cacert.pem          -keyout          cakey.pem          -subj
/CN=$realm/C=$ca_country/ST=$ca_state/L=$ca_city/O=$ca_company/OU=$ca_unit
        make_one_cert $hss_name
        make_one_cert $mme_name

  eurecom_certs:
    type: OS::Heat::SoftwareConfig
    depends_on: build_vm
    properties:
      group: script
      config: |
        #!/bin/bash
        DEBIAN_FRONTEND=noninteractive apt-get -y -q install openssl
        if [ ! -f /etc/ssl/certs/ca-certificates.crt.bak ] ; then cp
/etc/ssl/certs/ca-certificates.crt{,.bak} ; fi
        echo    -n    |    openssl    s_client    -showcerts    -connect
gitlab.eurecom.fr:443 2>/dev/null | sed -ne '/-BEGIN CERTIFICATE-/,/-END
```

```
CERTIFICATE-/p' >> /etc/ssl/certs/ca-certificates.crt

  compile_nettle:
    type: OS::Heat::SoftwareConfig
    depends_on: build_vm
    properties:
      group: script
      config: |
        #!/bin/bash
        DEBIAN_FRONTEND=noninteractive apt-get -y -q install autoconf
automake build-essential libgmp-dev wget
        cd /tmp
        rm -rf /tmp/nettle-2.5.tar.gz /tmp/nettle-2.5
        wget https://ftp.gnu.org/gnu/nettle/nettle-2.5.tar.gz
        tar -xzf /tmp/nettle-2.5.tar.gz
        cd /tmp/nettle-2.5
        ./configure --disable-openssl --enable-shared --prefix=/usr/local
        make
        make check
        make install
        cd /tmp
        rm -rf /tmp/nettle-2.5.tar.gz /tmp/nettle-2.5
        ldconfig

  compile_gnutls:
    type: OS::Heat::SoftwareConfig
    depends_on: build_vm
    properties:
      group: script
      config: |
        #!/bin/bash
        DEBIAN_FRONTEND=noninteractive apt-get -y -q purge libgnutls-dev
'libgnutlsxx2?'
        DEBIAN_FRONTEND=noninteractive apt-get -y -q install libtasn1-6-
dev libp11-kit-dev libtspi-dev libidn11-dev wget
        cd /tmp
        rm -rf /tmp/gnutls-3.1.23.tar.xz* /tmp/gnutls-3.1.23
        wget ftp://ftp.gnutls.org/gcrypt/gnutls/v3.1/gnutls-3.1.23.tar.xz
        tar -xJf /tmp/gnutls-3.1.23.tar.xz
        cd /tmp/gnutls-3.1.23
        ./configure --prefix=/usr/local
        make
        make install
        cd /tmp
        rm -rf /tmp/gnutls-3.1.23 /tmp/gnutls-3.1.23.tar.xz
        ldconfig

  compile_freediameter:
    type: OS::Heat::SoftwareConfig
    depends_on: build_vm
    properties:
      group: script
      config: |
        #!/bin/bash
        DEBIAN_FRONTEND=noninteractive apt-get -y -q install autoconf
automake bison flex build-essential cmake libsctp-dev libidn11-dev
libgcrypt-dev
```

```
        cd /tmp
        rm -rf /tmp/freediameter
        git    clone    https://gitlab.eurecom.fr/oai/freediameter.git    -b
eurecom-1.2.0
        cd /tmp/freediameter
        mkdir build
        cd build
        cmake -DCMAKE_INSTALL_PREFIX:PATH=/usr/local ../
        make
        make install
        cd /tmp
        rm -rf /tmp/freediameter
        ldconfig

  compile_asn1c:
    type: OS::Heat::SoftwareConfig
    depends_on: build_vm
    properties:
      group: script
      config: |
        #!/bin/bash
        DEBIAN_FRONTEND=noninteractive  apt-get  -y  -q  install  autoconf
automake bison flex build-essential libtool
        cd /tmp
        rm -rf /tmp/asn1c
        git clone https://gitlab.eurecom.fr/oai/asn1c.git
        cd /tmp/asn1c
        ./configure --prefix=/usr/local
        make
        make install
        cd /tmp
        rm -rf /tmp/asn1c

  compile_libgtpnl:
    type: OS::Heat::SoftwareConfig
    depends_on: build_vm
    properties:
      group: script
      config: |
        #!/bin/bash
        DEBIAN_FRONTEND=noninteractive  apt-get  -y  -q  install  autoconf
automake bison flex build-essential libtool libmnl-dev
        cd /tmp
        rm -rf /tmp/libgtpnl
        git clone git://git.osmocom.org/libgtpnl
        cd /tmp/libgtpnl
        autoreconf -fi
        ./configure --prefix=/usr/local
        make
        make install
        cd /tmp
        rm -rf /tmp/libgtpnl
        ldconfig

  compile_openair_cn:
    type: OS::Heat::SoftwareConfig
    depends_on: build_vm
```

```
    properties:
      group: script
      config: |
        #!/bin/bash
        DEBIAN_FRONTEND=noninteractive apt-get -y -q install autoconf
automake bison flex build-essential cmake libsctp-dev libconfig8-dev
libgmp-dev libsctp-dev libssl-dev libxml2-dev mscgen openssl mariadb-
client libmysqlclient-dev check
        cd /tmp
        rm -rf openair-cn
        git clone https://gitlab.eurecom.fr/oai/openair-cn.git
        cd /tmp/openair-cn/SCRIPTS
        ./build_mme --clean
        ./build_mme --clean --daemon
        ./build_spgw --clean
        ./build_spgw --clean --daemon
        ./build_hss --clean
        ./build_hss --clean --daemon

  mme_conf:
    type: OS::Heat::SoftwareConfig
    depends_on: mme_vm
    properties:
      group: script
      inputs:
      - name: hss_name
        default: { get_param: hss_name }
      - name: mme_name
        default: { get_param: mme_name }
      - name: realm
        default: { get_param: realm }
      - name: enb_count
        default: { get_param: enb_count }
      - name: int_cidr
        default: { get_param: int_cidr }
      - name: enb_cidr
        default: { get_param: enb_cidr }
      - name: mcc
        default: { get_param: mcc }
      - name: mnc
        default: { get_param: mnc }
      - name: tac
        default: { get_param: tac }
      - name: mme_gid
        default: { get_param: mme_gid }
      - name: mme_code
        default: { get_param: mme_code }
      - name: spgw_ip
        default: { get_attr: [ spgw_vm, first_address ] }
      - name: mme_ip
        default: { get_attr: [ mme_vm, first_address ] }
      - name: hss_ip
        default: { get_attr: [ hss_vm, first_address ] }
      config: |
        #!/bin/bash
        logger "$0: Creating MME configuration..."
        mkdir -p /etc/oai
```

```
        intnet=$int_cidr
        enbnet=$enb_cidr
        cat > /etc/oai/mme.conf <<__EOF
        MME :
        {
          REALM                                 = "$realm";
          MAXENB                                = $enb_count;
          MAXUE                                 = 16;
          RELATIVE_CAPACITY                     = 10;
          EMERGENCY_ATTACH_SUPPORTED            = "no";
          UNAUTHENTICATED_IMSI_SUPPORTED        = "no";
          EPS_NETWORK_FEATURE_SUPPORT_IMS_VOICE_OVER_PS_SESSION_IN_S1
= "no";
          EPS_NETWORK_FEATURE_SUPPORT_EMERGENCY_BEARER_SERVICES_IN_S1_MODE
= "no";
          EPS_NETWORK_FEATURE_SUPPORT_LOCATION_SERVICES_VIA_EPC
= "no";
          EPS_NETWORK_FEATURE_SUPPORT_EXTENDED_SERVICE_REQUEST
= "no";
          IP_CAPABILITY
= "IPV4V6";
          MME_STATISTIC_TIMER                   = 10;
          INTERTASK_INTERFACE :
          {
            ITTI_QUEUE_SIZE                     = 2000000;
          };
          S6A :
          {
            S6A_CONF                            = "/etc/oai/mme_fd.conf";
            HSS_HOSTNAME                        = "$hss_name";
          };
          SCTP :
          {
            SCTP_INSTREAMS                      = 8;
            SCTP_OUTSTREAMS                     = 8;
          };
          S1AP :
          {
            S1AP_OUTCOME_TIMER                  = 10;
          };
          GUMMEI_LIST = (
            {    MCC="$mcc"    ;    MNC="$mnc";    MME_GID="$mme_gid"    ;
MME_CODE="$mme_code"; }
          );
          TAI_LIST = (
            { MCC="$mcc" ; MNC="$mnc"; TAC="$tac"; }
          );
          NAS :
          {
            ORDERED_SUPPORTED_INTEGRITY_ALGORITHM_LIST = [ "EIA2" , "EIA1"
, "EIA0" ];
            ORDERED_SUPPORTED_CIPHERING_ALGORITHM_LIST = [ "EEA0" , "EEA1"
, "EEA2" ];
            T3402                               = 1  #in minutes
            T3412                               = 54 #in minutes
            T3422                               = 6
            T3450                               = 6
```

```
            T3460                                   = 6
            T3470                                   = 6
            T3485                                   = 8
            T3486                                   = 8
            T3489                                   = 4
            T3495                                   = 8
          };
          LOGGING :
          {
            OUTPUT                                  = "CONSOLE";
            COLOR                                   = "no";
            SCTP_LOG_LEVEL                          = "TRACE";
            S1AP_LOG_LEVEL                          = "TRACE";
            NAS_LOG_LEVEL                           = "TRACE";
            MME_APP_LOG_LEVEL                       = "TRACE";
            S6A_LOG_LEVEL                           = "TRACE";
            UTIL_LOG_LEVEL                          = "TRACE";
            MSC_LOG_LEVEL                           = "ERROR";
            ITTI_LOG_LEVEL                          = "ERROR";
            ASN1_VERBOSITY                          = "none";
          };
          NETWORK_INTERFACES :
          {
            MME_INTERFACE_NAME_FOR_S1_MME           = "eth0";
            MME_IPV4_ADDRESS_FOR_S1_MME                                     =
"${enbnet%.*}.2/${enbnet#*/}";
            MME_INTERFACE_NAME_FOR_S11_MME          = "eth0";
            MME_IPV4_ADDRESS_FOR_S11_MME            = "$mme_ip/${intnet#*/}";
            MME_PORT_FOR_S11_MME                    = 2123;
          };
        };
        S-GW :
        {
          SGW_IPV4_ADDRESS_FOR_S11                                          =
"$spgw_ip/${intnet#*/}";
        };
        __EOF
        cat > /etc/oai/mme_fd.conf << __EOF
        Identity       = "$mme_name.$realm";
        Realm          = "$realm";
        TLS_Cred                        =    "/etc/oai/$mme_name.cert.pem",
"/etc/oai/$mme_name.key.pem";
        TLS_CA         = "/etc/oai/cacert.pem";
        AppServThreads = 4;
        SCTP_streams   = 8;
        LoadExtension  = "dict_nas_mipv6.fdx";
        LoadExtension  = "dict_s6a.fdx";
        No_TCP;
        No_IPv6;
        NoRelay;
        ConnectPeer= "$hss_name.$realm" { ConnectTo = "$hss_ip"; No_IPv6;
No_TLS; port = 3868; realm = "$realm";};
        __EOF

  spgw_conf:
    type: OS::Heat::SoftwareConfig
    depends_on: spgw_vm
```

```
    properties:
      group: script
      inputs:
      - name: ue_cidr
        default: { get_param: ue_cidr }
      - name: ue_dns
        default: { get_param: ue_dns }
      - name: ue_mtu
        default: { get_param: ue_mtu }
      - name: int_cidr
        default: { get_param: int_cidr }
      - name: enb_cidr
        default: { get_param: enb_cidr }
      - name: spgw_ip
        default: { get_attr: [ spgw_vm, first_address ] }
      config: |
        #!/bin/bash
        logger "$0: Creating SPGW configuration..."
        mkdir -p /etc/oai
        intnet=$int_cidr
        enbnet=$enb_cidr
        cat > /etc/oai/spgw.conf << __EOF
        S-GW :
        {
          NETWORK_INTERFACES :
          {
            SGW_INTERFACE_NAME_FOR_S11               = "eth0";
            SGW_IPV4_ADDRESS_FOR_S11                                 =
"$spgw_ip/${intnet#*/}";
            SGW_INTERFACE_NAME_FOR_S1U_S12_S4_UP     = "tun0";
            SGW_IPV4_ADDRESS_FOR_S1U_S12_S4_UP                       =
"${enbnet%.*}.1/${enbnet#*/}";
            SGW_IPV4_PORT_FOR_S1U_S12_S4_UP          = 2152;
            SGW_INTERFACE_NAME_FOR_S5_S8_UP          = "none";
            SGW_IPV4_ADDRESS_FOR_S5_S8_UP            = "0.0.0.0/24";
          };

          INTERTASK_INTERFACE :
          {
            ITTI_QUEUE_SIZE                          = 2000000;
          };

          LOGGING :
          {
            OUTPUT                                   = "CONSOLE";
            THREAD_SAFE                              = "yes";
            COLOR                                    = "no";
            UDP_LOG_LEVEL                            = "TRACE";
            GTPV1U_LOG_LEVEL                         = "TRACE";
            GTPV2C_LOG_LEVEL                         = "TRACE";
            SPGW_APP_LOG_LEVEL                       = "TRACE";
            S11_LOG_LEVEL                            = "TRACE";
          };
        };
        P-GW =
        {
          NETWORK_INTERFACES :
```

```
          {
            PGW_INTERFACE_NAME_FOR_S5_S8             = "none";
            PGW_INTERFACE_NAME_FOR_SGI               = "eth0";
            PGW_MASQUERADE_SGI                       = "yes";
            UE_TCP_MSS_CLAMPING                      = "no";
          };

          IP_ADDRESS_POOL :
          {
            IPV4_LIST = (
                                                     "$ue_cidr"
                        );
          };

          DEFAULT_DNS_IPV4_ADDRESS                   = "$ue_dns";
          DEFAULT_DNS_SEC_IPV4_ADDRESS               = "8.8.4.4";
          FORCE_PUSH_PROTOCOL_CONFIGURATION_OPTIONS  = "no";
          UE_MTU                                     = $ue_mtu;
        };
        __EOF

  hss_conf:
    type: OS::Heat::SoftwareConfig
    depends_on: hss_vm
    properties:
      group: script
      inputs:
      - name: db_pass
        default: { get_param: db_pass }
      - name: operator_key
        default: { get_param: operator_key }
      - name: hss_name
        default: { get_param: hss_name }
      - name: realm
        default: { get_param: realm }
      - name: hss_ip
        default: { get_attr: [ hss_vm, first_address ] }
      config: |
        #!/bin/bash
        logger "$0: Creating HSS configuration..."
        mkdir -p /etc/oai
        cat > /etc/oai/hss.conf << __EOF
        HSS :
        {
          MYSQL_server = "127.0.0.1";
          MYSQL_user   = "hssadmin";
          MYSQL_pass   = "$db_pass";
          MYSQL_db     = "oai_db";
          OPERATOR_key = "$operator_key";
          RANDOM       = "true";
          FD_conf      = "/etc/oai/hss_fd.conf";
        };
        __EOF
        cat > /etc/oai/hss_fd.conf << __EOF
        Identity       = "$hss_name.$realm";
        Realm          = "$realm";
        TLS_Cred                        =    "/etc/oai/$hss_name.cert.pem",
```

```
"/etc/oai/$hss_name.key.pem";
        TLS_CA         = "/etc/oai/cacert.pem";
        AppServThreads = 4;
        SCTP_streams   = 8;
        ListenOn       = "$hss_ip";
        Port           = 3868;
        SecPort        = 5868;
        LoadExtension  = "acl_wl.fdx" : "/etc/oai/hss_acl.conf";
        LoadExtension  = "dict_nas_mipv6.fdx";
        LoadExtension  = "dict_s6a.fdx";
        No_TCP;
        No_IPv6;
        NoRelay;
        __EOF
        cat > /etc/oai/hss_acl.conf <<__EOF
        ALLOW_OLD_TLS   *.$realm
        __EOF

  install_nettle:
    type: OS::Heat::SoftwareConfig
    depends_on: [ hss_vm, mme_vm ]
    properties:
      group: script
      inputs:
      - name: build_name
        default: { get_param: build_name }
      config: |
        #!/bin/bash
        scp -r $build_name:/usr/local/lib /tmp
        install  -v  -m  0644  -o  root  -g  root  /tmp/lib/libnettle.so.4.4
/tmp/lib/libhogweed.so.2.2 /usr/local/lib/
        ln -sfv libnettle.so.4.4 /usr/local/lib/libnettle.so.4
        ln -sfv libnettle.so.4.4 /usr/local/lib/libnettle.so
        ln -sfv libhogweed.so.2.2 /usr/local/lib/libhogweed.so.2
        ln -sfv libhogweed.so.2.2 /usr/local/lib/libhogweed.so
        ldconfig
        rm /tmp/lib -rf

  install_gnutls:
    type: OS::Heat::SoftwareConfig
    depends_on: [ hss_vm, mme_vm ]
    properties:
      group: script
      inputs:
      - name: build_name
        default: { get_param: build_name }
      config: |
        #!/bin/bash
        scp -r $build_name:/usr/local/lib /tmp
        install -v -m 0644 -o root -g root /tmp/lib/libgnutls.so.28.21.3
/usr/local/lib/
        ln -sfv libgnutls.so.28.21.3 /usr/local/lib/libgnutls.so.28
        ln -sfv libgnutls.so.28.21.3 /usr/local/lib/libgnutls.so
        ldconfig
        rm /tmp/lib -rf

  install_freediameter:
```

```
    type: OS::Heat::SoftwareConfig
    depends_on: [ hss_vm, mme_vm ]
    properties:
      group: script
      inputs:
      - name: build_name
        default: { get_param: build_name }
      config: |
        #!/bin/bash
        scp -r $build_name:/usr/local/lib /tmp
        install -v -m 0644 -o root -g root /tmp/lib/libfdcore.so.1.2.0
/tmp/lib/libfdproto.so.1.2.0 /usr/local/lib/
        ln -sfv libfdproto.so.1.2.0 /usr/local/lib/libfdproto.so.6
        ln -sfv libfdproto.so.6 /usr/local/lib/libfdproto.so
        ln -sfv libfdcore.so.1.2.0 /usr/local/lib/libfdcore.so.6
        ln -sfv libfdcore.so.6 /usr/local/lib/libfdcore.so
        install -v -d -m 0755 -o root -g root /usr/local/lib/freeDiameter
        install -v -m 0644 -o root -g root /tmp/lib/freeDiameter/*
/usr/local/lib/freeDiameter
        ldconfig
        rm /tmp/lib -rf

  install_libgtpnl:
    type: OS::Heat::SoftwareConfig
    depends_on: spgw_vm
    properties:
      group: script
      inputs:
      - name: build_name
        default: { get_param: build_name }
      config: |
        #!/bin/bash
        scp -r $build_name:/usr/local/lib /tmp
        install -v -m 0644 -o root -g root /tmp/lib/libgtpnl.so.0.0.0
/usr/local/lib/
        ln -sfv libgtpnl.so.0.0.0 /usr/local/lib/libgtpnl.so.0
        ln -sfv libgtpnl.so.0.0.0 /usr/local/lib/libgtpnl.so
        ldconfig
        rm /tmp/lib -rf

  install_kernel:
    type: OS::Heat::SoftwareConfig
    depends_on: spgw_vm
    properties:
      group: script
      inputs:
      - name: spgw_kernel_version
        default: { get_param: spgw_kernel_version }
      - name: build_name
        default: { get_param: build_name }
      config: |
        #!/bin/bash
        #sleep 90 # wait for build to reboot  # we dont reboot on ubuntu14
        scp $build_name:/boot/*$spgw_kernel_version* /tmp
        install -o root -g root -m 0644 -v /tmp/*$spgw_kernel_version*
/boot/
        rm /tmp/*$spgw_kernel_version*
```

```
        ssh $build_name "tar -c /lib/modules/$spgw_kernel_version" | dd
of=/tmp/modules.tar
        cd /tmp
        tar -xf modules.tar
        cp -av lib/modules/$spgw_kernel_version /lib/modules
        rm modules.tar lib -rf
        update-initramfs -c -k $spgw_kernel_version
        update-grub
        # should reboot to use new kernel. Hangs on ubuntu14
        #shutdown -r +1
        #sleep 55

  install_freediameter_certs:
    type: OS::Heat::SoftwareConfig
    depends_on: [ hss_vm, mme_vm ]
    properties:
      group: script
      inputs:
      - name: hostname
      - name: build_name
        default: { get_param: build_name }
      config: |
        #!/bin/bash
        scp -r $build_name:/root/certs /tmp
        install -v -m 0600 -o root -g root /tmp/certs/cacert.pem /etc/oai
        install -v -m 0600 -o root -g root /tmp/certs/$hostname.key.pem
/etc/oai
        install -v -m 0600 -o root -g root /tmp/certs/$hostname.cert.pem
/etc/oai
        rm /tmp/certs -rf

  install_database:
    type: OS::Heat::SoftwareConfig
    depends_on: hss_vm
    properties:
      group: script
      inputs:
      - name: db_file
        default: { get_param: db_file }
      - name: base_url
        default: { get_param: base_url }
      - name: db_pass
        default: { get_param: db_pass }
      config: |
        #!/bin/bash
        # this was the only way I found to let it work on xenial and
trusty
        function reply_mysql_server_questions() {
          echo
          echo
          echo
        }
        reply_mysql_server_questions | DEBIAN_FRONTEND=noninteractive apt-
get -y -q install autoconf automake bison flex build-essential cmake
libsctp-dev libconfig8-dev libgmp-dev libsctp-dev libssl-dev libxml2-dev
mscgen openssl mariadb-server mariadb-client libmysqlclient-dev check
        sleep 5
```

```
        echo "CREATE DATABASE oai_db; GRANT ALL PRIVILEGES ON oai_db.* TO
'hssadmin'@'localhost' IDENTIFIED BY '$db_pass';" | mysql -u root
        cd /root
        curl -s -O $base_url/$db_file
        mysql -u hssadmin -p$db_pass -D oai_db < /root/$db_file

  install_hss:
    type: OS::Heat::SoftwareConfig
    depends_on: hss_vm
    properties:
      group: script
      inputs:
      - name: build_name
        default: { get_param: build_name }
      config: |
        #!/bin/bash
        DEBIAN_FRONTEND=noninteractive apt-get -y -q install libconfig9
libsctp1 libtspi1 libgmp10
        scp $build_name:/usr/local/bin/oai_hss /tmp
        scp $build_name:/usr/sbin/oai_hssd /tmp
        install -v -m 0755 -o root -g root /tmp/oai_hss /tmp/oai_hssd
/usr/local/sbin
        rm /tmp/oai_hss*

  install_mme:
    type: OS::Heat::SoftwareConfig
    depends_on: mme_vm
    properties:
      group: script
      inputs:
      - name: build_name
        default: { get_param: build_name }
      config: |
        #!/bin/bash
        DEBIAN_FRONTEND=noninteractive apt-get -y -q install libconfig9
libsctp1 libtspi1 libgmp10
        scp $build_name:/usr/local/bin/mme /tmp
        scp $build_name:/usr/sbin/mmed /tmp
        install -v -m 0755 -o root -g root /tmp/mme /tmp/mmed
/usr/local/sbin
        rm /tmp/mme*

  install_spgw:
    type: OS::Heat::SoftwareConfig
    depends_on: spgw_vm
    properties:
      group: script
      inputs:
      - name: build_name
        default: { get_param: build_name }
      config: |
        #!/bin/bash
        DEBIAN_FRONTEND=noninteractive apt-get -y -q install libconfig9
libmnl0
        scp $build_name:/usr/local/bin/spgw /tmp
        scp $build_name:/usr/sbin/spgwd /tmp
        install -v -m 0755 -o root -g root /tmp/spgw /tmp/spgwd
```

```
/usr/local/sbin
        rm /tmp/spgw*

### BUILD DEPLOYMENTS
  create_etc_hosts_build:
    type: OS::Heat::SoftwareDeployment
    depends_on: etc_hosts
    properties:
      config:
        get_resource: etc_hosts
      server:
        get_resource: build_vm

  update_system_build:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ create_etc_hosts_build, update_system ]
    properties:
      config:
        get_resource: update_system
      server:
        get_resource: build_vm

  ssh_keys_build:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ create_etc_hosts_build, update_system_build, ssh_keys ]
    properties:
      config:
        get_resource: ssh_keys
      server:
        get_resource: build_vm

  compile_kernel_build:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ ssh_keys_build, compile_kernel ]
    properties:
      config:
        get_resource: compile_kernel
      server:
        get_resource: build_vm

  eurecom_certs_build:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ compile_kernel_build, eurecom_certs ]
    properties:
      config:
        get_resource: eurecom_certs
      server:
        get_resource: build_vm

  create_freediameter_certs_build:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ compile_kernel_build, create_freediameter_certs ]
    properties:
      config:
        get_resource: create_freediameter_certs
      server:
        get_resource: build_vm
```

```
  compile_libgtpnl_build:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ compile_kernel_build, compile_libgtpnl ]
    properties:
      config:
        get_resource: compile_libgtpnl
      server:
        get_resource: build_vm

  compile_nettle_build:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ compile_kernel_build, compile_nettle ]
    properties:
      config:
        get_resource: compile_nettle
      server:
        get_resource: build_vm

  compile_gnutls_build:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ compile_nettle_build, compile_gnutls ]
    properties:
      config:
        get_resource: compile_gnutls
      server:
        get_resource: build_vm

  compile_freediameter_build:
    type: OS::Heat::SoftwareDeployment
    depends_on:     [     compile_nettle_build,     compile_gnutls_build,
eurecom_certs_build, compile_freediameter ]
    properties:
      config:
        get_resource: compile_freediameter
      server:
        get_resource: build_vm

  compile_asn1c_build:
    type: OS::Heat::SoftwareDeployment
    depends_on:    [    compile_freediameter_build,    eurecom_certs_build,
compile_asn1c ]
    properties:
      config:
        get_resource: compile_asn1c
      server:
        get_resource: build_vm

  compile_openair_cn_build:
    type: OS::Heat::SoftwareDeployment
    depends_on:   [   compile_freediameter_build,   compile_libgtpnl_build,
compile_asn1c_build, create_freediameter_certs_build, eurecom_certs_build,
compile_openair_cn ]
    properties:
      config:
        get_resource: compile_openair_cn
      server:
```

```
        get_resource: build_vm

### HSS DEPLOYMENTS
  create_etc_hosts_hss:
    type: OS::Heat::SoftwareDeployment
    depends_on: etc_hosts
    properties:
      config:
        get_resource: etc_hosts
      server:
        get_resource: hss_vm

  update_system_hss:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ create_etc_hosts_hss, update_system ]
    properties:
      config:
        get_resource: update_system
      server:
        get_resource: hss_vm

  create_hss_conf:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ create_etc_hosts_hss, update_system_hss, hss_conf ]
    properties:
      config:
        get_resource: hss_conf
      server:
        get_resource: hss_vm

  ssh_keys_hss:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ create_etc_hosts_build, update_system_build, ssh_keys ]
    properties:
      config:
        get_resource: ssh_keys
      server:
        get_resource: hss_vm

  install_database_hss:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ ssh_keys_hss, create_hss_conf, install_database ]
    properties:
      config:
        get_resource: install_database
      server:
        get_resource: hss_vm

  install_nettle_hss:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ ssh_keys_hss, ssh_keys_build, compile_openair_cn_build,
install_nettle ]
    properties:
      config:
        get_resource: install_nettle
      server:
        get_resource: hss_vm
```

```
  install_gnutls_hss:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ ssh_keys_hss, ssh_keys_build, compile_openair_cn_build,
install_nettle_hss, install_gnutls ]
    properties:
      config:
        get_resource: install_gnutls
      server:
        get_resource: hss_vm

  install_freediameter_hss:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ ssh_keys_hss, ssh_keys_build, compile_openair_cn_build,
install_nettle_hss, install_gnutls_hss, install_freediameter ]
    properties:
      config:
        get_resource: install_freediameter
      server:
        get_resource: hss_vm

  install_freediameter_certs_hss:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ ssh_keys_mme, ssh_keys_build, compile_openair_cn_build,
install_freediameter_hss, create_hss_conf, install_freediameter_certs ]
    properties:
      config:
        get_resource: install_freediameter_certs
      server:
        get_resource: hss_vm
      input_values:
        hostname: { get_param: hss_name }

  deploy_hss:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ ssh_keys_hss, ssh_keys_build, compile_openair_cn_build,
install_freediameter_certs_hss, install_database_hss, install_hss ]
    properties:
      config:
        get_resource: install_hss
      server:
        get_resource: hss_vm

### MME DEPLOYMENTS
  create_etc_hosts_mme:
    type: OS::Heat::SoftwareDeployment
    depends_on: etc_hosts
    properties:
      config:
        get_resource: etc_hosts
      server:
        get_resource: mme_vm

  update_system_mme:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ create_etc_hosts_mme, update_system ]
    properties:
```

```
      config:
        get_resource: update_system
      server:
        get_resource: mme_vm

  create_mme_conf:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ create_etc_hosts_mme, update_system_mme, mme_conf ]
    properties:
      config:
        get_resource: mme_conf
      server:
        get_resource: mme_vm

  ssh_keys_mme:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ create_etc_hosts_mme, update_system_mme, ssh_keys ]
    properties:
      config:
        get_resource: ssh_keys
      server:
        get_resource: mme_vm

  vpn_client_mme:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ ssh_keys_mme, ssh_keys_spgw, vpn_server_spgw, vpn_client
]
    properties:
      config:
        get_resource: vpn_client
      server:
        get_resource: mme_vm
      input_values:
        client_name: { get_param: mme_name }

  install_nettle_mme:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ ssh_keys_mme, ssh_keys_build, compile_openair_cn_build,
install_nettle ]
    properties:
      config:
        get_resource: install_nettle
      server:
        get_resource: mme_vm

  install_gnutls_mme:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ ssh_keys_mme, ssh_keys_build, compile_openair_cn_build,
install_nettle_mme, install_gnutls ]
    properties:
      config:
        get_resource: install_gnutls
      server:
        get_resource: mme_vm

  install_freediameter_mme:
    type: OS::Heat::SoftwareDeployment
```

```
    depends_on: [ ssh_keys_mme, ssh_keys_build, compile_openair_cn_build,
install_nettle_mme, install_gnutls_mme, install_freediameter ]
    properties:
      config:
        get_resource: install_freediameter
      server:
        get_resource: mme_vm

  install_freediameter_certs_mme:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ ssh_keys_mme, ssh_keys_build, compile_openair_cn_build,
install_freediameter_mme, create_mme_conf, install_freediameter_certs ]
    properties:
      config:
        get_resource: install_freediameter_certs
      server:
        get_resource: mme_vm
      input_values:
        hostname: { get_param: mme_name }

  deploy_mme:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ ssh_keys_mme, ssh_keys_build, compile_openair_cn_build,
install_freediameter_mme, install_freediameter_certs_mme, create_mme_conf,
vpn_client_mme, install_mme ]
    properties:
      config:
        get_resource: install_mme
      server:
        get_resource: mme_vm

### SPGW DEPLOYMENTS
  create_etc_hosts_spgw:
    type: OS::Heat::SoftwareDeployment
    depends_on: etc_hosts
    properties:
      config:
        get_resource: etc_hosts
      server:
        get_resource: spgw_vm

  update_system_spgw:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ create_etc_hosts_spgw, update_system ]
    properties:
      config:
        get_resource: update_system
      server:
        get_resource: spgw_vm

  create_spgw_conf:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ create_etc_hosts_spgw, update_system_spgw, spgw_conf ]
    properties:
      config:
        get_resource: spgw_conf
      server:
```

```
        get_resource: spgw_vm

  vpn_server_spgw:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ create_etc_hosts_spgw, update_system_spgw, vpn_server ]
    properties:
      config:
        get_resource: vpn_server
      server:
        get_resource: spgw_vm

  ssh_keys_spgw:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ create_etc_hosts_spgw, update_system_spgw, ssh_keys ]
    properties:
      config:
        get_resource: ssh_keys
      server:
        get_resource: spgw_vm

  install_kernel_spgw:
    type: OS::Heat::SoftwareDeployment
    depends_on:  [  ssh_keys_spgw,  ssh_keys_build,  compile_kernel_build,
install_kernel ]
    properties:
      config:
        get_resource: install_kernel
      server:
        get_resource: spgw_vm

  install_libgtpnl_spgw:
    type: OS::Heat::SoftwareDeployment
    depends_on: [ ssh_keys_spgw, ssh_keys_build, compile_openair_cn_build,
install_kernel_spgw, install_libgtpnl ]
    properties:
      config:
        get_resource: install_libgtpnl
      server:
        get_resource: spgw_vm

  deploy_spgw:
    type: OS::Heat::SoftwareDeployment
    depends_on:   [   ssh_keys_spgw,   ssh_keys_build,   create_spgw_conf,
install_libgtpnl_spgw,      vpn_server_spgw,      compile_openair_cn_build,
install_kernel_spgw, install_spgw ]
    properties:
      config:
        get_resource: install_spgw
      server:
        get_resource: spgw_vm

### STACK OUTPUTS
outputs:
  public_ip:
    description: Floating IP address of SPGW instance in external network
    value: { get_attr: [ floating_ip, floating_ip_address ] }
  ssh_spgw:
```

```
    description: SSH connect string for SPWG host
    value:
      str_replace:
        template: |
          ssh   -o   ForwardAgent=yes   -o   StrictHostKeyChecking=no   -o
UserKnownHostsFile=/dev/null -o User=ubuntu $floating_ip
        params:
          $floating_ip : { get_attr: [ floating_ip, floating_ip_address ]
}
  ssh_hss:
    description: SSH connect string for HSS host
    value:
      str_replace:
        template: |
          ssh   -o   ForwardAgent=yes   -o   StrictHostKeyChecking=no   -o
UserKnownHostsFile=/dev/null   -o   User=ubuntu   -o   ProxyCommand="ssh   -o
ForwardAgent=yes          -o          StrictHostKeyChecking=no          -o
UserKnownHostsFile=/dev/null   -o   User=ubuntu   -q   $floating_ip   nc   -q0
$hss_name 22" $floating_ip
        params:
          $floating_ip : { get_attr: [ floating_ip, floating_ip_address ]
}
          $hss_name: { get_param: hss_name }
  ssh_mme:
    description: SSH connect string for MME host
    value:
      str_replace:
        template: |
          ssh   -o   ForwardAgent=yes   -o   StrictHostKeyChecking=no   -o
UserKnownHostsFile=/dev/null   -o   User=ubuntu   -o   ProxyCommand="ssh   -o
ForwardAgent=yes          -o          StrictHostKeyChecking=no          -o
UserKnownHostsFile=/dev/null   -o   User=ubuntu   -q   $floating_ip   nc   -q0
$mme_name 22" $floating_ip
        params:
          $floating_ip : { get_attr: [ floating_ip, floating_ip_address ]
}
          $mme_name: { get_param: mme_name }
  ssh_build:
    description: SSH connect string for Build host
    value:
      str_replace:
        template: |
          ssh   -o   ForwardAgent=yes   -o   StrictHostKeyChecking=no   -o
UserKnownHostsFile=/dev/null   -o   User=ubuntu   -o   ProxyCommand="ssh   -o
ForwardAgent=yes          -o          StrictHostKeyChecking=no          -o
UserKnownHostsFile=/dev/null   -o   User=ubuntu   -q   $floating_ip   nc   -q0
$build_name 22" $floating_ip
        params:
          $floating_ip : { get_attr: [ floating_ip, floating_ip_address ]
}
          $build_name: { get_param: build_name }
```