A PROPOSAL FOR A SECURED, EFFICIENT AND SCALABLE LAYER 2 NETWORK VIRTUALISATION MECHANISM

JON MATIAS FRAILE

eman ta zabal zazu

Universidad del País Vasco

Euskal Herriko Unibertsitatea

Supervisor: Eduardo Jacob Taquet
Department of Communications Engineering
Faculty of Engineering
University of the Basque Country (UPV/EHU)

December 2015

ABSTRACT

The Internet has become the essential medium for information and for communication of both social and business interactions. However, several limitations of the current Internet have been identified to meet the expectations generated by its resounding success. In this context, the Future Internet initiative has emerged as a research effort to overcome these shortcomings.

To solve the limitations of the Internet (some of them due to its original design), it is necessary to research on novel network architectures and proposals, either evolutionary or clean-slate approaches. Moreover, the experimental facilities emerge to provide a realistic environment to validate these new approaches at large-scale conditions, and become the fundamental artefact for robust experimentation and testing of these proposals. As a design principle, the experimental facility must be orthogonal to the experimentation and its impact must be reduced to the minimum. Due to the need of sharing the same infrastructure and resources to test different networking proposals at the same time, the network virtualisation is key for success.

There is no unique definition of network virtualisation, since it depends on the target scenario, which could impose specific requirements due to its particularities. To properly analyse and compare the different approaches to network virtualisation a new taxonomy is proposed. Based on this taxonomy, a survey of different network virtualisation proposals is presented. As a result, three main types of virtualization have been identified and grouped in two categories: node centric and network centric approaches. In the former, the nodes are exposed as individual elements to the tenants (researchers in this case); whereas in the latter, the underlying network infrastructure is abstracted as a whole. With regard to the node centric approaches, two types of network virtualisation are proposed: Virtual Node (vNode) and SDN-enabled Virtualisation (SDNeV). On the one hand, the vNode solutions expose virtual network devices with an associated functionality implemented on them. On the other hand, the SDNeV proposals expose the programmability

of individual (either physical or virtual) network devices. Concerning the network centric approaches, just a single type of network virtualisation is proposed: overlay. The overlay solutions are built on the edge devices, which encapsulate the external addressing scheme into the addressing space used internally by the physical infrastructure.

The open innovation in networking demands the programmability of the networking resources to be exposed to the researchers. As a consequence, the Software Defined Networking (SDN) technology becomes the essential enabler of network innovation by means of the separation of data and control planes through a standard interface, and thus, the SDNeV type of network virtualisation is the most appropriate to build the experimental facilities.

Additionally, the most relevant experimental facilities are presented, with a special focus on how each of them enables the research on networking. The survey is divided in two groups: pre-SDN approaches based on legacy technology and SDN-enabled facilities. The latter is the relevant group due to the type of network virtualisation selected.

Based on the previous analyses and the research activities to be conducted on the experimental facility, a list of requirements is defined: the isolation between experiments (including the secured isolation between them), the flexibility to experiment on novel architectures and clean slate approaches, the scalability of the facility, the stability of the experiment slice definition, the transparency of the facility with regard to the experiments (including the efficiency, avoiding any overhead), and the support for research on networking. The available experimental facilities supported by the currently proposed network virtualisation approaches do not meet all these requirements. Therefore, a new experimental facility orthogonal to the experimentation is needed, as well as a new network virtualisation proposal that transparently supports the research on novel network architectures and approaches.

One of the responsibilities of any experimental facility is to deploy new services and resources to enable the experimentation. In this regard, the ongoing work on Network Functions Virtualisation (NFV) is relevant to this goal. NFV has been proposed to innovate in the service delivery arena by using standard computing virtualisation technology to consolidate in commodity hardware the functions previously performed by specific hardware appliances.

The main contributions of this thesis are also the key building blocks for the experimental facility: Layer 2 Prefix-based Network Virtualisation (L2PNV), MAC Address Configuration Protocol (MACP), and Flow-based Network Access Control (FlowNAC).

L2PNV is an efficient and scalable network virtualisation mechanism that relies on the MAC address prefixes to univocally identify the corresponding virtual network instances. This proposal is based on a novel MAC addressing scheme, which reserves part of the MAC address (a prefix) to encode the virtual instance identifier.

As a consequence of not using the globally administered MAC addresses, the end devices must update their MAC addresses to be compliant with the new addressing scheme. Therefore, a new mechanism, called MACP, is proposed to facilitate the assignment of MAC addresses. In addition, this proposal could be also used by other Layer 2 proposals based on alternative MAC addressing schemes, thus enabling the innovation in this research direction.

Furthermore, the overall security of both the experimental facility and the network virtualisation proposal is improved by adding a new mechanism, called FlowNAC, for fine-grained access control based on the identity of the users and the policy defined by the tenants (researchers in this case).

In relation to the aforementioned contributions, new design principles based on SDN-enabled architectures have been adopted to implement these proposals as Virtual Network Functions (VNFs). The goals of using SDN-enabled architectures for service delivery are to improve the performance of the VNFs and to facilitate the deployment of these new services (i.e., MACP and FlowNAC) over the experimental facility.

As a result (and an additional contribution), a new experimental facility is deployed at the University of the Basque Country (UPV/EHU), the EHU OpenFlow Enabled Facility (EHU-OEF), to experimentally validate these approaches. The main objective of EHU-OEF is to enable the innovation on networking architectures and proposals.

In this context, L2PNV has been experimentally validated and its fundamental mechanism for virtualising the network (the MAC prefix matching) has been evaluated. It has been also compared to other similar mechanisms, all of them with similar performance, which indicates that the proposed mechanism is technically feasible for production on currently available hardware and software solutions.

Moreover, the performance of MACP and FlowNAC, based on SDN-enabled architectures, have been also evaluated and demonstrate the usability of these solutions, with similar order of magnitude than other comparable proposals.

Each proposal can be used individually (i.e., as a network virtualization solution, as a mechanism to assign MAC addresses, or as a fine-grained access control solution, respectively) or in conjunction to build a virtualised platform for experimentation (as EHU-OEF) or for production (data center, campus and operator networks).

Additional experimentation has been also conducted in EHU-OEF with promising results, such as a novel forwarding mechanism based on MAC prefixes, called Prefix-based Forwarding Decision (PFD).

As a final remark, this work not only constitutes a valuable research to propose new network virtualization architectures leveraging new emerging technologies SDN and NFV, but it also assists the understanding of past, present and future evolution of these topics.

RESUMEN

Internet se ha convertido en un medio fundamental como fuente de información y comunicación tanto para la interacción social como de negocios. Sin embargo, se han identificado varias limitaciones de la actual Internet a la hora de cumplir con las expectativas generadas debido a su rotundo éxito. En este contexto, la iniciativa de la Internet del Futuro ha emergido como un esfuerzo investigador para superar estas limitaciones.

Para solucionar las limitaciones de Internet (algunas son debidas a su diseño original), es necesario investigar en arquitecturas y propuestas novedosas, tanto aproximaciones evolutivas como totalmente rompedoras. Además, las plataformas de experimentación surgen para proporcionar un entorno realista para validar estas nuevas propuestas a gran escala, y convertirse en una herramienta fundamental para una robusta experimentación y testeo de dichas propuestas. Como principio de diseño, las plataformas experimentales deben de ser ortogonales a la experimentación y su impacto debe reducirse al mínimo. Debido a la necesidad de compartir la misma infraestructura y recursos para testear simultáneamente diversas propuestas de red, la virtualización de red es la clave del éxito.

No hay una definición única para la virtualización de red, ya que ésta depende del escenario objetivo, el cual puede imponer requerimientos específicos debido a sus particularidades. Para poder analizar y comparar adecuadamente las diferentes aproximaciones propuestas para la virtualización de red, se propone una taxonomía nueva. Basado en esta taxonomía, se presenta un estudio de las diversas propuestas realizadas para la virtualización de red. Como resultado de dicho estudio, se han identificado tres tipos principales de virtualización, los cuales se agrupan en dos grandes grupos: las propuestas centradas en el nodo, y las centradas en la red. En el primer grupo los nodos son expuestos como elementos individuales a los operadores virtuales (que en este caso son los investigadores), mientras que en el segundo grupo la infraestructura de red subyacente se abstrae como un todo. Con respecto a

las aproximaciones centradas en el nodo, se proponen dos tipos de virtualización de red: el Nodo Virtual (vNode) y la Virtualización posibilitada por SDN (SDNeV). Por una parte, las soluciones de tipo vNode exponen los dispositivos de red virtualizados con una funcionalidad asociada implementada sobre el propio dispositivo. Por otro lado, las propuestas de tipo SDNeV exponen la programabilidad de los dispositivos de red individuales (tanto físicos como virtuales). Con respecto a las aproximaciones centradas en la red, únicamente se propone un tipo de virtualización de red: el overlay. Las soluciones de overlay se construyen sobre los dispositivos de frontera, los cuales encapsulan el esquema de direccionamiento externo en el espacio de direccionamiento usado internamente por la infraestructura física.

La innovación abierta en la red demanda que la progamabilidad de los recursos de red se exponga a los investigadores. Como consecuencia, la tecnología de Redes Definidas por Software (Software Defined Networks, SDN) se convierte en un facilitador esencial para la innovación en la red mediante la separación de los planos de datos y control a través de un interfaz estándar, y por lo tanto, el tipo de virtualización de red SDNeV es el más apropiado para construir las plataformas experimentales.

Además, se presentan las plataformas experimentales más relevantes, con un foco especial en la forma en la que cada una de ellas permite la investigación en propuestas de red. Este estudio se divide en dos grandes grupos: las propuestas previas a SDN basadas en tecnología tradicional y las plataformas experimentales activadas por SDN. El segundo grupo es el más relevante para el tipo de virtualización de red seleccionado.

Apoyado en los análisis previos y en las actividades de investigación a desarrollar sobre la plataforma de experimentación, se define una lista de requisitos: el aislamiento entre los experimentos (incluyendo el aislamiento seguro entre los mismos), la flexibilidad para experimentar en arquitecturas novedosas y aproximaciones rompedoras, la escalabilidad de la plataforma, la estabilidad de la definición de la capa de experimentación, la transparencia de la plataforma con respecto a los experimentos (incluyendo la eficiencia de la solución evitando cualquier sobrecarga), y el soporte para la investigación en propuestas de red. Las plataformas de experimentación disponibles basadas en las propuestas de virtualización de red actuales no cumplen todos estos requisitos. Por lo tanto, una nueva plataforma de experimentación ortogonal a la experimentación es necesaria, así como una propuesta de virtualización de red que, de forma transparente, soporte la investigación en arquitecturas de red y propuestas novedosas.

Una de las responsabilidades de cualquier plataforma de experimentación es desplegar nuevos servicios y recursos para permitir la experimentación. En este respecto, el trabajo que se está realizando en relación a la Virtualización de Funciones de Red (Network Function Virtualisation, NFV) es relevante para este objetivo. NFV ha sido propuesto para innovar en el área de provisión de servicios, mediante el uso de tecnología estándar de virtualización de computación para consolidar en hardware estándar las funciones de red previamente desplegadas sobre aplicaciones hardware específicas.

Las principales contribuciones de esta tesis son también los elementos clave para construir la plataforma de experimentación: la Virtualización de Red basada en Prefijos de Nivel 2 (Layer 2 Prefix-based Network Virtualisation, L2PNV), un Protocolo para la Configuración de Direcciones MAC (MAC Address Configuration Protocol, MACP), y un sistema de Control de Acceso a Red basado en Flujos (Flow-based Network Access Control, FlowNAC).

L2PNV es un mecanismo de virtualización de red eficiente y escalable que delega en los prefijos de dirección MAC la identificación unívoca de las instancias de red virtual correspondientes. Esta propuesta está basada en un esquema de direccionamiento MAC novedoso, el cual reserva parte de la dirección MAC (un prefijo) para la codificación del identificador de la instancia virtual.

Como consecuencia de no usar las direcciones MAC globalmente administradas, los dispositivos finales tienen que actualizar sus direcciones MAC para ser compatibles con el nuevo esquema de direccionamiento. Por lo tanto, se propone un nuevo mecanismo, llamado MACP, para facilitar la asignación de direcciones MAC. Además, este mecanismo puede también ser usado por otras propuestas de Nivel 2 basadas en esquemas de direccionamiento MAC alternativos, permitiendo de esta forma la innovación en esta línea de investigación.

Por último, la seguridad global tanto de la plataforma de experimentación como de la propuesta para la virtualización de red se mejora añadiendo un nuevo mecanismo, llamado FlowNAC, para controlar de forma granular el acceso basándose en la identidad del usuario y en la política definida por el operador de la red virtual (el investigador en este caso).

En relación con las contribuciones anteriormente citadas, se adoptan unos nuevos principios de diseño basados en arquitecturas posibilitadas por SDN para la implementación de dichas propuestas como Funciones de Red Virtualizadas (Virtual Network Functions, VNFs). Los objetivos de usar estas arquitecturas posibilitadas por SDN para la entrega de servicios son la mejora del rendimiento de las VNFs y facilitar el despliegue de estos nuevos servicios (MACP y FlowNAC) sobre la plataforma de experimentación.

Como resultado (y una contribución adicional), se ha desplegado en la Universidad del Pais Vasco (UPV/EHU) una nueva plataforma experimental, la Plataforma Activada por OpenFlow de EHU (EHU OpenFlow Enabled Facility, EHU-OEF), para experimentar y validar estas propuestas. El principal objetivo de EHU-OEF es permitir la innovación en arquitecturas y propuestas de red.

En este contexto, L2PNV se ha validado de forma experimental y su principal mecanismo para virtualizar la red (la asociación de prefijos MAC) ha sido evaluado. Este mecanismo se ha comparado con otros mecanismos similares, todos ellos con un rendimiento similar, lo que indica que el mecanismo propuesto es técnicamente viable para su puesta en producción con las soluciones hardware y software disponibles actualmente.

Además, el rendimiento de MACP y FlowNAC, basados en arquitecturas posibilitadas por SDN, ha sido evaluado y demuestra la usabilidad de las soluciones, con un orden de magnitud parecido a otras propuestas similares.

Cada propuesta puede ser usada de forma individual (como solución para la virtualización de red, como mecanismo para la asignación de direcciones MAC, o como una solución para el control de acceso granular, respectivamente) o en conjunto para construir una plataforma virtualizada para la experimentación (como EHU-OEF) o para producción (en redes de centro de datos, de campus o de operador).

De forma adicional, se han llevado acabo labores de experimentación en EHU-OEF con resultados prometedores, como es el caso de un novedoso mecanismo para la conmutación de tramas basado en prefijos MAC, llamado Decisión de Conmutación basado en Prefijos (Prefix-based Forwarding Decision, PFD).

Como apunte final, este trabajo no solo constituye una investigación destacable con la propuesta de nuevas arquitecturas para la virtualización de red que se sustentan sobre tecnologías emergentes como SDN y NFV, sino que además permite un mejor entendimiento del pasado, presente y evolución futura de estas líneas de investigación.

LABURPENA

Internet, informaziorako eta komunikaziorako baliabiderik garrantzitsuena bilakatu da, bai elkarrekintza sozialetarako, baita negozioetarako ere. Hala ere, gaur egungo Interneten arrakasta handiak eragindako beharrizanak asetzerakoan, zenbait muga identifikatu dira. Testuinguru horretan, Etorkizuneko Interneta edo Future Internet delako ekimena sortu da, muga horiek gainditu ahal izateko.

Esandako Internetaren muga horiek gainditzeko (zenbait bere jatorrizko diseinuaren ondorio direnak), beharrezkoa da sareen arkitektura eta proposamen berriak ikertzea, gutxikako eboluzio bezala planteatutakoak zein guztiz berritzaileak liratekeenak ere. Horrela, proposamen berri horiek eskala-handian balioztatzeko ingurune errealista modura, esperimentaziorako plataformak sortu dira, sareen esperimentuak egiteko eta testerako baliabiderik garrantzitsuenak bilakatu direnak. Diseinurako printzipio bezala, esperimentaziorako plataforma batek zera jarraitu behar du, esperimentazioarekiko ortogonala izatea eta bere eragina ahalik eta txikiena izatea. Hortaz, sareetarako proposamen ezberdinak azpiegitura eta baliabide berdinetan aldi berean probatu ahal izateko, sare horien birtualizazioa egin ahal izatea funtsezkoa da.

Ez dago sare-birtualizaziorako definizio bateraturik, erabilpen egoeraren araberakoa delako; egoera bakoitzak beharrizan zehatzak izan ditzake, bere ezaugarrien araberakoak. Sare-birtualizaziorako proposamenen azterketa eta alderatzea egokiro egin ahal izateko, taxonomia berri bat aurkezten dugu hemen. Taxonomia horretan oinarrituta, sare-birtualizaziorako proposamenen ikerketa bat azaltzen da ere. Ondorioz, hiru birtualizazio modu nagusi identifikatu dira, bi kategoriatan taldekatuak: nodoetan oinarrituak eta sareetan oinarrituak. Lehenengo birtualizazio moduan, sareko nodoak banakako elementu bezala aurkezten zaizkie operadore birtualei (kasu honetan, ikertzaileei); aldiz, bigarrenean, azpiegiturako sarea guztiz ezkutatzen da. Nodoetan oinarritutako birtualizazioetan, bi modu proposatu dira: Virtual Node (vNode) modua eta SDN-

enabled Virtualiation (SDNeV) deiturikoa. Alde batetik, vNode modukoetan sareko gailu birtualak adierazten dira, bakoitzak funtzionaltasun bat inplementatua duena beregan. Aldiz, SDNeV proposamenetan, sareko gailuen (fisikoen zein birtualen) programagarritasuna ahalbidetzen da. Sareetan oinarritutako birtualizazioetan, sare-birtualizazio modu bakarra proposatu da: overlay deiturikoa. Overlay moduko proposamenek ertzeko gailuak bezala ezagutzen direnak erabiltzen dituzte, kanpoko helbideratze eskemak azpiegitura fisikoan erabiltzen diren barne-helbideetan kaptsulatzen dituztenak.

Sareen berrikuntzak, ikertzaileei eskainitako baliabideen programagarritasuna ahalbidetzea eskatzen du. Ondorioz, Software Defined Networking edo SDN teknologia nahitaezkoa bilakatzen da sareen berrikuntzan datuen eta kontrolaren planoak interfaze estandarrarekin banandu ahal izateko, eta horregatik, SDNeV moduko sare-birtualizazioa da egokiena esperimentaziorako plataformak eraikitzeko.

Gainera, dokumentu honetan gaur egungo esperimentaziorako plataformarik garrantzitsuenak aurkezten dira, bakoitzak sareekiko ikerkuntza nola ahalbidetzen duen zehaztuz. Hortaz, ikerketa bi ataletan banatua dago: SDN-aurreko proposamenak (aurreko teknologietan oinarrituak) eta SDN duten plataformak. Azkena da talderik garrantzitsuena, aukeratutako sare-birtualizaziorako moduaren arabera.

Aurreko azterketetan eta esperimentaziorako plataformetan egin beharreko ikerkuntza-ekintzetan oinarrituta, zenbait beharrizan definitzen dira: esperimentuen arteko isolamendua (isolamendu segurua barnean hartzen duena), arkitektura berrietan eta proposamen guztiz berritzaileetan ikertzeko malgutasuna, plataformaren eskalagarritasuna, esperimentu bakoitzari dagokion slice edo geruzaren egonkortasuna, plataformaren esperimentuekiko gardentasuna (eraginkortasuna barne, gainkargak ekidinez), eta sareen ikerkuntzarako laguntza. Gaur egungo sare-birtualizaziorako proposamenetan oinarritutako plataforma erabilgarriek, ez dituzte aipatutako beharrizan guztiak betetzen. Hortaz, esperimentaziorako plataforma ortogonal berri bat behar da, eta aldi berean, sare-birtualizaziorako proposamen berri bat, sareen arkitektura eta proposamen berrien ikerkuntza modu gardenean ahalbidetzen duena.

Edozein esperimentaziorako plataformak bete beharreko ezaugarri bat, baliabide eta zerbitzu berriak hedatu ahal izatea da, esperimentazioa ahalbidetzeko. Zentzu horretan, Network Functions Virtualisation edo NFV delakoaren inguruko lanak oso garrantzitsuak dira. NFV, zerbitzuen hornitzea egiteko modua berritzeko proposatua izan da; konputazioaren birtualizaziorako teknologia estandarraren bidez, helburu orokorreko hardwaretan lehen neurrira egindako produktuetan inplementatzen ziren funtzioak gauzatzen ditu.

Tesi honen ekarpenik garrantzitsuenak esperimentaziorako plataformak eratzeko oinarrizko zatiak dira: Layer 2 Prefix-based Network Virtualisation (L2PNV) delakoa, MAC Address Configuration Protocol (MACP) eta Flow-based Network Access Control (FlowNAC).

L2PNV sare-birtualizaziorako mekanismo eraginkor eta eskalagarri bat da, MAC helbideen aurrizkietan oinarritzen dena sareko instantzia birtualak identifikatu ahal izateko. Proposamen honetan MAC helbideratze eskema berri bat ere aurkezten da, MAC helbidearen zati bat (aurrizki bat) erabiltzen duena instantzia birtualen identifikatzaile modura.

MAC helbide globalak ez erabiltzearen ondorio bezala, amaierako gailuek MAC helbide propioak eguneratu behar dituzte helbideratze eskema berriarekiko bateragarriak izan daitezen. Horretarako, beraz, mekanismo berri bat proposatzen da, MACP deiturikoa, MAC helbideak egokitzeko. Gainera, proposamen hau beste MAC helbideratze eskema alternatiboetan oinarritutako Layer 2 edo 2. mailako proposamenetan erabili ahalko litzateke, arlo horretan ikerkuntza berria baliatuz.

Are gehiago, esperimentaziorako plataformaren zein sare-birtualizazioaren segurtasun orokorra ere hobetzen da, mekanismo berri bati esker, FlowNAC deiturikoa; zehaztasun handiko sarbide-kontrola egiten du horrek, erabiltzaileen identitatean eta operadore birtualen (ikertzaileen) politiketan oinarrituta.

Aurretik esandako ekarpenekin lotuta, SDN darabilten arkitekturetan oinarritutako diseinu printzipio berriak erabili dira proposamen berriak gauzatzeko, Virtual Network Functions (VNFs) direlakoen bitartez. SDN darabilten arkitektura horiek zerbitzuak hornitzeko erabiltzearen helburua zera da, VNFen eraginkortasuna hobetzea eta zerbitzu berrien (hots, MACP eta FlowNAC zerbitzuen) hornikuntza eta hedapena ahalbidetzea esperimentaziorako plataforman.

Ondorioz (eta tesiaren ekarpen gehigarri bezala), Euskal Herriko Unibertsitatean (UPV/EHUn) esperimentaziorako plataforma berria hedatu da, EHU OpenFlow Enabled Facility (EHU-OEF) deiturikoa, proposamen guztiak balioztatu ahal izateko. EHU-OEFren helburu nagusia da sareetarako arkitekturen eta proposamenen berrikuntza ahalbidetzea.

Testuinguru horretan, L2PNV esperimentuekin balioztatua izan da eta sare-birtualizaziorako mekanismo nagusia (MAC helbideen alderatzea) ebaluatua izan da. Antzeko beste mekanismoekin ere alderatu izan da, neurriko eraginkortasuna dutenak; horrek adierazten duena zera da, mekanismoa teknikoki erabilgarria dela produkzio inguruneetan gaur egungo hardware eta software elementuekin.

Bestalde, MACP eta FlowNACen eraginkortasuna ebaluatua izan da SDN darabilten arkitekturetan, eta hortaz, ebazpen horien erabilgarritasuna demostratua izan da, alderagarriak diren beste proposamenen antzeko emaitzekin.

Proposamen bakoitza bakarka erabili daiteke (sare-birtualizaziorako proposamen bezala, MAC helbideen egokitze-mekanismo modura edo zehaztasun handiko sarbide-kontrolerako, hurrenez hurren) edo guztiak batera, esperimentaziorako plataforma birtualak sortzeko (EHU-OEF bezala) edo produkzio inguruneetan (datacenter, campus edo hornitzaileen sareetan).

EHU-OEF plataforman esperimentazio gehiago ere egin da, emaitza esanguratsuekin: zehazki, MAC aurrizkietan oinarritutako bidalketa-mekanismo berri bat, Prefix-based Forwarding Decision (PFD) deiturikoa.

Amaitzeko, lan hau ez da soilik sare-birtualizaziorako arkitektura berriak proposatzen dituen ikerketa baliotsu bat, SDN eta NFV teknologia berriak darabiltzana; aldiz, landutako ikerkuntza lerroen aurreko, egungo eta etorkizuneko egoerak ulertzeko tresna lagungarria da.

RESUMEN EXTENDIDO

1) CONTEXTUALIZACIÓN

Internet se ha convertido en un medio fundamental como fuente de información y comunicación tanto para la interacción social como de negocios. Sin embargo, se han identificado varias limitaciones de la actual Internet a la hora de cumplir con las expectativas generadas debido a su rotundo éxito. En este contexto, la iniciativa de la Internet del Futuro ha emergido como un esfuerzo investigador para superar estas limitaciones.

Para solucionar las limitaciones de Internet (algunas son debidas a su diseño original), es necesario investigar en arquitecturas y propuestas novedosas, tanto aproximaciones evolutivas como totalmente rompedoras. Además, las plataformas de experimentación surgen para proporcionar un entorno realista para validar estas nuevas propuestas a gran escala, y convertirse en una herramienta fundamental para una robusta experimentación y testeo de dichas propuestas. Como principio de diseño, las plataformas experimentales deben de ser ortogonales a la experimentación y su impacto debe reducirse al mínimo. Debido a la necesidad de compartir la misma infraestructura y recursos para testear simultáneamente diversas propuestas de red, la virtualización de red es la clave del éxito.

2) OBJETIVOS DE LA INVESTIGACIÓN

El principal objeto de la investigación desarrollada en esta tesis es doble: por una parte, la investigación en nuevas arquitecturas y propuestas de red, y por otro lado, facilitar la posibilidad de investigar en dichas propuestas y arquitecturas rompedoras. A continuación se detallan los objetivos concretos que se van a cubrir:

- Despliegue de una plataforma experimental en el campus de la Universidad del Pais Vasco (UPV/EHU).

- Definición de una taxonomía para facilitar la comprensión y comparativa del estado de arte actual de las diferentes propuestas para la virtualización de red.
- Diseño e implementación de una nueva propuesta para la virtualización de red que cubra los requisitos planteados.
- Diseño e implementación de un mecanismo que facilite la asignación de nuevos esquemas de direccionamiento MAC.
- Diseño e implementación de un mecanismo seguro para el control de acceso a la plataforma de experimentación basado en la identidad de los usuarios y en la política definida por los investigadores.
- Definición de los principios básicos para el diseño óptimo de nuevos servicios y protocolos como funciones de red virtualizadas.

3) ANÁLISIS DEL ESTADO DEL ARTE

No hay una definición única para la virtualización de red, ya que ésta depende del escenario objetivo, el cual puede imponer requerimientos específicos debido a sus particularidades. La Figura 1 demuestra la arquitectura definida.

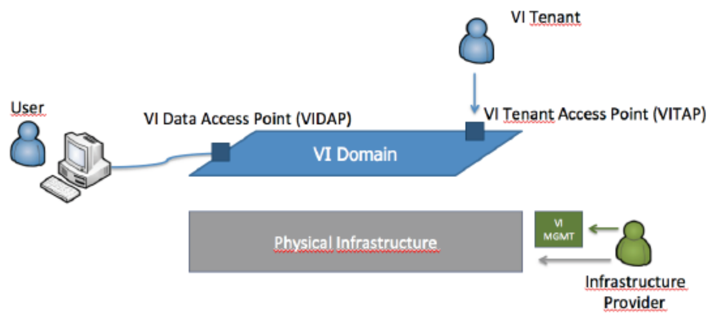


Figura 1: Arquitectura definida para la Virtualización de Red

Para poder analizar y comparar adecuadamente las diferentes aproximaciones propuestas para la virtualización de red, se propone una taxonomía nueva. Basado en esta taxonomía, se presenta un estudio de las diversas propuestas realizadas para la virtualización de red.

Como resultado de dicho estudio, se han identificado tres tipos principales de virtualización, los cuales se agrupan en dos grandes grupos: las propuestas centradas en el nodo, y las centradas en la red. En el primer grupo los nodos son expuestos como elementos individuales a los operadores virtuales (que en este caso son los investigadores), mientras que en el segundo grupo la infraestructura de red subyacente se abstrae como un todo. Con respecto a las aproximaciones centradas en el nodo, se proponen dos tipos de virtualización de red: el Nodo Virtual (vNode) y la Virtualización posibilitada por SDN (SDNeV). Por una parte, las soluciones de tipo vNode exponen los dispositivos de red virtualizados con una funcionalidad asociada implementada sobre el propio dispositivo. Por otro lado, las propuestas de tipo SDNeV exponen la programabilidad de los dispositivos de red individuales (tanto físicos como virtuales). Con respecto a las aproximaciones centradas en la red, únicamente se propone un tipo de virtualización de red: el overlay. Las soluciones de overlay se construyen sobre los dispositivos de frontera, los cuales encapsulan el esquema de direccionamiento externo en el espacio de direccionamiento usado internamente por la infraestructura física.

La innovación abierta en la red demanda que la progamabilidad de los recursos de red se exponga a los investigadores. Como consecuencia, la tecnología de Redes Definidas por Software (Software Defined Networks, SDN) se convierte en un facilitador esencial para la innovación en la red mediante la separación de los planos de datos y control a través de un interfaz estándar, y por lo tanto, el tipo de virtualización de red SDNeV es el más apropiado para construir las plataformas experimentales.

Además, se presentan las plataformas experimentales más relevantes, con un foco especial en la forma en la que cada una de ellas permite la investigación en propuestas de red. Este estudio se divide en dos grandes grupos: las propuestas previas a SDN basadas en tecnología tradicional y las plataformas experimentales activadas por SDN. El segundo grupo es el más relevante para el tipo de virtualización de red seleccionado.

Apoyado en los análisis previos y en las actividades de investigación a desarrollar sobre la plataforma de experimentación, se define una lista de requisitos: el aislamiento entre los experimentos (incluyendo el aislamiento seguro entre los mismos), la flexibilidad para experimentar en arquitecturas novedosas y aproximaciones rompedoras, la escalabilidad de la plataforma, la estabilidad de la definición de la capa de experimentación, la transparencia de la plataforma con respecto a los experimentos (incluyendo la eficiencia de la solución evitando cualquier sobrecarga), y el soporte para la investigación en propuestas de red. Las plataformas de experimentación disponibles basadas en las propuestas de virtualización de red actuales no cumplen todos estos requisitos. Por lo tanto, una nueva plataforma de experimentación ortogonal a la experimentación es necesaria, así como una propuesta de virtualización de red que, de forma transparente, soporte la investigación en arquitecturas de red y propuestas novedosas.

Una de las responsabilidades de cualquier plataforma de experimentación es desplegar nuevos servicios y recursos para permitir la experimentación. En este respecto, el trabajo que se está realizando en relación a la Virtualización de Funciones de Red (Network Function Virtualisation, NFV) es relevante para este objetivo. NFV ha sido propuesto para innovar en el área de provisión de servicios, mediante el uso de tecnología estándar de virtualización de computación para consolidar en hardware estándar las funciones de red previamente desplegadas sobre aplicaciones hardware específicas.

4) CONTRIBUCIONES

Las principales contribuciones de esta tesis son también los elementos clave para construir la plataforma de experimentación: la Virtualización de Red basada en Prefijos de Nivel 2 (Layer 2 Prefix-based Network Virtualisation, L2PNV), un Protocolo para la Configuración de Direcciones MAC (MAC Address Configuration Protocol, MACP), y un sistema de Control de Acceso a Red basado en Flujos (Flow-based Network Access Control, FlowNAC).

L2PNV es un mecanismo de virtualización de red eficiente y escalable que delega en los prefijos de dirección MAC la identificación unívoca de las instancias de red virtual correspondientes (Figura 2).

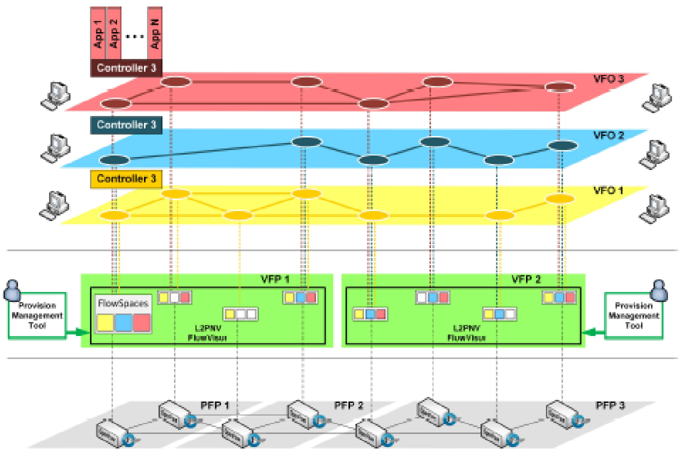


Figura 2: Redes Virtuales definidas por L2PNV

Esta propuesta está basada en un esquema de direccionamiento MAC novedoso, el cual reserva parte de la dirección MAC (un prefijo) para la codificación del identificador de la instancia virtual (Figura 3).

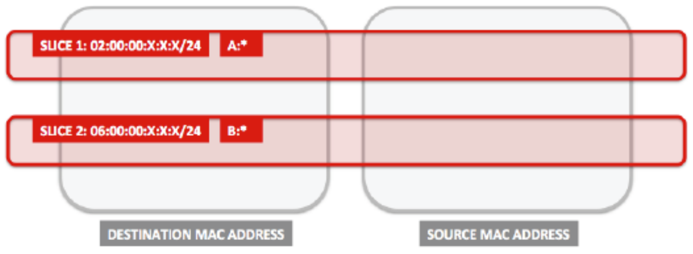


Figura 3: Definición de instancia virtual en L2PNV

Como consecuencia de no usar las direcciones MAC globalmente administradas, los dispositivos finales tienen que actualizar sus direcciones MAC para ser compatibles con el nuevo esquema de direccionamiento, tal y como se muestra en la Figura 4. Por lo tanto, se propone un nuevo mecanismo, llamado MACP, para facilitar la asignación de direcciones MAC.

Además, este mecanismo puede también ser usado por otras propuestas de Nivel 2 basadas en esquemas de direccionamiento MAC alternativos, permitiendo de esta forma la innovación en esta línea de investigación.

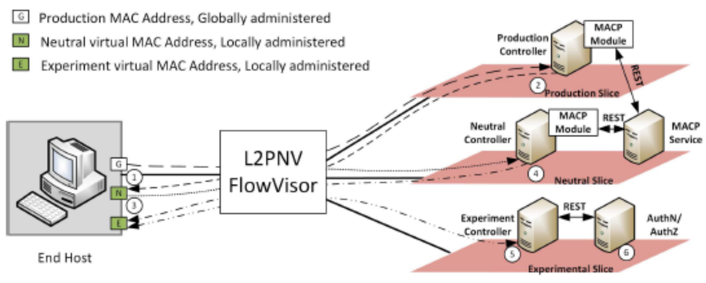


Figura 4: Proceso de asignación de direcciones MAC a través de MACP

Por último, la seguridad global tanto de la plataforma de experimentación como de la propuesta para la virtualización de red se mejora añadiendo un nuevo mecanismo, llamado FlowNAC, para controlar de forma granular el acceso basándose en la identidad del usuario y en la política definida por el operador de la red virtual (el investigador en este caso). El proceso completo se muestra en la Figura 5.

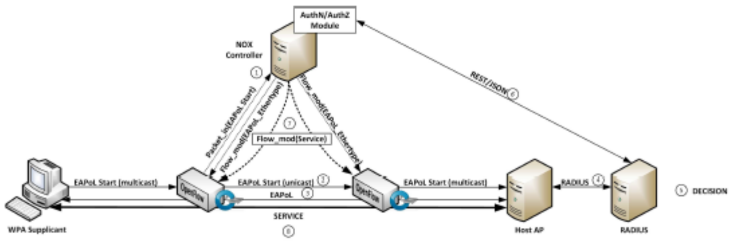


Figura 5: Proceso de autenticación y autorización definido por FlowNAC

En relación con las contribuciones anteriormente citadas, se adoptan unos nuevos principios de diseño basados en arquitecturas posibilitadas por SDN para la implementación de dichas propuestas como Funciones de Red Virtualizadas

(Virtual Network Functions, VNFs). Los objetivos de usar estas arquitecturas posibilitadas por SDN para la entrega de servicios son la mejora del rendimiento de las VNFs y facilitar el despliegue de estos nuevos servicios (MACP y FlowNAC) sobre la plataforma de experimentación.

5) VALIDACIÓN

Como resultado (y una contribución adicional), se ha desplegado en la Universidad del Pais Vasco (UPV/EHU) una nueva plataforma experimental (Figura 6), la Plataforma Activada por OpenFlow de EHU (EHU OpenFlow Enabled Facility, EHU-OEF), para experimentar y validar estas propuestas. El principal objetivo de EHU-OEF es permitir la innovación en arquitecturas y propuestas de red.

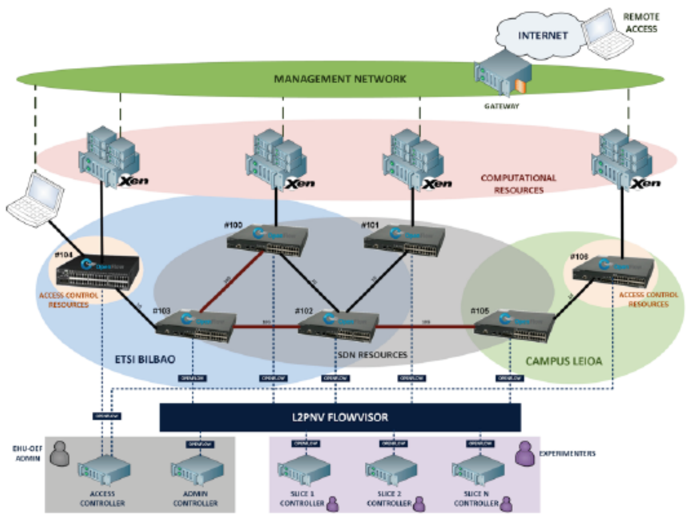


Figura 6: Plataforma experimental EHU-OEF

En este contexto, L2PNV se ha validado de forma experimental y su principal mecanismo para virtualizar la red (la asociación de prefijos MAC) ha sido evaluado. Este mecanismo se ha comparado con otros mecanismos similares, todos ellos con un rendimiento similar, lo que indica que el mecanismo propuesto es técnicamente viable para su puesta en producción con las soluciones hardware y software disponibles actualmente.

Además, el rendimiento de MACP y FlowNAC, basados en arquitecturas posibilitadas por SDN, ha sido evaluado y demuestra la usabilidad de las soluciones, con un orden de magnitud parecido a otras propuestas similares.

Cada propuesta puede ser usada de forma individual (como solución para la virtualización de red, como mecanismo para la asignación de direcciones MAC, o como una solución para el control de acceso granular, respectivamente) o en conjunto para construir una plataforma virtualizada para la experimentación (como EHU-OEF) o para producción (en redes de centro de datos, de campus o de operador).

De forma adicional, se han llevado acabo labores de experimentación en EHU-OEF con resultados prometedores, como es el caso de un novedoso mecanismo para la conmutación de tramas basado en prefijos MAC, llamado Decisión de Conmutación basado en Prefijos (Prefix-based Forwarding Decision, PFD).

6) CONCLUSIONES

Este apartado detalla la lista completa de contribuciones aportadas por esta tesis. De forma adicional a las tres contribuciones anteriormente detalladas, hay otra serie de contribuciones que caben destacar.

- La plataforma experimental EHU-OEF desplegada en la Universidad del País Vasco (UPV/EHU).
- Una nueva taxonomía para la virtualización de red.
- La propuesta L2PNV para virtualización de red.
- La propuesta MACP para la asignación y configuración de direcciones MAC.
- La propuesta PFD para la conmutación de tramas basadas en prefijos MAC.
- La propuesta FlowNAC para el control de acceso granular a la red basada en flujos.
- Una arquitectura para soluciones NFV apoyada en la tecnología SDN.

Los resultados de la presente investigación han sido publicados en 17 artículos (12 como primer autor), de los cuales tres son en revistas con JCR y el resto en conferencias internacionales. Además, se han publicado los resultados en 8 conferencias naciones. Como resultados de la implementación, se han presentado también 4 demostradores en conferencias internaciones. Finalmente, el trabajo presentado ha sido pieza fundamental en el contexto de dos proyectos europeos (FP7) y 6 proyectos de ámbito nacional.

Como apunte final, este trabajo no solo constituye una investigación destacable con la propuesta de nuevas arquitecturas para la virtualización de red que se sustentan sobre tecnologías emergentes como SDN y NFV, sino que además permite un mejor entendimiento del pasado, presente y evolución futura de estas líneas de investigación.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

ACL	Access Control List
DAL	Device and resource Abstraction Layer
EHU-OEF	EHU OpenFlow Enabled Facility
FlowNAC	Flow-based Network Access Control
InP	Infrastructure Provider (as defined by the 4WARD FP7 project)
L2PNV	Layer 2 Prefix-based Network Virtualisation
MACP	MAC Address Configuration Protocol
NAC	Network Access Control
NFV	Network Functions Virtualization
NIC	Network Interface Card
NSAL	Network Services Abstraction Layer
NV	Network Virtualisation
PFD	Prefix-based Forwarding Decision
PFP	Physical Facility Provider
PIP	Physical Infrastructure Provider (as defined by the GEYSERS FP7 project)
PNAC	Port-based Network Access Control
PR	Physical Resources
SDN	Software Defined Networking
SDNeV	SDN-enabled Virtualization
VFP	Virtual Facility Provider
VFO	Virtual Facility Operator

VI	Virtual Instance
VIAP	Virtual Instance Access Point
VIDAP	Virtual Instance Data Access Point
VIO	Virtual Infrastructure Operator (as defined by the GEYSERS FP7 project)
VIP	Virtual Infrastructure Provider (as defined by the GEYSERS FP7 project)
VITAP	Virtual Instance Tenant Access Point
VM	Virtual Machine
VNF	Virtual Network Function
VNO	virtual network operator (as defined by the 4WARD FP7 project)
vNode	Virtual Node
VNP	virtual network provider (as defined by the 4WARD FP7 project)
VPN	Virtual Private Network
VR	Virtual Resources
VRF	Virtual Routing and Forwarding

1 INTRODUCTION

This thesis presents results of the research on novel network architectures and proposals, and about enabling the research on novel network architectures and proposals. To this aim, Network Virtualisation (NV) is perceived as the key enabler to achieve both objectives. More concretely, the main research topics covered in this thesis are NV, Security, Experimental Facilities, Software Defined Networking (SDN) and Network Functions Virtualization (NFV).

Therefore, one of the main contributions from this thesis is a novel proposal for virtualising the network. To test and validate this proposal the experimental facility use case is selected as the main target scenario to consider, due to the fact that it's one of the most challenging scenarios for any NV approach.

This introduction first provides background of the research work, then presents the motivations and research objectives from this thesis, and finally, outlines the remainder of this document.

1.1 CONTEXTUALISATION

For the recent decades, the Internet has become an universal enabler for continuous innovation in all areas of IT and human activity. It is become the essential medium for information and for communication of both social and business interactions. This remarkable success has generated higher expectations for future services, such as reliability, availability and interoperability, which the Internet may not be capable of addressing properly. Moreover, the number of nodes is expected to increase beyond 100 billion soon [1], resulting in an even more challenging situation.

Currently, increasing numbers of users interact with each other and are acting as real content providers, which has greatly increased the amount of content stored and transported and has created another challenging issue for the Internet. The resulting increase in demand for capacity can no longer be addressed through overdimensioning. In this context, the Future Internet (FI) [2] initiative has emerged as a research effort aimed at generating the tools to overcome the current limitations of the Internet [3], such as the processing and handling of large amounts of data, deficiencies of storage and transmission and restrictions in the ability to control system functions.

In a nutshell, the FI is a set of research activities, projects and frameworks intended to promote and to develop new architectures and tools that replace the current Internet. Significant efforts have been conducted worldwide to develop the FI. The most relevant funding frameworks and research projects in the FI arena are detailed in a survey [2] and include the Future Internet Architecture[1] in the USA (the NSF FIA programme built on a prior programme, Future Internet Design, FIND[2]), the Future Internet Assembly (FIA) in Europe[3]and the AKARI project [4] in Japan.

To solve the current limitations of the Internet, it is necessary to promote innovation and the success of the FI; however, any novel proposal demands robust experimentation and tests of large-scale conditions. In this context, the experimental facilities emerge to provide a realistic environment to validate these new approaches. Both GENI [5] in USA and EU FIRE [6] in Europe are the prime examples of these facilities. There are several other facilities spread around the world, and each of them had a different set of requirements, which makes them optimal for experimentation on the research topics that were considered at design time.

Deploying an experimental facility is challenging from a technical standpoint. At a minimum, the facility has to address the following requirements: (1) the flexibility to support clean slate proposals imposing the smallest number of conditions possible, (2) the capacity to run multiple experiments simultaneously in the same infrastructure, (3) a complete isolation between experiments (including the secured isolation between them) and (4) the ability to virtualise the physical

1 NSF Future Internet Architecture: http://www.nets-fia.net/
2 NSF Future Internet Design: http://www.nets-find.net/
3 Future Internet Assembly: http://www.future-internet.eu

infrastructure as needed to share it properly among researchers. Legacy technology, however, lacks full support for these requirements. For instance, an implementation based on the VLAN Bridging standard (IEEE 802.1Q [7]) imposes several hard restrictions to any novel approach, such as the support of a broadcast domain, the learning of MAC addresses or the Spanning Tree Protocol family (i.e., STP, RSTP and MSTP) to avoid loops. Therefore, a more generic facility without such restrictions is desirable.

Two of the most important requirements in any experimental facility are flexibility and isolation. The concept of SDN [8] provides the flexibility lacking in networking equipment with behaviour that is defined by hardware or a vendor-dependent firmware. The SDN technology is an enabler to achieve the expected flexibility and isolation. This isolation is necessary for any type of experiment so the experiment is not affected by external entities. This way, independence and a minimum variability will be assured. However, the isolation is not an exclusive requirement for experimental facilities, and there are other scenarios that could also benefit from it, such as campus, operator, and data center networks.

SDN has been one of the pillars of innovation in network infrastructures, allowing the decoupling of the control and data planes through an open and standard interface that enables the programmability of the network. OpenFlow [9], ForCES [10], and I2RS [11] are some examples of SDN technology. SDN has also contributed to the virtualisation of the network infrastructure, providing the foundation to isolate, abstract, and share the network resources. Modifying the behaviour of networking equipment allows researchers to conduct all types of experiments involving novel architectures and protocols to overcome current limitations (e.g., mobility and security features) with radical new approaches instead of trying to patch the current Internet with only incremental changes layered on top of the existing architecture. By means of supporting NV, different approaches demanding distinct requirements could be simultaneous deployed in the same infrastructure obtaining the optimal behaviour from the network.

OpenFlow [12] is one of the most successful SDN technologies. Created at Stanford University, this technology is currently controlled by the Open Networking Foundation (ONF)[4], which is composed of hardware vendors, service providers and telecommunication companies. OpenFlow tries to strip the high-level routing decisions (control paths/plane) from the fast packet forwarding (data paths/plane) in a switch or router and thus moves the control path/plane to a separate controller. The protocol defines how the OpenFlow Switch and the OpenFlow Controller communicate by specifying the messages involved. OpenFlow is a technology designed to achieve flow level programmability, which is defined as a set of matching fields, instructions and counters.

To optimise the usage of the network infrastructure, similarly to the optimisation of computing resources, the idea of virtualisation has jumped from the host to the network. Although different definitions have been proposed for NV depending on the target scenario, such as the VNRG[5], [13], [14], the basic idea behind this concept is the abstraction of the actual network as a virtual network instance delegated to another single administrative entity. Many different approaches have been proposed with dissimilar requirements, but there are some fundamental properties common to all of them: the sharing of physical resources concurrently between all the virtual networks, the isolation between different virtual network instances, and the abstraction of resources as virtual instances. Depending on the target scenario (e.g., experimental facility, campus, operator or data center networks) some additional characteristics could be included as mandatory. Due to the dissimilar criteria proposed when organising the NV proposals, in some occasions oriented to specific scenarios, it could be hard to differentiate and properly compare all the approaches.

In this context, OpenFlow arises as a promising technology that enables the virtualisation of the network. In this case, virtualising a network involves the gathering of all hardware/software and network functionalities under a single administrative entity that can delegate parts of those resources, known as slices. Network virtualisation can be achieved in OpenFlow by distributing the flows in slices, where the control

4 Open Networking Foundation (ONF): https://www.opennetworking.org
5 Virtual Networks Research Group (VNRG): https://irtf.org/concluded/vnrg

plane associated to each slice is delegated to a different controller. Thus, exposing the programmability of the virtual network instance to the corresponding administrative entity. Due to this capability, using OpenFlow to build the NV approach is highly beneficial to several areas. For instance, its adoption in current experimental facilities (e.g., OFELIA [15], GENI [5]) allows several experiments to run at the same time on the same physical infrastructure. In this scenario, OpenFlow enables new protocols to be tested without affecting other experiments or the production traffic. However, the available facilities based on current NV proposals can affect the experimentation depending on the target scenario and the actual proposal under test, and therefore, they present some limitations, including some security concerns. This effect should be minimised to be as transparent as possible to the experiments, as if there were no virtual environment. This also means that the experimental facilities are designed based on a set of requirements, with the main goal to be orthogonal to the research itself.

One of the responsibilities of any experimental facility is to deploy new services and resources to enable the experimentation. In relation to these supporting activities, the ongoing work on NFV[6] must be considered as a hot research topic in the area of service delivery. Service provisioning is often based on proprietary hardware appliances, which imposes some restrictions when trying to deploy new network services, such as capacity and availability. In this scenario, the network infrastructure is not flexible enough to accommodate new services or migrate them to other locations due to its dependence on the physical appliances. NFV has been proposed to innovate in the service delivery arena by using standard computing virtualisation technology to consolidate in commodity hardware (i.e., standard high volume servers, storage, and switches) the functions previously performed by specific hardware appliances. Virtual Network Functions (VNFs), which compose the service chain, are the basic elements to achieve the complete virtualisation of service delivery and are commonly based on computing resources.

6 Network Functions Virtualisation (NFV): `http://www.etsi.org/technologies-clusters/technologies/nfv`

Major standardisation efforts of the emerging NFV technology are being led by the European Telecommunications Standards Institute (ETSI), where the NFV Industry Specification Group (ISG) has recently published 11 NFV specifications, including the NFV architecture [16]. The defined architecture focuses on the aspects unique to virtualisation, such as the transformation of the management and orchestration of VNFs, rather than common challenges to both physical and virtualised NFs, such as the control and operation of the end-to-end network service. Moreover, the NFV ISG is also coordinating and promoting public demonstrations of proofs of concept (PoCs)[7] that illustrate key aspects of NFVs, such as scalability, multi-tenancy, and migration issues.

This interest in NFV is twofold. On the one hand, the deployment of services as VNFs on the experimental facility could benefit from the outcomes of NFV related proposals. On the other hand, experimentation of NFV proposals must be supported by the experimental facility. Moreover, SDN and NFV are complementary technologies, and each one can leverage off the other to improve the flexibility and simplicity of networks and service delivery over them. For this aim, new architectures and interfaces between them are needed, and several proposals are emerging (e.g., NFV Proofs of Concept, T-NOVA [17] and UNIFY [18]).

1.2 MOTIVATION

The motivation of this thesis comes from the analysis of the aforementioned contextualisation. As previously exposed, the current Internet presents some architectural limitations that requires either drastic changes with new radical approaches or clean slate proposals. One of the keys to success for these novel solutions is the extensive testing in a realistic scenario to both validate the approaches and demonstrate their benefits for real deployments. Although, at least in Europe, the FI initiative has evolved into different specific research areas (e.g., IoT, 5G, and Big Data) and the idea of one-for-all experimental facilities

7 ETSI ISG for NFV, NFV Proofs of Concept: http://www.etsi.org/technologies-clusters/technologies/nfv/nfv-poc

seems to be quite utopic, the reality is that some kind of environment for large-scale experimentation would be needed. The experimental facility is the response to this demand.

Currently, several experimental facilities have been deployed worldwide (e.g., in USA and in Europe). They have been designed based on a different set of requirements depending on the target scenario and the focus of experimentation to be supported. However, these facilities present some limitations for networking research due to the NV solution adopted. Most of them rely on VLAN tags to isolate the virtual network instances, which, for instance, prevents to research on campus and operator networks (concretely VLAN tags are extensively used on access networks), since this tag is not further exposed transparently to the experiment. Moreover, the actual list of requirements established as preliminary conditions is not fulfiled by any of those experimental facilities. Due to the target scenarios to be investigated and the research topics to be covered, the imposed requirements are: the isolation between experiments (including the secured isolation between them), the flexibility to experiment on novel architectures and clean slate approaches, the scalability of the facility, the stability of the experiment slice definition, the transparency of the facility with regard to the experiments (including the efficiency, avoiding any overhead), and the support for research on networking. Following with the example of VLAN tags used above: first, the flexibility is restricted since the VLAN space cannot be exposed to the experiment (at least, transparently); second, the scalability is limited for large-scale experimentation (12 bit, up to 4096 slices); and third, the transparency is compromised both at data plane and control plane (a tag is added).

As previously mentioned, the NV approach used is closely related to the ability of the experimental facility to research on networking. Therefore, one of the most fundamental aspects to consider when designing the experimental facility to really support novel architectures and networking proposals, is the NV solution. There are lots of NV proposals and due to the different aspects and scenarios that they cover, it is difficult to properly understand and compare all these proposals. Therefore, a means to categorize and organise the current state of the art is desired. Based on this study, a novel NV approach to deal with all the requirements imposed to the experimental facility is also needed.

As a research topic on clean slate approaches, the investigation on novel MAC addressing schemes (such as [19]) has been considered. As a consequence, on the one hand, the NV solution must support the experimentation on new addressing schemes. On the other hand, a mechanism to facilitate the deployment of these novel schemes is desired. As a result of this research line, new proposals are expected to demonstrate goodness of this approach.

Considering the currently deployed experimental facilities, there are some security concerns that must be overcome. Concretely, although the isolation within the network is provided by the NV approach, the data plane access from end devices to the slice is not properly secured. In those facilities, is quite simple to introduce data traffic to other slices, since the end user is the only responsible to properly configure the access to the slice, and no further check is performed. Therefore, a mechanism to enhance the overall security of the experimental facility is needed (mainly related to the data plane, since some support is available at control plane). The main challenge is how this access control could be supported and integrated with the NV solution proposed for the facility.

Also related to this latter idea of how to integrate and deploy new functionalities and services as VNFs on the experimental facility, some kind of guidance is desired. Additionally, it could be beneficial to learn from the NFV related proposals already demonstrated to improve the design of the facility. Moreover, the design of new services as VNFs can benefit from leveraging on SDN technology to improve their adoption by NFV based platforms.

1.3 RESEARCH OBJECTIVES

The principal objective of this thesis is twofold: the research on novel network architectures and solutions, and enabling experimental validation of the proposed solutions. This means that both the results of the research on networking topics and an experimental facility on which they could be tested in a realistic environment are expected outcomes from this thesis. Although it seems that the chronological order is the just the opposite (first the experimental facility, and then the networking research), the truth is that building the experimental facility is really a huge challenge from the networking perspective. As a consequence, the research on novel networking proposals (e.g., a NV approach) has been the key of success to actually assemble the experimental facility, and vice versa, building the facility to meet the imposed requirements has driven the research activity.

Based on the previous analysis, there are several aspects to improve and some limitations to overcome. Therefore, some concrete research objectives and research questions *(RQ)* of the presented research are defined as follows:

- Build a campus-wide experimental facility that covers all the requirements imposed to enable the research on novel networking architectures and proposals. The facility must support the secured isolation between experiments, be flexible, be scalable, be stable at slice definition, be transparent to the focus of the experiment, and basically, support the research on networking approaches by exposing the programmability of the network to the researchers. It must also support both experimental and production traffic over the same physical infrastructure. Some fundamental pieces (i.e., a NV approach, a secured access control system and an automatic mechanism for end host configuration) are needed and detailed as additional concrete objectives.

- Define a NV taxonomy to facilitate the comprehension and comparison of the current state of the art of NV approaches and its evolution. The taxonomy must document the common characteristics and attributes from these proposals and clarify them to guide through the process of understanding the past, current and future solutions to virtualise the network.

- Design and implement a novel NV approach that fulfils all the imposed requirements. The solution must be efficient, both by reducing the overhead at data plane (not tunneled) and by being transparent on the end to end communication (e.g., avoiding push/pop actions).

RQ1: How to design efficient a (SDN based) NV method and architecture to reduce the overhead at data plane and be transparent on the end to end communication (optionally) for dynamically changed user groups and workflows?

- Design and implement a mechanism to facilitate the deployment of novel MAC addressing schemes both for physical and virtual devices. Due to the fact that the innovation on MAC addressing schemes is perceived as a research topic, the distribution/update of MAC addresses to the end hosts is not solved.

  RQ2: What are specific needs of new MAC address scheme to be efficiently deployed over a network?

- Design and implement a secured mechanism to control the access to the experimental facility based on the identity of the users and the policies defined by the researchers. This scheme could be extrapolated to a generic NV approach, whose aim is not focused on enabling the experimentation. The control access is focused on the data plane and must be granular enough (i.e., fine or coarse granularity) to meet the policy defined by the researchers.

  RQ3: How the access to a virtual network could be secured based on the identity of end users and policies defined by the corresponding tenant?

- Define basic architectural principles for an optimal design of new services and protocols as VNFs based on SDN technology to better fit into a facility based on NFV concepts. Apart from facilitating the adoption of new services and its deployment on the experimental facility, other scenarios, such as operator or data center networks, could benefit from these VNF design principles to improve and optimise the service delivery in production environments.

  RQ4: Which are the design principles that improves the performance of VNFs (based on SDN technology) while minimizing their deployment effort?

The specifics and challenges in research on new networking technologies and the construction of the experimental facility requires that the above described objectives are addressed in a complex and integrated way. Accordingly, the experimental facility relies on the novel NV approach, the mechanism for deploying alternative MAC addressing schemes, and the secured access control proposal. Moreover, the innovative proposals based on new MAC address schemes and the architectural principles for optimal design of services will be tested on the experimental facility.

Finally, an additional objective, which is transversal to all the previous ones, is the experimental validation and performance evaluation of these proposals under realistic circumstances (i.e., tested in the experimental facility).

1.4 DOCUMENT ORGANISATION

The remainder of this document is organised as follows. Chapter 2 presents the state of the art related to NV and experimental facilities. It also proposes a novel NV taxonomy to organise the NV survey analyzed. Then, Chapter 3 details three novel proposals as individual research contributions of the presented research: Layer 2 Prefix-based Network Virtualisation (L2PNV), MAC Address Configuration Protocol (MACP) and Flow-based Network Access Control (FlowNAC). Afterward, Chapter 4 experimentally validates these proposals and presents the EHU OpenFlow Enabled Facility (EHU-OEF), which is one of the main contributions from this thesis. This chapter also analytically demonstrates the benefits from the L2PNV proposal. Finally, Chapter 5 summarises the content of this document and details the main contributions obtained as outcomes from this thesis, as well as the dissemination of results related to this research work. To conclude the future work as the next steps for this thesis are outlined.

2 STATE OF THE ART

This chapter focuses on analysing the state of the art related to Network Virtualisation (NV) (in Sections 2.1, 2.2 and 2.3) and experimental facilities (in Section 2.4). These two topics are relevant to contextualise and clearly identify the main contributions from this thesis. However, prior to introduce both analyses, the Software Defined Networking (SDN) technology is also contextualised due to its relevance for the content of this thesis.

SOFTWARE DEFINED NETWORKING (SDN)

Therefore, some SDN related concepts and previous work must be introduced. As defined by RFC 7426 [8], SDN is a programmable networks approach that supports the decoupling of control and forwarding planes by means of standardised interfaces.

Although the SDN paradigm has revitalised the idea of programmable networks, these concepts have been around for a long time, as confirmed by this survey [20] of programmable networks published in 1992. In this paper, the authors present a model for programmable networks and a number of projects are reviewed based on that model. The idea of programmable virtual networking is also introduced as closely related to the network programmability.

A few years later, in 2008, McKeown et al. [12] presented OpenFlow as a way to enable the innovation in networks based on similar ideas. By means of OpenFlow, the programmability of the network is exposed to the researchers to run their experimental protocols by using the same infrastructure deployed for the production traffic. The authors introduce OpenFlow as a useful campus component to build large-scale testbeds like GENI, and claim a first deployment of a running OpenFlow network at Stanford University. Afterward, the complete process of maturing the OpenFlow technology divided in four phases (from the proof of concept to production development), the lessons learned and outcomes from real deployments are presented in [21]. In this process, SDN demonstrates its potential for different target scenarios, such as data center networks, service providers, enterprises and home networks.

Recently, several publications have presented an historical perspective of SDN and its relation with other programmable network approaches, such as in [22] and [23].

On the one hand, [22] analyses the past, present and future of programmable networks (i.e., open signaling; active networking; DCAN; 4D Project; NETCONF; and Ethane/SANE) and presents a survey of SDN. The paper details the SDN architecture and the OpenFlow standard in particular, and examines different alternatives proposed for implementing SDN-based protocols and services. The virtualisation is also analysed in the context of SDN proposals.

On the other hand, [23] also presents an historical approach to the evolution of programmable networks (i.e., active networks; separation of control and data planes; and OpenFlow and network operating systems). This paper also highlights the NV as a prominent use case for SDN. NV is described as an abstraction of the network that is decoupled from the underlying physical equipment, allowing multiple virtual networks share the same physical infrastructure. The legacy support for NV before SDN (e.g. VLANs and virtual private networks) is characterised by two fundamental principles: (1) only the network administrators are able to create these virtual networks, and (2) the virtual networks are limited to the already existing network protocols. Therefore, the innovation in legacy approaches is quite limited.

Apart from this historical view, some other recent publications have surveyed the last SDN developments, such as in [24] and [25].

The former [24] presents a comprehensive survey on SDN (with 579 references). The paper introduces the motivation for SDN and the most relevant concepts and standardisation activities (from different SDOs, such as ONF, IETF, IRTF, ITU-T, BBF, MEF, IEEE and some others) related to this technology. Then, the key elements of an SDN architecture are presented using a bottom-up layered approach. Moreover, the paper presents a deep analysis of hardware devices, southbound and northbound APIs, NV layers, network operating systems (NOS, also known as SDN controllers), network programming languages and network applications. The problem of debugging and troubleshooting in SDN is also covered in the survey. Furthermore, some ongoing research efforts and challenges are also envisioned. Also relevant for this thesis, an OpenFlow security assessment is detailed and some countermeasures are mentioned for SDN approaches based on OpenFlow. In this regard, the survey also covers the current proposals for security applications based on SDN technology.

The latter [25] also presents a survey on SDN based on a layered taxonomy proposed by the authors (i.e., infrastructure layer, control layer, application layer, control/infrastructure layers, application/control layers, and application/control/infrastructure layers). First, the SDN architecture and main components are presented. Then, the paper analyses and categorizes the most relevant research works related to the SDN technology based on the proposed taxonomy. The main issues and research directions are also detailed, as well as some scenarios where the use of SDN could be beneficial.

NETWORK VIRTUALISATION (NV)

As previously mentioned in the introduction, and also confirmed by aforementioned surveys, NV is one the most relevant use cases of the SDN technology. However, there are other NV proposals based on legacy technology (i.e., limited to already existing network protocols), which are also widely used in current deployments. Several NV surveys have been published as detailed below, but they do not fit the objectives of this thesis, because either they are fundamentally focused on legacy approaches or they are specific to a concrete scenario (e.g. data center networks). Moreover, the already proposed NV taxonomies do not properly cover all the different NV approaches presented so far.

In 2009, the authors in [13] (and also in [26]) presented a state of the art of NV. Nevertheless, the SDN technology was not yet mature enough at that time, and its relevance on the survey is marginal. The main contributions from this paper are the NV architectural principles, the NV design goals and the research challenges for NV. First, the paper presents the NV architectural principles, which are coexistence, recursion, inheritance, and revisitation. Then, the complete list of NV design goals are detailed, including flexibility, manageability, scalability, isolation, stability/convergence, programmability, heterogeneity, and legacy support. Finally, the most relevant research challenges for NV are identified, such as interfacing, signaling/bootstrapping, resource/topology discovery, resource allocation, admission control/usage policing, vNodes/vLinks, naming/addressing, mobility management, monitoring/configuring/fault handling, security/privacy, interoperability issues, and NV economics.

There are also some other NV surveys focusing on specific scenarios, such as data center networks. Related to this scenario, the authors in [27] cover the most relevant NV proposals for data center networks. They analyse their characteristics and classifies the different approaches based on different features, such as the forwarding scheme, bandwidth guarantee support, multi-path support, and relative bandwidth sharing. Then, the comparison between the different proposals is presented based on the following criteria: scalability, fault-tolerance, deployability, QoS support, and load-balancing.

Moreover, the authors in [28] present a survey in NV and SDN for cloud computing. This paper analyses the benefits from NV for this scenario, including the sharing of resources, isolation, aggregation, dynamics, and easy of management. Then, the different elements to be considered for the overall NV solution are detailed, such as the virtual Network Interface Card (NIC), virtual switch, virtual LANs in the cloud, virtualisation for multi-site data centers, and network function virtualisation. The SDN technology is detailed as the latest revolution in networking innovations and programmable networks, becoming the key element to build the NV solutions for the cloud. Finally, the OpenADN proposal is presented.

In relation to the NV analysis presented in this chapter, first of all, some concepts related to NV and its definition are introduced in Section 2.1. Based on this, a characterisation and NV taxonomy is presented in Section 2.2. Finally, some of the most relevant proposals are analysed in Section 2.3 considering the proposed taxonomy. This effort does not aim to be a thorough analysis of all the NV proposals, but a means to detail on the NV taxonomy and elaborate with specifics.

EXPERIMENTAL FACILITIES

The research efforts on the future internet demand a robust experimentation and tests at large-scale. The experimental facilities provides a realistic environment to validate the new architectures and proposals. A survey of future internet research activities is presented in [2]. The most relevant activities in the USA (the FIA and FIND programmes, and the GENI testbed), Europe (the FIRE testbed), and Japan (the AKARI project, and the JGN2plus and JGN-X testbeds) are detailed. In this regard, [6] compares the FIRE initiative with the GENI testbed, and concludes that while the US approach is to build a separate network infrastructure for researching, the EU approach assumes that a large-scale platform cannot be planned before the new architectural concepts are previously validated. As a consequence, the progressive federation of testbeds is the EU approach as shown in [29].

A special issue on Future Internet Testbeds [30] (split across two volumes: Part I and Part II) focuses on the design, building and experimentation on the most relevant testbeds worldwide, such as GENI [5], the G-Lab [31], the Global Lambda Integrated Facility [32], OFELIA [15], FEDERICA [33], the GEYSERS optical testbed [34], and SmartSantander [35]. Apart from describing the architecture and infrastructure of all these Future Internet testbeds, the special issue also covers the experimentation and usage of these testbeds. In relation to GENI, a number of projects that operate over this testbed are also described, such as the InstaGENI initiative [36], the K-GENI experiment over GENI and KREONET [37] and the GpENI experimentation [38]. Moreover, the results from some other experiments related to Smart Grids [39] and the development of an instrumentation and measurement system called INSTOOLS [40] used in GENI are also included. Regarding PlanetLab [41], some results from the experimentation on end-to-end performance in heterogeneous testbeds are presented in [42]. The SmartSantander paper [35] also details the experimentation on IoT over this testbed. Finally, with regard to the research activities at Stanford University, the process of maturing OpenFlow through real deployments is detailed in [21].

In the literature there are some other surveys concrete to a specific research topic, such as [43], which focuses on experimental facilities for researching on IoT under realistic conditions. For instance, MoteLab [44] is a testbed for wireless sensor networks. However, the supporting for experimentation on wireless technologies and IoT approaches is out of the scope of this thesis.

In relation to the analysis of the most relevant Experimental Facilities related to the goal of this thesis detailed in Section 2.4, the survey has been grouped in two categories: non-SDN facilities (Section 2.4.1) and SDN enabled facilities (Section 2.4.2).

2.1 NETWORK VIRTUALISATION

Although NV is not a new concept, it has been related to different approaches and has evolved during the last years, thus it remains unclear what it really means and the supported features it must provide. The NV proposals, their characteristics and which are considered as mandatory features depends on the scenario and target use case. This document considers NV in its broader sense, trying to be inclusive with any approach to the virtualisation of the network.

The remainder of this section is organised as follows. First, the reference architecture of network devices is presented in Section 2.1.1. Then, the proposed architecture for NV is detailed in Section 2.1.2. Finally, the reference definition for NV used in this document is presented in Section 2.1.3.

2.1.1 Network Device Architecture

Before getting into details, the overall architecture must be depicted to avoid a partial approach to the NV problem. Moreover, some features could be assumed to be generic, but some others could become fundamental depending on the target scenario, whereas in other scenarios these later features are no mandatory or even relevant. Therefore, the global architecture is depicted as a reference for the on going discussion.

The architecture presented in [8] (shown in Figure 2.1) is used as the reference architecture for properly introducing and describing the possible options to implement NV. Although this architecture is presented in the context of SDN, it does not mean that the overall architecture cannot deal with legacy solutions. What a SDN related architecture introduces is an explicit interface and separation between the forwarding plane and the control plane. In legacy systems this separation is not exposed and it always remains internal to the network device. Moreover, this architecture clearly claims that there is no distinction between physical and virtual resources or hardware and software implementation of those elements described in the document. It is also important to consider that it follows a network-device-centric approach, which means that the architecture is related to a single network device. This is relevant for later on characterisation, since the abstraction of a complete network is out of the scope of this RFC.

The document presents the SDN layer architecture for a network device, which introduces three layers separated by two abstraction layers. On the bottom, the network device has two main components, the forwarding plane and the operational plane, which are mainly related to the control and management planes respectively. The distinction between control and management is controversial and out of the scope of this discussion, but some pointers can be found in the RFC in section 3.5. The forwarding plane, also known as data plane, is responsible for handling the packets in the data path based on the decisions made in the control plane. The operational plane is responsible for managing the operational state (e.g. active/inactive, status of ports and so on) of the network device. The Device and resource Abstraction Layer (DAL) exposes the forwarding and operational plane through the control plane southbound interface and the management plane southbound interface.

On top of the network device and the DAL, the second layer has two main components, the control plane and the management plane. The control plane is responsible for making decisions on how the packets should be forwarded in the

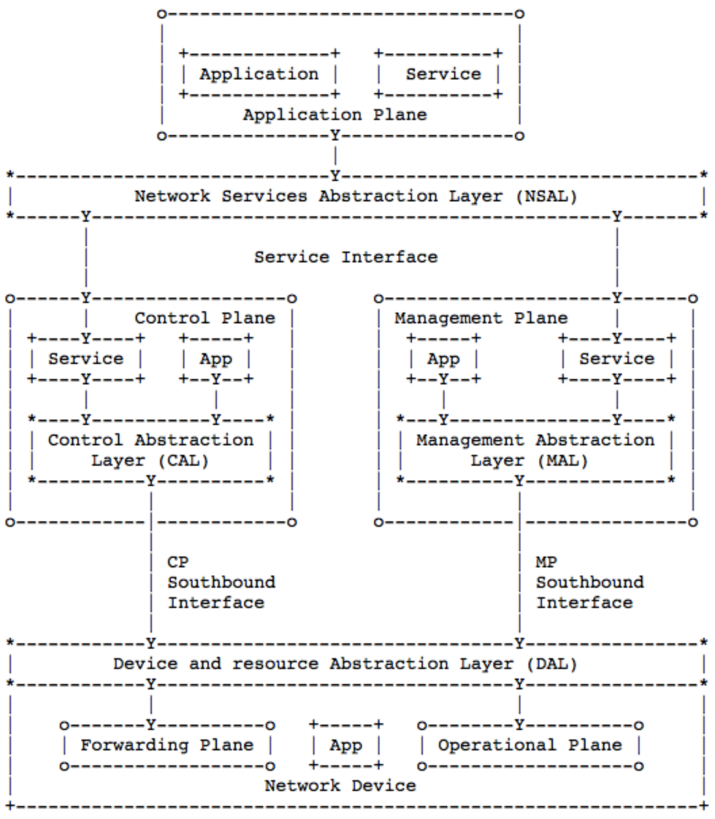


Figure 2.1: The SDN layer architecture defined by RFC 7426 [8]

data plane, whereas the management plane is responsible for monitoring, configuring and maintaining the network device. Due to the relevance that the control plane and its functions has for the following characterisation, it is worth to mention some of the responsibilities and functionalities that are assigned to the control plane: topology discovery and maintenance, packet route selection (i.e., routing processes) and path failover mechanisms. Moreover, it is important to distinguish between the control actions performed through the control plane southbound interface (detailed above) and the east-west communication between control entities. The depicted architecture considers the former communication (i.e., between the control and the forwarding planes), while the later interface is out of the scope of this discussion and usually implemented through an in-band protocol. The Network Services Abstraction Layer (NSAL) exposes northbound both the control and management planes through a service interface.

On top of the NSAL, the third layer comprises the application plane, where the behaviour of the network is defined by applications and services. In the end, the application plane is the final user of the virtual instance exposed by the NV solution. One of the main goals of any NV approach is to be as transparent as possible to the application, so it behaves in the same way as if it is interacting with a real network instance.

As previously mentioned, this SDN architecture is the reference for NV solutions characterisation and the main factor to distinguish between NV approaches.

2.1.2 Network Virtualisation Architecture

As in any other discussion, to clarify the nomenclature used to discuss the different NV approaches is essential for a proper understanding. Moreover, some general components and entities that comprises any NV approach are introduced to avoid any misunderstanding.

There are several proposals to describe the components and entities of a NV solution. In the context of the GEYSERS FP7 project[1], a generic architecture for the cloud infrastructure as a service (IaaS) provisioning model [45] is proposed, which details the main actors and functional layers in on-demand infrastructure services provisioning. The model consists of three layers (shown in Figure 2.2), each managed by a different actor: the Physical Infrastructure Provider (as defined by the GEYSERS FP7 project) (PIP), the Virtual Infrastructure Provider (as defined by the GEYSERS FP7 project) (VIP) and the Virtual Infrastructure Operator (as defined by the GEYSERS FP7 project) (VIO). From a bottom-up perspective, the physical infrastructure is provided by the PIP, which is virtualised by the VIP, and which is finally operated by the VIO.

There can be several network domains, each normally (but not always) managed by a single PIP. Each PIP can provide Physical Resources (PR) to several VIPs, whilst each VIP can request PRs

1 The GEYSERS Project: `http://cordis.europa.eu/project/rcn/93786_en.html`

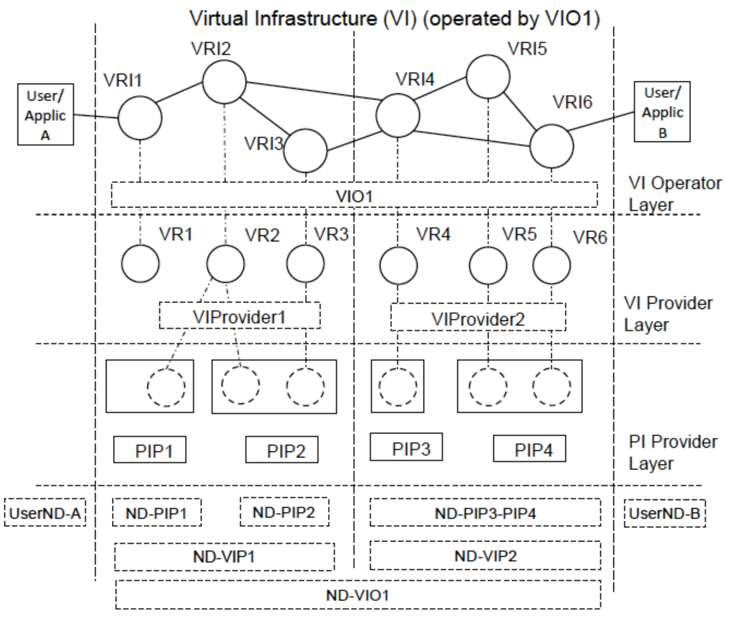


Figure 2.2: The general architecture for the cloud IaaS [45]

from several PIPs. Both entities are responsible for ensuring that resources are provided in the range of capacity. The PRs are virtualised as Virtual Resources (VR) by the abstraction layer. The VRs are the elements managed by the VIPs and provided to the VIOs. Each VIP can provide, in this way, VRs to several VIOs, while each VIO can request VRs from several VIPs. Again, both entities must ensure that the total capacity of the physical resources is not exceeded. Between each pair of entities (PIP-VIP and VIP-VIO) there is an adaptation layer, which is responsible for hiding the underlying complexity to the upper layer. The virtual infrastructure requested by users or applications is eventually managed by the VIO. In this way, each VIO operates a different virtual network slice, which must be isolated from the others. Finally, the user is the entity who sends the data packets through the Virtual Infrastructure.

The 4WARD Virtual Network (VNet) architecture [46], in the context of the 4WARD FP7 project[2], also proposes a similar set of roles and actors for a NV architecture. The Infrastructure Provider (as defined by the 4WARD FP7 project)

2 The 4WARD Project: http://www.4ward-project.eu/

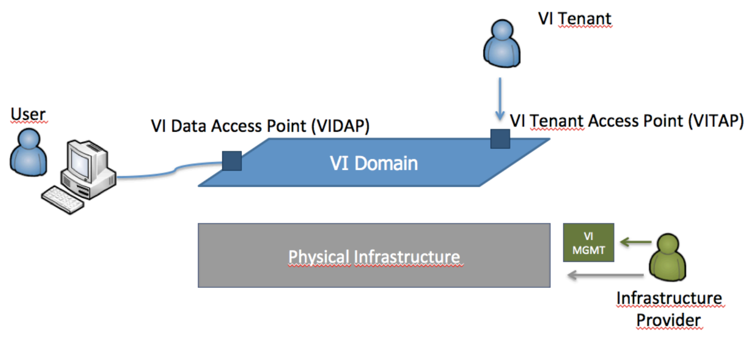


Figure 2.3: The two-layered model for NV architecture

(InP), which owns and operates the physical network, the virtual network provider (as defined by the 4WARD FP7 project) (VNP), which combines the slices of virtualised infrastructure from InPs together into a functional VNet, and virtual network operator (as defined by the 4WARD FP7 project) (VNO), which operates and manages the instantiated VNets. The end-user is the entity who inserts the data traffic into the VNet instance.

Apart from the details and specific functionality associated to each entity, and some nomenclature distinction, there is almost a 1:1 mapping between both approaches in relation to the main roles and business boundaries that could be found in any NV approach.

A simplified two-layer NV architecture (shown in Figure 2.3) is proposed for two main reasons, because this simplification is enough for proper characterisation of NV approaches and because using two layers is more generic and more proposals could fit into this model. Some proposals could not be mapped into a three-layer model, since the differentiation between the VIP and the VIO (or VNP and VNO) is not always clear or even considered.

- The **bottom layer** is the physical infrastructure layer, which provides the actual network devices and physical links to interconnect them.
  - The **infrastructure provider** is the owner of the physical resources and is the responsible for creating

the virtual instances exposed to the layer above by means of some kind of abstraction.

At this layer a virtual instance management functionality is needed, which is used to define and instantiate the virtual resources. For simplicity, only one domain is considered in this architecture.

- The **upper layer** is the virtual instance domain (Virtual Instance (VI) domain) and represents all the virtual instances created over the same physical domain (i.e., a set of physical resources). The VI domain must fulfil some rules imposed by the NV solution to enforce the isolation between the instances and allow the sharing of resources, such as a means to uniquely identify each of the virtual instances. It is assumed that multiple virtual instances could be sharing the same physical infrastructure, otherwise the virtualisation overhead makes no sense. A virtual instance is broader concept than a virtual infrastructure, which not only refers to an infrastructure or a set of resources, it could be an abstract view exposed to the upper layer.

  - The **virtual instance tenant** (referred also as tenant to simplify the notation) is the entity who operates, manages and/or controls the virtual instance, depending on the type of the virtualisation performed.
  - The **user** is the entity who sends and receives the data packets by using the appropriate virtual instance provided for this aim.

  There is a relation between the user and the corresponding tenant that provides the networking service. Due to the fact that the physical infrastructure is shared between multiple tenants, the users must be securely bounded to the associated tenant to avoid unauthorised access to the network services provided by other tenants.

In this NV architecture, there are two reference points that need special attention: Virtual Instance Data Access Point (VIDAP) and Virtual Instance Tenant Access Point (VITAP). The Virtual Instance Access Point (VIAP) is a generic term that comprises both reference points.

- The **virtual instance data access point** (VIDAP) is the entry point for data packets to the virtual instance domain. The data traffic related to a given user needs a VIDAP to send and receive data packets to/from the virtual instance. It is important to highlight that the resources between the user and the VIDAP are not virtualised per se (or at least, they are not part of the target VI domain) and need to be properly analysed to avoid isolation problems on the virtual instances.
- The **virtual instance tenant access point** (VITAP) is the entry point for the tenant to operate, manage and/or control the virtual instance. Once again, this interface must be accurately defined, deeply analysed, and securely exposed to the corresponding tenant. Some deployments consider an out-of-band network for this purpose, although there are also in-band solutions.

Both the VIDAP and VITAP need the support of sharing the corresponding access points in the physical infrastructure.

2.1.2.1 *Virtual Instance Namespace*

The virtual instance namespace is a key concept for implementing a NV solution. In general, the concept of namespaces appears to avoid name collisions when different identifiers share the same name. The namespace introduces the uniqueness to the identifier by including a distinctive part which adds a context to the name. The combination of both, the namespace identifier and the local name, is designed to be unique.

In the context of VN, it is fundamental to uniquely identify the virtual instance to which a data packet belongs. Therefore, there must be a kind of an identifier in the data packet that allows to make such a distinction. In a nutshell, a namespace refers to the portion of the packet headers (e.g. a field of L2-L7 header, a set of bits from these headers, an address or prefix, a tag or any combination of them) with a specific function assigned, the identification of the virtual instance. Since the physical resources are shared, a predefined namespace must be assigned to identify the target virtual instance. Each VI domain must determine the namespace to be used for this task, and it must be consistent in the domain for all the virtual instances to avoid collisions. This means that the VIDAP is characterised by a unique VI namespace value. The adjacent VI domain could determine another namespace to identify the virtual instances, which implies that some adaptation function must be performed at data level between the VIDAPs of each VI domain to ensure the isolation of each virtual instance when transversing different domains.

2.1.3 Network Virtualisation Definition

Depending on the target scenario, available technology and specific needs, the NV concept has evolved and could represent different abstractions. This means that there is not only a single valid definition for NV. To avoid confusion on the scope covered by this document, a NV definition is proposed, which tries to be generic enough to include the majority of past and future proposals. The Virtual Networks Research Group[3] (VNRG) from the Internet Research Task Force (IRTF) was focused on the topic of NV, whose activity ended in 2012. The Virtual Network (VN) was introduced as an instance of a physical network, which attempts to better utilise the networking resources (e.g. routers, hosts, links and services) through reusing the physical infrastructure. Multiple VNs can be instantiated concurrently on the same single physical network. The isolation between VNs must be supported, as well as the sharing of the same physical infrastructure for different purposes. The main fundamental properties of VNs are: (a) the sharing of resources concurrently between multiple VN instances, (b) the isolation between VNs, and (c) the abstraction of resources (i.e., virtual resources) that could not be directly related to the physical one.

Another attempt to define NV is presented in [13] and [26] (from the same authors) as a means to support the coexistence of multiple VNs on the same physical substrate. Each VN is a collection of virtual nodes and virtual links, being essentially a subset of the underlying physical network resources. In this work, the authors focus on an operator network as the target

3 Virtual Networks Research Group (VNRG): https://irtf.org/concluded/vnrg

scenario, and explain the NV as a decoupling of the role of a traditional Internet Service Provider (ISP) into two: (a) the provider of the infrastructure (InP) managing the physical resources, and (b) the service provider (SP), who aggregates network resources from different InPs to provide the end-to-end network service.

In [14], the authors introduced the NV as an abstraction of a shared physical infrastructure that allows each tenant to define its own network topology, addressing space, and custom logic from an isolated set of network resources. The data center is the target scenario, and the main reason because the arbitrary definition of network topology is so important. In this type of scenario, the flexibility and capability of easy migration of VNs are essential properties.

As previously mentioned, the definition of NV proposed in this document tries to be generic, inclusive and agnostic from the target scenario. *NV provides the ability to create a logical or virtual (abstract rather than actual) instance of a network that can be delegated to a tenant, who represents a single administrative entity.* Depending on the NV solution, the interface to the virtual instance exposed to the tenant can be different in scope and functionality. Each virtual instance, also known as virtual network (VN), must be isolated from the other instances that share the same physical infrastructure. The main fundamental properties of any NV solution are:

- The abstraction of the physical infrastructure into a virtual one.
- The sharing of the physical substrate.
- The isolation of virtual instances.

This NV definition could fit into a wide range of possible target scenarios, such as operator networks, data center networks, campus networks and experimental facilities, each of which could include additional properties as mandatory to deal with the specifics of the scenario.

One important aspect that is considered fundamental to be included in any NV proposal is the security related to how the different entities (i.e., infrastructure provider, VI tenant and user) are authenticated and authorised to get access to the VI domain through the reference points (i.e., VIDAP and VITAP) previously defined. The issue to be addressed is how a VI domain can assure, for instance, that data packets come from a valid and authorised user. The same applies to the relation between the tenant and the VI domain.

2.2 NETWORK VIRTUALISATION TAXONOMY

In a context in which multiple NV solutions are proposed, a classification system is a fundamental element needed for proper comparison and understanding of what each proposal is providing. To this end, a NV taxonomy is presented, which is the foundation for the following study of the state of the art of NV proposals. Once again, the taxonomy is designed to be generic enough to cover past and future proposals, as well as agnostic to the target scenario. Some proposals for NV classification are more related to legacy proposals, such as in [13], whereas some others are related to a specific target scenario (e.g. Data Center), such as in [27], or a specific technology (e.g. OpenFlow) for building the NV proposal, such as in [47].

The following subsections introduce and detail the different aspects to analyse from any NV approach: NV type, VI namespace, DP mechanism, VIAP classification, network topology, VI creation point, and isolation level.

2.2.1 Network Virtualisation Type

The NV solutions fall into three main categories that distinguish the foundation level at which the virtualisation is exposed to the tenants, and the level at which the network abstraction is done: SDN-enabled Virtualisation (SDNeV), Virtual Node (vNode), and Overlay. These three categories can be grouped in two different approaches: node centric and network centric. In the former, the nodes are exposed as individual elements to the tenants; whereas in the latter, the underlying network infrastructure is abstracted as a whole. In relation to the NV types, SDNeV and vNode are node centric approaches, while

Overlay is a network centric approach. This categorization relies on the architecture and interfaces previously described in Section 2.1.2 that will not be further detailed.

2.2.1.1 *Software Defined Networking-enabled Virtualisation (SD-NeV)*

This category refers to those solutions that take advantage from the separation between the forwarding plane and the control plane proposed by SDN. This separation enables the programability of the network by exposing the control plane southbound interface. The forwarding plane is abstracted through this interface, then virtualised and exposed to the different tenants. Since this VN type is built on top of the control interface, it is related to the network device, which means that the virtualisation is performed at node level, and more concretely at the DAL from the network device architecture defined in RFC 7426 [8]. Since the virtualisation is performed on top of the DAL, there is no control plane-related functionality implemented on the virtual instance of the network resource, which means that this layer must be added by the tenant. This gives the tenant the freedom to design the behaviour of its own virtual instances, which also provides a better environment for innovation and clean slate proposals with novel network architectures.

In this approach it is essential to properly enforce the isolation between tenants in the virtualised control interface and avoid collisions in the forwarding plane by using the appropriate definition of the VI namespace. Moreover, the access to this interface must be secured and only authorised tenants must be allowed.

2.2.1.2 *Virtual Node (vNode)*

This category refers to those approaches that are built op top of network devices with an associated functionality that defines its behaviour. This means that both the forwarding and control planes are joint together and abstracted as a virtual instance. A Virtual Node (vNode) is a general term to refer to any networking functionality that could be provided by the node, e.g. vSwitch. vRouter, vFirewall and so on. Each physical node could implement a different behaviour based on the control plane functionality, which is then virtualised and exposed to the tenants. The tenants cannot change or update the control logic provided by the vNode, but they are able to configure the virtual instance delegated to them. In this category, the virtualisation is also performed at node level, and more concretely at the NSAL from the network device architecture defined in RFC 7426 [8].

Since legacy devices (i.e., previous to SDN) do not separate the forwarding and control planes, the vNode is the typical (or the only) approach to NV when performed at node level. In this legacy approaches, the same physical node only implements one control logic, which is the same for all the vNodes instantiated on the network device. It is also quite common in this scenarios that the infrastructure provider and the tenant are the same entity, which is responsible to properly configure each virtual instance and avoid collisions. In this case, being the same entity the security concerns are limited and the interface between the infrastructure provider and the tenant is not so critical. However, there are other vNode scenarios in which the roles are played by different entities and the interface exposed to the tenants is critical to avoid misconfigurations that could impact in other tenants. The isolation between the tenants and the security become critical topics, only authorised tenants can configure their virtual instance and there must be a mechanism to avoid collisions. Again, the VI namespace must be clearly defined and enforced.

2.2.1.3 *Overlay*

This category refers to those proposals which are built on top of another network. In overlay solutions the resource which is abstracted is the network itself, opposite to the previous categories in which the abstracted resource is the network device. These NV approaches do not have the notion of individual elements exposed by the underlying physical infrastructure, since the whole infrastructure is abstracted. The virtual instance is built based on the edge elements (e.g. Network Virtualisation Edge defined in RFC 7364 [48]) which encapsulate the external addressing scheme into the addressing space used internally by the physical infrastructure, and each virtual instance is implemented as an overlay. The edge nodes are the only elements that have the notion of both domains and are capable to properly encapsulate (and de-encapsulate) one domain into another, which means adding (and removing) the internal header which makes use of the internal addressing namespace. Therefore, the virtualisation is performed at the edge, and the core network is not aware of taking part in any virtualisation solution. This makes the core network independent from the NV solution, which means that it does not depend on the number of virtual instances that are deployed on it. Therefore, the scalability of overlay approaches is higher than other proposals in which the number of virtual instances have a direct impact on the solution. However, this high scalability and independence between the physical and the virtual instances are not free and imply on overhead in the data plane. Depending on the encapsulation procedure used and the size of the packets sent it could have more or less impact on the effective data throughput.

Since this NV approach is not performed at node level, the reference architecture defined in RFC 7426 [8] cannot be used to define the overlay solutions. A network abstraction is out of the scope of this network device architecture, however, the underlying infrastructure is built with individual nodes which could be described through this architecture. In the end, the underlying network is pure functional (e.g. a switched or routed network) and both the forwarding and control planes are present.

2.2.1.4 *Conclusions*

As a conclusion, three different NV types are proposed to categorize past and future NV approaches. The first two are based on a node level abstraction, whereas the third one is based on a network level abstraction. Moreover, the first approach allows the tenant to actually define the behaviour of the virtual node, whereas in the second and the third types the behaviour of the abstracted element is defined by the infrastructure provider. Depending on the target scenario, some of the characteristics of each of this NV types fits better than the others, so this should be the first decision to make before choosing a concrete NV proposal.

2.2.2 Virtual Instance Namespace

As previously introduced, a namespace provides the uniqueness to an identifier and a VI namespace allows to uniquely identify the virtual instance to which a data packet belongs. The VI namespace is a parameter of the VI domain and the basis to avoid collisions between the virtual instances. Therefore, one of the design decisions to define a NV proposal is to select the VI namespace to be used, which means that the identification of the virtual instance will be coded in a portion of the packet headers. This namespace must be reserved and not be modified or used for any other aim. Once again, the target scenario could impose some restrictions to the possible or more appropriate namespace to be used.

Due to its role of packet classification at data level, the VI namespace must be part of the packet header, that can be easily identified when the packet comes into the physical infrastructure. Then, the classification process must decide which is the corresponding virtual instance it belongs to. This process takes place just before the VIDAP, since the VIDAP is the entry point for those packets which belong to a given virtual instance. In principle, the VI namespace could be any field from L2 to L7 headers (e.g. address), any set/combination of bits from these headers (e.g. prefix) or a specific tag. Therefore, the VIDAP is characterised by a specific value of the VI namespace, the one associated to the virtual instance.

- On a node centric approach (i.e., SDN-enabled Virtualization (SDNeV) and vNode) it is typical to use a tag as the VI namespace, e.g. VLAN or MPLS. One of the benefits is that most of the network devices are able to add and remove this type of tags. However, this means that the usage of those tags is then restricted at the virtual instance level. Although some agreements on the value range of this tags that could be used by each virtual instance or the encapsulation of tag-in-tag could be used, this add complexity at the virtual instance level. In the first case, the range of values must be enforced and it could be a restriction for the target scenario. In the second case, it is not so trivial to use a virtual instance tag by legacy devices, since this means that each time that the virtual instance needs to add, remove or modify their own tags the tag of the VI namespace must be also processed (e.g. additional pop and push actions are needed on the VI namespace, which is a feature not widely supported).

Another possible VI namespace for node centric approaches is the use of the IP prefix. By doing so, each node could easily identify the associated virtual instance based on the source IP prefix. However, this approach could be more feasible when a private IP addressing scheme is used, than in an operator deployment when assigning public IP addresses.

- On a network centric approach, when an overlay is used, the first aspect to decide is the encapsulation mechanism to be used (e.g. VXLAN [49], NVGRE [50], STT [51], Geneve [52] and so on). Associated to the encapsulation solution, the virtual instance identifier is defined, VXLAN Network Identifier (VNI) in VXLAN, Virtual Subnet Identifier (VSID) in NVGRE, Context ID in STT or Virtual Network Identifier (VNI) in Geneve. The overlay approach is typical in data center scenario (RFC 7364 [48]) due to its scalability and also due to the presence of L3 datacenter networks. Apart from standard encapsulation proposals there are other interesting proposals, such as Segment-oriented Connexion-Less protocol [53] (SCLP), which claims performance improvements compared to standard VXLAN.

The above examples are just some of the most used alternatives selected as VI namespace, but this does not mean these are the only possibilities. Each proposal must consider which could be the best suited, depending on the target scenario and possible restrictions imposed by this scenario.

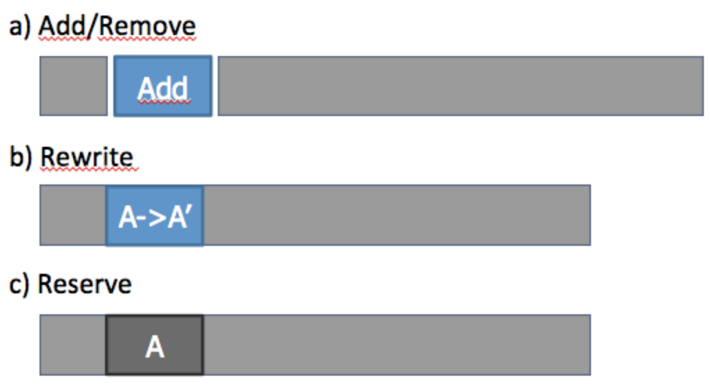


Figure 2.4: The alternatives for data plane adaptation mechanism

2.2.3 Data plane Adaptation Mechanism

The NV architecture defines the VI domain as the group of virtual instances that share the same set of physical resources. The entry point of data packets to a VI domain is the VIDAP, which must assure that each packet coming from a user enters in the corresponding virtual instance. In principle, the VI namespace must be the key to perform this task, although the VI domain must assure that its value has been securely set. It must be highlighted that this namespace is related to the target virtual instance, which means that some kind of procedure or mechanism related to the virtualisation solution must be performed to set the proper value in this namespace. Where and who is responsible for setting the VI namespace is analysed in Section 2.2.4.

There are three possible mechanisms (shown in Figure 2.4) to set the VI namespace value in the data packet: add/remove, rewrite and reserve. This procedure must take place between the user and the virtual instance, and must be assured and enforced at the VIDAP.

2.2.3.1 *Add/Remove*

The VI namespace can be added on ingress to and removed on egress from the virtual instance. By means of this mechanism a new set of bits is added in the user's packet to identify the target virtual instance. The most common way of adding this identifier is using a tag, such as VLAN ID or MPLS, as the VI namespace. On egress this tag must be removed since its meaning is limited to the VI domain. When adding a tag the addressing space is shared between the user and the physical infrastructure, which means that the actual network devices must forward the packets based on the addressing scheme used by the users and could impose some restrictions and scalability issues. In general, any other field or header, standard or not, could be added as a VI namespace. The main problem of adding a non standard field is the impact when transversing standard network devices, which could discard the packet.

Another typical case of adding a VI namespace is the overlay. In all the overlay solutions, the packet sent by the user (inner packet) is encapsulated into a new packet header (outer packet headers) and the VI namespace is added in the outer packet header as an additional field to identify the target virtual instance. The rest of the outer headers are fundamental to forward the packet through the underlying network to the appropriate destination, but they are not directly related to the virtual instance. In this case, the user and the underlying infrastructure are using independent addressing schemes, which means that physical network devices can be optimised and the traffic from virtual instances can be aggregated to avoid scalability issues. However, this imposes an overhead due to the encapsulation in the new packet header.

2.2.3.2 *Rewrite*

The VI namespace can be also rewritten given a new meaning to the selected namespace. This can be done either because the namespace does not have a previous meaning for the end-to-end communication or because the mapping between the previous value and the VI identifier is stored on ingress and restored on egress. There are NV solutions that rewrites the entire addressing space to isolate the user's addresses from the addresses used by the virtual instance. This separation allows that users from different virtual instances can use the same addressing space without any collision on the VI domain, since the addresses are rewritten using an alternative value. The main limitation of this approach comes when this mapping is 1:1 and no possible aggregation is possible. One possible aggregation could be the use of IP prefixes (if all the virtual instances has the same prefix length), which allows to aggregate the mapping and perform it at prefix level. The aggregation reduces the information that must be stored to perform the task of namespace rewriting.

2.2.3.3 *Reserve*

The VI namespace can be reserved to this aim, and the packets can be sent with the appropriate value already set. This means that the namespace cannot be modified by any element internal or external to the virtual instance. The main limitation is that all the users must be coordinated to avoid collisions because they are using the incorrect value. Therefore, the users are somehow aware that they are using a virtual instance, since the value is set by the source. One possible solution which makes use of this approach is a VI domain that uses the IP prefix of the source IP address to identify the virtual instance. The addresses, with the corresponding IP prefix associated to the target virtual instance, are assigned depending on the user identity. This procedure allows to differentiate the packets and the source IP prefix that is related to the virtual instance. The main benefit from this approach is that no extra overhead is added and the packets are not modified (not even rewrite) from the source to its destination all the path through the VI domain.

2.2.3.4 *Conclusions*

Although the possibilities for the VI namespace are numerous, the mechanisms to set the appropriate value in the packet are just three. As previously mentioned, the VI namespace must remain stable for the whole VI domain to avoid complex actions and possible collisions. For a similar reason, the mechanism chosen for setting the VI identifier must be also coherent in the VI domain, for instance, if a tag is added (or a field is rewritten) on the ingress it must be removed on the egress. However, the exact point at which this process happens could vary from one side to the other, which is related to the VIAP classification that is analysed later on.

When the control plane southbound interface is exposed to the tenants, i.e., the SDNeV type, a similar mechanism could be needed and the same study applies. Since the actual packets can be exposed to the tenant through the control interface and the tenant can also insert packets into the virtual instance, the VI namespace value must follow the same rules. Then, the VI namespace can be added/removed, rewritten, or reserved when the packets go up and down through the control interface. This process can be transparent or not to the tenant, who must process the packet due to the virtualisation solution just in the latter case. If the tenant is not aware of this process, then the infrastructure provider must perform the appropriate action in-between the actual forwarding plane and the control plane built by the tenant. Although introduced here, the exact location where this mechanisms take place and if the tenant is aware or not of it are questions related to the VIAP classification analysed afterward.

2.2.4 VIAP Classification

The VI access point is the reference point that separates the internal domain, where the NV is performed, from the external domain, where the physical infrastructure is not virtualised. Therefore, the VIAP is a generic term that encompasses both the data plane (VIDAP) and the interface toward the tenant (VITAP). However, this section is more oriented to the VIDAP reference point, although similar conclusions could apply when referring to the VITAP. Regarding the DP mechanisms introduced previously, the remaining questions are who is responsible for performing such a process and where does this process takes place. These two questions are relevant for any NV proposal since this also defines the relation between the three entities already identified, i.e. the infrastructure provider, the tenant and the user.

The are three main types of VIAP that could be present in any VI domain: transparent, non-transparent and semi-transparent. This three categories refers to the perception that the outside entity (i.e. the tenant or the user) has with regards to the virtual instance, if this entity is aware or not of being part of a VI domain. Although this seems to be something intangible, the consequences are clear and concrete as explained below.

2.2.4.1 *Transparent VIAP*

This type of VIAP refers to the fact that the external entity is not aware of taking part into a VI domain, and therefore this entity does not perform any action on the packets to set the appropriate value in the predefined VI namespace. As a consequence, the VIAP is responsible for processing the packet to handle the VI namespace appropriate for this entity. This means that, although the mechanism is related to the VI domain, the VIAP must be able to securely identify the target virtual instance associated to the external entity who is sending the packets in order to set the appropriate value in the corresponding namespace. How this VIAP knows which is the target virtual instance is something that each NV proposal must also define in relation to the VIAP definition. In a simplified scenario, all the traffic coming from a specific physical port could be related to just one virtual instance.

2.2.4.2 *Non-transparent VIAP*

In this case, the external entity is aware of the VI domain and some actions must be performed by this entity in cooperation with the VIAP in order to properly identify the target virtual instance for each packet. This means that the VIAP is not transparent for the external entity who is using the virtualisation services. Therefore, the external entity is responsible for processing the packet to handle the VI namespace and set the VI identifier delegated to this entity. As a consequence, the VI identifier must be delegated to the external entity through a process that must be defined by the NV solution. The VIAP is responsible for enforcing that the packets are authorised and ensuring that they reach the corresponding virtual instance. This type of VIAP is less complex since the source of the packets is identifying the target virtual instance, but a misconfiguration or a malicious entity could be harder to detect. Preventing data traffic from one virtual instance accessing a non-authorised virtual instance is the responsibility of the VIAP, then some authentication and authorisation process must be performed.

2.2.4.3 *Semi-transparent VIAP*

This type of VIAP refers to the cases where neither the external entity nor the VIAP directly performs the packet processing, but an intermediate process in-between both entities. This is the case when a certain piece of software/hardware can be configured to adequately process the packets to handle the VI namespace and set the corresponding VI identifier. This intermediate process could be located on the external entity or in another network device before accessing the VIAP. An example of this type of VIAP could be a tunneling process running on the hypervisor, or a network device which implements the edge functionality of an overlay solution. In any case, this functionality is considered out of the VI domain and not implemented by the external entity, who is not conscious of being part of a NV solution.

2.2.4.4 *Conclusions*

After introducing the three possible types of VIAP, there is one point that must be clarified. In contrast to the VI namespace and the DP mechanism that must remain constant for the whole VI domain, the VIAP type could vary from one access point to another. This means that the same VI domain could implement, for instance, transparent and non-transparent VIAPs depending on the attaching point to the VI domain. There is no contradiction with the previous reasonings, since in both alternatives the packets are properly processed before entering in the virtual instance, and the VI namespace remains coherent in the whole domain.

2.2.5 Network Topology

One fundamental characteristic of a network that differs from other resources that are abstracted and virtualised by other type of virtualisation solutions (e.g. CPU, memory and storage) is the fact that the network has a topology that affects its behaviour. For instance, not all the connexions between all the physical ports of a network have the same capacity or latency, it depends on different aspects, such as the selected path, the number of network devices that must be crossed (could be none if the source and destination port are on the same network device) and the available resources. In a nutshell, the topology is an essential attribute of a network.

When considering a NV solution, there are two options with regard to the network topology exposed to the virtual instance: if it is exposed or not. This exposition should not be confused with the fact that the virtual instance defines its own topology. The point here is if the infrastructure provider presents a topology from the underlying resources. Both options are analysed below.

2.2.5.1 *Network topology exposed*

The infrastructure provider exposes the topology of the network provided to the tenant. This means that the interface between the infrastructure provider and the tenant is able to exchange information related to the topology of the underlaying network. On a node centric approach (i.e., SDNeV and vNode) the topology could even be exposed dynamically by the network devices to the tenants, and each tenant could have a different topology based on the actual network devices included in the virtual instance.

The exposed network topology could be either physical or virtual. Depending on the target scenario, one could be more adequate than the other.

- **Physical topology**: the topology of a network varies depending on the layer being analysed, e.g. the optical topology could be different from the L2 topology, which could be different from the L3 topology. This means that, although the physical topology term refers to the idea that it is the same topology managed by the infrastructure provider, there is not a single topology view of a network. In this case, the physical topology exposed to the tenant will be the same that the infrastructure provider manages at the level of abstraction that network is virtualised and exposed to the tenant.

- **Virtual topology**: taking into account what has been analysed previously regarding the physical topology, when exposing a virtual topology to the tenants a new abstraction layer is added on the infrastructure provider that is responsible for mapping the virtual topology to the physical one. This also imposes some overhead on the VITAP, since this mapping must be enforced and some functionality must be also performed to hide the actual topology. Depending on the scenario, the virtual topology could be defined by the infrastructure provider and exposed to the tenant, or the other way around, the tenant could define the topology and the infrastructure provider must be able to map this virtual topology to the physical one. Therefore, the virtual topology could be exposed to or requested by the tenant. The NV solution must define how this process is performed and how the topology is described (e.g. a topology language).

2.2.5.2 *Network topology not exposed*

The topology of the underlying network is not exposed to the virtual instance. This case is typical when the whole network is abstracted as in the overlay NV type. When using an overlay, the underlaying network has its own addressing scheme and behaviour which is not exposed to the tenants. On the other hand, the tenants build its own virtual network, with its own addressing scheme, based on the functionality provided by the edge devices and tunnels between them. The virtual topology built by the overlay network is not related to the underlying infrastructure and is the result of the interconnexion between the edge devices, while the topology of the underlying network is managed by the infrastructure provider.

2.2.5.3 *Conclusions*

The exposure of the network topology is a characteristic of the NV approach and its relevance depends on the target scenario in which the NV solution is deployed. There are different reasons to hide the underlying topology. Sometimes it is a matter of not exposing the internal details to a third party. In other occasions, there is the need to be completely unrelated to the deployment infrastructure to improve the flexibility of the virtual instance to migrate to another physical infrastructure. However, there are also some other reasons to expose the underlying topology. For instance, it the tenant wants to have complete control over the network (e.g. a network operator) and the sharing of resources is a matter of sharing costs or a regulatory decision. When testing novel proposals and generating performance figures, it is also quite relevant to deal with the actual topology to properly interpret the results.

2.2.6 Virtual Instance Creation Point

The NV types distinguish between the different possible foundation levels at which the virtualisation is exposed to the tenants with regard to the network device architecture detailed above. However, there is a gap between a reference architecture and the physical architecture that actually implements the functionality in the physical elements. The virtual instance creation point (VICP) is more related to the latter one, which means that it considers how and where are implemented the different components and where the needed abstractions take place to build the virtualisation solution. The point where the virtual instance is created could impact the possible alternatives for the characteristics of the NV solution.

Depending on the technology used for virtualising the network, the options could be further detailed (e.g. as in [47]). This classification tries to be agnostic to the technology and these are the most relevant options considered in this analysis, based on the network device as a reference point: internal, proxy, external and edge-based.

2.2.6.1 *Internal*

Virtualisation takes place internally in the network device, which means it is performed in a distributed manner. Each network device must implement the needed functionality and, as any other distributed solution, the consistence of the virtual instance definition across the whole VI domain is key. Both a SDNeV and a vNode type could fit into this implementation model, and most of the legacy proposals for NV implementation are based on this approach, since the network devices are monolithic (i.e., the forwarding and control planes are bundled).

2.2.6.2 *Proxy*

Virtualisation takes place in an external element which is responsible for virtualising the interface between the forwarding and the control plane. Therefore, this approach is mainly related to NV solutions based on SDN technology. However, this does not mean that this category is bound to the SDNeV type of NV, since a vNode approach could be also built with SDN technology. The external element, i.e. the proxy, could be just one (centralised) or a set of elements (distributed). In the latter case, the consistence of the virtual instance definition would be also critical. In general, a proxy-based approach is dependent on the actual protocol used in the control plane, since it needs to properly abstract, translate and virtualise the protocol.

2.2.6.3 *External*

Virtualisation takes place outside the network device, where the control plane resides (i.e., the controller). Once again, this approach is mainly related to NV solutions based on SDN technology, and requires the separation between the forwarding and the control planes, but this does not mean that only SDNeV type of approaches are possible. As in the previous case, a vNode approach could be also created with SDN technology. In this case, the functionality associated to the virtualisation process is implemented on the controller side, and therefore, the approach could be considered as centralised. However, a SDN solution does not need to be centralised and there are several proposals that demonstrate the opposite, such as HyperFlow [54], ONIX [55] and ONOS [56].

2.2.6.4 *Edge-based*

Virtualisation takes place in the edge elements of the network. This is mainly related to the overlay type of NV, which makes use of the edge nodes to implement the mapping between the outside domain and the inside (underlying) network. The virtualisation is implemented in this mapping and all the internal resources are hidden to the overlay. The appropriate coordination in the configuration of the edge devices is critical for the correct implementation of the overlay.

2.2.6.5 *Conclusions*

The options mentioned above are a generic manner to define where the virtual instance is physically created. However, a concrete proposal with a given technology and physical architecture could enrich this study with more options. Although, at first sight it seems to be quite related to the NV type, as previously explained there is not an overlap between both categories, since both SDNeV and vNode approaches could be based on SDN technology. The fourth category reflects the process how the overlays are created, on the edge devices. This differentiation is also needed because the abstraction is performed over the whole network, opposite to the individual network devices as in the other NV types.

2.2.7 Isolation level

One fundamental requirement for any NV solution is the isolation between the virtual instances, which generically means that one virtual instance should not interact with or affect the others. However, this is too generic and there are multiple ways in which one virtual instance could interfere other instances, as analysed in FlowVisor [57]. The VI namespace is just one of the aspects that are related to the isolation of the virtual instances, since the proper assignment of unique VI identifiers allows to isolate the data traffic associated to each instance. In a same way, there are other parameters related to the VI isolation that are analysed, although not all of them are always covered by the NV solutions. A list of possible parameters to consider are detailed below.

2.2.7.1 *QoS support*

When virtualising a resource, one important question to answer to the tenants is how it would perform. Although this is not always an easy question to answer even with dedicated resources, a shared resource needs to be controlled somehow to avoid that one instance is consuming the whole resource or preventing other instances to get access to the resource. When dealing with networks, the bandwidth is one of those resources that is basic to share and which must be properly isolated. Despite it seems to be easy to isolate quotas of the bandwidth of the physical links, there are two main concerns. On the one hand, each system (i.e., network device, vendor, operating system and so on) could have its own way and capacity to isolate the bandwidth that could be also based on different criteria (e.g. priority bits, tags) and, moreover, this does not mean that it is homogeneous in the whole VI domain. On the other hand, a virtual instance could be also defined by virtual links, which could expand several physical links crossing a number of network devices with different QoS support. As a consequence, the QoS support could be covered in the NV solution, but ultimately it will be dependent of the actual deployment. For instance, in SDN there a standard way to control the QoS queues of a network device, but there is not a standard way of defining and managing these queues, this latter aspect being vendor dependent.

The previous study is mainly related to the data plane, however, a similar analysis could be done for SDN-based approaches in relation to the control plane. Since the control traffic between the network device and the controller makes use of an actual network for communication (if not co-located), and even the data packets could be also sent to the controller, this interface must be also analysed from the QoS point of view. A similar policy of bandwidth isolation and traffic prioritisation should be also applied to the control plane southbound interface. The impact of the control plane load has been analysed as a DoS attack for SDN solutions [58]. Moreover, some proposals has been presented to deal with the problem of load distribution at the control plane [55].

2.2.7.2 *Computational resources*

The sharing of resources between the virtual instances also includes computational resources from the network devices and, if it is the case, from external components (e.g. the controller in SDN solutions). The forwarding plane could be performed by software (e.g. CPU) or specific purpose hardware (e.g. ASICs, NPUs). In any case, there is some additional actions related to the overall operation of the network device that requires some computational capacity, such as the internal state keeping and control plane related process. Depending if it is a SDN device or not, those control plane processes are different. In the former case, the network device must handle the requests from the controller, deploy the new flow entries and forward the slow path packets (i.e., those packets that cannot be processed by specialised hardware). In the latter case, the complete control plane must be running on the network device (as in legacy devices). In any case, the available CPU and memory resources must be fairly shared between the virtual instances and proper isolation mechanisms must be implemented. There is reliable technology to perform this task, but it must be deployed also on the network devices. The power consumption or the processing time devoted to each virtual instance could be other parameters to consider for measuring the resource sharing.

2.2.7.3 *Traffic isolation*

Once again, with regard to the traffic, two alternatives must be considered, the data traffic and the control traffic, and both of them must be properly isolated. Concerning the latter, it must be assured that the control traffic associated to each virtual instance is securely delivered to the corresponding tenant, as well as some specific identifiers that must be virtualised (e.g. transaction id and buffer id). In relation to the data traffic, the VI namespace is the key element to assure the isolation. In the context of SDN, the flowspace is a closely related concept, which is a mean to define and slice the header space. Moreover, the flow entry is the most granular way to isolate the traffic in SDN solutions and it is a valuable and limited resource. Therefore, the number of flow entries (e.g. TCAM entries) and the number of forwarding tables (i.e. flowtables) must be fairly divided between the virtual instances.

2.2.7.4 *Topology isolation*

In those NV approaches where topology, either virtual or physical, is exposed to the tenant, it must be assured that only the delegated topology must be visible by each virtual instance. This means that the appropriate mechanisms must be enforced to avoid that a tenant could detect additional network devices that are not exposed through that topology. Furthermore, all the network devices exposed through the topology must be detected by the tenant. This could imply that specific protocols need to be properly emulated or adapted. In proxy based approaches, the proxy must consider the topology exposed to each tenant and implement the needed logic. When dealing with a virtual topologies, it is quite common to implement specific logic to interact with network topology discovery mechanisms and protocols. In this aspect, novel protocols developed for this purpose, i.e. topology discovery, could be a challenge for NV proposals.

2.2.7.5 *Conclusions*

There could be additional parameters relevant for a concrete NV proposal, with regard to the isolation of the virtual instance, that could be also added for the proper characterisation of the approach.

Similarly to some other characteristics, the isolation of resources could be changeable for the whole VI domain, since some aspects could depend on the specifics of the network devices deployed. Therefore, this could be considered as a list of parameters to analyse by any NV solution that must be adapted to the actual deployment. How this aspects are fulfiled with the available hardware depends on the support offered by the physical resources (e.g. support of QoS queues).

2.3 NETWORK VIRTUALISATION SURVEY

# To Be Completed for final version

This introduction needs to be completed for the final version. The content of this introduction is sketched below.

- Focus on NV proposals, not projects or complete scenario description
- Not complete survey, for the purpose of understand the past, current (and future) proposals and the proposed taxonomy
- Apart from the aforementioned taxonomy, the main target scenario is added to the analysis, since it has a fundamental influence on the selected options and required characteristics
- Scenario could be relevant for design decisions/principles
- The main scenarios considered (although not exclusive): operator network, data center network, campus network, and experimental facilities
- Solutions grouped by NV type, and for historical reasons/better introduction and exposition of differences: vNode, overlay, and SDNeV.
- Summary Table: NV name + taxonomy + main target scenario

2.3.1 vNode approaches

Following a similar approach to computing virtualisation proposals, such as VMware, KVM, Xen or VirtualBox, a network can be virtualised by means of the individual virtualisation of the network devices. As a result, each network device is virtualised and multiple instances of the node logic are isolated on the same physical resources. Both virtualisation solutions, i.e., computing and network, share the physical network interfaces between different instances, which means that the traffic must be somehow classified as belonging to one virtual instance or another. At this point is where the VI namespace comes into the scene.

The virtualisation of the network device functionality (including the control logic) internally in the node falls into the vNode category. However, this is not the only way of implementing a vNode solution. Exposing a complete network device (i.e., forwarding and control planes) to the tenant can be also implemented through the SDN technology on the external controller. Both alternatives are analysed through the following examples.

2.3.1.1 *IEEE 802.1Q - VLAN Bridging*

One of the most well know technologies used to virtualise the network is the IEEE 802.1Q standard [7]. This standard defines the VLAN Bridging, which is the process that allows the virtualisation of a bridged IEEE 802 network. This solution is generically known as VLAN (term defined in RFC 4026 [59]), but taking into account the previously introduced taxonomy, the VLAN is just the VI namespace used to identify the associated virtual instance. Following the same characterisation of NV, the IEEE 802.1Q introduces a vNode type of approach by the instantiation of vBridge elements.

The VLAN Bridging network operation is decomposed in the following elements stacked to provide the virtual instances. At the bottom, the physical topology provides the capability of bidirectional connectivity for MAC frames by means of physical links interconnecting the network nodes. On top of the physical topology, the active topology is a logical subset of the physical topology that provides a loop-free topology. There could be several active topologies on top of the physical one, this also allows to recover from link failures. The active topology is typically obtained as a result of a distributed protocol, such as RSTP, MSTP or SPB. The VLAN topology is built on top of an active topology as a subset of this latter topology, which includes those Bridge Ports that are attached to the same virtual instance identified by the VLAN ID. After building the VLAN topology, the end stations that take part in this virtual instance must be located through the station location procedure, which includes the learning of source addresses. On top of this process, the relay functionality is implemented, which comprises the following steps: (1) classification of frames on ingress as belonging to just one VLAN (based on untagged, priority-tagged or VLAN-tagged options); (2) forwarding the frame based on the VLAN topology and station location information (i.e., the result of the learning process); and (3) queuing of frames on egress and determination of the appropriate frame format (i.e., untagged or tagged).

The actual internal isolation of vBridge instances can be configured, since some management decisions determine if some of the process of the bridge are or can be shared. As an example, there is a means to define through the management plane if the MAC learning process is shared by all the VLAN instances or independent per VLAN. In the former, duplicate MAC addresses are not possible, whereas in the latter each VLAN instance could learn the same MAC on different locations without interactions at the VLAN bridging.

Based on the previously introduced taxonomy, the VLAN Bridging is categorized as shown in Table 2.1.

NV Taxonomy	**VLAN Bridging**
NV Type	vNode - vBridge
Main target scenario	campus networks
VI namespace	VLAN identifier
DP Adaptation Mechanism	add/remove VLAN tags on shared physical links
VIAP classification	transparent with untagged ports (e.g. access interface), and non-transparent with tagged ports (e.g. trunk interface)
Network topology	subset of physical topology
VI creation point	internal to the network device
Isolation level	QoS through PCP (Priority Code Point) field of VLAN tags (priority queues), VLAN associated topology, MAC learning per VLAN

Table 2.1: VLAN Bridging categorization based on NV Taxonomy.

The VLAN Bridging is a vNode type of NV, since it builds virtual instances of bridging (vBridge) that are isolated per tenant. This solution has been widely deployed in campus networks as a simple and effective way of creating isolated broadcast domains. Each network node must be properly configured to obtain the desired isolation at network level, which means that the appropriate configuration needs to be distributed through all the nodes. Thus, the virtual network is created by individual configuration on each and every network node.

The VLAN identifier is the VI namespace that uniquely identifies each virtual instance and the common field used network-wide to univocally define and identify the tenant.

The adaptation mechanism used at data plane is the addition/removal of the VLAN tags to the user's frames on those physical links that are shared between several virtual instances (by using trunk interfaces). If a physical link is not shared with other virtual instances, there is no need to add/remove the VLAN tag, since the whole interface is dedicated to one virtual instance. Anyway, the network node must know the VLAN identifier associated to this untagged interface for proper isolation at node level.

Regarding the classification of the VIAP, there are two possible options: transparent and non-transparent. The former refers to those cases in which the interface directly attached to the user is exclusive to one virtual instance, and then, there is no need to tag the frames on ingress (i.e., frames are untagged on interfaces in mode access). The latter refers to those occasions in which the user needs to tag the frames, either because the physical link is shared by multiple virtual instances or because it is decided by the network administrator (i.e., frames must be tagged on interfaces in mode trunk).

Concerning the network topology, each virtual instance could use a different active network, but all of them are a loop-free topology, which is a subset of the physical one. In summary, the VLAN topology of each virtual instance is a subset of the physical topology that connects the physical ports of users associated to the instance. Moreover, this VLAN topology automatically recovers from link failures if there is an alternative loop-free tree in the physical topology.

In relation to the VI creation point, the virtual instance is created internally in the network device. In fact, the VLAN identifier is also used internally to differentiate the internal processes, such as the MAC learning (which could be done per VLAN) or the rely (classification, forwarding and queuing based on the VLAN identifier).

With regard to the isolation level of the VLAN Bridging solution, several aspects can be highlighted. Firstly, the QoS support is defined in the standard, which is related to the Priority Code Point (PCP) bits included in the VLAN tag. This PCP bits allows to accordingly process the priority queues in which the frames are buffered based on its value. Secondly, the topology associated to each VLAN identifier is isolated from the others and does not interfere with them. Moreover, the MAC learning process can be performed per VLAN (by proper configuration). In principle, the computing resources at the network node are not typically isolated per VLAN and the number of leaned MACs are not ordinarily controlled per instance.

2.3.1.2 *Virtual Routing and Forwarding*

With regard to the Virtual Routing and Forwarding (VRF) technology, there is some controversy related to the terminology. From the IETF perspective, the RFC 4026 [59] introduces the VRF as "VPN Routing and Forwarding" to refer to the capability of PE routers to maintain a per-site forwarding table in the Virtual Private Network (VPN) use case. The virtual table used in each case is selected depending on its source site. Additionally, in the context of VPNs based on BGP/MPLS technology (defined by RFC 4364 [60]), the "VPN Routing and Forwarding" tables (VRFs) are defined as separate forwarding tables on each PE router (i.e service provider's edge router). However, vendors typically refer to the VRF as "Virtual Routing and Forwarding", not making explicit reference to the main target use case, the VPNs, being a more generic concept of creating individual routing and forwarding tables per virtual instance. In [61] there is an explicit reference to the VRF acronym, which explains that VRF is used equally for "VPN Routing and Forwarding" and "Virtual Routing and Forwarding" in Cisco literature, although the latter one is more used by vendors (e.g. Cisco, Juniper, Brocade and so on).

In [62], VRF is defined as an IP technology that enables the instantiation of multiple routing tables on the same router simultaneously. Each of these instances are isolated, and allows to define overlapping IP addresses without conflict between the different routing tables. It is also mentioned that VRF is also used to refer to the multiple routing table instances per each VPN that coexists on a Provider Edge (PE) router, referring to the VPN use case for VRF mainly related to MPLS core networks. A different routing protocol could be run between neighbouring routing instances to individually populate the tables. Then, the concept of VRF-lite is introduced as an extension of the VRF concept to the Customer Edge (CE) router, that also supports multiple, independent and overlapping routing and forwarding tables per each customer. In this latter case, there is no MPLS functionality on the CE, and no label exchange is done between the CE and the PE. In this case, the VRF uses the input interfaces to differentiate the routes associated to each virtual instance, which then populate the virtual forwarding tables by attaching one or more Layer 3 interfaces to each VRF instance. This interfaces can be physical or logical (e.g. VLAN interface), but they can only belong to one VRF instance at one time. As an evolution of the VRF-Lite solution, Cisco introduced the Easy Virtualisation Network (EVN) [63] to simplify the VRF configuration process by simply creating a trunk, called Virtual Network (VNET) trunk, between the routers. The EVN automates the process of uniquely tagging the packets belonging to each different virtual instance.

Based on the previously introduced taxonomy, the VRF is generically (no separate study is performed for VRF and VRF-Lite) categorized as shown in Table 2.2.

The VRF proposal is a vNode type of NV, in which isolated virtual instances of routing and forwarding tables (vRouter) are created per tenant. This is a legacy solution widely used for VPN deployments in operator networks, as an effective way to share the physical resources from the service provider (PEs) and the customer (CEs) to create virtual private networks, initially based on MPLS technology. Later on, this dependence on the MPLS support was removed, and the use case was also extended to campus networks, in which the routers can be easily shared by creating multiple isolated vRouters. Due to the distributed nature of any NV approach based on network devices, the proper configuration of each individual node is fundamental for the coherent virtualisation at network-wide level. EVN approach tries to simplify and automate precisely this configuration process.

Regarding the VI namespace, both VLAN identifiers and MPLS labels are used to identify the virtual instance depending on the actual implementation (e.g. VRF or VRF-Lite) and the location (e.g. customer's site or service provider's network). For instance, at the routing protocol level it is also important to differentiate the virtual instance.

At data plane, the adaptation mechanism depends on if a tag/label is needed, and in case it is required (when the interface is shared between several instances), the tag/label is added/removed on the user's traffic. When the interface is associated to just one virtual instance (i.e., the link is not shared), the add/remove mechanism is not needed and the interface is uniquely associated to one VRF instance.

With regard to the VIAP classification, both transparent and non-transparent solutions are possible. When the interface is

NV Taxonomy	**Virtual Routing and Forwarding**
NV Type	vNode - vRouter
Main target scenario	operator networks and campus networks
VI namespace	VLAN identifier (VRF-Lite) and MPLS labels (VRF)
DP Adaptation Mechanism	add/remove VLAN tags/MPLS labels on shared physical links
VIAP classification	transparent with dedicated interfaces, and non-transparent with logical interfaces (e.g. VLAN tagged)
Network topology	subset of physical topology
VI creation point	internal to the network device
Isolation level	QoS mechanisms supported at the MPLS router and priority queues based on PCP (Priority Code Point) field of VLAN tags, VRF associated topology, isolated routing processes

Table 2.2: VRF categorization based on NV Taxonomy.

exclusively associated to one virtual instance, there is no need to perform any action on the user's side and the process is transparent. On the other hand, when the same interface is shared between different users (or because of an administrative decision even if the interface is not shared), each user must add/remove the appropriate VI identifier to the traffic.

Regarding the network topology, each virtual instance could have a different topology based on the routing decisions and the location of the end users associated to each instance. However, all of them are a subset of the physical topology.

Concerning the VI creation point, the virtual instances are generated internally in the network device, and individual routing processes and isolated routing and forwarding tables are assigned to each instance.

In relation to the isolation level that can be achieved with the VRF solution, there are some aspects to consider. Firstly, as previously said, parallel and isolated routing processes are run per virtual instance. Moreover, the routing and forwarding tables are virtualised and each VRF instance has its own tables (not shared). Secondly, the QoS support from legacy technologies such as MPLS and IEEE 802.1Q (i.e., priority queues) can be exploited by the VRF instances, and isolated virtual networks could have different associated SLAs. Additionally, the topology associated to each VRF is isolated from the others and does not interfere with them. In principle, the computing resources at the network node are not commonly isolated per VRF and the size of routing and forwarding tables are not typically controlled per instance.

2.3.1.3 *VIrtual Network Infrastructure (VINI)*

The authors in [64] presents a virtual network infrastructure (VINI), which allows the network researchers to deploy their novel protocols and services in a realistic scenario. The infrastructure provides real routing software, network events and traffic loads to test their proposals over isolated virtual networks that share the same physical resources at the same time. It allows to build flexible network topologies, which is fundamental to evaluate new routing protocols. To achieve this flexibility, the infrastructure provides a wide variety of nodes and links that could be configured to construct the desired topology. Moreover, the nodes provide the ability to attach an arbitrary number of interfaces that could be connected to other nodes. This flexibility is based on the usage of virtual interfaces, virtual nodes and virtual links. On top of these flexible topologies, VINI also provides a flexible forwarding and routing, which implies that the traffic associated to each virtual instance must be isolated along a particular path, and the distributed information that dictates how the traffic is forwarded must be also isolated.

An implementation of VINI in PlanetLab [41], called PL-VINI, demonstrates the viability of this proposal on a real deployment. PlanetLab is an overlay testbed that provides virtual servers (VServers) as lightweight slice of each node available in the testbed. Then, the computational resources are properly isolated as VServers in order to create the PL-VINI nodes. The slice topology is built as as overlay topology that must be exposed to the tenants and user traffic is tunneled between the virtual nodes. Moreover, the network isolation on PlanetLab relies on the VNET [65] module, which multiplexes the incoming and outgoing traffic and provides the illusion of root-level access to the underlying network interfaces. This module also allows to create an arbitrary number of interfaces on the virtual node, although they could be sharing the same physical interface. Based on these two building blocks, i.e., VServers and VNET, the PL-VINI nodes construct the solution. Each PL-VINI node provides an isolated virtual router to the tenant and is composed of a routing process and a forwarding data path. On the one hand, the routing software used to build the control plane is XORP [XORP], which runs inside User-Mode Linux [UML] to assure its isolation from other routing processes. On the other hand, the Click modular software router [Click] is used as the forwarding engine (and additionally as a network address translation for egress traffic). In order to build the virtual network devices attached to the virtual router, a modified version of Linux TUN/TAP driver is used to isolate the traffic between different slices on PlanetLab.

Based on the previously introduced taxonomy, the VINI approach is categorized as shown in Table 2.3.

The VINI proposal is a vNode type of NV that allows to create complete virtual routing instances (vRouter) isolated per tenant. The main aim of this approach is to provide a framework for network researchers to test and validate new routing protocols. Therefore, the principal target scenario is the experimental facilities. In fact, a proof of concept of VINI has been implemented for PlanetLab (PL-VINI), which is a testbed for researchers.

There is no specific VI identifier defined to identify the virtual instance, but UDP tunnels are used to create virtual links building an overlay, in which the end-to-end tunnel is the means to isolate the traffic and associate it to the appropriate PL-VINI instance.

As a consequence of this overlay, the adaptation mechanism used at data plane is the addition/removal of the UDP tunnel to identify on the other end the PL-VINI instance. This process of UDP encapsulation is extended even until the users, which need to install some software, e.g. OpenVPN, to perform the proper UDP encapsulation. Moreover, this could be additionally used to secure the access to the slice, and thus, authenticate and authorise the access to the virtual instance at data plane. However, as any other encapsulation mechanism, this adds an overhead at data level.

NV Taxonomy	**VLAN Bridging**
NV Type	vNode - vBridge
Main target scenario	campus networks
VI namespace	VLAN identifier
DP Adaptation Mechanism	add/remove VLAN tags on shared physical links
VIAP classification	transparent with untagged ports (e.g. access interface), and non-transparent with tagged ports (e.g. trunk interface)
Network topology	subset of physical topology
VI creation point	internal to the network device
Isolation level	QoS through PCP (Priority Code Point) field of VLAN tags (priority queues), VLAN associated topology, MAC learning per VLAN

Table 2.3: VINI categorization based on NV Taxonomy.

Accordingly, the VIAP classification is non-transparent, since the users need to install and configure the tunneling software on the end device.

In relation to the network topology, the proposal was designed to provide the ability to define an arbitrary topology. The flexibility at the network topology is based on an overlay solution, which though the proper encapsulation, allows to define virtual links between any physical nodes. As a consequence the topology is virtualised through the overlay and exposed to the tenant (i.e., it is not hidden).

With regard to the VI creation point, the PL-VINI instance is created internally in the VServer node. Virtualisation techniques such as UML and VNET are used to this aim, allowing isolated routing and packet forwarding processes.

Regarding the isolation level, there are some features that are inherited from PlanetLab and not exclusively related to VINI (e.g. VServer and VNET). As previously mentioned, the both the routing and forwarding tables and processes are isolated per tenant based on UML and Click software. The overlay based on UDP tunnels allows to isolate the topology defined by each tenant, not interfering with other topologies. However, the virtual topology would be affected by the underlaying physical topology on which it is build. This also allows the routing protocols to be tested under real failures and events. How this physical failures are propagated into the virtual topology is not an easy task. Moreover, the virtual links can be mapped into several underlying physical links, which may be using different technologies, and not all of them implement a proper sharing mechanism. This means that the traffic from one experiment could affect the network conditions from another experiment. Concerning the computing resources, they are properly virtualised at the VServer and each vRouter has its own isolated environment to run the routing processes. Moreover, the virtual interfaces are properly isolated by VNET.

2.3.1.4 *Programmable-Flow*

The Programmable-Flow (P-Flow) [66] is a NV solution proposed by NEC, in which multiple interconnected switches can be virtualised as a resource pool of networking elements that can be dynamically controlled. The forwarding behaviour of the physical switch devices can be adapted and controlled to implement Layer 2 to Layer 4 functionalities. In summary, this technology is based on the separation of the forwarding plane and control plane and the decoupling of the logical network design and the physical network operation. In P-Flow the control plane related functionalities are extracted from the network device to be implemented in an external entity, the controller, which typically runs on a server appliance. The OpenFlow protocol is used to interconnect the controller with the physical network devices. By using an standard protocol, the P-Flow is able to support switches from other vendors, including software-based switches, such as Open vSwitch (OVS).

On the one hand, P-Flow allows to overcome some of the limitations of other vNode approaches (e.g. IEEE 802.1X or VRF) based on internal VI creation point, such as the individual configuration and management of the network devices. On the other hand, in the case of overlays, they could become difficult to manage when the interconnexion between end nodes scales in number of tunnels. By introducing a network level virtualisation, P-Flow allows to hide the underlying physical network to expose a simple virtual network view, which is easier to configure, manage and operate. Then, this virtual network is decoupled from the physical one, which allows to have multiple virtual networks sharing the same resources while isolates the behaviour of each virtual instance.

The separation between the logical and the physical plane allows P-Flow to introduce a unique network design, in which a logical abstraction plane capture the high level network functionality that can be mapped automatically into any underlying physical network topology. The high level functionalities are dynamically created by adding/removing/modifying flow rules at individual switches through their OpenFlow interface exposed to the controller. The logical abstraction introduced by P-Flow is called the Virtual Tenant Network (VTN), a concept that can be also exported and implemented in other software projects such as in OpenDaylight (ODL)[4] (by NEC developers).

4 OpenDaylight Virtual Tenant Network (VTN): https://wiki.opendaylight.org/view/OpenDaylight_Virtual_Tenant_Network_%28VTN%29:Main

This VTN application provides a multi-tenant virtual network on the ODL controller.

Based on the previously introduced taxonomy, the P-Flow approach is categorized as shown in Table 2.4.

The P-Flow is a vNode type of NV that allows to implement different logic on the vNode. Some of the control logic available in P-Flow are: vBridge, vRouter, vExternals, vFilter, and vRedirect. The vBridge allows to build L2 networks, while the vRouter is the basic component to build a L3 network. Moreover, the vExternals allows to interconnect the network to the end points, which could be either a server (i.e., a physical or virtual machine) or a legacy network device (e.g. router or switch). The vFilter implements any L2-L4 based Access Control List (ACL) by the proper definition of the matching on the header fields and associated action, e.g. allow or deny (implemented as output or drop). Finally, the vRedirect is associated with a vFilter, and allows to create an explicit forwarding path from a vBridge/vRouter to a vExternals. Any of these vNodes are isolated per tenant and interconnected based on the logical network design provided by the tenant. This solution is mainly focused on data center networks as a means to avoid exposing the physical complexity of a data center (e.g. topology) to the tenants. Moreover, this approach allows to simplify

NV Taxonomy	Programmable-Flow
NV Type	vNode
Main target scenario	data center networks
VI namespace	VLAN identifier
DP Adaptation Mechanism	add/remove VLAN tags on shared physical links
VIAP classification	transparent with untagged ports (i.e., not shared interface), and non-transparent with tagged ports (e.g. shared interface)
Network topology	virtual topology
VI creation point	external in the controller
Isolation level	support existing legacy features, control logic isolated (different behaviours), virtual topology decoupled

Table 2.4: P-Flow categorization based on NV Taxonomy.

the management of the network by removing the location-dependent constrains, for instance in the common scenario of virtual machine migration, which has a great impact on the network resources.

Although not explicitly mentioned in technical descriptions (i.e., it is a product from a vendor), the VLAN identifier is shown in some deployment scenarios as the VI namespace used to univocally identify the associated virtual instance and also the means to uniquely identify the tenant at network-wide level. This information has been also contrasted with the VTN implementation in ODL.

Also based on this source of information, the adaptation mechanism used at data plane is the addition/removal of the VLAN tags to the frames of the users. This process is needed when the physical link is shared between multiple virtual instances, but it is not needed when this interface is dedicated exclusively to one virtual instance. Although this procedure is not needed in the latter case, the network node must associate a VLAN identifier to the untagged interface to properly isolate at node level.

Regarding the classification of the VIAP, two options are feasible: transparent and non-transparent. The transparent VIAP allows untagged frames coming from the user to be uniquely associated to the target virtual instance by two means: the port of the physical switch is dedicated exclusively to the virtual instance or the MAC address is associated to the tenant. The non-transparent VIAP is used typically when the physical port is shared between multiple virtual instances, and the user needs to tag the frames to identify the tenant responsible for their control and forwarding.

Concerning the network topology, the P-Flow introduces the VTN as a logical representation of the network that is a virtual representation that needs to be mapped later on to the physical infrastructure. This virtual topology representation allows the virtual instance to be decoupled from the physical deployment, which is a fundamental requirement in scenarios like data center networks. The controller is responsible for performing this mapping between the virtual and physical topologies in a dynamic way.

In relation to the VI creation point, the virtual instance is created external to the network device, specifically in the controller. The mapping between both topologies is just one of the responsibilities of the controller, it must also implement and isolate the logic associated to the virtual nodes and then map this logic to the physical resources.

With regard to the isolation level of the P-Flow approach, it is built to support existing legacy features, which means that the QoS support offered by the network devices must be available to build the solution. Moreover, the control logic associated to the vNodes must be isolated on the physical devices (i.e., multiple instances on the same network element) and per tenant (i.e., the vNodes from one tenant must not interfere with the vNodes from other tenants). Additionally, the control logic is different between vNodes, and this logic must be properly isolated and no collision in the behaviour must be enforced. As one of its main contributions, the virtual topology is decoupled (i.e., VTN) from the physical infrastructure and does not interfere with other virtual topologies from other tenants. In principle, the computing resources are not isolated at network node and the impact on the number of flow entires actually deployed on the physical resources depends on the mapping algorithm, which must be optimised on the controller side, but a flow entry quota is not specify.

2.3.2 Overlay approaches

There are a set of NV approaches that have some common characteristics and similar overall requirements which are based on the abstraction of the network as a whole (opposite to network device abstraction). This means that there is no exposure of the internals of the underlying network to the tenants and the solutions are built on top of some functionality added on the edge network devices (i.e., basically mapping and encapsulation). All these proposals are categorized as overlay type of NV for the purposes of this study. As a result, the underlying network is shared between multiple virtual instances, which are isolated per tenant. Moreover, the topology is not exposed and each virtual instance defines its own topology based on the interconnexion defined between the edge devices.

The addressing space from the underlying network is decoupled from the addressing space used by the tenants, and the mapping between both addressing spaces is mapped on the edge device. Besides, the addressing space from one tenant is isolated from the addressing space used by another tenant, thus, enforcing the isolation at tenant level.

The basic idea behind the implementation of an overlay is straightforward. First of all, a new overlay is provided each time a virtual network is instantiated. Then, the edge device that provides access to the underlying network performs the encapsulation (and de-encapsulation on the egress) on the packet on the ingress. This encapsulation is defined in such a way that the destination of the encapsulated packet is the edge device that must de-encapsulate the packet just before it is delivered to the final endpoint. The addressing space used in the encapsulation header is focused on the interconnexion between the edge devices, and has no knowledge about the tenant's addressing space. It is the edge device the entity who knows how to encapsulate one addressing space into the other in order to reach the destination endpoints defined by the tenant. The underlying network forwards the packets based on its own addressing space (used in the encapsulation header), which simplifies the core network and improves the scalability of the solution, since the forwarding process does not need to consider and learn the information from the tenants. This means that the number of the virtual instances do not affect the underlying network.

As previously mentioned, the main common characteristics of any overlay approach are: there is no topology exposed to the tenant, the edge device performs the encapsulation/de-encapsulation of packets, the addressing spaces are isolated, and the mapping between both addressing spaces is performed on the edge device. However, there are some differences between the overlay proposals depending on the target scenario. Here, there are two main target scenarios that group the following study: operator networks (RFC 4026 [59]) and data center networks (RFC 7364 [48]). The former refers to the provider provisioned VPN that will be referred as VPN approaches, whereas the latter refers to the overlays for virtual networks for providing multi-tenancy in large data centers.

One business case that has been a source of revenue for many operators is the VPN solutions. A VPN can be categorized as an overlay with some security aspects added (e.g. authentication, authorisation, cyphering, and integrity), although not all the VPN solutions cover the security attributes. In any VPN architecture defined by the IETF, such as in RFC 4664 [67] and RFC 4110 [68], there is a Customer Edge device (CE) and a Provider Edge device (PE), which distinguished between the tenant's edge device (CE) and the operator's edge device (PE) that are connected and have a specific logic to build the VPN solution. This solution is related to the operator networks that provide a means to interconnect distributed sites from the same tenant in a secure and isolated way by using the operator's network. A huge plethora of technologies are available to provide this type of solution. In order to organise and group all these proposals, they are further categorized into L1VPN, L2VPN, L3VPN, and higher-Layer VPN. The L2VPN and L3VPN are the most typical implementations for provisioning a VPN, which differentiate the type of connectivity that is transported through the VPN service. The former defines that a Layer 2 connectivity is provides between the tenant's sites, whereas the latter implies that this connectivity is performed at Layer 3 (i.e., IP). This Layer 2 / Layer 3 does not refer to the technology used by the underlying network to forward and isolate (by means of the encapsulation) the traffic sent to/from the tenants. This means that a L2VPN can be encapsulated for instance on a Layer 2, Layer 3 or Layer 4 header. There are also L1VPN and higher-Layer VPN proposals, but they are not so common. A L1VPN means that the tenant's sites are connected at Layer 1, whereas higher-Layer VPN, such as SSL VPN, means that the connexion is performed on top of SSL/TLS (RFC 5246 [69]).

The data center network scenario has its own particularities that make to different from the VPN service in operator networks. The multi-tenancy has been one design requirement imposed to any data center network approach, in which each tenant is able to define their virtual networks that must be isolated from other tenants. The solution must scale to hundreds of thousands of endpoints (i.e., the number of Virtual Machine (VM)s expected to be deployed in large data centers), and this requires that the amount of state associated to the tunneling mechanism (i.e., the overlay encapsulation) must be less than in the case of the VPN solutions. Moreover, it is typical to have a virtualised environment (i.e., computing virtualisation solution) in which the end users are deployed as VMs in a virtualised server (e.g. VMware, KVM, Xen, or Virtual Box). In this cases, the hypervisor responsible for virtualising the hardware of the servers is a key component to build the network overlay in data center scenarios. The virtual switches (i.e., software switches) available in the hypervisor (or sometimes the adjacent device) are responsible for mapping and encapsulating (and de-encapsulating) the traffic from the VMs. The main difference with the previous scenario is that the VMs (and their virtual NICs), the hypervisor and the data center network are provisioned, controlled and administered by the same entity. How the virtual network is implemented is responsibility of the data center operator and the tenant does not care about it. The tenant wants an isolated network mainly performing some bridging or routing functionality between its VMs (i.e., Layer 2 and Layer 3 overlay).

Due to the aforementioned distinction between VPN services and data center overlays they are analysed separately, then some standard implementations are mentioned to illustrate possible solutions. Afterward, some concrete examples are individually analysed for particular reasons: NSX VMware [70], NetLord [71], Diverter [72], and PortLand [19]. The NSX VMware is a complete approach for a data center software (i.e., computing and networking are integrated in the same solution). NetLord implements a novel MAC in IP encapsulation including the VI identifier in the destination IP address. Diverter performs a MAC address rewrite using VNET module [65] as the encapsulation mechanism. Finally, PortLand proposes to forward the frames based on MAC prefixes. Although this last proposal does not support NV, it is analysed because the relation of a MAC prefix approach with the Layer 2 Prefix-based Network Virtualisation (L2PNV) contribution described in Section 3.1.

2.3.2.1 *Virtual Private Network (VPN)*

As previously explained, the VPNs are categorized as an overlay with some security aspects added. It is considered as an overlay because the VPN is a virtual network that makes use of the underlying networking to provide a secure communication that is built on the edge devices (i.e., the topology is not exposed). The VPN addressing space is isolated from the underlying addressing space by some kind of encapsulation, which is performed on the edge device after some kind of mapping. Although not all the VPN solutions implements a secure connexion between the tenant's sites, the authentication, authorisation, cyphering and integrity are the most common security aspects covered by a VPN proposal.

The characterisation of VPN solutions as a NV approach does not vary so much from one proposal to another, so this study is performed once and any particularization will be highlighted if needed. However, in order to clarify the scope of this study, the VPN proposals covered on it are described. Generically, the provider provisioned VPN solutions are categorized into L1VPN, L2VPN, L3VPN and higher-Layer VPNs. Just a few insights on each category before their analysis as a NV solution that is mainly focused on L2VPN and L3VPN [73] (there other two categories are briefly introduced).

The L1VPN (RFC 4847 [74]) is a service that provides Layer 1 connectivity between two or more tenant's sites that is offered by a Layer 1 core network. Moreover, the tenant has some kind of control over the establishment of the connexion and the type of this Layer 1 connectivity. For clarification, a Layer 1 network is a transport network that typically performs spatial switching between fibers, time-division multiplexing (TDM), or lambda switching. This category of VPN provides high bandwidth, but less granularity and flexibility than other solutions.

Regarding the higher-Layer VPNs, the National Institute of Standards and Technology (NIST) from the U.S. Department of Commerce presented a guide to the SSL VPNs [75]. In this study they present the justification for higher-Layer VPNs as an alternative to IPsec (RFC 4303 [76]), and introduce two types of solutions: SSL portal VPN and SSL tunnel VPN. The former allows a user to securely access multiple network services through a single standard SSL/TLS connexion to a Web site, whereas the latter allows to securely access the same services through a SSL/TLS tunnel that is running on a typical Web browser with some active content.

The RFC 4026 [59] is an informational standard that presents the terminology used in provider provisioned VPNs, mainly related to L2VPNs and L3VPNs.

The L2VPNs (RFC 4664 [67] and [73]) can be further classified in two types of services: virtual private wire service (VPWS) and virtual private LAN service (VPLS). A third category, IP-only LAN service (IPLS), is a subcategory of the VPLS service. The VPWS provides a Layer 2 point-to-point service, whereas the VPLS emulates a LAN service (i.e., multipoint connectivity). The former could take into account the VLAN or deliver a transparent port-based service, transported over a pseudowire. The latter is also based on pseudowire creation between the PE devices, which also perform MAC address learning and MAC forwarding similar to a other switch on the LAN from the user's perspective. In the case of IPLS, the MAC forwarding tables are populated through a protocol (rather than learning) and the ARP is proxied (instead of transparently forwarded). For this reasons, the IPLS imposes a router as a CE, instead of a switch possible in VPLS solutions. The actual technology used to provide those services could be either IP, MPLS or Ethernet. For instance, there is a set of control protocols and encapsulation mechanisms available to provide Layer 2 connexions between IP nodes, such as L2TPv3 (RFC 3931 [77]), PPTP (RFC 2637 [78], although it is not a IETF standard, it was developed by a vendor consortium) or GRE (RFC 2784 [79], generic protocol to encapsulate network protocols). The IEEE also provides technology to deliver L2VPN services through the Provider Bridges (IEEE 802.1ad amendment, also known as Q-in-Q) and Provider Backbone Bridges (IEEE 802.1ah amendment, also known as MAC-in-MAC) , which are now part of the IEEE 802.1Q-2014 standard [7]. The former proposes to add an additional VLAN tag for the service provider, whereas the latter proposes to encapsulate the user's MAC header into a new MAC header managed by the service provider.

The L3VPNs (RFC 4110 [68], and [73]) can be classified in two different types of services: CE-based and PE-based. The former can be deployed by the tenant without any specific support from the infrastructure provider of the underlying network beyond the access to Internet. However, in this type of CE-based L3VPN services the CE device must provide all the functionality needed to build the VPN. In the latter service, the infrastructure provider (i.e., the operator of the underlying network) must configure the PE devices to provide the VPN connexions between the tenant's sites. In this case, the CE devices does not need to perform any special functionality apart from the ordinary routing functionality. Both the BGP/MPLS IP VPNs and virtual router IP VPNs solutions are used to build PE-based L2VPNs, whereas the IPsec VPNs (RFC 4303 [76]) are typically a solution for CE-based L2VPNs.

Based on the previously introduced taxonomy, the VPNs are generically categorized as shown in Table 2.5.

The VPNs are generically categorized as an overlay type of NV, since they build a virtual network over the underlying infrastructure by means of the edge devices that are isolated per tenant. Some security support could be added to the overlay, since it is typically deployed over a shared and unreliable network (e.g. Internet). The VPNs have been a successful business service deployed in operator networks as a cheaper alternative to private networks and leased lines between the tenant's sites.

NV Taxonomy	**Virtual Private Network**
NV Type	overlay
Main target scenario	operator networks
VI namespace	dependent on the tunneling mechanism
DP Adaptation Mechanism	add/remove encapsulation header
VIAP classification	transparent in PE-based VPNs, and non-transparent in CE-based VPNs
Network topology	not exposed
VI creation point	edge device (either in the PE or in the CE)
Isolation level	QoS support depending on the underlying technology, addressing space is isolated (between infrastructure provider and tenants)

Table 2.5: VPN categorization based on NV Taxonomy.

The VI namespace, or the exact manner in which the virtual instances are uniquely identified, depends on the actual tunneling technology used to build the VPN, which it is also related to the type of VPN that is provided and the actual technology used by the underlying network. As previously mentioned, there are a lot of possible tunneling technologies that can be used to provide, for instance, L2VPN and L3VPN services. Each technology provides a means to uniquely identify the virtual instance (i.e., the VPN tenant) both at the ingress and at the egress. The study of all these possible encapsulations are out of the scope of this study, but some of them are: MPLS, GRE, L2TPv3, PPTP, Q-in-Q, MAC-in-MAC, IPsec and so on.

In all these mechanisms, the data plane adaptation mechanism is the same, the addition/removal of the encapsulation header (which is dependent on the technology) to the user's traffic on the edge devices. This procedure is implemented on the PE/CE and some additional functionality would be needed (e.g. MAC learning, ARP proxy and so on) depending on the VPN solution and the technology used. This encapsulation mechanism also provides the ability to isolate the addressing space of the tenants from the one used by the underlying network.

With regard to the VIAP classification, two possible options are considered: transparent and non-transparent. On the one hand, when the VPN is built on the PE devices, the service is transparent to the users, since all the configuration and management is performed by the operator. It is on the PE where the encapsulation process (and some others) takes place. On the other hand, when the VPN is configured and managed on the CE devices, the service is not transparent to the users. In this latter case, the operator does not need to perform any action since the encapsulation and some other associated processes take place.

Regarding the network topology, as any other overlay solution, the underlying topology is not exposed to the tenants, since the network as a whole (including its functionality) is abstracted and the internal devices are not exposed at all. However, this does not mean that the overlay could define its own topology based on the tunnels stablished over the underlying network.

Concerning the creation point of the VI, the virtual instances are created on the edge devices, either on the PE or on the CE (depending on the approach). The underlying network is dedicated to forward the encapsulated packets as any other packets not related to the VPN and it does not take part in building the VI instance.

In relation to the isolation level, the different VPN solutions could make use of the QoS support provided by the underlying technology (if it is supported). The QoS could be an additional parameter of the VPN service to be provided. As previously mentioned, the addressing space is isolated between the tenant and the infrastructure provides, as well as between the different tenants (or virtual instances). Finally, the computing resources of the edge devices could be isolated, although it is not so common in current implementations.

2.3.2.2 *Data Center overlay*

Large data center networks are challenging because they need to scale to hundreds of thousands of endpoints and multi-tenancy is a mandatory requirement. A overlay-based NV approach has been proposed to support this multi-tenancy at scale providing isolated virtual networks (through some king of encapsulation) that are built on the edge elements of the data center.

RFC 7364 [48] presents the overlays as a mechanism to implement NV solutions for large data centers. One of the main particularities of data centers is that the same entity is responsible for providing computing, storage and networking resources. With regard to the network, the data center operator needs to provide the isolated virtual instances to the tenants, and in this case, these virtual networks are overlays on top of the underlying infrastructure. Some additional functionality must be provided to deploy the virtual network instances, such as the association of the VM's network interfaces with the corresponding virtual network. This relationship must be maintained when the VM is activated, migrated or even deactivated. In this context, the hypervisor covers a key role in the adequate interconnexion between the virtual interfaces and the overlay, thus, the hypervisor becomes the edge element responsible for encapsulating (and de-encapsulating) the user's traffic.

In any multi-tenancy solution, the traffic isolation between tenants is a key requirement. One tenant must not be able to access traffic from another tenant. Moreover, the address space isolation allows to different tenants use the same addressing scheme on different virtual network instances. This means that the same network addresses can be used at the same time in different virtual network instances. Furthermore, the address space used by the tenants is isolated from the addressing used by the underlying infrastructure. In order to achieve this level of isolation, each tenant is able to instantiate one or several network instances. The only way to sent traffic from one virtual instance to another is through a security gateway or a shared router (accessible simultaneously by several virtual instances). The same control is performed when the traffic from one tenant needs to exit the virtual network to Internet, it can only be done through controlled exit points and the corresponding policy.

In general, the way in which the virtual network is implemented does not matter to the tenant. The important aspect is that the semantics and performance of the network service provided (e.g. Layer 2 or Layer 3) is preserved. Then, it could be implemented by means of a bridged or routed network (or a combination).

As previously mentioned, the scalability of the solution is an issue. For instance, some overlay proposals (e.g. L2TPv3) require significant state associated to the tunnel that must be stored on the endpoints for encapsulating and de-encapsulating the user's traffic. The overlay approaches to be used in large data center networks require less state to be associated to the tunnel. Scaling the solution to hundreds of thousands of endpoint is not easy when the associated state is significant. In the end, the virtual switches provided in the hypervisor must be responsible for performing the encapsulation and de-encapsulation of all the user's traffic, accomplishing a basic function as the overlay edge component. When the hypervisor is not present (e.g. physical server) or the virtual switch is nor able to perform such functionality, the adjacent device will be responsible for mapping and encapsulating the traffic.

The specific encapsulation mechanism is not relevant for the categorization of data center overlays as a NV proposal based on the proposed taxonomy. However, this mechanism is fundamental from the networking and performance point of view. The requirements imposed by any of these encapsulations must be analysed to select the most adequate to be deployed. Some of the most relevant overlay-based proposals for data center NV are: VXLAN (RFC 7348 [49]), Geneve [52], NVGRE (RFC 7637 [50]), and STT [51].

Virtual eXtensible Local Area Network (VXLAN, RFC 7348 [49]) is an overly technology that allows to virtualise Layer 2 networks over Layer 3 networks. It is mainly focused on data center scenario to enable multi-tenancy support. The main issues to overcome are: the limitation VLAN space (i.e., limit of 4094) to isolate the tenant's virtual networks at scale in a data center, the limitation of Spanning Tree Protocol (STP) to distribute the traffic through redundant paths (i.e., large number of disabled links), and the inadequate table sizes at Top of the Rack (ToR) switches that must scale to hundreds of thousands of VMs. Relying on a Layer 3 network, i.e., IP, and an encapsulation scheme (VXLAN header) allow to overcome those problems. Each VXLAN segment (i.e., virtual network instance) is identified through a 24-bit identifier, the VXLAN Network Identifier (VNI). The VXLAN header is encapsulated into a UDP outer header with a destination port (value 4789) assigned by the IANA for VXLAN. Moreover, VXLAN defines unicast traffic for VM to VM communication, whereas the broadcast communication is mapped into multicast traffic. The support of IP multicast is one of the main limitations of VXLAN. The fragmentation of VXLAN packets on the source is avoided, however, intermediate routers could fragment the packets due to the larger frame size, and Path MTU discovery is proposed to avoid fragmentation. There are not specific security measures cover by the standard.

The GRE header encapsulation has been proposed as a mechanism to support NV by the NV using Generic Routing Encapsulation (NVGRE, RFC 7637 [50]) standard from the IETF. The decoupling of virtual networks and addressing space from the underlying network infrastructure to provide multi-tenancy is the main objective of NVGRE. The standard is mainly focused on the data plane aspects of the solution and how it provides the desired isolation between multiple virtual networks. The VLAN and RSTP limitations and huge utilisation of the network capacity in data centers are the main issues to overcome. The design goals are a location independent addressing scheme, the scalability of the number of virtual networks, the broadcast isolation and the preservation of the Layer 2 semantics on the tenant. The GRE header is directly encapsulated into the IP outer header and a 24-bit Virtual Subnet Identifier (VSID) is included to identify the virtual instance. The outer IP header protocol is 0x2F used for GRE encapsulation, and the GRE header protocol type is 0x6558 to specify that a transparent Ethernet bridging service is provided. The broadcast and multicast on the virtual networks can be either mapped into multicast traffic (one per virtual instance or the same for all the instances) or into N-way unicast (encapsulated into N unicast packets) to avoid the requirement of IP multicast support on the underlying network. The IP fragmentation in NVGRE is avoided through Path MTU discovery mechanism. IPsec or similar IP-based mechanism are proposed to mitigate spoofing attacks.

The Stateless Transport Tunneling (STT) protocol [51] was proposed as an internet draft to implement a NV solution. Although it was not finally considered as a standard, STT was a pragmatic approach which faces the improvement and acceleration of the encapsulation process relying on hardware offloading of TCP packets. This proposal has the historical value of being the core development of Nicira to build data center overlays with SDN technology. This draft standard was originally promoted by Nicira and continued at VMware by the same authors (after the 1 billion acquisition). The main benefits from STT are: overlapping addresses between multiple tenants, decoupling of virtual topology from the physical network by means of an overlay, VM mobility support, scalability of the number of virtual instances, decoupling of tenant technology from the underlying technology, and isolation of the physical devices from the addressing used by the virtual networks. The novelty of this approach is the usage of a TCP-like outer header to encapsulate the STT header. However, the encapsulation process is stateless although it is based on a TCP header, i.e., there is no TCP connexion state on the tunnels. The reason for this is to leverage the offloading capabilities of existing NICs to accelerate the TCP segmentation processing. The segmentation is performed as for any other TCP packet, so IP fragmentation would not be an additional issue. This draft considers the encapsulation/de-encapsulation process to be performed at the virtual switch in the hypervisor, physical switch or some other appliance. The STT header includes a 64-bit Context identifier as a generalised way of identify the virtual instance on the tunnel endpoint, which allows STT to scale to address the large data center requirements. If the underlying network supports IP multicast, this capacity can be exploited by STT to encapsulate broadcast and multicast traffic from the tenants. Although not specific security aspects are covered by the draft, IPsec is mentioned as a possible mechanism to add security to STT packets.

Rather than yet another encapsulation header, Generic Network Virtualisation Encapsulation (Geneve [52]) was designed as a framework for tunneling specifically designed to meet the requirements of NV at scale for large data centers. Geneve supports NV and understands the tunnels as a backplane between the virtual switches in hypervisors, physical switches and middleboxes, where the underlying network is an arbitrary IP network. As design requirements, the data plane is generic and extensible to support current and future control protocols, the tunnel must be efficiently implemented in hardware and software, and it must provide high performance over existing IP fabrics. As a result, Geneve is a pure tunnel format specification capable to be adapted to any control protocol and the options are implemented in Type-Length-Value (TLV) format. The Geneve header defines a Virtual Network Identifier (VNI) of 24-bits and is then encapsulated into a UDP outer header that uses a destination port (value 6081) assigned by the IANA to Geneve. Geneve tunnels can be unicast or multicast, and opposite to VXLAN, unicast point-to-point connexions can be used to transmit broadcast traffic that is previously replicated into multiple packets. This option removes the limitation of VXLAN to support IP multicast in the underlying network. Moreover, NIC offloads are also considered to improve the performance of the encapsulation process. Path MTU discovery os proposed to avoid IP fragmentation of Geneve packets. The security issues are not specifically considered by the proposal.

Due to the scale of large data centers and the expected load in this scenarios, the performance of the encapsulation process (beyond the inherent overhead added in the data plane) is relevant for overlay solutions. In this context, the hardware offloading support in the NIC is fundamental, such as TCP Segmentation Offload (TSO) [TSO], Generic Receive Offload (GRO) [GRO], Generic Segmentation Offload (GSO) [GSO] or Large Receive Offload (LSO) [LSO]. [53] analyses this offloading effect on the aforementioned proposals and propose Segment-oriented Connexion-less protocol (SCLP) as an efficient encapsulation process that improves the performance of VXLAN when using this protocol instead of UDP. The offloading capabilities for UDP-based and TCP-based encapsulation while sending and on reception are different, and thus, the overall performance of each proposal varies. For instance, STT makes use of TSO, while SCLP exploits GRO capabilities.

Based on the previously introduced taxonomy, the Data Center overlays are generically categorized as shown in Table 2.6.

The Data Center overlays are generically categorized as an overlay type of NV, since the virtual network instances and multi-tenancy are created by providing isolated overlay networks per tenant. This type of solutions have been widely deployed in data center networks as an optimal approach to achieve the multi-tenancy at scale in large data centers based on IP networks.

The specific VI namespace used to identify the virtual network instance depends on the technology selected to provide the overlay. As previously mentioned, there are different possible overlay technologies, and each of them defines an analogous

NV Taxonomy	**Data Center overlay**
NV Type	overlay
Main target scenario	data center networks
VI namespace	dependent on the tunneling mechanism
DP Adaptation Mechanism	add/remove encapsulation header
VIAP classification	semi-transparent in the hypervisor
Network topology	not exposed
VI creation point	edge device (hypervisor)
Isolation level	QoS support depending on the underlying technology, addressing space is isolated (between infrastructure provider and tenants)

Table 2.6: Data Center overlay categorization based on NV Taxonomy.

identifier to this aim: VXLAN Network Identifier (VNI) in VXLAN, Virtual Subnet Identifier (VSID) in NVGRE, Context ID in STT and Virtual Network Identifier (VNI) in Geneve. This identifier is used to associate the traffic to the corresponding virtual network instance at the edge element (e.g. hypervisor).

Therefore, as an overlay technology, the adaptation mechanism used at data plane is the addition/removal of the encapsulation header (different header depending on the technology) to the user's traffic on the edge elements. This procedure could be done at the virtual switch on the hypervisor, a physical switch or an appliance. Some additional functionality is needed to learn the user's MAC address (in Layer 2 services), create the multicast groups, discover the MTU of end-to-end paths and some others depending on the technology.

In relation to the classification of the VIAP, the data center overlays are semi-transparent, since the data plane adaptation is performed in the hypervisor on the end node, but in a separate and isolated context. This process is transparent for the user and the packets are adapted before accessing the underlying network. The mapping and encapsulation mechanism are implemented in the virtual switch that is part of the global virtualisation solution for computing, storage and networking.

With regard to the network topology, the details of the underlying network are not exposed to the overlay, since the entire network is abstracted and exposed as a whole to the tenants. However, the overlay is able to define its own topology which is constructed by using tunnels through the underlying network.

Regarding the VI creation point, as any other overlay, the virtual instances are created on the edge elements (i.e., the virtual switch in the hypervisor). The underlying network is not responsible to create the instances and is devoted to forward the encapsulated packets from one edge element to another.

Concerning the isolation level, the different Data Center overlay solutions could provide a distinct QoS support depending on the actual solution and the support provided by the underlaying infrastructure. Moreover, the addressing space from the tenants are completely isolated by means of the encapsulation, as well as the underlying addressing scheme used by the data center infrastructure provider. Although the computing resources delegated to the tenant (i.e., VMs) are completely isolated on the data center, the computing resources associated to the networking functionality (i.e., mapping and encapsulation processes on the edge element) are not typically isolated on per tenant basis.

2.3.2.3 *VMware NSX: the Network Virtualisation Platform (NVP)*

The VMware NSX [70] is the unified platform for network and security virtualisation that joints the best of VMware vCloud Network and Security (vCNS) and Nicira Network Virtualisation Platform (NVP). As a result, NSX provides a set of logical networking elements, such as switches, routers, firewalls, load balancers, and so on, that can be combined in any topology with multi-tenancy support and isolation between virtual instances provided to the tenants. Moreover, all this support can be exposed through programmable APIs that can adapted to any cloud management system. Just with this definition, it seems that this approach is more related to a vNode type of NV, which in fact it is partially true. However, the complete solution can also be deployed on top of any physical IP network fabric, and in order to achieve this support, NSX makes use of an overlay approach to deal with this underlying network. Precisely, because of this latter support, NSX could be also considered as an overlay type of NV. The two elements are complementary to build the complete NV platform for data carter scenarios.

In order to better understand the overall picture, two key components of the NSX architecture must be highlighted: the hypervisor (i.e., VMware vSwitch) and the controller cluster, which orchestrates the resources to fulfil the tenant's requests. It must be mentioned that VMware focuses on building the data center solutions based on edge virtualisation software, the hypervisor. First of all, the controller cluster exposes a northbound interface to other management platforms (e.g. vCloud) to receive the logical network descriptions coming from the tenants. This controller processes the requests and proactively programs the vSwitches on the hypervisors. On the one hand, the vNode related logic is implemented on the vSwitches distributed along the data center (the best location is selected by the controller cluster). On the other hand, the distributed nature of the vSwitches requires from an isolated inter-vSwitch communication, which is solved by means of the data center overlay approach. This implies that the controller cluster must dynamically program the overlays through the appropriate encapsulation tunnels between the hypervisors. STT and VXLAN technology are currently supported by NSX, which allows to decouple the addressing space used by the VMs from the underlying network fabric (i.e., an IP network).

For completeness, NSX also provides Gateway services to securely control the traffic coming into or going out the data center. The NSX Gateway nodes are also programmable and provides services such as MPLS, IP routing, load balancing, firewall, NAT, and VPN.

Based on the previously introduced taxonomy, the VMware NSX is categorized as shown in Table 2.7.

The VMware NSX provides a complete solution for data center NV with a balanced approach between a vNode type and overlay type of NV. The vNode components deployed on the edge elements (i.e., the hypervisor) are interconnected by using an

NV Taxonomy	**VMware NSX**
NV Type	vNode | overlay
Main target scenario	data center networks
VI namespace	flowspace for vNode | VNI or Context ID for overlay
DP Adaptation Mechanism	reservation of flowspace for vNode | add/remove encapsulation header for overlay
VIAP classification	semi-transparent in the hypervisor
Network topology	virtual topology for vNode | not exposed for overlay
VI creation point	edge device (hypervisor)
Isolation level	QoS support depending on the underlying technology, control logic (and computing resources) from vNodes isolated, virtual topology decoupled for vNodes, addressing space is isolated (between infrastructure provider and tenants)

Table 2.7: VMware NSX categorization based on NV Taxonomy.

overlay on top of an IP network fabric. This composed solution covers the needs from data center networks with a software based approach build on the edge servers.

From the vNode perspective, the VI namespace rely on the flowspaces (including the internal virtual ports) to identify the traffic associated to each vNode, whereas from the overlay perspective, the VI namespace is the VNI (in VXLAN) or the Context ID (in STT) depending on the encapsulation technology used.

The data plane adaptation mechanism is the reservation of the appropriate flowspace in the case of vNodes and the addition/removal of the encapsulation header in the case of overlays.

Concerning the VIAP classification, the hypervisor performs the needed functionality in a semi-transparent way on the edge servers without any modification on the user side.

In relation to the network topology, the vNode is capable of defining an arbitrary virtual topology to be built on the hypervisors, whereas the overlay does not expose any topology from the underlying network.

With regard to the creation point of the VI, the virtual instance is created on the edge device in the hypervisor, both the vNode and the overlay.

Regarding the isolation level, the overlay can make use of the QoS support provided by the underlying network. Moreover, the control logic of the vNodes is isolated and isolated computing resources could be also assigned to each of these vNodes. The virtual topology defined between the vNodes is decoupled from the underlying resources and implemented in the hypervisor. This virtual topology could be physically distributed in several hypervisors, however, this physical distribution could have no relation to the virtual one. Finally, the addressing space used by the tenants is isolated from the underlying infrastructure, as well as from other tenants.

2.3.2.4 *NetLord*

NetLord [71] presents a novel architecture for multi-tenancy support at scale in data centers. It allows to isolate the addressing space of tenants by encapsulating the user's traffic on the edge elements (the hypervisor). One of the main difference with the previously introduced data center overlays is that NetLord relies on a Layer 2 fabric (instead of Layer 3). It claims that NetLord can be deployed over inexpensive commodity Ethernet switches.

The NetLord Agent (NLA) is a piece of software included in the hypervisor that is responsible for encapsulating and de-encapsulating the Layer 2 frames sent by the user to achieve the isolation and increase the scalability of the solution. An additional Layer 3 and Layer 2 header is attached by the NLA to the traffic coming from the VMs, and by means of this encapsulation, the Layer 2 addresses from the source and destination VMs are not visible for the underlying Layer 2 network.

The encapsulation details are relevant to understand how NetLord performs the virtualisation of the network and the requirements imposed due to this mechanism. First of all, the virtual interface of the tenant's VM is identified by three parameters: the Tenant ID, the MAC address space ID and the MAC address. This means that each tenant is able to define multiple addressing schemes, which is the context need to uniquely identify the MAC address of a VM. When a frame is sent from one VM which belongs to a tenant, this frame is encapsulated by the NLA in the following way. On the one hand, the additional Layer 2 header uses the MAC address of the ingress switch (that belongs to the underlying network and to which the server where the VM is running) and the MAC address of the egress switch (to which the server with the destination VM is connected) as source and destination MAC addresses respectively. On the other hand, the additional Layer 3 header uses the MAC address space ID as the source IP, and encode the physical output port from the egress switch (toward the server where the destination VM is running) and the Tenant ID as the destination IP. At the egress switch a Layer 3 lookup takes place, and the destination IP is decoded to rewrite the Layer 2 header as follows: the MAC address of the destination NLA is used as the destination MAC address, and the MAC address of the edge switch is used as the source MAC address. On reception, the NLA de-encapsulates the frame and extracts the Tenant ID, MAC address space ID and destination MAC address in order to forward the user frame to the corresponding destination VM.

In [71] the authors also propose to explore the benefits of adding SPAIN [80] support in the underlying Layer 2 fabric to achieve multipath forwarding based on VLAN tagging. Although this support is orthogonal to the NetLord proposal, it increases the overall usage of the Layer 2 fabric to reduce the possible congestion of some links. On the other hand, the drawback is that this support is very granular on a per-flow basis and not scalable. Moreover, the NL-ARP is included to combine the IP routing and ARP functionalities to maintain an NL-ARP table that maps the user MAC address to the egress switch and port number. Then, the NLA explores this table to proxy the ARP requests from VMs and avoid broadcast traffic from users on the Layer 2 fabric. With regard to the IP fragmentation that can be an issue when adding an encapsulation header (L2+L3 in this case), NetLord proposes to control the MTU of the virtual interfaces though the NLA and use jumbo frames to avoid this fragmentation. Finally, the proposal claims to be supported by commodity and inexpensive Ethernet switches on the Layer 2 fabric, however, the support for the IP forwarding (Layer 3 lookup) on the egress switch is not always a feature present on all switches.

Based on the previously introduced taxonomy, the NetLord approach is categorized as shown in Table 2.8.

NV Taxonomy	**NetLord**
NV Type	overlay
Main target scenario	data center networks
VI namespace	Tenant ID on L3 destination
DP Adaptation Mechanism	add/remove the NetLoad encapsulation header
VIAP classification	semi-transparent by the NLA in the hypervisor
Network topology	not exposed
VI creation point	NLA in the hypervisor (edge-based)
Isolation level	addressing space is isolated per tenant and per MAC address space instance, multipath support based on SPAIN (complementary)

Table 2.8: NetLord categorization based on NV Taxonomy.

The NV performed by NetLord is an overlay type, since the virtual instances isolated per tenant are built on the edge elements (i.e., the NLA) by using an underlying Layer 2 fabric with an isolated addressing scheme. This solution focuses on data center networks to provide multi-tenancy at scale that improves the utilisation of the underlying resources.

The VI namespace is the Tenant ID, which in combination with the MAC addressing space ID, allows to identify the target virtual instance associated to the tenant.

The addition/removal of the NetLord header is used as the adaptation mechanism at data plane level. The NLA is responsible to perform the encapsulation/de-encapsulation of user's frames in the hypervisor, which means that this process is classified as a semi-transparent VIAP. Moreover, the topology of the Layer 2 fabric is not exposed to the tenants, which is used as a forwarding plane that makes use of the outer header to perform such a process.

In relation to the VI creation point, the NLA software component deployed on all the hypervisors is responsible for creating the virtual instance on the edge elements.

With regard to the isolation level, the approach mainly focuses on isolating the tenant's addressing space, not only from other tenants and the underlying fabric, but also from multiple addressing schemes that can be defined by the tenant (via the MAC address space ID). Moreover, the cooperation with SPAIN allows to generate multiple isolated paths between the same source and destination hypervisors (relying on VLAN tags). However, there is no QoS support considered in the proposal and the computing resources are not isolated on per tenant basis (e.g. the NLA is a shared component).

2.3.2.5 *Diverter*

Diverter [72] is another proposal for providing an efficient multi-tenant support in data center networks at scale. It is a software-based approach to be deployed on the hypervisor that focuses on partitioning the IP address space to achieve the multi-tenancy. The means, agreements and user frame processing are quite different from the previous approaches.

The virtualisation is performed at Layer 3, based on the IP addresses, and provides isolated sets of subnets to the tenants. This means that the addressing space is not completely isolated, but an agreement on how to distribute the IP addressing space is assumed to enable the tenant isolation. Concretely, the IP addresses delegated to the VMs maintain the following format: 10.f.s.h, where f is used to identify the tenant (called farms in the proposal, the virtual network in the end), s is used to define the subnet, and h is used to identity the host (VM). This means that each tenant would have a 10.f.X.X/16 subnet assigned.

The solution is implemented by a software module, called VNET (a kernel module), which resides in the hypervisor and intercepts the frames sent by the VMs and process them appropriately. The basic processing of these frames is the following. First the an anti-spoofing filter ensures that the VMs use the proper IP addresses and prefixes (with the correct tenant identifier) that enforce separation of tenants. Then, the destination IP address is examined to decide which is the associated destination hypervisor. A mapping table resolves the IP addresses of the VM to the MAC address of the associated hypervisor where the VM resides. This mapping is performed and maintained by the VNET ARP engine, and if no appropriate mapping exists in the table, the user's frame is queued until this engine resolves the ARP. If the user's frame needs to be sent to a remote hypervisor, the source and destination MAC addresses of the user's frame are rewritten to the MAC address of source hypervisor and the MAC address of the remote hypervisor respectively. As a result, the underlying Layer 2 fabric does not see any MAC address from the end VMs, but from the edge hypervisors. Moreover, the frames coming from the VMs does not increase their size, since the action performed to isolate the Layer 2 addressing scheme is a rewrite, rather than an addition of a new outer header. From the possible IP fragmentation in the underlying network, this adaptation process is transparent and it does not have an impact. From the security perspective, the VNET performs an anti-spoofing check before sending the frames from the VM.

Based on the previously introduced taxonomy, the Diverter approach is categorized as shown in Table 2.9.

The Diverter approach is an overlay type of NV, since the underlying Layer 2 fabric is abstracted and a software (i.e., VNET) deployed on the edge hypervisors is responsible for building the virtual network instance delegated to the tenant. This approach focuses on the requirements imposed by data center networks and provides multi-tenant support at scale.

However, the Diverter proposal is quite different from all the previous solutions, since the VI namespace used is an IP prefix. Each tenant is assigned with a unique IP prefix (i.e., 10.f.X.X/16, where the f value identifies the tenant) that cannot be used by the rest of the tenants.

This adaptation mechanism at data plane actually implies a kind of reservation process by which each IP prefix is reserved for each tenant. In fact, this process does not scale too much, since only 8-bits are used to identify the tenant (i.e., the customer virtual network or farm, as called by the authors). Moreover, in conjunction with the reservation of the IP prefix, a rewriting process of the MAC address of the user's frames (i.e., the MAC addresses assigned to the VMs) is performed to adapt these frames to the underlying Layer 2 fabric.

With regard to the classification of the VIAP, it is considered as semi-transparent because the adaptation process is performed

NV Taxonomy	**Diverter**
NV Type	overlay
Main target scenario	data center networks
VI namespace	IP prefix
DP Adaptation Mechanism	reservation of IP prefix and rewriting of MAC addresses
VIAP classification	semi-transparent by VNET in the hypervisor
Network topology	not exposed
VI creation point	VNET in the hypervisor (edge-based)
Isolation level	Layer 3 address space not completely isolated (IP prefix reservation), Layer 2 address space isolated (Layer 2 fabric isolated from tenant's VM)

Table 2.9: Diverter categorization based on NV Taxonomy.

by the VNET in the hypervisor, neither at the user's VM nor at the Layer 2 fabric level.

Regarding the network topology, as any other overlay solution, the underlying topology is not exposed to the tenant and it is just used to forward the frames from the source to the destination hypervisor.

Concerning the creation point of the VI, it is performed by the VNET in the hypervisor on the edge element.

In relation to the isolation level, the Diverter approach isolates the Layer 2 address space used by the tenant's VMs from the addressing used in the underlying Layer 2 fabric. However, the Layer 3 address space is not completely isolated and the isolation is performed by a common agreement on the usage of IP prefixes. Moreover, the solution does not provide any QoS support mechanism and the computing resources devoted to the networking processing are not isolated.

2.3.2.6 *PortLand*

Just before performing any analysis related to the PortLand approach [19], it must be highlighted that this proposal does not support any NV at all. The reason for presenting this study is because of the usage of MAC address prefixes as a key design element of one of the contributions of this thesis, and the reason to include it here is because it focuses on data center networks to improve the scalability of the Layer 2 fabric.

The PortLand approach presents a fault tolerant and scalable Layer 2 routing, forwarding, and addressing for data center networks. It is intended to be plug and play, and no configuration is needed in the fabric. One fundamental requirement is the seamless migration of VMs without breaking the ongoing connexions (i.e., TCP session and application-level state), which implies that the IP address of the VM must remain the same. On the one hand, a Layer 3 fabric scale by means of aggregation (e.g. IP prefix assigned to Top of the Rack switches), but the IP addresses are modified when migrating the VM from one ToR to another. In this scenario, a Layer 2 fabric is an appropriate alternative, however, it has some scalability issues and not scale well due to the broadcast support and flat addressing scheme. VLAN segments could limit the broadcast traffic, but it does not scale properly for large data centers (e.g. 4094 limit) and manual configuration is needed. Morover, the fabric must avoid forwarding loops and, in this proposal, it is related to fat-tree topology data centers (which could be generalised to traditional multi-rooted tree topology).

PortLand proposes a centralised fabric manager responsible for maintaining the state of the network configuration (e.g. the actual topology), the ARP resolution, and fault tolerance and multicast support. The efficient routing and forwarding, as well as the support for seamless VM migration, leverage on the positional Pseudo MAC addresses (PMAC). The solution assigns one PMAC to each VM that encodes its location in the topology, while the VM does not need to change their MAC address, called the Actual MAC address (AMAC). Additionally, the fabric manager is responsible for AMAC (and IP) to PMAC mapping, and it also performs a proxy ARP to resolve the VM's IP to the associated PMAC. This means that the VMs resolve the PMAC of the destination IP through ARP, which is then proxied to the fabric manager. This process implies that the end VMs are not completely isolated from the fabric, since they learn the PMACs instead of the AMACs of the remote VMs.

The PMAC is coded as follows: pod.position.port.vmid, where the pod number (16-bits) is unique in the fat-tree topology, the position (8-bits) is the number assigned within the pod, the port (8-bits) is the number related to the physical host, and the vmid (16-bits) is the VM identifier. The PMAC is created incrementally with a timeout and then it could be reassigned.

In relation to the fabric, the forwarding is completely performed based on the PMACs, which allows to reduce the forwarding tables through aggregation (only the pod and position number is learned) since the PMAC encodes the location through a prefix. The switches are populated with forwarding entries with longest prefix match against the destination PMAC. As a result, switches with k ports only need O(k) state to be stored, although if some load balancing is needed on per-flow basis it requires some additional state. PortLand allows the separation of the host location from the host identifier in a transparent way to the end VMs without any overhead included in the data plane, since no additional header is added. Moreover, on the one hand, the ingress switch detects new source MAC addresses and notifies the fabric manager to create an associated PMAC and map it to the corresponding AMAC and IP, which is then used to reply the ARP requests. On the other hand, the fabric manager (acting as a controller) inserts a flow entry on this switch to rewrite the destination PMAC to the associated AMAC value. When this edge switch performs the destination MAC rewrite action, it is in fact acting as the egress switch. Consequently, a VM migration implies to update the PMAC value, and in order to achieve a seamless migration, a technique based on sending gratuitous ARPs is proposed to distribute the PMAC changes to the end VMs.

Based on the previously introduced taxonomy, the PortLand approach is categorized as shown in Table 2.10 (although this proposal does not support NV).

As mentioned above, the PortLand approach is not a solution for NV and focuses on improving the Layer 2 forwarding on the fabric at scale. Therefore, the NV type, VI namespace, and VI creation point are not considered in this analysis. The target scenario is the data center networks, and that is the reason for including this study here.

In relation to the adaptation mechanism used at data plane, the egress switches rewrite the destination MAC addresses with the corresponding AMAC, while the ingress switch does not need to perform the associated MAC rewrite because the end VMs learn the PMACs.

NV Taxonomy	**PortLand**
NV Type	none
Main target scenario	data center networks
VI namespace	none
DP Adaptation Mechanism	rewrite destination MAC address
VIAP classification	transparent on the edge switches
Network topology	real topology - Layer 2 forwarding
VI creation point	none
Isolation level	NV not supported, Layer 2 addressing not completely isolated (PMAC learned by VMs)

Table 2.10: PortLand categorization based on NV Taxonomy.

With regard to the access point classification, the traffic is processed transparently on the edge switches and the users are not aware of it. Regarding the network topology, as a proposal for improving the Layer 2 forwarding, PortLand deals with the actual topology of the fabric.

Concerning the isolation level, the main outcome is that there is no isolation since this is not a proposal for virtualising the network. However, in relation to the L2 address space, it is not completely isolated between the users and the fabric, because the PMACs are learned by the VMs.

The most relevant conclusion of this study is that the usage of MAC prefixes in PortLand is not related to the virtualisation of the network, but related to the forwarding on the Layer 2 fabric.

2.3.3 SDNeV approaches

As a result of the separation between the forwarding and control planes proposed by SDN, a new way of implementing NV is possible: the SDN-enabled Virtualization (SDNeV). SDN proposes the definition of a new interface between the forwarding plane and the control plane, which crystallizes in a protocol that allows an external entity, the controller, to define the behaviour of the network devices. With an open/standard protocol, such as OpenFlow, the vendor lock-in is prevented and both planes (i.e., forwarding and control) can be provided by different vendors while assuring their compatibility.

When building the NV approach based on this separation, the interface between both planes must be isolated to enable the virtualisation of the forwarding plane and allow each tenant to add their own control logic. This interface exposes the programmability of the network devices to the tenants and the behaviour defined by one tenant must not interfere others. Moreover, this interface (i.e., the VITAP) also provides a means to access the data packets from the control plane (e.g. receive data packets in the controller or sent data packets by the controller), which makes it more complex to manage and properly isolate, since an enforcing mechanism must be set at the VITAP. Furthermore, as in previous NV approaches, the isolation at the VIDAP must be also enforced.

In this type of NV, the individual network devices can be virtualised through the control interface, in a similar way to the vNode approaches, but in this case, the control logic is not part of the network devices and must be added by the tenants. To achieve the virtualisation of the whole network, a coherent definition of the virtual instance across the individual devices is required. On the one hand, a vNode approach can also be implemented with SDN technology (e.g. Programmable Flow), however, the control logic is already provided and the tenant cannot modify it. On the other hand, an overlay can be also implemented by means of SDN (e.g. Nicira extensions to Open vSwitch), but it controls the setup of the tunnels and the mapping on the edge devices, and the control plane is not exposed to the tenants.

Although the study of SDNeV proposals is mainly related to OpenFlow-based solutions, this is due to the availability of NV proposals which relies on this technology rather than a correlation between SDNeV and OpenFlow. OpenFlow is just one of the possible protocols to implement SDN solutions, but it is not the only option (e.g. ForCES, I2RS and some others). However, OpenFlow has a strong relation with NV since its beginning, being one of the reasons for developing this protocol [81], [82], [12], [83]. Moreover, the NV has been a research topic for this technology and lots of proposals has been presented, but some other pre-SDN proposals (in relation to the time when they were presented and the time when the concept of SDN was introduced) for implementing NV have some similarities and has been included in this study.

2.3.3.1 *FlowVisor*

FlowVisor [57] has been one of the most widely used solutions to virtualise OpenFlow networks and a common way of running networking related experiments in parallel with production traffic. It allows multiple isolated logical networks to define different addressing schemes and forwarding mechanisms while sharing the same physical infrastructure based on OpenFlow-enables network devices. The forwarding plane is shared between the tenants and implement a different control logic defined by the virtual instance. Compared to computing virtualisation, FlowVisor is located between the underlying hardware and the software that defines its behaviour. Moreover, as any other operating system it uses an instruction set to control this hardware, the OpenFlow protocol, which exposes the programmability of the forwarding plane to the OpenFlow controller. In this case, FlowVisor allows multiple controllers (one per instance) to share the control interface, while ensuring the isolation between the virtual network instances.

The design goals of FlowVisor are the following. The virtualisation of the network must be transparent to the controller of the virtual instance, in order to enable the design of controllers for non-virtualised environments and to debug network protocols in realistic topologies. It must also isolate the virtual network instances both at data and control planes. Finally, the virtual network instance (also known as slice) definition must be extensible to allocate as many instances as possible.

FlowVisor can be categorized as a specialised OpenFlow controller acting as a transparent proxy between the OpenFlow-enabled devices and controllers from the virtual instances. This means that all the OpenFlow messages are sent through FlowVisor, which is responsible to enforce the isolation between the instances on the control plane. As a design requirement, the virtual instance's controller does not need any modification, and assumes that it is directly connected to the network device. The inspection, rewriting and policing of all the OpenFlow messages is the mechanism to enforce the isolation and transparency. FlowVisor ensures that all the messages sent from the controller to the switches only affect the traffic and resources assigned to the appropriate slice, whereas the messages from the switches are analysed to determine the corresponding controller to which each message should be forwarded. Due to its relation to the transparent processing of OpenFlow messages, FlowVisor could be stacked to further virtualise the resources associated to the tenant.

In FlowVisor a slice is defined as a set of flows that constitute a well-defined subspace of the complete geometric space of packet headers, from the physical port to TCP/UDP port numbers. In this context, a wildcard is a bit mask that can be used to define a region from this space. As a consequence, the slice can comprise a set of these regions, even non-contiguous, which is called the flowspace. Based on this flowspace, each packet can be associated to one or another slice depending on the packet header values. It is important to avoid that two flowspaces overlap in order to assure the isolation between the virtual instances.

From the isolation perspective, FlowVisor supports bandwidth, topology, computing, flowspace and control plane isolation between the virtual network instances. Regarding the bandwidth isolation, FlowVisor considers to use VLAN priority bits (PCP) or IP's ToS to assure a minimum QoS to each slice. Although OpenFlow does not expose properly the QoS support from the underlying resources, the QoS queues could be used to implement the isolation at this level. Concerning the topology, the controllers can rely on OpenFlow to discover the nodes and links associated to the virtual instance and FlowVisor assures that each controller only sees the resources corresponding to its view. Moreover, some well known protocols related to the topology discovery (e.g. link layer discovery protocol, LLDP) are properly adapted to only report the assigned virtual topology. With regard to the composing isolation in the network device, there is some OpenFlow related load that must be properly isolated to avoid the over-load of the resource: new flow setup message, request handling from the controller, forwarding of slow path packets, and internal state keeping. In relation to the flowspace isolation, the appropriate rewriting of messages is performed transparently to assure the isolation between the slices. Moreover, the number of flow entries per slice is controlled and each virtual instance cannot exceed a predefined limit. Finally, the OpenFlow control plane is isolated and virtualised in a proper way, which implies the rewrite of the transaction ID or the buffer identifier. Status related messages are also appropriately duplicated and sent to all the affected controllers.

In some forums, FlowVisor is considered as a slicing mechanism, rather than a virtualisation solution, because the address space and topology are not virtualised. However, the NV concept used in this study is more generic and the virtualisation of the addressing and the topology are just a characteristic that identifies a NV approach, more than a requirement. There are some target scenarios in which the virtualisation of these two aspects, i.e., addressing and topology, is fundamental to properly cover the needs (e.g. in data center networks).

Due to its design goals, FlowVisor as a NV mechanism is quite open (and flexible) in its definition. Therefore, FlowVisor can be seen as a framework to construct NV solutions. One reason for this consideration is the fact that the namespace used for defining the virtual instance is not fixed, there is not a flowspace selected for that aim. A concrete NV proposal must define the VI namespace to avoid possible collisions, and it should not be delegated to the tenants. As a consequence, there are a lot of (concrete) NV proposals that are based on FlowVisor.

As previously mentioned, FlowVisor has been used in a lot of deployments and research projects to build experimental facilities, such as the Stanford OpenFlow testbed, GENI, GEANT OpenFlow Facility, FP7 OFELIA, FIBRE and FELIX projects.

Based on the previously introduced taxonomy, FlowVisor is categorized as shown in Table 2.11.

FlowVisor is categorized as a SDNeV type of NV, since it relies on OpenFlow to virtualise the forwarding plane of network devices. As previously mentioned, it could be considered as a framework for building NV proposals due to its lack of definition of some basic virtualisation aspects. Its main target scenarios

NV Taxonomy	FlowVisor
NV Type	SDNeV
Main target scenario	campus networks and experimental facilities
VI namespace	flexible flowspace
DP Adaptation Mechanism	reservation of flowspace
VIAP classification	transparent based on the flowspace definition
Network topology	subset of physical topology
VI creation point	proxy at control plane
Isolation level	QoS through PCP (Priority Code Point) field of VLAN tags or IP ToS bits (priority queues), isolated topology, computing isolation on the OpenFlow-enabled devices, flowspace isolation enforcing, limit of number of flow entries, control plane isolation

Table 2.11: FlowVisor categorization based on NV Taxonomy.

are the campus networks and the experimental facilities, both scenarios in which FlowVisor has been widely deployed to experiment through novel networking proposals while enabling and isolating the production traffic.

The VI namespace is flexible and it has been generically defined as the flowspace, although it is not a concrete proposal. The specific flowspace used by the NV approach is relevant to understand how and where it could be deployed and if there is any chance to collide between the network instances.

The adaptation mechanism used at data plane is the reservation of the flowspace assigned to the slice. This means that, in principle, the data plane is not modified and the packet header determines the associated virtual instance. This procedure also implies that the traffic from the users must not interfere with each other, and FlowVisor must enforces this isolation.

As a consequence, the classification of the VIAP is transparent and based on the flowspace definition, which should not collide with other virtual instances. There are some deployments based on FlowVisor, that make use of the VLAN identifier as the flowspace to uniquely identify the virtual instance. In these cases, such as in OFELIA, the user must configure the data plane interface to tag the traffic before sending it to the facility. Moreover, it must be considered that the VLAN tags are also present in the control plane (i.e., packet_in and packet_out), which means that the tenant's controller is aware of this process, not being completely transparent (one of the design goals from FlowVisor). Therefore, in this type of solutions, the VIAP classification is non-transparent, since the user must tag the traffic. However, generically, FlowVisor relies on transparently on the flowspace to identify the target slice.

In relation to the topology, FlowVisor assigns a subset of the physical topology to each slice and it is bundled to the underlying hardware, although some functionality much be implemented to isolate the topology and hide some network devices not assigned to the virtual intense.

With regard to the VI creation point, the virtual instances are created in the proxy element where the FlowVisor logic resides. It is typically located on an external device, although this is not mandatory. When just a single box is performing the global virtualisation of the network it is considered as a single point of failure, but FlowVisor also allows to create a distributed setup with multiple proxies controlling part of the network, as it has been done within OFELIA with the concept of OpenFlow islands.

Regarding the isolation level, as previously detailed, FlowVisor considers bandwidth, topology, computing, flowspace, flow entry and control plane isolation.

2.3.3.2 *VeRTIGO*

VeRTIGO [84] (ViRtual TopologIes Generalisation in OpenFlow networks) is a proposal for NV which relies on SDN, concretely on OpenFlow. In fact, it is based on FlowVisor as a network slicing system, to which VeRTIGO adds the capability to expose virtual topologies to the tenants. It could be used to either expose a single abstract node virtualising the whole underlying network or a logically connected network with an arbitrary topology.

ADVisor [85] is a prior work from the same authors and an initial implementation of the same concept and written on a C version of FlowVisor. VeRTIGO is implemented in Java and extends that work by adding extra features to improve the virtual topology support. Both proposals basically add virtual links to FlowVisor.

Depending on the requirements and necessities of the tenant, VeRTIGO can instantiate either a virtual network with virtual network devices (i.e., forwarding plane) and virtual links or a single virtual network device (i.e., forwarding plane) that collapses the whole underlying network. On the former, the tenant has full control of the virtual network devices, while the infrastructure provider must deal with the virtual topology assigned to the tenant. On the latter, the tenant defines the routing policies, while the infrastructure provider manages the underlying resources accordingly and offers different service level agreements to each tenant, such as maximum latency or packet loss. In both cases the failures and network congestion must be addressed by the infrastructure provider.

In a similar way than FlowVisor, VeRTIGO sits between the underlying network devices and the controllers and interacts via the control interface based on OpenFlow. The two basic elements provided by VeRTIGO are the virtual links and the virtual ports, which are used to construct arbitrary topologies exposed to the tenants. The virtual links aggregate a set of physical links and OpenFlow-enabled devices, whereas the virtual ports are just a virtual identifier for a physical port. In order to virtualise the physical links, the same physical port can be mapped into multiple virtual ports, each one associated to a different virtual link. VeRTIGO uses a database to store the header sequence associated to the flows of each virtual link, opposite to a tag-based identification (e.g. VLAN or MPLS) as proposed in ADVisor.

A set of components have been added to FlowVisor to implement this abstraction: classifier, node virtualiser, port mapper, internal controller, storage, VT planner and UI and Control framework. The classifier identifies is an OpenFlow message must be forwarded to the tenant's controller depending if the incoming port is visible in the associated virtual topology. The node virtualised is responsible to multiplex the control channels between the underlying network devices and VeRTIGO, and then between VeRTIGO and the controller. In this process the datapath identifiers are remapped. The port mapper is responsible for abstracting and remapping the physical port identifiers into the virtual ones. The internal controller is responsible for those network devices that are hidden to the controller, which are never communicated to the tenant's controller. The storage module concentrates the definition and configuration of virtual nodes and network topologies. The VT planner is the component that associates the virtual network instances to the physical infrastructure and implements the path selection algorithm to appropriately map the virtual links to the physical devices and links. Internally, a monitoring module is responsible for collecting the flows statistics to feed the VT planner algorithms. Finally, the UI and Control framework is provided to simplify the configuration and definition of virtual topologies and associated requirements, such as the flowspace.

VeRTIGO has been mainly deployed in OpenFlow-based experimental facilities, such as FP7 OFELIA, to add the capacity to expose virtual topologies to the tenants in addition to the capacities provided by FlowVisor to enforce the isolation of flowspaces to virtualise the physical network devices.

Based on the previously introduced taxonomy, VeRTIGO is categorized as shown in Table 2.12.

VeRTIGO is a SDNeV type of NV, since the forwarding plane is virtualised and exposed to the tenants by means of the proper isolation of the OpenFlow interface. The main target scenario is the experimental facilities used to test and validate novel architectures and protocols to overcome the ossification of Internet. Moreover, the definition of new business models beyond the basic connectivity is an objective that motivates this proposal.

Due to its dependency on FlowVisor, several characteristics are inherited, such as the flexible VI namespace definition, which relies on the flowspace concept. There is no concrete proposal to avoid collisions in its definition, and it relies on the policy logic to check possible isolation issues. The main focus of VeRTIGO is to uniquely differentiate the traffic associated to each of the virtual links defined by the tenants. To this aim, the database storing the header sequence of each flow associated to each virtual link is proposed to transparently identify the virtual links. Previously, ADVisor makes use of explicit tagging (e.g. VLAN or MPLS) to perform this identification.

Additionally, the adaptation mechanism used at data plane is the reservation of the flowspace defined to identify each slice, which implies that the traffic does not need to be modified and

NV Taxonomy	VeRTIGO
NV Type	SDNeV
Main target scenario	experimental facilities
VI namespace	flexible flowspace (header sequence database for virtual links, previously VLAN/MPLS in ADVisor)
DP Adaptation Mechanism	reservation of flowspace
VIAP classification	transparent based on the flowspace definition
Network topology	virtual topology
VI creation point	proxy at control plane
Isolation level	FlowVisor isolation inherited

Table 2.12: VeRTIGO categorization based on NV Taxonomy.

the packet headers are analysed to identify the target flowspace and slice.

Regarding the VIAP classification, it is transparent and no action or modification is needed on the user's traffic. This implies that the flowspace is properly defined and the user's traffic only matches its flowspace, which does not collide with the flowspace from any other slice. The use of VLAN tags to identify each slice is also possible with similar consequences as in the case of FlowVisor.

Concerning the network topology, VeRTIGO provides a virtual topology to each tenant, which could be either a single network device (forwarding plane) or a fully connected network of virtual network devices and virtual links.

In relation to the creation point of the VI, a proxy element is used to build the virtual instances. In order to create arbitrary topologies it is likely to be located on an external device to have the full flexibility to either collapse or spread out the virtual elements.

With regard to the isolation level, as previously detailed, the main characteristics are inherited from FlowVisor, such as the bandwidth, topology, computing, flowspace, flow entry and control plane isolation.

2.3.3.3 *OpenVirteX*

OpenVirteX [86] is a NV platform that provides to the tenants a means to define their virtual topology and addressing scheme, while running their own Network Operating System (NOS) to control the virtual network devices (i.e., forwarding plane). Due to the decoupling of the virtual network instance from the underlying physical network, it allows to add resiliency to network devices and links, and support the ability to create network snapshots and the migration of the virtual network. Similarly to FlowVisor, OpenVirteX sits between the physical network devices and the control logic (adding very limited overhead), acting as a proxy in the control plane between the underlying network and multiple tenants (each with its own NOS). Opposite to FlowVisor that performs network slicing, OpenVirteX provides a fully virtualised network with an arbitrary topology and a full header space to each tenant.

The main aim of OpenVirteX is to provide an Infrastructure as a Service (IaaS) solution for SDN devices. As a consequence, the priorities and requirements are different from previous proposals, and must enforce strong isolation between virtual network instances with the ability to snapshot, migrate and define the slice topology. In this context, the virtual network and the compute resources could be instantiated at the same time. SDN provides the proper flexibility and dynamicity to address the tenants' demands while improving the utilisation of the network resources and reducing the management complexity.

OpenVirteX is implemented as a network hypervisor that allows to provide virtual network instances to their tenants. This network hypervisor provides virtualised addresses to isolate the traffic from each tenant, virtual topologies defined by the tenants, and exposes the network programmability of the virtual network devices to the tenant's NOS.

The virtual network requests coming from the tenants via API calls are processed and passed to the network embedder. By means of this API, the tenants specify the desired addressing scheme, topology and a link to the tenant's NOS. Then, the embedder maps the virtual resources to the physical ones using the available information collected from the physical infrastructure. Once the embedding process is completed, OpenVirteX instantiates the virtual network on the physical infrastructure. The embedding process is not the focus of OpenVirteX and a modular architecture allows to implement and test different embedding algorithms.

OpenVirteX decouples the virtual elements from the physical network devices and maintains a mapping between them, which is in fact the output from the embedder. As a requirement, each virtual element must be mapped to at least one physical network element. This mapping does not specify the mechanism to implement the virtualisation, and each element can implement (if coherent) in a different manner, for instance using MPLS tags or MAC rewriting. The virtual network element is just a pointer to the corresponding physical network element that implement that resource. Therefore, the virtual network elements can be disabled, enabled or modified at runtime.

The virtualisation of an arbitrary topology requires special support from OpenVirteX. As a design limitation, the physical network device cannot be partitioned into multiple network elements inside the same instance. Moreover, the LLDP messages needs to be properly processed by OpenVirteX to expose the appropriate virtual view to each tenant. This is achieved by implementing separate LLDP domains for the physical and each virtual topology. As a consequence, the LLDP packets in the physical network data plane are limited to those generated by OpenVirteX, while OpenVirteX maintains an isolated and virtual LLDP domain for each virtual instance.

Additionally, OpenVirteX provides the ability to define the addressing scheme used by each tenant, even allowing concurrent overlapping address spaces. A globally tenant ID is generated by OpenVirteX to uniquely identify each tenant, then, each host is mapped to a physical address that encodes the tenant ID to easily identify its owner. This physical address is a combination of MAC and IP headers, and then, the MAC and IP address are rewritten on ingress to isolate Layer 2 and Layer 3 rules pushed to the network devices. In this way, the semantics of the rules are not modified, while both Layer 2 and Layer 3 virtual networks are supported. The collision of the addressing schemes (and therefore, the collision of flow entries from different virtual instances) is avoided through the address rewriting. In order to make this enforcing mechanism transparent to the tenants and users, the traffic is rewritten on the edge network devices at data plane and the OpenFlow messages are modified accordingly to implement the same mapping between the tenant/user addresses and the physical addresses at control plane. This process imposes some overhead on both the data and control planes, but it is negligible if a software-based switch is used on the edge, such as in data center developments (i.e., virtual switches in server hypervisors).

Apart from enforcing the address rewriting in the control plane, OpenVirteX is also responsible for mapping other control functions to the physical network, and some simple control functions on the virtual network could imply multiple actions on the physical network. The tenant's controller must be unaware of the underlying topology and of sharing the physical resources with other tenants.

The main features provided by OpenVirteX rely on its loose coupling with the physical network elements. As a consequence, the topology can be customized and it does not need to be limited to a subset of the physical one. Moreover, the virtual links and nodes can be mapped to different physical elements to achieve some level of redundancy on the virtual instance. Furthermore, the virtual network can be dynamically reconfigured and mapped to alternative resources, since this mapping is reduced to the manipulation of key-value pairs and it does not store any networking state. As a consequence, the virtual instance is portable to other infrastructure providers, while enabling network persistence and troubleshooting.

Although the experimental facilities is not its main target scenario, OpenVirteX has been successfully deployed on GENI to test and validate the development.

Based on the previously introduced taxonomy, OpenVirteX is categorized as shown in Table 2.13.

OpenVirteX is categorized as a SDNeV type of NV. It provides a virtual network with an arbitrary topology defined by the tenant and exposes the control plane of the virtual network devices to the tenant's NOS. Therefore, the control logic is defined by the tenant, who is responsible to program the network behaviour. The main target scenarios identified for OpenVirteX, and also the motivation for this novel approach due to the requirements imposed, are the operator and data center networks.

NV Taxonomy	**OpenVirteX**
NV Type	SDNeV
Main target scenario	operator and data center networks
VI namespace	tenant ID encoded in the physical address
DP Adaptation Mechanism	Layer 2/Layer 3 address rewrite
VIAP classification	semi-transparent in the hypervisor
Network topology	virtual topology
VI creation point	proxy at control plane
Isolation level	resources mapping isolated per tenant, virtual topology isolated, addressing scheme isolation, and control functions isolated per tenant

Table 2.13: OpenVirteX categorization based on NV Taxonomy.

The VI namespace is the tenant ID which is encoded in the physical address to uniquely identify the associated virtual network instance.

In order to let the tenants define its own addressing schemes, the data plane must be adapted to include this tenant ID in the physical addresses. Therefore, the edge elements need to rewrite the Layer2 an Layer 3 addresses to properly isolate the traffic and identify the corresponding tenant. The same rewriting process must be also performed on the control plane to be transparent to the tenant's NOS.

With regard to the classification of the VIAP, it is semi-transparent since the virtual switch in the hypervisor must rewrite the addresses to enforce the isolation at data plane. A similar consideration could be assumed on the control plane, since the proxy must rewrite the addresses on the control traffic to isolate the virtual instances exposed to the tenant's NOS.

Regarding the topology, OpenVirteX is able to provide a virtual topology based on the request coming from the tenant. The only limitation is that a physical network element cannot be instantiated multiple times on the same instance, that imposes some restrictions to the virtual instance's topology depending on the actual physical infrastructure.

Concerning the VI creation point, the virtual instances are created in the proxy element based on the embedding process and the associated mapping between the virtual and the physical network devices. Moreover, the virtualisation of the addressing is defined by the proxy, which performs the address rewriting at control plane and populate the corresponding flow entries on the edge devices to rewrite the addresses at data plane.

In relation to the isolation level, as previously descibed, OpenVirteX isolates the mapping of virtual resources to physical elements per tenant. It also isolates the virtual topology and implements the appropriate functionality to isolate the LLDP domain per tenant. The address isolation is also provided and the same addressing scheme can be used by different tenants without collisions. Finally, the control functions are adapted and isolated per virtual instance depending on the corresponding mapping between the virtual and physical elements.

2.3.3.4 *FlowN*

FlowN [14] is a NV proposal that provides each tenant with a virtual network instance with virtualised address space, virtual topology and control logic defined by the tenant. In order to efficiently map between the physical and virtual resources, the latest database technology is used. This mapping is key for any NV approach and its performance is crucial in real deployments. In the context of data center networks, exposing the control interface to the tenants that define their own control logic and allowing the tenant to define their own topology and addressing scheme are basic requirements. SDN provides the appropriate interface to address this requirements and build the NV solution, while providing an API to popular the tenant's flow entries, reporting statistics and topology changes. However, supporting a huge number of tenants brings some scalability issues that must be considered, such as maintaining individual topologies running their own control logic and learning about physical changes that affect the virtual topology.

FlowN leverage on SDN technology to expose the programmability of the virtual network devices to the tenants. Each tenant can define its own topology, address space and control logic. In order to provide an efficient and scalable NV approach, FlowN relies on the latest database technology to improve the mapping between the physical and virtual resources. Moreover, a lightweight controller environment (similar to Linux containers, LXC) is provided to the tenants to improve the overall scalability and reduce the resource utilisation. As a consequence, each tenant is able to run its own controller application, which implements the control logic defined by the tenant. Some default applications (e.g. basic switching or routing logic) are offered to the tenants that does not want to innovate in this area.

The main performance issues to consider in a SDNeV type of NV are the overhead and latency introduced in the control plane, and the scalability issues derived from the mapping process. The key design decisions in FlowN to overcome this problems are the following. On the one hand, a shared controller platform (based on NOX) is provided, instead of separate controller instances per tenant. Each tenant runs its own application on the controller, which consists of a set of handlers related to the network events. Therefore, FlowN is a modified NOX controller that is able to run several applications with its own address space, virtual topology and event handlers. The container-based virtualisation is supported through the mapping at NOX API calls, rather than at the OpenFlow messages (e.g. FlowVisor). On the other hand, the database technology is explored to efficiently implement the mapping process. The mapping between the virtual and physical spaces can easily become the bottleneck, and FlowN leverage on the latest advances in database technology to overcome the scalability issues. Instead of using in-memory data structure to map the virtual topology to the physical infrastructure, FlowN relies on a database-style relational model to efficiently perform the mapping. Each virtual topology has a unique identifier and stores the mapping information in two tables, the node assignments to each virtual node and the path assignments to each virtual link.

Regarding the virtual network topology, the tenants are able to define an arbitrary topology for their virtual instance. The topology is described as a set of virtual nodes (VMs or OpenFlow-enabled network devices) with a set of interfaces interconnected by virtual links. Each resource can impose some constrains, such as the number or cores for VMs, the maximum number of flow entries for network devices, or the bandwidth for virtual links. Then, an embedding algorithm must be run to map the virtual description to the physical resources. In FlowN the mappings are internal to the controller platform and not exposed to the tenants. By decoupling the virtual topology and physical infrastructure the basic connectivity could be improved with some redundancy or failure recovery support.

In relation to the address space, the tenants are able to define their own addressing scheme and FlowN provides a virtual address space to each application, rather than dividing it between the tenants. This allows different tenants to use the same IP addresses in different virtual instances, while controlling the full header space. In order to achieve this, FlowN distinguishes the traffic from different tenants by encapsulating the incoming packets in the edge network devices with a transparent and protocol-agnostic header (e.g. VLAN tag), which is used just to identify the target tenant.

The bandwidth isolation per tenant is also considered although not implemented due to the limited support available in OpenFlow and lack of enforcing mechanisms. The embedding algorithms must be also updated to consider the bandwidth assigned to each virtual link.

Based on the previously introduced taxonomy, the FlowN approach is categorized as shown in Table 2.14.

The FlowN approach is a SDNeV type of NV, since it exposes the programmability of the network devices to the tenant through isolated API calls in the controller, instead of using the OpenFlow interface. The main target scenario is the data center networks and cloud computing.

The VI namespace used in the current implementation is the VLAN identifier, however, any other header field orthogonal to the control logic could be used in other deployments. The idea is to add a unique identifier to all the packets associated to each virtual network instance in order to easily determine the corresponding tenant.

Therefore, the data plane adaptation mechanism is the addition/removal of the VLAN tag (with the proper value) on the ingress/egress to the network. The VLAN push action is performed by the first network element, i.e., the edge device, that receives the packet. Then, the last element in the path to the destination must perform the VLAN pop action. A similar action to the VLAN push/pop must be performed by the API

NV Taxonomy	**FlowN**
NV Type	SDNeV
Main target scenario	data center networks
VI namespace	VLAN identifier
DP Adaptation Mechanism	add/remove VLAN tags in the edge device
VIAP classification	semi-transparent in the hypervisor
Network topology	virtual topology
VI creation point	external in the controller
Isolation level	bandwidth isolation considered, address space isolated per tenant, virtual topology isolated, control plane isolated in containers

Table 2.14: FlowN categorization based on NV Taxonomy.

calls to transparently expose the programmability of the virtual devices to the tenants.

In relation to the classification of the VIAP, the adaptation mechanism is semi-transparently performed in the hypervisor on the edge device (i.e., virtual switch).

With regard to the network topology, each tenant is able to define their own virtual topology, which is then mapped into the physical one by the embedding algorithm.

Regarding to the VI creation point, the virtual instance is created in the external controller and stored in the database. The VLAN identifier is the key to identify the target tenant's control application.

Concerning the isolation level, FlowN considers different aspects. Firstly, although not yet supported, the bandwidth isolation is mentioned as future support and the key aspects to update are identified, i.e., the embedding algorithm and OpenFlow bandwidth enforcing mechanisms. Moreover, the address space is isolated per tenant and different tenants can use the same addressing schemes without collisions. Furthermore, the virtual topologies are isolated per tenant and internally separated in the controller. Finally, the control plane is isolated and exposed to the tenants as API calls (rather than OpenFlow interface), while the execution environment is similar to a container running the control logic with its own virtual topology, address space and event handlers.

2.3.3.5 *Switchlets/Tempest*

In 1997, [87] presented an approach to virtualise an ATM switch by allowing different control architectures to be simultaneous operational in the same ATM network by using the same physical resources. Each ATM switch can be divided into switchlets, which encapsulate a subset of the physical resources of the ATM switch. Through the combination of a group of switchless from different network devices, a virtual ATM network can be constructed. This process of isolating resources from the network devices is the basis to build the virtualisation of the network resources and fully relies on the ATM technology. Moreover, the control and management mechanisms are external to the network device and rely on the Ariel interface, an open and generic switch control interface, to externally control the ATM switch behaviour. This interface allows the programability of the ATM network by exposing the control plane (similar to the SDN concept introduced some years later), as presented in Tempest [88] by the same authors. This means that different control architectures can be simultaneously running and using the same physical resources, each control plane accessing its own switchlets. The Switch Divider Controller enables the virtualisation and isolation of the Ariel interface.

The isolation of switchlets requires the specification of which resources can be partitioned as a subset of the physical resource, which include ports, VPI/VCI space, bandwidth, buffer space, and queueing and scheduling policies. The ports and VPI/VCI space can be partitioned with different granularity, and the partitioning of the VCI level (the more general option) is the one chosen to specify the switchlets. The rest of the resources, i.e., the bandwidth, buffer space, and queueing and scheduling policies are combined to define the capacity of the switch. In this proposal, the QoS details are hidden behind the five service categories defined by ATM. Therefore, a switchlet is defined as a set of ports, a range of VPIs, a range of VCIs per VPI, a set of service categories and the capacity per service category.

Although this technology is obsolete, this approach demonstrates that some basic concepts of SDNeV virtualisation are common to other proposals, and that the proposed taxonomy is applicable to past NV solutions.

Based on the previously introduced taxonomy, the Switchlets proposal is categorized as shown in Table 2.15.

The Switchlets proposal is a SDNeV type of NV, which allows to create isolated switchlets controlled by different tenants with their own control logic. Since the control logic is provided by the tenants, which means that the programmability is exposed to them, the type of NV is SDNeV (as in Tempest). However, if the infrastructure provider would be responsible for providing the control logic and the tenants are only able to select (and configure) which logic they want, the approach would be considered as a vNode type of NV (similar to comparing FlowVisor and Programmable Flow proposals). This Switchlet approach is mainly oriented to operator network to deploy different control architectures over the same physical infrastructure based on ATM technology. By properly coordinating the control and management of individual switchlets, a virtual network can be instantiate and operate.

The VI namespace relies on the association of ports and VPI/VCI values to univocally refers to a switchlet instance on the node.

Therefore, the adaptation mechanism used at the data plane is performed by the reservation of the proper VPI/VCI values, which means that the VPI/VCI spaces are divided and assigned to each virtual instance. Once this values are assigned, they cannot be reused by another virtual instance at the same time. Since this values has a local meaning (i.e., hop-by-hop), the VPI/VCI values must be divided per port.

In relation to the classification of the VIAP, the mechanism is transparent for the user, since is a similar process to a generic ATM network without virtualisation. The user needs to configure the ATM interface with the proper VPI/VCI values, which is then mapped into the associated switchlet.

With regard to the network topology, the topology of each virtual instance could be different, depending on the location of the set of corresponding switchlets that are part of the instance, but in any case, it is a subset of the physical topology. Moreover, there should be switchlets on all the physical ATM switches that interconnect the users. This means that if two switchlets from

NV Taxonomy	**Switchlets**
NV Type	SDNeV
Main target scenario	operator networks
VI namespace	ports and VPI/VCI space
DP Adaptation Mechanism	reservation of the VPI/VCI space
VIAP classification	transparent though the appropriate VPI/VCI
Network topology	subset of physical topology
VI creation point	external to the network device
Isolation level	QoS through ATM service categories, isolated control architecture (control and management mechanisms)

Table 2.15: Switchlets categorization based on NV Taxonomy.

two ATM switches are interconnected through another ATM switches and the virtual instance does not have a switchlet on the latter one, there is no connexion between the former switchlets.

Regarding the creation point of the virtual instance, it is performed external to the ATM switch, where the control plane resides. It is at the control plane where the Switch Divider Controller virtualises the ATM switch resources and ensures their isolation between the different virtual instances.

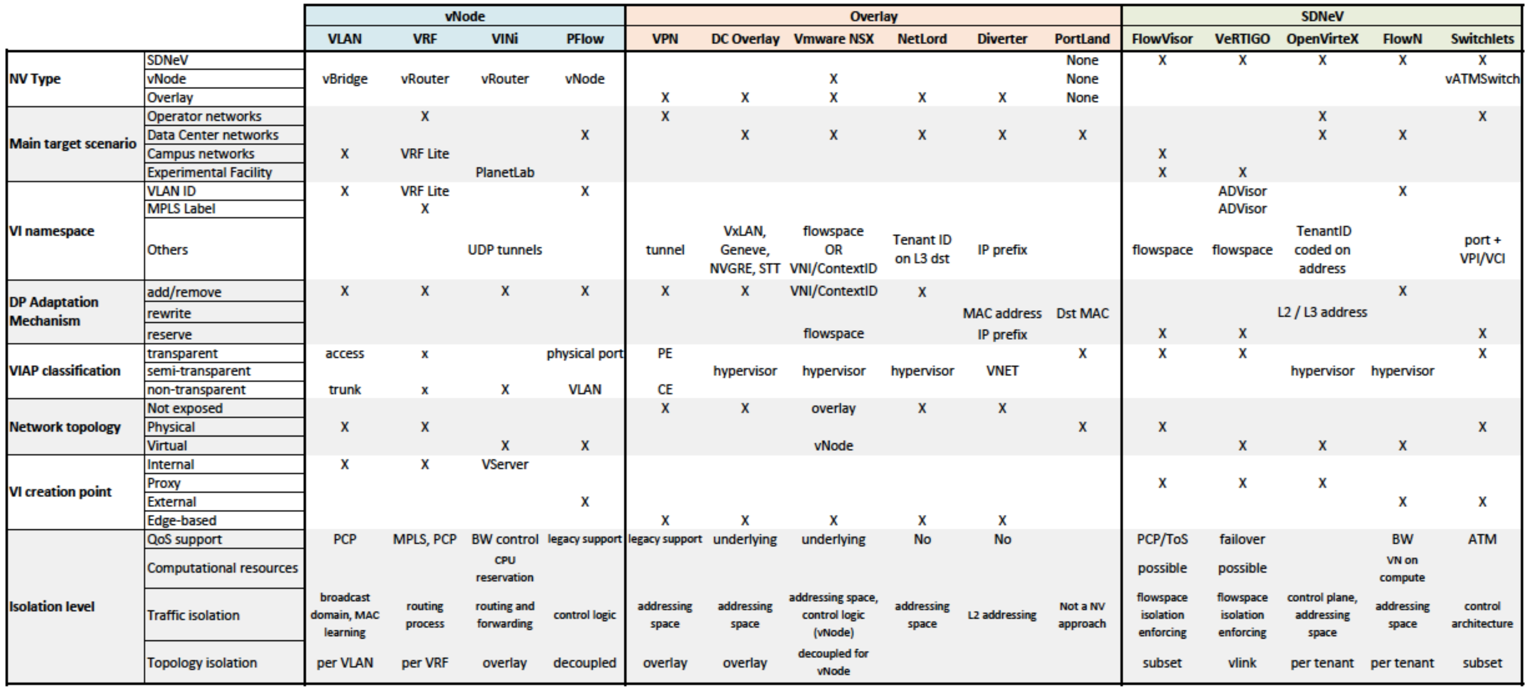
Concerning the isolation level, this approach allows to use the service categories defined by ATM to implement QoS at the virtual instance level. Additionally, the control architecture is isolated on the external entity, which allows different control and management mechanisms to be applied on the shared physical infrastructure. The virtualisation of the computing resources that run the control architecture is not considered, although it is possible.

2.3.4 Conclusions from NV survey

The following Table 2.16 summarises the main characteristics from the different approaches analysed in the Section 2.3 based on the proposed NV taxonomy described in Section 2.2. The main conclusions extracted from this table are detailed below.

- Regarding the use of legacy or SDN technology, both vNode and overlay NV types presents some proposals using either legacy or SDN technology. For instance, regarding vNode type, there are some legacy proposals, such as VLAN and VRF, and some SDN proposals, such as PFlow. With regard to overlay type, there are some legacy proposals, such as VPN and DC Overlays, and some SDN proposals, such as VMware NSX. However, the SDNeV proposals are all based on SDN technology (including the pre-SDN proposal Switchlets).

- There are two proposals that could be in two groups depending on the actual deployment, and one proposal that is not implementing a NV approach, which is Portland. One the one hand, VMware NSX could be either an overlay or a vNode type of approach, with different characteristics



		vNode				Overlay						SDNeV				
		VLAN	VRF	VINi	PFlow	VPN	DC Overlay	Vmware NSX	NetLord	Diverter	PortLand	FlowVisor	VeRTIGO	OpenVirteX	FlowN	Switchlets
NV Type	SDNeV										None	X	X	X	X	X
	vNode	vBridge	vRouter	vRouter	vNode			X			None					vATMSwitch
	Overlay					X	X	X	X	X	None					
Main target scenario	Operator networks		X			X								X		X
	Data Center networks				X		X	X	X	X	X			X	X	
	Campus networks	X	VRF Lite									X				
	Experimental Facility			PlanetLab								X	X			
VI namespace	VLAN ID	X	VRF Lite		X								ADVisor		X	
	MPLS Label		X										ADVisor			
	Others			UDP tunnels		tunnel	VxLAN, Geneve, NVGRE, STT	flowspace OR VNI/ContextID	Tenant ID on L3 dst	IP prefix		flowspace	flowspace	TenantID coded on address		port + VPI/VCI
DP Adaptation Mechanism	add/remove	X	X	X	X	X	X	VNI/ContextID	X						X	
	rewrite									MAC address	Dst MAC			L2 / L3 address		
	reserve							flowspace		IP prefix		X	X			X
VIAP classification	transparent	access	x		physical port	PE					X	X	X			X
	semi-transparent						hypervisor	hypervisor	hypervisor	VNET				hypervisor	hypervisor	
	non-transparent	trunk	x	X	VLAN	CE										
Network topology	Not exposed					X	X	overlay	X	X						
	Physical	X	X								X	X				X
	Virtual			X	X			vNode					X	X	X	
VI creation point	Internal	X	X	VServer												
	Proxy											X	X	X		
	External				X										X	X
	Edge-based					X	X	X	X	X						
Isolation level	QoS support	PCP	MPLS, PCP	BW control	legacy support	legacy support	underlying	underlying	No	No		PCP/ToS	failover		BW	ATM
	Computational resources			CPU reservation								possible	possible		VN on compute	
	Traffic isolation	broadcast domain, MAC learning	routing process	routing and forwarding	control logic	addressing space	addressing space	addressing space, control logic (vNode)	addressing space	L2 addressing	Not a NV approach	flowspace isolation enforcing	flowspace isolation enforcing	control plane, addressing space	addressing space	control architecture
	Topology isolation	per VLAN	per VRF	overlay	decoupled	overlay	overlay	decoupled for vNode				subset	vlink	per tenant	per tenant	subset

Table 2.16: Comparison of NV Survey based on the proposed NV taxonomy

depending on the case. On the other hand, the Switchlets approach could be either a SDNeV or a vNode type of NV depending on whether the programmability of the switchlets is exposed or not to the tenants.

- Both the vNode and SDNeV types of NV propose different approaches to cover all the target scenarios. However, the overlay type of approaches are mainly oriented to data center networks (except VPN that is oriented to operator networks).
- Both the overlay and SDNeV types of NV proposals could use the three types of data plane adaptation mechanism. It seems that the vNode type of approaches are bound to an adaptation mechanism based on add/remove. However, this is not an intrinsic characteristic, and it could be considered as a common mechanism, but the other two (rewrite and reserve) are also perfectly valid. As an example, the vNode approach proposed by VMware NSX makes use of the reservation of flowspaces as the mechanism to adapt the data plane.
- The overlay type of approaches are characterised by not exposing the network topology to the tenants, and defining the VI creation point on edge devices.
- The isolation level study is variable and the actual implementation depends on the specific deployment and the capabilities offered by the physical infrastructure.
- As a general observation, it must be mentioned that compared with SDN based approaches, the tenant interface from legacy devices is not always properly isolated, because it is typical that the same entity performs the infrastructure provider and the tenant roles. This is also the reason why the VIAP classification study mainly focuses on the data plane.

Because its relevance for the L2PNV proposal detailed in Section 3.1, the SDNeV approaches are further compared.

- The main difference between FlowVisor/OpenVirteX and FlowN is that the first two exposes the control plane to an external OpenFlow controller (as the Network Operating System to build the solution), whereas the latter allows to define the network behaviour developing a Control Application to be installed on the single OpenFlow controller.

- The main difference between FlowVisor and OpenVirteX is the way they enforce the virtual network instance isolation. FlowVisor reserves a flowspace for each virtual instance and inspects the OpenFlow control plane to ensure that each controller only access its assigned flowspace (described by the policy), both on matching and actions (OpenFlow version dependent, as well as the flowspace). However, OpenVirteX defines a simple action (available on all OpenFlow versions) to be performed on the edge of the network, an address rewrite (both MAC and IP). The idea is that this simple action ensure the isolation and the mapping between tenant's addresses (virtual) and infrastructure's addresses (physical) is a straightforward process (both at data and control planes).

2.4 EXPERIMENTAL FACILITIES SURVEY

Large-scale networking research has been nearly impossible in the past due to the unwillingness of many vendors to include non-standardised technologies in their equipment. Because of these business decisions, many innovative and remarkable ideas fall by the wayside due to the lack of infrastructure to test them properly. The Future Internet initiative has emerged to solve the current limitations of the Internet in its current incarnation.

During the past decade, the platforms available to researchers have increased considerably in number. Even before the appearance of OpenFlow and the Software Defined Networking concept, several platforms had been deployed, providing a large scale service to test networking experiments. One essential requirement for these type of facilities is to remain as generic as possible to be able to support new, clean slate approaches. Furthermore, it is mandatory to accept several experiments at the same time despite having the same resources. In this regard, the isolation between experiments is fundamental; therefore, some type of virtualisation of the physical infrastructure is needed.

2.4.1 Non-SDN facilities

This section focuses on some of the most relevant pre-SDN experimental facilities based on legacy technology and provides an overview of the work conducted up to this point. Due to the relevance of SDN for the contributions of this thesis, this section has been reduced to the minimum.

2.4.1.1 *The PlanetLab testbed*

PlanetLab [89] is an overlay-based testbed that appeared in mid-2002 and that supports the development of new network services. This research facility includes more than 600 nodes spread over the world, and it has been shared by more than 1100 researchers running experiments. It provides virtual servers (VServers) as lightweight slice of each node available in the testbed. Then, the computational resources are properly isolated.

PlanetLab provides virtualisation at the application layer by relying on the IP as a networking technology and manages the relationship between the owners of the nodes and the researchers while promoting solidarity in the use of limited resources. From the network's point of view, the slices are defined as a set of VMs running in computer nodes connected through the Internet, which implies that researching lower-layer network protocols is not supported. The distributed virtualisation model implemented in PlanetLab enforces resource and security isolation allowing experiments assigned to different slices to run concurrently. This isolation at network level relies on the VNET [65] module, which multiplexes the incoming and outgoing traffic. VNET provides each slice with the illusion of exclusive access to the physical network device.

The slice management approach used in this facility requires a trusted intermediary who, on the one hand, will guarantee node owners the proper management of their resources and, on the other hand, will provide access to the adequate set of nodes able to support the users' services. Through the trusted intermediary PLC (PlanetLab Central), the node owners establish what resources will be redistributed among all slices and which ones will be granted to a specific slice. One of the most useful services in the facility is its front end, which researchers can use to create slices from both a GUI and a programmatic interface. This flexibility greatly simplifies the creation of the slices.

As an example of experimentation over PlanetLab, the PL-VINI [64] demonstrates the viability of the VINI proposal (detailed in Section 2.3.1.3) on a real deployment. Moreover, the CoMon [90] is presented as a scalable monitoring system for PlanetLab. Furthermore, in [91] the integration of a wireless mesh network in PlanetLab is detailed, as well as the results from an innovative peer-to-peer traffic optimisation technique.

2.4.1.2 *The FEDERICA project*

In 2008, the FIRE programme in Europe launched the FP7 FEDERICA (Federated E-infrastructure Dedicated to European Researchers Innovating in Computing network Architectures) [92] project as a tool for researchers trying to validate Future Internet technologies in an existing large-scale facility. The main goal of this project [33] was to create a Europe-wide research infrastructure supported by virtualised computing resources and wired networks, offering virtual testbeds as a service to researchers.

This facility is based on the NETCONF (Network Configuration) protocol, which enables the configuration of network devices based on XML (Extensible Markup Language). The facility is composed of several programmable high-end routers and switches that can be logically instantiated in the core nodes, multiprotocol switches that extend virtual links to non-core nodes and several PCs that run software routers or act as end users. The connectivity between these entities is established over the GÉANT[5] pan-European backbone network. Because

5 GÉANT, The European Research and Education Backbone: http://www.geant.net

the researcher has the ability to manage both the network and the computer elements, it allows experimentation in the lower layers; however, the network virtualisation at Layer 2 is based on VLANs, which are not intended to work with several technologies without tunnels including Q-in-Q or MAC-in-MAC.

Regarding the experimentation in FEDERICA, the authors in [93] summarise the most relevant research projects that have used this facility to validate their proposals and obtain results in a realistic environment. As an example, in order to test the network performance and the isolation between the slices, the ISOLDA slice was created to measure bandwidth, latency, jitter, and packet loss parameters in two virtual connexions, each one using a different VLAN.

2.4.2 SDN enabled facilities

As mentioned in the introduction of this chapter, due to its relevance for the contributions presented in this thesis, the SDN technology has a prominent role in this analysis. Therefore, this section focuses on some of the most relevant SDN-enabled experimental facilities and provides an overview of the research activities related to these testbeds.

In all the facilities detailed below, the OpenFlow protocol is the selected technology to implement the SDN-enabled facility. Additionally, most of them make use of FlowVisor (detailed in Section 2.3.3.1) to virtualise and enforce the slice isolation between the OpenFlow resources. By means of FlowVisor, an OpenFlow interface is exposed to the researchers, thus, exposing the programmability of the network to the experiments. Each slice has its own view of the network nodes and the connectivity between them, as well as of its own bandwidth fraction on each link.

2.4.2.1 *The Stanford facility*

In 2008, the authors in [12] presented their plans to deploy a large OpenFlow network in the Computer Science and Electrical Engineering departments at Stanford University. This idea finally crystallized in the Stanford Deployment[6] deployed in the 3A wing of William Gates Building at Stanford University.

The most recent publicly available information states that it contains six 48-port 1 GE OpenFlow-enabled switches from HP, NEC and Toroki, 30 WiFi APs based on the ALIX PCEngine boxes, and 1 NEC WiMAX base-station, which provides connectivity to their users. At this facility, three different OpenFlow networks are used to run production, experimental and demonstration traffic with each of them identified by a different VLAN tag. Only the demonstration traffic is sliced in the FlowVisor to avoid interference with production and research traffic when the FlowVisor is under development. The facility enforces isolation in the four main slicing dimensions: bandwidth, topology, device CPU and forwarding tables.

The authors in [21] present the OpenFlow maturing process over a three-year period. The paper details the four-phased deployments and demonstrations conducted during that period, including the first-ever SDN enabled lab, the Stanford Deployment. These deployments and lessons learned were crucial to evolve the SDN technology and its ecosystem through the experimentation. They also influenced the successive versions of OpenFlow specification, the evolution of its architecture, and the improvement of performance of several SDN components.

2.4.2.2 *The GENI facility*

The GENI (Global Environment for Network Innovations) [5] facility is one of the most important virtual laboratories for large-scale networking experimentation available to researchers. This facility is supported by the National Science Foundation in the United States of America, and aims to allow clean-slate architectures and proposals to experiment under real conditions with real users. The research and implementation plan of GENI consists of continuous spirals, which lasts for 12 months.

GENI [2] is able to build multiple virtualised slices that contains two key components: the physical network elements and the global control and management framework to construct

6 The Standford deployment: http://archive.openflow.org/wp/stanford-deployment/

the slices. Therefore, the main activities involved in GENI testbeds are the deployment of the prototype (by means of federation) and observe, control and execute the experiments on it.

There are five competing control frameworks for GENI, called clusters: Cluster A (Trial Integration Environment based on DETER, TIED) focusing on federation, trust, and security; Cluster B (based on PlanetLab) running experiments with VMs over the Internet; Cluster C (ProtoGENI, based on Emulab) dealing with network control and management; Cluster D (Open Resource Control Architecture, ORCA) considering resource allocation and integration of sensor networks; Cluster E (Open-Access Research Testbed for Next-Generation Wireless Networks, ORBIT) dealing with mobile and wireless testbed networks.

Researchers can allocate configurable resources of the computer network, including wireless and sensor aggregates, to create an isolated slice controlled by an OpenFlow controller and to run experimental protocols or new technologies. GENI has been deployed in several campus networks and testbeds where the GENI racks are located. Each one of those racks includes a computed aggregate to manage programmable VMs and a network aggregate to manage a programmable Layer 2 switch with OpenFlow features provides connectivity between GENI backbones. The deployment of these GENI racks in campus networks allows users to be directly connected to service experiments.

To provide an easy-to-use interface to configure the slices while ensuring there is no interference between slices, a management software named Expedient[7] is available to the facility's researchers. It consists of several subsystems: aggregate and components, clearinghouse, research organisations, experiment support service, opt-in end users, and GENI operation and management.

Additionally, GENI is connected to the Internet without interfering with it and enables other users to connect to experimental services in a simple way. Furthermore, it is extensible and allows research communities to connect and to integrate their networks to the facility. Finally, it provides

7 Expedient: http://www.openflow.org/wk/index.php/Expedient

the mechanisms required to include additional technologies as well as the support needed for measurement-based quantitative research.

Two different paradigms can be used to virtualise the network resources in GENI. The virtual local area networks (VLANs) is the most basic strategy, which provides a well-known data isolation level. However, this approach cannot offer performance isolation or network programmability to the researchers. A more flexible approach incorporates SDN in the slice via OpenFlow-enabled network components. In this latter case, similarly as in the Stanford facility, FlowVisor determines the resources that belong to each slice. The most common approach is to assign an IP subnet to a slice, and in the case of non-IP experiments, an Ethertype identifies the traffic, which implies the configuration of the end hosts used by the researchers. As a consequence, the allocated VMs could be interconnected at data plane using IP, vanilla VLANs, and VLANs managed by OpenFlow.

2.4.2.3 *The OFELIA facility*

OFELIA (OpenFlow in Europe: Linking Infrastructure and Applications) [15] is an experimental facility where researchers are able to run experiments in an already deployed OpenFlow network. As a feature added by the SDN technology, it allows researchers to control and to extend the network itself dynamically and precisely. The latest reported setup of OFELIA facility is formed by 10 islands connected to each other.

Regarding the experimentation process, the OFELIA Control Framework (OCF)[8] is the management software to use, and it is based on the Expedient software developed by GENI. One of the extensions introduced by OCF adds computing resources to the management framework. Every experiment must be previously registered and accepted such that it can be run in an isolated slice that the researcher will be able to manage and to configure. This slice consists of a set of virtual machines and end hosts that are deployed on physical servers, an OpenFlow controller

8 OFELIA Control Framework (OCF): http://www.fp7-ofelia.eu/ocf-ofelia-control-framework

(typically located on one of those VMs) and the allocated network resources.

Concerning the mechanism used to virtualise the control plane exposed to the experiments, either FlowVisor or VeRTIGO (detailed in Section 2.3.3.2) are used to slice the experimental network (a decision made at each island). An Optical FlowVisor is also available to slice the optical devices.

FlowVisor is the most used mechanism to slice a facility based on OpenFlow and implement the virtualisation of the networking resources. This tool is integrated within the OCF (through the OpenFlow Aggregate Manager component) and is used by the experimenters to manually select the desired topology. However, it does not allow to instantiate virtual topologies (only a subset from the physical topology is possible). VeRTIGO overcomes this limitation and allows to request a virtual topology on top of the OFELIA facility. Moreover, the integration with the OCF allows the automatic setup of arbitrary topologies defined by the researchers.

After testing different slices schemes, the VLAN based slicing was proposed for OFELIA to offer the maximum flexibility to the experiments (although other alternative slicing mechanism could be used). The reasoning was twofold: the scalability issues with OpenFlow version 1.0 and the limited number of TCAM entries on hardware switches available in the facility. To improve the flexibility offered to the experiments, either one VLAN ID or a group of VLAN IDs could be assigned to the slice, enabling the use of VLAN tags at the experiment level. This proposed slicing mechanism guarantees the isolation at Layer 2 while limiting the impact on the number of TCAM entries used per slice, and thus, improving the overall scalability of the facility.

It is important to note that OFELIA and GENI are research facilities with no production traffic unlike the Stanford facility, which supports both research and production traffic.

2.4.2.4 *The GEANT Testbed as a Service (TaaS) service*

Currently, the GÉANT Testbed Facility [94] (GTS) is the new service offered by GÉANT to researchers who want to set up and customize their own experimental platform to test new concepts in networking. Previously, the GÉANT OpenFlow Facility [95]

(GOF) was proposed to offer an OpenFlow based experimental facility to researchers.

THE GÉANT OPENFLOW FACILITY (GOF)

The GÉANT OpenFlow Facility (GOF) [95] was deployed on top of the GÉANT backbone network[9]. The aim was to demonstrate its potential to deliver Testbed as a Service (TaaS) capabilities over GÉANT. GOF supports the announcement of the available network and computing resources, a reservation mechanism and the exposure of both management and control plane to the researchers.

The GOF PoPs are collocated with five of the GÉANT PoPs in Vienna, Zagreb, London, Amsterdam and Frankfurt. Regarding the resources deployed in this PoPs, the computing resources are VMs provided by Xen hypervisor, and the networking resources are software-based switches based on Open vSwitch (OVS). The physical topology is a full mesh graph connected using pseudowires (L2MPLS VPNs) over the GÉANT backbone (avoiding VLAN switching support).

Similarly to OFELIA, GOF assigns one or a range of VLAN IDs to each slice in order to virtualise the network. As a consequence, the traffic from each user is identified by the VLAN ID. The experimentation with VLANs is only exposed to the researcher when a set of VLANs are assigned to the slice.

The GÉANT OpenFlow Control Framework (GOCF), which is an extension of OCF (developed within the OFELIA project), is located in the Frankfurt PoP. Moreover, FlowVisor is also used to implement the virtualisation of the network, which is also located in the same PoP.

THE GÉANT TESTBED FACILITY (GTS)

The GÉANT Testbed Facility [94] (GTS) is the new production service from GÉANT to deliver TaaS capabilities. The main advantage over other similar facilities (such as PlanetLab or GENI) is that a variety of resources is exposed to the researchers as any resource can be included in GTS through

9 GÉANT, The European Research and Education Backbone: `http://www.geant.net`

the appropriate Resource Control Agents (RCAs), which is the specific management software developed for that resource. Moreover, GTS offers the capability to describe the network topology and virtualised resources to be programmed and reserved using a Domain Specific Language (DSL). The DSL description can be saved and reused in larger experiments. Additionally, DSL allows to pre-define and schedule the testbed settings, and also specify the start and end times.

The GTS PoPs are collocated with four of the GÉANT PoPs in Amsterdam, Prague, Ljubljana and Bratislava. The GÉANT backbone network is used to interconnect these locations by means of Virtual Circuits over the GÉANT Lambda service. Currently, the testbeds could be extended beyond the GTS boundaries through the External Domains Ports.

Regarding the infrastructure, it consist of several servers with standard operating systems and virtualisation software (providing VMs), and standard Ethernet switches and hardware based OpenFlow enabled devices (OpenFlow 1.3 based on HP5900 devices).

The composition and orchestration of the experiment testbeds y provided by GTS services. The requests from the users are translated into resource-specific actions on the virtualised platforms.

2.4.2.5 *The FIBRE facility*

The FIBRE [96] (Future Internet testbeds/experimentation between Brazil and Europe) testbed is a large-scale experimental facility for researching on Future Internet launched in 2011. The facility is the result of the federation of experimental facilities deployed in Brazil and Europe, both at the data plane and the control and monitoring framework levels.

The federation of different testbeds is beneficial for the research community to permit experiments that span several facilities. Concretely, FIBRE is the result of the federation of 13 separate experimental facilitates, known as islands. The main challenge is to provide a unified view to manage all the exposed resources. Thus, the management of FIBRE is the main challenge to overcome, and the control and monitoring framework (CMF) is the key element to facilitate this task.

As a design decision, FIBRE includes three CMFs: OFELIA Control Framework (OCF), OMF and ProtoGENI, which respectively allows to orchestrate three different types of resources: OpenFlow enabled devices, wireless resources, and emulated resources. In order to deal with the particularities of FIBRE, the three CMFs have been extended. As a result, FIBER offers the possibility to research with heterogeneous physical resources, including OpenFlow, wireless and optical communications.

Regarding the OCF (developed within the OFELIA project), this framework makes use of FlowVisor through the appropriate Aggregate Manager to handle the slicing of OpenFlow resources, as flowspaces, and expose them to the corresponding researcher. In relation to this, there is nothing new or specific to FIBRE.

The resources available at each island include OpenFlow enabled switches (and OpenFlow controllers), a cluster of compute and storage servers (with virtualisation software), and a cluster of wireless nodes (also virtualised). Moreover, each island could add specific resources to make them available to the researchers, such as optical networks, WiMax nodes and 3G/4G testbeds.

2.4.2.6 *The FELIX facility*

The FELIX [97] (FEderated Testbeds for Large-scale Infrastructure eXperiments) facility aims to enable the experimentation of Future Internet through the federation advanced programmable network testbeds deployed in Europe and Japan. FELIX defines a common framework for federated Future Internet testbeds to dynamically request resources, manage and control their interconnexion and monitor the assigned resources.

Each domain is controlled by a dedicated software that exposes different interfaces to the federation framework, which is in charge of orchestrating the resources when dealing with a multi-domain scenario. Each island is a set of virtualised computing and networking resources under a single administrative domain, which consists of multiple SDN zones, each with specific control tools and interfaces (such as Layer 2 switching zone, optical switching zone, OpenFlow controlled zone, and transit domain zones). The transit network domain exposes the control of the connectivity services by means of Network Service Interface (NSI). Finally, a slice is an isolated subset of virtual computing and networking resources defined by the researcher.

Both networking and computing resources available at different facilities are exposed to the researchers. The federated resources allows to create a virtual infrastructure that spans several domains. One of the research topics covered by FELIX is the transit domains, more concretely, the integration of SDN enabled facilities with transit domains controlled by NSI. This solution makes possible to dynamically establish and tear-down network flows across multiple domains that use different technologies.

On the one hand, the FELIX Space is the management and control framework that coordinates the creation of the slices over heterogeneous technologies and multi-domain facilities. In this context, the Resource Orchestators (RO) orchestrates the end-to-end provisioning, while the Resource Managers (RM) are used to manage and control specific type of resources (e.g. Computing RM). On the other hand, the User Space is able to dynamically manage and control the resources assigned to the associated slice.

The architecture defined in FELIX is the result of the analysis of other relevant research projects, such as OFELIA, FIBRE, BonFIRE, FED4FIRE, GridARS and RISE. In relation to OFELIA and FIBRE, the main differences are the idea of a Resource Orchestrator for multi-domain provisioning and the removal of human intervention on some configuration and provisioning operations.

As detailed in [98], most of the architectural components have been developed from scratch, such as the logic and interfaces of RO. However, some key artifacts developed by OFELIA have been used as the starting point. In this regard, several RM have reused the Aggregate Managers developed in OFELIA. In addition, some components from the Graphical User Interface have been reused from OFELIA (which are also based on Expedient developments).

Regarding the SDN Resource Manager (SDN RM) responsible of configuring and controlling the OpenFlow-enabled devices, this module interacts with FlowVisor to proxy the

OpenFlow messages to the appropriate controller (exposing the programmability of the network to the experiments). As a result, the slices are properly isolated in a similar way to previously analysed facilities.

2.5 CONCLUSIONS

This chapter focuses on two different topics: NV and experimental facilities. On the one hand, with regard to NV, the reference architecture and definition are presented to contextualise the terminology and scope in this thesis. Then, a NV taxonomy is proposed to properly categorize the NV approaches. Finally, a NV survey is presented and analysed using the proposed taxonomy. On the other hand, in relation to the second topic, a survey of the most relevant experimental facilities in the context of this thesis is presented.

To conclude, as a result from the analysis of these surveys, Section 2.5.1 presents the list of requirements to be fulfiled by the contributions presented in Chapter 3.

2.5.1 Requirements

One of the main contributions from this thesis is a new NV proposal, which is presented in Section 3.1. As previously mentioned, even the definition of what the NV really means, and thus, which are the design principles to consider, depend on the target scenario, which for the purposes of this thesis is the experimental facility. Apart from some of the common requirements imposed to any experimental facility, such as the isolation between slices (i.e. experiments), flexibility to support novel approaches (e.g. non-IP proposals) and scalability of the facility, there are some other requirements imposed to the NV approach that have their roots in the research activities conducted in the actual experimental facility, which in this case are focused on security and access networks technologies. The complete list of imposed requirements are detailed next.

R1 ISOLATION

One of the most important steps in the design of a shared experimental facility, such as the EHU OpenFlow Enabled Facility (EHU-OEF) detailed in Section 4.1, is deciding how to define the network slices to be experiment-agnostic. The slice definition must not collide with the network proposals to be tested. As previously stated in Section 2.3.3.1, the FlowVisor relies on the same header fields to define its rules as OpenFlow flows. These rules define the flowspace, which is the set of flows that make up the slice. Although implementation dependent, the concept of flowspaces can be generalised to any NV approach and its possible implementation. The flowspaces can concern a single layer, either the physical, link, network or transport layer, or be a combination of all four. The definition of each slice can be made independently from the others; however, having a common infrastructure with slices defined at different layers (i.e. by means of different parameters) can make it challenging to separate them all. For example, in the case that one slice is defined as a destination IP address and another slice is defined as a destination TCP port, there is a conflict between both slices when a packet goes to that IP address and that TCP port. Therefore, the slices cannot be completely isolated, and this does not comply with one of the requirements of the facility. In this multi-level slice definition, an extra verification step must be enforced to avoid the collision between slice definitions. As a consequence, it is easier to use a common layer and an even parameter (or set of parameters) to define the slice, which further simplifies the traceability. In this case, it is straightforward to ensure isolation, the only condition is that each possible value of that parameter can only be used once.

R2 FLEXIBILITY

Another important requirement imposed by the experimental facility is the flexibility to support novel approaches. It must impose as few restrictions as possible to the experiments conducted over its infrastructure. This flexibility for experiments collides with the flexibility provided by the NV approach (e.g. FlowVisor) to define the slices, as shown in Figure 2.5. If

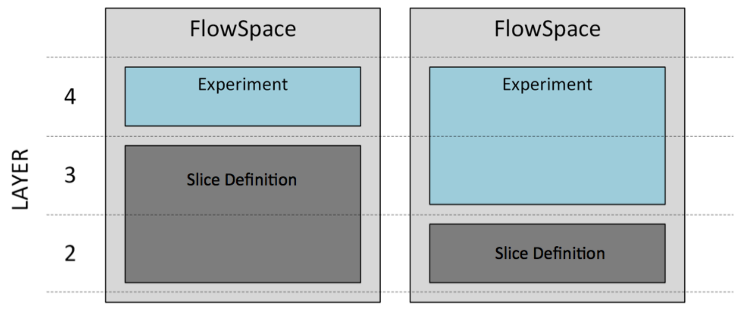


Figure 2.5: The flexibility at slice definition vs. the flexibility at experiment level

a researcher uses the flexibility at the slice definition, which requires several parameters and different layers to be used to define the slice, there would be fewer degrees of freedom (i.e. the parameters not selected at slice definition) in the experiment. For example, if the slice is defined by setting a value for all the parameters from layer 1 to layer 3, the experiment could only use the layer 4 ports to conduct the experiments because the rest of the parameters must remain the same in the definition of the slice. To support clean-slate approaches (e.g. post-IP), it is better to use the lower layers for slice definition to avoid the assumption of legacy protocols, such as IP, within the experiments. Layer 2 is a good compromise to support post-IP proposals (Layer 3) and to avoid a slice definition based on physical ports (Layer 1), which is quite restrictive because the physical links cannot be shared. This means that specific links and physical ports must be deployed for each slice, which does scale to the facility size (i.e., just a few experiments would be possible). A pure optical backbone network (such as NV based on CE- or PE-based L1VPNs [99]) is discarded for economic (optical equipment is very expensive) and scalability reasons. As explained in [100], optical equipment (e.g. OXCs and ROADMs) is able to switch optical signals at individual wavelength level (i.e. lambda), which is the highest granularity offered by this devices to implement the optical NV solution and does not scale enough.

R3 SCALABILITY

The scalability of the NV solution is a requirement for any experimental facility, in order to support a huge number of researchers and experiments running on it. Moreover, it is also expected that the proposed NV approach could be deployed in data center networks, where this requirement is even more challenging and must scale to hundreds of thousands of tenants.

Regarding the scalability of the implementation some other considerations must be also taken into account. For instance, using VLANs for slicing the facility, the theoretical maximum number of slices is 4096 (the VLAN identifier has 12 bits); however, one of the main limitation factors currently in OpenFlow deployments is the total number of flow entries supported by the devices [101]. Moreover, when using FlowVisor, the expansion of the rules explained in [102] impairs the scalability of the facility.

R4 STABILITY

In general, the definition of the virtual network instance should remain stable and should not evolve over time. If the slice definition changes, it is possible that previously deployed flow entries could not be updated to cover the new flowspace. For instance, if the slice definition is a list of MAC addresses of users (list1) and a certain flow entry is deployed and enforced to list1 (with a possible explosion of flow entries), when a new user is added (list1'), the previously deployed flow entries will probably not be updated to list1'. It could be complex to track all these changes and some inconsistencies could arise. In order to avoid this problems and simplify the traceability of the virtual instance, the slice description must not change from its original definition.

R5 TRANSPARENCY

The experimental facility must be as transparent as possible to the experiments and expose the virtual network instance with the minimum overhead. From the application's point of view, which will run on top of the OpenFlow controller, it is desirable to process the packets in a transparent manner. This way, the application does not need to be aware of the facility. This means that packets arriving at the application should not be tagged with additional fields (e.g. a VLAN tag to identify the slice) because this could imply that the application needs to be aware of the corresponding tag. Even more so, some third-party applications may be unable to adapt (e.g. closed code). To this end, the data plane should not be modified in the path from the source to the destination, or at the very least, this modification should not be exposed to the application. Additionally, the transparency also makes easier the traceability and DevOps processes (e.g. root-cause analysis, observability, troubleshooting and verification).

R6 SUPPORT FOR RESEARCH ON NETWORKING

The main goal of EHU-OEF is to allow the research on networking and enable a facility for testing novel proposals and network architectures. In this case, this requirement is twofold: on the one hand, it must expose the network programmability to support the novel protocols, and on the other hand, it must support the research on operator networks. In some scenarios, such as in access networks (e.g., DOCSIS or GPON technologies), it is typical to make use of VLAN tags and MPLS labels to identify and aggregate traffic. Thus, to make research possible in those scenarios, it is necessary to expose the full space of VLANs and MPLS to the experiments. Consequently, the facility must support the ability to address VLAN tags and MPLS labels at the experimental level without any restriction, which requires the whole VLAN and MPLS space to be available for each experiment.

5 CONCLUSIONS

This chapter concludes with a summary of the content presented throughout this document in Section 5.1. Then, the main contributions as the outcomes from this thesis are highlighted in Section 5.2. Afterward, the dissemination of results related to the content of this thesis are listed in Section 5.3. Finally, the future work and research directions are detailed in Section 5.4.

5.1 SUMMARY

The main outcome from this thesis is a Layer 2 Network Virtualisation (NV) mechanism, which is secured, efficient and scalable. The security is provided by Flow-based Network Access Control (FlowNAC), which is an identity based access control proposal that authenticates and authorises the users before granting access at data plane level to the virtual network instance. Moreover, the efficiency is achieved by a transparent end to end communication (without any additional header/tag or action performed) that does not add any extra overhead. The virtual network identifier is coded on the MAC prefix, being a namespace orthogonal to the networking research supported at the virtual network instance. The performance numbers from MAC prefix matching also supports the idea of an efficient approach to NV. Furthermore, the scalability comes from the capability of supporting the isolation between the different virtual network instances. The most important parameter is the length of this identifier, which in Layer 2 Prefix-based Network Virtualisation (L2PNV) is configurable. As in EHU OpenFlow Enabled Facility (EHU-OEF), it could be perfectly defined as 22 bit length (using the first three bytes for the MAC prefix assigned to L2PNV), being scalable enough even for the largest data center scenarios.

The introduction section contextualises this research work and presents the research questions and objectives covered by this thesis. It also motivates the need for this research and its relevance in the current context.

Afterward, the state of the art section analyses the literature with regard to the two main topics covered in this thesis, the NV approaches and the current experimental facilities (as the main target scenario related to the proposed NV approach). In this regard, a NV taxonomy has been proposed to properly organise and understand the past, present and future NV proposals analysed in the NV survey. The conclusions from this analysis based on the proposed taxonomy are relevant for this work. A definition for NV in the context of this thesis and a NV architecture are also presented. The survey of experimental facilities allows to contextualise the work performed on EHU-OEF.

Once the state of the art is presented, the three individual contributions are detailed. First, the L2PNV approach is explained as the main objective from this thesis. Since the main limitation from L2PNV is the definition of a new MAC addressing scheme, the MAC Address Configuration Protocol (MACP) protocol is proposed to facilitate the dynamic assignment of MAC addresses to either physical or virtual end devices. Finally, the FlowNAC is proposed as a mechanism to add security at data plane, by controlling the access to the network at flow level. In conjunction, the three components are the missing and most relevant pieces to build the solution.

Then, the validation of the aforementioned contributions is presented. Basically, two different types of validation have been performed: experimental and analytical. The EHU-OEF demonstrates the viability of the proposals and experimentally validates the proposed experimental facility. EHU-OEF is based on L2PNV, MACP and FlowNAC. With regard to L2PNV, EHU-OEF demonstrates that it works fulfiling all the requirements: a stable definition of slices, the isolation between virtual network instance, scalable for great number of experiments, flexible to experiment on novel architectures and proposals, and the VLAN space is available at experimental level. Moreover, the deployment of MACP has permitted to assign novel MAC addressing schemes quite easily. Furthermore, the FlowNAC has enhanced the facility with a mechanism to secure the access to the experiments at data level. The three proposals have been experimentally validated on EHU-OEF as necessary building blocks to construct the facility.

Regarding the analytical validation, the performed studies demonstrated the reduction of flow entries as a result of enforcing the isolation between the virtual network instances (just one MAC prefix per slice), and the reduction of flow entries as a consequence of using less rules to add support for legacy protocols at experiment level.

Finally, the complete list of contributions obtained as the outcomes from this thesis are detailed: the EHU-OEF facility, the NV taxonomy, the L2PNV proposal, the MACP protocol, the Prefix-based Forwarding Decision (PFD) approach, the FlowNAC proposal and the Software Defined Networking (SDN)-enabled Network Functions Virtualization (NFV) architecture. Additionally, the dissemination activities related to this research work are listed grouped by journals with JCR, international publications, demonstrators, contribution to research projects and Spanish publications. To conclude, the future work is presented and separated in future activities and future research lines.

5.2 CONTRIBUTIONS

This section details the complete list of contributions from this thesis. As previously demonstrated, apart from the specific contributions detailed in Chapter 3, there are other additional contributions that can be considered as a result of this thesis. Some of them are a direct consequence of writing the actual document (e.g., the NV taxonomy) and some others are outcomes from the actual means developed to validate the novel proposals (e.g., the EHU-OEF facility). It is worth to enumerate all of them:

1. The EHU-OEF facility: the campus-wide experimental facility deployed on the University of the Basque Country (UPV/EHU) that fulfils all the imposed requirements is one of the main contributions of this thesis. An active and successful facility that incorporates other original contributions from this thesis and experimentally validates

the research work, is a valuable achievement. In addition, it has been employed and demonstrated in several demo-tracks in relevant international conferences and final reviews from EU projects. As an experimental facility its value is twofold. On the one hand, it has been the means to validate the innovative proposals (also used to construct the facility). On the other hand, the facility has been the enabler for testing these novel networking proposals, as well as, for managing the lifecycle of the experiments performed.

2. The NV Taxonomy: although the NV has been a research topic for years, recently, the emergence of the SDN technology has shaken its basic principles. Moreover, depending on the target scenario, the requirements imposed to the NV solution are different, crystalising in distinct manners of defining what the NV should be. As a consequence, most of the NV surveys and proposed classifications are related to a specific target scenario, or they are partial studies not considering the latest advances. Due to all these reasons, a novel NV taxonomy is proposed, that allows to organise the state of the art related to NV and its evolution to better understand the past, present and future proposals. It also allows to properly categorize the L2PNV approach and clearly identify the contribution of this work.

3. The L2PNV proposal: one of the main contributions of this thesis, addressing *RQ1*, is the novel NV approach, which relies on the MAC address prefix as the virtual instance namespace to identify the corresponding virtual network instance. The use of MAC prefixes allows the aggregation of MAC addresses associated to each virtual instance, thereby improving the scalability of the solution and reducing the number of rules to define the virtual network. Additionally, the virtual instance namespace is orthogonal to the experiments (or in general, to the namespaces exposed to the virtual instances), being transparent for both the data and control planes. Moreover, it does not introduce any overhead at data plane (neither a tunnel or a tag is added) or at network device level (no action

is required, e.g., push/pop). The isolation is achieved by simply enforcing a matching on the MAC address field.

4. The MACP protocol: this protocol was designed to overcome the main limitation of L2PNV, i.e. it is based on a novel MAC addressing scheme, while addressing *RQ2*. This means that the globally administered address assigned by Network Interface Card (NIC) manufacturers, ending in a flat addressing scheme, are not valid for this novel NV proposal. As a consequence, the MAC addresses of end devices must be updated either manually or automatically depending on the target virtual network instance. Although the manual configuration is always possible, and the automatic provisioning of properly configured virtual interfaces is possible in data center environments, there is no solution for other target scenarios with physical NICs, such as campus and operator networks. As a consequence, a novel protocol for assigning and dynamically configure MAC addresses is proposed to fill the gap with a generic approach (valid for both physical and virtual NICs). As a side effect, the MACP becomes as an enabler to facilitate the adoption of novel MAC addressing schemes, used for instance in L2PNV and PFD proposals.

5. The PFD approach: as a way to demonstrate the opportunities that novel MAC addressing schemes could generate, a novel forwarding mechanism based on MAC address prefixes is presented. This proposal leverages on a mechanism for aggregating the MAC addresses of end devices to reduce the size of the forwarding tables (i.e. the number of flow entries), and consequently, improve the overall scalability of Layer 2 networks. Similarly to L2PNV, the MAC addresses from all the interfaces must be updated, and once again, MACP becomes the enabler to facilitate the proper assignment of addresses to end devices. Moreover, PFD has been deployed and tested over EHU-OEF following the proposed methodology for experimentation. Therefore, in addition to a novel forwarding proposal for campus, operator and data center networks, PFD has covered two important objectives: (1) the experimental validation of EHU-OEF to support research

on novel networking proposals, and (2) the relevance and usefulness of researching on novel MAC addressing schemes.

6. The FlowNAC proposal: this proposal addresses *RQ3* and allows to overcome one of the main security issues of most of the current experimental facilities, the data plane access to the facility (the Virtual Instance Data Access Point (VIDAP) reference point). This topic has been only partially solved in those cases in which a remote access to the facility is secured by a Virtual Private Network (VPN) access. A similar approach is not available when the access from end devices is internal to the facility (such as a direct physical connexion or a virtual interface from a Virtual Machine (VM) deployed on one of the virtualised servers). FlowNAC provides a granular (fine or coarse granularity, based on flows) mechanism to authenticate and authorise individually the access to each network service (e.g. a virtual network instance) offered by the network. As a result, on the one hand, the end user is able to request access to a specific slice, and on the other hand, the experimental facility is able to control the access to each slice based on the identity of the user and the policies defined by the researcher (the owner of the slice). In a nutshell, FlowNAC provides a secured mechanism to control the access at the VIDAP reference point (data plane), similarly to what L2PNV-FlowVisor (through an OpenFlow interface) provides at the Virtual Instance Tenant Access Point (VITAP) reference point (control plane) when securely exposes the programmability of the network to the researchers (tenants). In this latter case, the secured solution relies on TLS (based on certificates) on the control channel and a policy defined on the proxy controller (i.e. the L2PNV-FlowVisor). In addition to its applicability to experimental facilities (EHU-OEF) and NV approaches (L2PNV), FlowNAC was designed as a generic tool to secure the access to the networks (similar to an stateless firewall dynamically configured) that can be used in other scenarios, such as campus, operator and data center networks.

7. The SDN-enabled NFV architecture: as a consequence of the adoption of some NFV principles in the design of the experimental facility to simplify the deployment of new services (as Virtual Network Functions (VNFs)) to support the research activity, several VNFs have been designed and developed on EHU-OEF. In this process, the VNF design has become the key factor to success, evolving from SDN-agnostic approaches to fully SDN-enabled proposals. As a result, a set of VNF design principles have been identified, and a SDN-enabled architecture is proposed to implement the VNFs based on the separation of stateless and stateful components to improve the overall performance of the deployed VNF, thus, addressing *RQ4*. In relation to this approach, the FlowNAC case has been used to illustrate the benefits from this architecture. The modular design allows not only to improve the performance (stateful processing at the forwarding engine level) but also to individually scale each component depending on the actual demand. The exposure of the network programmability to build the VNFs is a step forward in this evolution that introduces new challenges (how to share the forwarding resources), but has a great future (performance improvements).

Regarding the specific contributions described in Chapter 3, it is important to highlight that L2PNV, MACP and FlowNAC have been designed to work independently from each other. As a consequence of a modular design, the proposals are more versatile and could be adopted individually in different scenarios. For instance, MACP could be used for PFD or FlowNAC could improve the access control in campus networks. However, as demonstrated by EHU-OEF, the cooperation between them is worthwhile, thus, L2PNV rely on MACP to facilitate the assignment of the new MAC addressing scheme and FlowNAC improves the security of L2PNV adding a novel access control system.

As demonstrated by this research work (mainly with regard to L2PNV and PFD), a better use of MAC address namespace is feasible. The research on novel MAC addressing schemes is relevant to achieve some improvements in networking (such as NV, forwarding or scalability). As a result, the Layer 2 can be further enhanced with additional functionalities. In this regard, MACP becomes the enabler for the innovation in novel addressing schemes allowing clean slate proposals at Layer 2.

In relation to the objective of researching on networking, SDN technology is an actual enabler for innovation in different aspects. First of all, it allows to expose the programmability of the network devices to researches, which is characteristic from this technology (it was not possible before) and opens new possibilities. Additionally, SDN overcomes the kidnapping of networking innovation by vendors, since they do not allow external innovation on their products (they implement either standard proposals or specific vendor features as competitive advantage). Moreover, SDN allows to reduce the development cycles of new features (similar to software development) and both hardware and software based solutions share the same standard interface.

With regard to the latest advances on NFV concepts, the approach to experimental facilities could evolve with innovative architectures and proposals. There are several design principles (such as isolation, flexibility, scalability, and sharing of resources) that are common in both cases. In this sense, the outcomes from the research on NFV can boost the deployment of more advanced experimental facilities.

5.3 DISSEMINATION OF RESULTS

The complete list of dissemination activities related to the outcomes of this thesis is detailed below. They are organised in five categories: (1) journals with JCR, (2) papers on international conferences, (3) demonstrators on international conferences, (4) contributions to research projects and (5) papers in Spanish conferences.

1. Publications in Journals with JCR:
    - 2015, "An architecture for dynamic QoS management at Layer 2 for DOCSIS access networks using OpenFlow", Computer Networks, Elsevier (ISSN: 1389-1286) (Impact Factor JCR=1.282); http://dx.doi.org/10.1016/j.comnet.2015.11.017;

Mendiola, A; Fuentes, V; Matias, J; Astorga, J; Toledo, N; Jacob, E; Huarte, M.

- 2015, "Toward an SDN-enabled NFV architecture", Network and Service Vistualization; IEEE Communications Magazine, vol 53, no. 4, pp 187-193; (ISSN: 0163-6804) (Impact Factor JCR=4.460); http://dx.doi.org/10.1109/MCOM.2015.7081093; Matias, J; Garay, J; Toledo, N; Unzilla, J.; Jacob, E.
- 2014, "The EHU-OEF: An OpenFlow-based Layer-2 experimental facility", Special issue on Future Internet Testbeds
  - Part II, vol 63, pp 101-127; Computer Networks, Elsevier (ISSN: 1389-1286) (Impact Factor JCR=1.282); http://dx.doi.org/10.1016/j.bjp.2013.11.013; Matias, J; Mendiola, A.; Toledo, B.; Tornero, B.; Jacob, E.

2. Publications on International Conferences:
   - September 2014, "FlowNAC: Flow-based Network Access Control", Third European Workshop on Software Defined Networks (EWSDN 2014); Matias, J.; Garay, J.; Mendiola, A.; Toledo, N.; Jacob, E.; Budapest, Hungary (September 2014).
   - September 2014, "Integrating complex legacy systems under OpenFlow control: The DOCSIS use", Third European Workshop on Software Defined Networks (EWSDN 2014); Fuentes, V.; Matias, J.; Mendiola, A.; Huarte, M.; Unzilla, J.; Jacob, E.; Budapest, Hungary (September 2014).
   - May 2014, "Deploying a Virtual Network Function over a SDN infrastructure", Terena Networking Conference (TNC 2014); Jacob, E.; Matias, J.; Mendiola, A.; Fuentes, V.; Garay, J.; Pinedo, C.; Dublin, Ireland (Mayo 2014).
   - October 2012, "Implementing Layer 2 Network Virtualisation using OpenFlow: Challenges and Solutions", European Workshop on Software Defined Networks (EWSDN 2012); Matias, J.; Tornero, B.; Mendiola, A.; Jacob, E.; Toledo, N.; Darmstadt, Germany (Octubre 2012).

- June 2012, "Extending Neutrality to Experimental Facilities", 3rd International Conferences on Access Networks, Services and Technologies (ACCESS 2012 - InfoWare 2012); Matias, J.; Jacob, E.; Higuero, M.V.; Toledo, N.; Venecia, Italy (June 2012). Best Paper Award and invitation for International Journal On Advances in Network and Services, v6 n 1&2 2013.
- May 2012, "Deploying OpenFlow in production at the University of the Basque Country: Identity based network infrastructure configuration", TERENA Networking Conference 2012 (TNC 2012); Jacob, E.; Matias, J.; Reykjavik, Iceland (May 2012). Selected Paper for TNC2012 Proceedings (ISBN 978-90-77559-00-0).
- November 2011, "An OpenFlow based Network Virtualisation Framework for the Cloud", Network Infrastructure Services as part of Cloud Computing (NetCloud 2011 - CloudCom 2011); Matias, J.; Jacob, E.; Demchenko, Y.; Sanchez, D.; Atenas, Greece (November 2011).
- June 2011, "Towards Neutrality in Access Networks: A NANDO Deployment with OpenFlow", 2nd International Conferences on Access Networks, Services and Technologies (ACCESS 2011 - InfoWare 2011); Matias, J.; Jacob, E.; Toledo, N.; Astorga, J.; Luxemburgo (June 2011).
- September 2010, "Extending AAA Operational Model for Profile-based Access Control in Ethernet-based Neutral Access Networks", 1st International Conferences on Access Networks, Services and Technologies (ACCESS 2010 - InfoWare 2010); Matias, J.; Jacob, E.; Demchenko, Y.; de Laat, C.; Gommans, L.; Valencia, Spain (September 2010).
- June 2010, "Access Control for Carrier Ethernet-based Service Delivery: The Service-Port Policy Enforcement", TERENA Networking Conference 2010 (TNC 2010); Matias, J.; Jacob, E.; Demchenko, Y.; de Laat, C.; Vilnius, Lithuania (June de 2010).

- June 2009, "Enhancing NGN's versatility for Multi-Service support: the Bridging Virtualisation approach", 10th International Conference on Telecommunications (ConTel 2009); Matias, J.; Jacob, E.; Aguado, M.; Astorga, J.; Zagreb, Croatia (June 2009).
- June 2009, "Network Convergence through Bridging Virtualisation Tool", TERENA Networking Conference 2009 (TNC 2009); Matias, J.; Jacob, E.; Málaga, Spain (June 2009).
- May 2009, "Towards a real convergence for Multi-play", IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB 20009); Matias, J.; Jacob, E.; Higuero, M.V.; Saiz, P.; Astorga, J.; Bilbao, Spain (Mayo 2009).
- December 2006, "Adaptation of IEEE 802.1X for secure session establishment between Ethernet peers", Second International Conference on Information Systems Security (ICISS06); Sáiz, P.; Matías, J.; Jacob, E.; Bustamante, J.; Astarloa, A.; Kolkata, India (Diciembre 2006).

3. Demonstrators on International Conferences:
    - June 2015, "Exploring the Service/Orchestrator interaction: Secure and dynamic deployment of services based on identity", Demonstrator in the "SDN Solutions Showcase", Open Networking Summit 2015 (ONS2015); Jacob, E.; Garay, J.; Matias, J.; Woesner, H.; Sune, M.; Jungel, T.; Santa Clara, EEUU (June 2015).
    - April 2015, "Self-deploying Service Graphs over ELwUD (EHU-OEF Lightweight UNIFY Domain)", Demonstrator in the 1st IEEE Conference on Network Softwarization (NetSoft 2015); Garay, J.; Matias, J.; Mendiola, A.; Astorga, J.; Jacob, E.; London, UK (April 2015).
    - May 2014, "Deploying a Virtual Network Function over a SDN infrastructure", Live demonstrator in the paper presentation at the Terena Networking Conference (TNC 2014); Jacob, E.; Matias, J.; Mendiola, A.;

Fuentes, V.; Garay, J.; Pinedo, C.; Dublin, Ireland (May 2014).

- March 2014, "Enriched slices in EHU-OEF empowered by NFV: the Access Control VNF use case", Demonstrator in the "Academic/Research Demo track", Open Networking Summit 2014 (ONS2014); Matias, J.; Garay, J.; Mendiola, A.; Fuentes, V.; Pinedo, C.; Jacob, E.; Santa Clara, USA (March 2014).
- December 2013, "Access Control NFV enforcement at the EHU-OpenFlow Enabled Facility", Network Functions Virtualisation Demonstrations (NFV-Demo), IEEE Global Communications Conference (GLOBECOM 2013); Matias, J.; Jacob, E.; Pinedo, C.; Mendiola, A.; Fuentes, V.; Garay, J.; Atlanta, USA (December 2013).

4. Contribution to research projects:
   - The FP7 UNIFY project ("Unifying Cloud and Carrier Networks", funded by the European Commission, FP7-ICT-2013-11, `https://www.fp7-unify.eu/`, 01/10/2013 - 31/04/2016): the EHU-OEF facility, L2PNV, MACP and FlowNAC have been demonstrated in several demo-tracks from international conferences (i.e., ONS2015, NetSoft2015, TNC2014, and ONS2014) in the context of this project. EHU-OEF (supported by L2PNV and MACP) has been used in different development activities during the project. Additionally, the state migration in FlowNAC (as a decoupled VNF: stateless and stateful) has become one of the use cases to be demonstrated in the final review of this project.
   - The FP7 ALIEN project ("Abstraction Layer for Implementation of Extensions in Programmable Networks", funded by the European Commission, `http://www.fp7-alien.eu/`, 01/10/2012 - 30/09/2014): the EHU-OEF facility was added as an island to the project's testbed. It was used during the final review of the project to demonstrate the DOCSIS integration in the SDN based facility. Additionally, EHU-OEF (including L2PNV, FlowNAC and MACP) was

used for project related demonstrators, testing of developments and validation activities.

- The S&N-SEC project ("Despliegue seguro de servicios con Redes Definidas por Software y Virtualisacion de Funciones de Red", funded by the Spanish Government, MINECO, 01/01/2014 - 31/12/2016): this project supports the further development of FlowNAC as a VNF and general security issues related to SDN and NFV architectures.
- The A3RAM-NG project ("Autenticación, Autorization y Registro de Actividades en Redes de Acceso Multiservicio de Nueva Generación", funded by the Spanish Government, MICINN TIC2010-21719-C02-01, 01/01/2011 - 31/12/2013): this project supported the initial implementation of the FlowNAC proposal.
- The Future Internet (funded by the Basque Government, ETORTEK IE08-227, 01/01/2008 - 31/12/2010) and Future Internet II projects (funded by the Basque Government, ETORTEK IE11-316, 01/01/2011 - 31/12/2013): these projects partially supported the initial deployment of the EHU-OEF facility and further testing of experiments over this experimental facility.
- The RedSOC project (funded by the Basque Government, SAIOTEK, 01/01/2011 - 21/12/2012)): this project partially supported the initial deployment of the EHU-OEF facility and the implementation of the L2PNV and the MACP proposals.

5. Publications on Spanish Conferences:
   - November 2012, "Solución de virtualisación de nivel 2 basada en prefijos para plataformas de experimentación", Jornadas Técnicas RedIRIS 2012 (JT2012); Matias, J.; Jacob, E.; Mendiola, A.; Tornero, B.; Pinedo, C.; Bilbao, Spain (November 2012).
   - November 2009, "Sistema para el control de acceso a red basado en servicios", Jornadas Técnicas RedIRIS 2009; Matias, J.; Jacob, E.; Santiago de Compostela, Spain (November 2009).

- October 2008, "Establecimiento dinámico de conexiones sobre PBB-TE", XVIII Jornadas Telecom I+D; Matias, J.; Jacob, E.; Higuero, M.V.; Astorga, J.; Bilbao, Spain (October 2008).
- October 2007, "Una propuesta basada en IEEE 802.1X para la autenticación y configuración de equipos finales en redes NGN", XVII Jornadas Telecom I+D; Matias, J.; Jacob, E.; Higuero, M.V.; Saiz, P.; Martinez de Salinas, J.; Valencia, Spain (October 2007). Third place in Best Paper Award from the congress.
- September 2007, "Propuesta para la configuración dinámica en redes NGN: Extended Configuration Protocol (ECP)", VI Jornadas de Ingeniería Telemática (Jitel 2007); Matias, J.; Jacob, E.; Higuero, M.V.; Saiz, P.; Martinez de Salinas, J.; Málaga, Spain (September 2007).
- November 2006, "Modelo de Red Orientado a Servicios Basado en Ethernet", XVI Jornadas Telecom I+D; Matías, J.; Jacob, E.; Sáiz, P.; Higuero, M.; Astarloa, A.; Madrid, Spain (November 2006). Third place in Best Paper Award from the congress.
- November 2006, "Hacia una arquitectura hardware de altas prestaciones para securización de Ethernet", XVI Jornadas Telecom I+D; Jacob, E.; Sáiz, P.; Astarloa, A.; Matías, J.; Aguado, M.; Pinedo, C.; Areizaga, E.; Madrid, Spain (November 2006).
- November 2005, "Problemática de seguridad en redes Ethernet extremo a extremo", XV Jornadas Telecom I+D; Saiz, P.; Matias, J.; Jacob, E.; Álvarez, A.; Areizaga, E.; Madrid, Spain (November 2005).

5.4 FUTURE WORK

This section details the foreseen future work as continuation of the research work presented in this thesis. Due to the difference in scope and goals, the future work can be separated in future activities and future research lines. The former details the expected next steps as the result of the research outcomes from this thesis, whereas the latter outlines the research directions identified in this process.

Concerning the future activities, several objectives have been identified to further disseminate the results from the work presented in this thesis. The main areas in this regard are the publication of novel contributions, the standardisation of the proposals and the adoption of the technology. Although part of the original contributions have been already published, which actually validates the originality and relevance of the ideas proposed in this thesis, there are some other innovative contributions that have not yet been published. This is in fact the most relevant next step. Additionally, some of this ideas are perfectly aligned with ongoing discussion on relevant standardisation bodies, such as the Internet Engineering Task Force (IETF). The standardisation of proposals is the best way for their widely adoption in real deployments, which is the main challenge for the outcome of any research activity. Moreover, prior to the adoption of these proposals, the current developments deserves the optimisation their implementation and the improvement of performance figures, which also contributes to justify their need.

With regard to the future research lines identified as the continuation of this research work, the most immediate ones are listed next. In relation to EHU-OEF, an upcoming evolution of the facility more influenced by the latest advances in NFV architectures is expected. Currently, the basic orchestration is a limitation for optimal provision of available resources, and smarter embedding algorithms are awaited. Moreover, new hardware (both computing and networking resources) is planned to be deployed extending the current infrastructure.

Regarding the L2PNV approach, there are several research lines to improve the solution. First of all, it is worth to mention that the actual proposal described in this thesis is a concrete NV approach (SDN-enabled Virtualization (SDNeV) type). However, the idea behind L2PNV is more general, the use of a novel MAC addressing scheme to virtualise the network based on MAC prefixes (as the Virtual Instance (VI) namespace) to identify the virtual network instances. Based on this criteria, other possible implementations or even other NV types are feasible, for instance, a Virtual Node (vNode) type of NV that makes use of MAC prefixes to differentiate the target instance. In addition, other alternative implementations for the SDNeV type (not based on FlowVisor) are considered, as a way to overcome the main limitation of FlowVisor and achieve L2PNV support beyond OpenFlow 1.0. An implementation based on OpenVirteX could be a plausible solution, consequently, instead of enforcing the isolation based on a MAC prefix matching on each and every device (which requires inspection at the OpenFlow control channel), a MAC rewrite action performed at the edge devices would be enough to assure the isolation. In this latter case, the OpenFlow control channel would be modified to perform MAC address translation between the virtual (at the tenant) and physical (at the infrastructure) addresses. Moreover, the simple matching (MAC prefix) needed at device level could be optimised and supported by specific hardware acceleration on the NIC (similarly to SR-IOV technology) to bypass the software switch provided by virtualisation solutions, and thus, improve the I/O performance for VNFs deployed on VMs. Also related to the support for NFV, L2PNV could be considered as a basic isolation mechanism on which novel steering proposals could rely.

In relation to FlowNAC, there is an ongoing research activity to improve the overall security provided to the target scenarios. According to [135], there are different security mechanisms (encryption, authentication, authorisation, and auditing) that can be used to overcome the distinct types of security threats (interception, interruption, modification, and fabrication). Given the actual implementation of FlowNAC, the encryption is not yet available, although the ability to distribute key material is provided (the key management and the establishment of security associations is supported by the IEEE 802.1X standard). In addition, the IEEE 802.1AE standard [136] (MAC Security) defines the MACsec PDUs and provides connectionless user data confidentiality, frame data integrity, and data origin authenticity. FlowNAC could be improved by leveraging on MACsec to either cypher or protect the integrity of the MAC frames.

One scenario that requires further investigation is the mobility of end nodes in EHU-OEF, because of either the migration of VMs or the mobility of physical machines. L2PNV, MACP and FlowNAC must be studied in detail to determine how this mobility really affect the experiments conducted on the facility and how this effect could be minimised. However, it is important to highlight that L2PNV does not bound any location information to the MAC address assigned to the end devices, only the virtual network instance identifier must be coded on the MAC address. By contrast, PFD actually codifies the location information on the MAC addresses in order to aggregate the end devices to reduce the forwarding table size. In this latter case, the use of gratuitous ARP (also used in PortLand [19]) could be used to minimise the impact of mobility scenarios. Depending on the case, a new MACP process could be needed after the migration of end devices. In addition, a new FlowNAC process is mandatory to preserve the security level, but it could be optimised (similarly to secure handover in IEEE 802.11 networks [137]). The outcomes from this line of research could be relevant not only for EHU-OEF, but also for other target scenarios such as campus, operator and data center networks.

BIBLIOGRAPHY

[1] "New Generation Network Architecture - AKARI Conceptual Design, AKARI project," 2007, http://www.nict.go.jp/photonic_nw/archi/akari/4otfsk00000cb3t3-att/4otfsk00000cb4g1.pdf.

[2] J. Pan, S. Paul, and R. Jain, "A survey of the research on future internet architectures," *Communications Magazine, IEEE*, vol. 49, no. 7, pp. 26–36, 2011.

[3] "Future Internet Assembly," 2011, http://www.future-internet.eu/uploads/media/FIArchCurrentInternetLimitationsMarch2011FINAL.pdf.

[4] H. Harai, "Designing new-generation network - overview of AKARI Architecture Design," in *Communications and Photonics Conference and Exhibition (ACP), 2009 Asia*, nov. 2009, pp. 1 –2.

[5] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "GENI: A federated testbed for innovative network experiments," *Computer Networks*, vol. 61, pp. 5–23, 2014.

[6] A. Gavras, A. Karila, S. Fdida, M. May, and M. Potts, "Future internet research and experimentation: the FIRE initiative," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 3, pp. 89–92, 2007.

[7] "IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks," *IEEE Std 802.1Q-2014*, 2014.

[8] E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology," RFC 7426 (Informational), Internet Engineering Task Force, Jan. 2015. [Online]. Available: http://www.ietf.org/rfc/rfc7426.txt

[9] "The OpenFlow Switch Specification, Version 1.4.0," 2015, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf.

[10] A. Doria, J. H. Salim, R. Haas, H. Khosravi, W. Wang, L. Dong, R. Gopal, and J. Halpern, "Forwarding and Control Element Separation (ForCES) Protocol Specification," RFC 5810 (Proposed Standard), Internet Engineering Task Force, Mar. 2010, updated by RFCs 7121, 7391. [Online]. Available: http://www.ietf.org/rfc/rfc5810.txt

[11] N. Bahadur, S. Kini, and J. Medved, "Routing Information Base Info Model," Active Draft, IETF Secretariat, Internet-Draft draft-ietf-i2rs-rib-info-model-08, 2015.

[12] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[13] N. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *Communications Magazine, IEEE*, vol. 47, no. 7, pp. 20–26, 2009.

[14] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined networks," *Internet Computing, IEEE*, vol. 17, no. 2, pp. 20–27, 2013.

[15] M. Suñé, L. Bergesio, H. Woesner, T. Rothe, A. Köpsel, D. Colle, B. Puype, D. Simeonidou, R. Nejabati, M. Channegowda *et al.*, "Design and implementation of the OFELIA FP7 facility: The European OpenFlow testbed," *Computer Networks*, vol. 61, pp. 132–150, 2014.

[16] "ETSI GS NFV 002: Network Functions Virtualisation (NFV); Architectural Framework," *ETSI ISG for NFV*, 2014.

[17] G. Xilouris, E. Trouva, F. Lobillo, J. M. Soares, J. Carapinha, M. McGrath, G. Gardikis, P. Paglierani, E. Pallis, L. Zuccaro *et al.*, "T-NOVA: a marketplace for virtualized

network functions," in *Networks and Communications (EuCNC), 2014 European Conference on*. IEEE, 2014, pp. 1–5.

[18] P. Skoldstrom, B. Sonkoly, A. Gulyás, F. Németh, M. Kind, F.-J. Westphal, W. John, J. Garay, E. Jacob, D. Jocha *et al.*, "Towards unified programmability of cloud and carrier infrastructure," in *Software Defined Networks (EWSDN), 2014 Third European Workshop on*. IEEE, 2014, pp. 55–60.

[19] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: a scalable fault-tolerant layer 2 data center network fabric," in *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4. ACM, 2009, pp. 39–50.

[20] A. T. Campbell, H. G. De Meer, M. E. Kounavis, K. Miki, J. B. Vicente, and D. Villela, "A survey of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 2, pp. 7–23, 1999.

[21] M. Kobayashi, S. Seetharaman, G. Parulkar, G. Appenzeller, J. Little, J. Van Reijendam, P. Weissmann, and N. McKeown, "Maturing of OpenFlow and Software-defined Networking through deployments," *Computer Networks*, vol. 61, pp. 151–175, 2014.

[22] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, T. Turletti *et al.*, "A survey of software-defined networking: Past, present, and future of programmable networks," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 3, pp. 1617–1634, 2014.

[23] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.

[24] D. Kreutz, F. M. Ramos, P. Esteves Verissimo, C. Esteve Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[25] Y. Jarraya, T. Madi, and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 4, pp. 1955–1980, 2014.

[26] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862–876, 2010.

[27] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani, "Data center network virtualization: A survey," *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 2, pp. 909–928, 2013.

[28] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: a survey," *Communications Magazine, IEEE*, vol. 51, no. 11, pp. 24–31, 2013.

[29] J. Jofre, C. Velayos, G. Landi, M. Giertych, A. C. Hume, G. Francis, and A. V. Oton, "Federation of the BonFIRE multi-cloud infrastructure with networking facilities," *Computer Networks*, vol. 61, pp. 184–196, 2014.

[30] J. P. Sterbenz, D. Hutchison, P. Müller, and C. Elliott, "Special issue on Future Internet Testbeds - Guest Editorial," *Computer Networks*, 2014.

[31] D. Schwerdel, B. Reuther, T. Zinner, P. Müller, and P. Tran-Gia, "Future Internet research and experimentation: The G-Lab approach," *Computer Networks*, vol. 61, pp. 102–117, 2014.

[32] J. Mambretti, J. Chen, and F. Yeh, "Creating environments for innovation: Designing and implementing advanced experimental network research testbeds based on the Global Lambda Integrated Facility and the StarLight Exchange," *Computer Networks*, vol. 61, pp. 118–131, 2014.

[33] M. Campanella and F. Farina, "The FEDERICA infrastructure and experience," *Computer Networks*, vol. 61, pp. 176–183, 2014.

[34] B. Belter, J. R. Martinez, J. I. Aznar, J. F. Riera, L. M. Contreras, M. Antoniak-Lewandowska, M. Biancani, J. Buysse, C. Develder, Y. Demchenko *et al.*, “The GEYSERS optical testbed: A platform for the integration, validation and demonstration of cloud-based infrastructure services,” *Computer Networks*, vol. 61, pp. 197–216, 2014.

[35] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis *et al.*, “SmartSantander: IoT experimentation over a smart city testbed,” *Computer Networks*, vol. 61, pp. 217–238, 2014.

[36] N. Bastin, A. Bavier, J. Blaine, J. Chen, N. Krishnan, J. Mambretti, R. McGeer, R. Ricci, and N. Watts, “The InstaGENI initiative: An architecture for distributed systems and advanced programmable networks,” *Computer Networks*, vol. 61, pp. 24–38, 2014.

[37] D. Kim, J. Kim, G. Wang, J.-H. Park, and S.-H. Kim, “K-GENI testbed deployment and federated meta operations experiment over GENI and KREONET,” *Computer Networks*, vol. 61, pp. 39–50, 2014.

[38] D. Medhi, B. Ramamurthy, C. Scoglio, J. P. Rohrer, E. K. Çetinkaya, R. Cherukuri, X. Liu, P. Angu, A. Bavier, C. Buffington *et al.*, “The GpENI testbed: network infrastructure, implementation experience, and experimentation,” *Computer Networks*, vol. 61, pp. 51–74, 2014.

[39] A. Sydney, D. S. Ochs, C. Scoglio, D. Gruenbacher, and R. Miller, “Using GENI for experimental evaluation of Software Defined Networking in smart grids,” *Computer Networks*, vol. 63, pp. 5–16, 2014.

[40] J. Griffioen, Z. Fei, H. Nasir, X. Wu, J. Reed, and C. Carpenter, “Measuring experiments in GENI,” *Computer Networks*, vol. 63, pp. 17–32, 2014.

[41] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, “Planetlab: an overlay testbed for broad-coverage services,” *ACM SIGCOMM*

Computer Communication Review, vol. 33, no. 3, pp. 3–12, 2003.

[42] S. Keranidis, D. Giatsios, T. Korakis, I. Koutsopoulos, L. Tassiulas, T. Rakotoarivelo, M. Ott, and T. Parmentelat, "Experimentation on end-to-end performance aware algorithms in the federated environment of the heterogeneous PlanetLab and NITOS testbeds," *Computer Networks*, vol. 63, pp. 48–67, 2014.

[43] A. Gluhak, S. Krco, M. Nati, D. Pfisterer, N. Mitton, and T. Razafindralambo, "A survey on facilities for experimental internet of things research," *Communications Magazine, IEEE*, vol. 49, no. 11, pp. 58–67, 2011.

[44] G. Werner-Allen, P. Swieskowski, and M. Welsh, "Motelab: A wireless sensor network testbed," in *Proceedings of the 4th international symposium on Information processing in sensor networks*. IEEE Press, 2005, p. 68.

[45] Y. Demchenko, J. Van der Ham, V. Yakovenko, C. De Laat, M. Ghijsen, and M. Cristea, "On-demand provisioning of Cloud and Grid based infrastructure services for collaborative projects and groups," in *Collaboration Technologies and Systems (CTS), 2011 International Conference on*. IEEE, 2011, pp. 134–142.

[46] P. Papadimitriou, O. Maennel, A. Greenhalgh, A. Feldmann, and L. Mathy, "Implementing network virtualization for a future internet," in *20th ITC Specialist Seminar on Network Virtualization-Concept and Performance Aspects*, 2009.

[47] P. Sköldström and K. Yedavalli, "Network virtualization and resource allocation in openflow-based wide area networks," in *Communications (ICC), 2012 IEEE International Conference on*. IEEE, 2012, pp. 6622–6626.

[48] T. Narten, E. Gray, D. Black, L. Fang, L. Kreeger, and M. Napierala, "Problem Statement: Overlays for Network Virtualization," RFC 7364 (Informational), Internet Engineering Task Force, Oct. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7364.txt

[49] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks," RFC 7348 (Informational), Internet Engineering Task Force, Aug. 2014. [Online]. Available: http://www.ietf.org/rfc/rfc7348.txt

[50] P. Garg and Y. Wang, "NVGRE: Network Virtualization Using Generic Routing Encapsulation," RFC 7637 (Informational), Internet Engineering Task Force, Sep. 2015. [Online]. Available: http://www.ietf.org/rfc/rfc7637.txt

[51] B. Davie and J. Gross, "A stateless transport tunneling protocol for network virtualization (STT)," Expired Draft, IETF Secretariat, Internet-Draft draft-davie-stt-04, 2014.

[52] J. Gross, T. Sridhar, P. Garg, C. Wright, I. Ganga, P. Agarwal, K. Duda, D. Dutt, and J. Hudson, "Geneve: Generic Network Virtualization Encapsulation," Expired Draft, IETF Secretariat, Internet-Draft draft-ietf-nvo3-geneve-00, May 2015.

[53] R. Kawashima, S. Muramatsu, H. Nakayama, T. Hayashi, and H. Matsuo, "SCLP: Segment-oriented Connection-less Protocol for high-performance software tunneling in datacenter networks," in *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*. IEEE, 2015, pp. 1–8.

[54] A. Tootoonchian and Y. Ganjali, "HyperFlow: A distributed control plane for OpenFlow," in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*. USENIX Association, 2010, pp. 3–3.

[55] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, "Towards an elastic distributed SDN controller," in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 7–12.

[56] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "ONOS: towards an open, distributed SDN OS," in

Proceedings of the third workshop on Hot topics in software defined networking. ACM, 2014, pp. 1–6.

[57] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, “Flowvisor: A network virtualization layer,” *OpenFlow Switch Consortium, Tech. Rep*, 2009.

[58] R. Kandoi and M. Antikainen, “Denial-of-service attacks in OpenFlow SDN networks,” in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*. IEEE, 2015, pp. 1322–1326.

[59] L. Andersson and T. Madsen, “Provider Provisioned Virtual Private Network (VPN) Terminology,” RFC 4026 (Informational), Internet Engineering Task Force, Mar. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4026.txt

[60] E. Rosen and Y. Rekhter, “BGP/MPLS IP Virtual Private Networks (VPNs),” RFC 4364 (Proposed Standard), Internet Engineering Task Force, Feb. 2006, updated by RFCs 4577, 4684, 5462. [Online]. Available: http://www.ietf.org/rfc/rfc4364.txt

[61] K. Wallace, *CCNP Routing and Switching ROUTE 300-101 Official Cert Guide*. Cisco Press, 2014.

[62] “Cisco, Cisco Active Network Abstraction 3.7.2 Reference Guide,” 2015, http://www.cisco.com/c/en/us/td/docs/net_mgmt/active_network_abstraction/3-7-2/reference/guide/ANA372RefGuide/vrf.html.

[63] “Cisco, Easy Virtual Network: Simplifying Layer 3 Network Virtualization,” 2015, http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/layer-3-vpns-l3vpn/whitepaper_c11-638769.pdf.

[64] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, “In VINI veritas: realistic and controlled network experimentation,” in *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4. ACM, 2006, pp. 3–14.

[65] M. Huang, “VNET: PlanetLab virtualized network access,” Tech. Rep. PDN–05–029, PlanetLab Consortium, Tech. Rep., 2005.

[66] “NEC ProgrammableFlow: Redefining Cloud Network Virtualization with OpenFlow (NEC white paper),” 2015, http://www.necam.com/sdn/doc.cfm?t=WhitePapers.

[67] L. Andersson and E. Rosen, “Framework for Layer 2 Virtual Private Networks (L2VPNs),” RFC 4664 (Informational), Internet Engineering Task Force, Sep. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4664.txt

[68] R. Callon and M. Suzuki, “A Framework for Layer 3 Provider-Provisioned Virtual Private Networks (PPVPNs),” RFC 4110 (Informational), Internet Engineering Task Force, Jul. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4110.txt

[69] T. Dierks and E. Rescorla, “The Transport Layer Security (TLS) Protocol Version 1.2,” RFC 5246 (Proposed Standard), Internet Engineering Task Force, Aug. 2008, updated by RFCs 5746, 5878, 6176, 7465, 7507, 7568, 7627, 7685. [Online]. Available: http://www.ietf.org/rfc/rfc5246.txt

[70] “VMware NSX Network Virtualization Platform,” 2015, https://www.vmware.com/files/pdf/products/nsx/VMware-NSX-Network-Virtualization-Platform-WP.pdf.

[71] J. Mudigonda, P. Yalagandula, J. Mogul, B. Stiekes, and Y. Pouffary, “NetLord: a scalable multi-tenant network architecture for virtualized datacenters,” in *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4. ACM, 2011, pp. 62–73.

[72] A. Edwards, A. Fischer, and A. Lain, “Diverter: a new approach to networking within virtualized infrastructures,” in *Proceedings of the 1st ACM workshop on Research on enterprise networking*. ACM, 2009, pp. 103–110.

[73] P. Knight and C. Lewis, “Layer 2 and 3 virtual private networks: taxonomy, technology, and standardization

efforts," *Communications Magazine, IEEE*, vol. 42, no. 6, pp. 124–131, 2004.

[74] T. Takeda, "Framework and Requirements for Layer 1 Virtual Private Networks," RFC 4847 (Informational), Internet Engineering Task Force, Apr. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4847.txt

[75] S. Frankel, P. Hoffman, A. Orebaugh, and R. Park, "Guide to SSL VPNs," *NIST Special Publication*, vol. 800, p. 113, 2008.

[76] S. Kent, "IP Encapsulating Security Payload (ESP)," RFC 4303 (Proposed Standard), Internet Engineering Task Force, Dec. 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4303.txt

[77] J. Lau, M. Townsley, and I. Goyret, "Layer Two Tunneling Protocol - Version 3 (L2TPv3)," RFC 3931 (Proposed Standard), Internet Engineering Task Force, Mar. 2005, updated by RFC 5641. [Online]. Available: http://www.ietf.org/rfc/rfc3931.txt

[78] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, and G. Zorn, "Point-to-Point Tunneling Protocol (PPTP)," RFC 2637 (Informational), Internet Engineering Task Force, Jul. 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2637.txt

[79] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina, "Generic Routing Encapsulation (GRE)," RFC 2784 (Proposed Standard), Internet Engineering Task Force, Mar. 2000, updated by RFC 2890. [Online]. Available: http://www.ietf.org/rfc/rfc2784.txt

[80] J. Mudigonda, P. Yalagandula, M. Al-Fares, and J. C. Mogul, "SPAIN: COTS Data-Center Ethernet for Multipathing over Arbitrary Topologies," in *NSDI*, 2010, pp. 265–280.

[81] M. Casado, T. Garfinkel, A. Akella, M. J. Freedman, D. Boneh, N. McKeown, and S. Shenker, "SANE: A Protection Architecture for Enterprise Networks," in *Usenix Security*, 2006.

[82] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, “Ethane: Taking control of the enterprise,” in *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4. ACM, 2007, pp. 1–12.

[83] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. M. Parulkar, “Can the production network be the testbed?” in *OSDI*, vol. 10, 2010, pp. 1–6.

[84] R. Doriguzzi Corin, M. Gerola, R. Riggio, F. De Pellegrini, and E. Salvadori, “Vertigo: Network virtualization and beyond,” in *Software Defined Networking (EWSDN), 2012 European Workshop on*. IEEE, 2012, pp. 24–29.

[85] E. Salvadori, R. Doriguzzi Corin, A. Broglio, and M. Gerola, “Generalizing virtual network topologies in OpenFlow-based networks,” in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*. IEEE, 2011, pp. 1–6.

[86] A. Al-Shabibi, M. De Leenheer, M. Gerola, A. Koshibe, G. Parulkar, E. Salvadori, and B. Snow, “OpenVirteX: Make your virtual SDNs programmable,” in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 25–30.

[87] J. E. van der Merwe and I. M. Leslie, “Switchlets and dynamic virtual ATM networks,” in *Integrated Network Management V*. Springer, 1997, pp. 355–368.

[88] J. E. Van der Merwe, S. Rooney, I. Leslie, and S. Crosby, “The Tempest-a practical framework for network programmability,” *Network, IEEE*, vol. 12, no. 3, pp. 20–28, 1998.

[89] L. Peterson, A. Bavier, M. E. Fiuczynski, and S. Muir, “Experiences building PlanetLab,” in *Proceedings of the 7th symposium on Operating systems design and implementation*, ser. OSDI ’06. Berkeley, CA, USA: USENIX Association, 2006, pp. 351–366. [Online]. Available: http://dl.acm.org/citation.cfm?id=1298455.1298489

[90] K. Park and V. S. Pai, "CoMon: a mostly-scalable monitoring system for PlanetLab," *ACM SIGOPS Operating Systems Review*, vol. 40, no. 1, pp. 65–74, 2006.

[91] G. D. Stasi, R. Bifulco, F. P. D'Elia, S. Avallone, R. Canonico, A. Apostolaras, N. Giallelis, T. Korakis, and L. Tassiulas, "Experimenting with P2P traffic optimization for wireless mesh networks in a federated OMF-PlanetLab environment," in *Wireless Communications and Networking Conference (WCNC), 2011 IEEE*. IEEE, 2011, pp. 719–724.

[92] P. Szegedi, S. Figuerola, M. Campanella, V. Maglaris, and C. Cervelló-Pastor, "With Evolution for Revolution: Managing FEDERICA for Future Internet Research," *Comm. Mag.*, vol. 47, no. 7, pp. 34–39, Jul. 2009. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2009.5183470

[93] P. Szegedi, J. F. Riera, J. García-Espín, M. Hidell, P. Sjödin, P. Söderman, M. Ruffini, D. O'Mahony, A. Bianco, L. Giraudo *et al.*, "Enabling future internet research: the FEDERICA case," *Communications Magazine, IEEE*, vol. 49, no. 7, pp. 54–61, 2011.

[94] J. Sobieski, F. Farina, S. Naegele-Jackson, K. Kramaric, B. Pietrzak, and M. Hazlinsky, "GÉANT Testbed Service External Domain Ports: A Demo on Multiple Domain Connectivity," in *Software Defined Networks (EWSDN), 2015 Fourth European Workshop on*. IEEE, 2015, pp. 113–114.

[95] "GÉANT, GÉANT OpenFlow Facility Design," 2015, http://geant3.archive.geant.net/Media_Centre/Media_Library/Media%20Library/GN3-13-003_DJ1-2-1_Technology-Investigation-of-OpenFlow-and-Testing.pdf.

[96] T. Salmito, L. Ciuffo, I. Machado, M. Salvador, M. Stanton, N. Rodriguez, A. Abelem, L. Bergesio, S. Sallent, and L. Baron, "FIBRE-An International Testbed for Future Internet Experimentation," in *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos-SBRC 2014*, 2014, pp. p–969.

[97] G. Carrozzo, R. Monno, B. Belter, R. Krzywania, K. Pentikousis, M. Broadbent, T. Kudoh, A. Takefusa, A. Vieo-Oton, C. Fernandez *et al.*, "Large-scale SDN experiments in federated environments," in *Smart Communications in Network Technologies (SaCoNeT), 2014 International Conference on*. IEEE, 2014, pp. 1–6.

[98] "The FELIX architecture for large scale SDN and NSI experiments (FELIX white paper)," 2015, http://www.ict-felix.eu/wp-content/uploads/2015/04/FELIX-whitepaper-architecture-final.pdf.

[99] K. Shiomoto, I. Inoue, and E. Oki, "Network virtualization in high-speed huge-bandwidth optical circuit switching network," in *INFOCOM Workshops 2008, IEEE*. IEEE, 2008, pp. 1–6.

[100] R. Nejabati, E. Escalona, S. Peng, and D. Simeonidou, "Optical network virtualization," in *Optical Network Design and Modeling (ONDM), 2011 15th International Conference on*. IEEE, 2011, pp. 1–5.

[101] K. Kannan and S. Banerjee, "Compact TCAM: Flow entry compaction in TCAM for power aware SDN," in *Distributed Computing and Networking*. Springer, 2013, pp. 439–444.

[102] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Can the Production Network be the Testbed?" in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, ser. OSDI'10. USENIX Association, 2010, pp. 1–6. [Online]. Available: http://dl.acm.org/citation.cfm?id=1924943.1924969

[103] "Big Virtual Switch and OpenStack, Big Switch Networks," 2015, http://www.bigswitch.com/sites/default/files/sdn_resources/openstack_aag.pdf.

[104] "IEEE Standard for Ethernet," *IEEE 802.3-2012*, 2012.

[105] C. Rotsos, N. Sarrar, S. Uhlig, R. Sherwood, and A. W. Moore, "OFLOPS: An Open Framework for

OpenFlow Switch Evaluation," in *Proceedings of the 13th international conference on Passive and Active Measurement*, ser. PAM'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 85–95. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-28537-0_9

[106] J.-W. Hu, W.-Y. Huang, H.-M. Tseng, H.-L. Lee, L.-C. Ku, S.-C. Lin, T.-L. Liu, and C.-S. Yang, "Future Internet in Taiwan: Design and Research Activities over TWAREN Network," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*. IEEE, 2012, pp. 329–333.

[107] W.-Y. Huang, J.-W. Hu, S.-C. Lin, T.-L. Liu, P.-W. Tsai, C.-S. Yang, F. I. Yeh, J. H. Chen, and J. J. Mambretti, "Design and Implementation of an Automatic Network Topology Discovery System for the Future Internet Across Different Domains," in *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*. IEEE, 2012, pp. 903–908.

[108] P. Kutch, "PCI-SIG SR-IOV primer: An introduction to SR-IOV technology," *Intel application note*, pp. 321 211–002, 2011.

[109] N. Chowdhury, F.-E. Zaheer, and R. Boutaba, "iMark: An identity management framework for network virtualization environment," in *Integrated Network Management, 2009. IM'09. IFIP/IEEE International Symposium on*. IEEE, 2009, pp. 335–342.

[110] "IEEE Organizationally Unique Identifier (OUI)," 2015, http://standardsoui.ieee.org/oui.txt.

[111] "Metro Ethernet Forum (MEF) Technical Specifications," 2015, https://www.mef.net/carrier-ethernet/technical-specifications.

[112] R. Droms, "Dynamic Host Configuration Protocol," RFC 2131 (Draft Standard), Internet Engineering Task Force, Mar. 1997, updated by RFCs 3396, 4361, 5494, 6842. [Online]. Available: http://www.ietf.org/rfc/rfc2131.txt

[113] "IEEE Standards for Local Area Networks: Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications," *IEEE 802.3-1985*, 1985.

[114] "VMware vSphere 6.0, MAC address management," 2015, https://pubs.vmware.com/vsphere-60/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-601-networking-guide.pdf.

[115] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," RFC 3315 (Proposed Standard), Internet Engineering Task Force, Jul. 2003, updated by RFCs 4361, 5494, 6221, 6422, 6644, 7083, 7227, 7283, 7550. [Online]. Available: http://www.ietf.org/rfc/rfc3315.txt

[116] S. Thomson, T. Narten, and T. Jinmei, "IPv6 Stateless Address Autoconfiguration," RFC 4862 (Draft Standard), Internet Engineering Task Force, Sep. 2007, updated by RFC 7527. [Online]. Available: http://www.ietf.org/rfc/rfc4862.txt

[117] K. Benton, L. J. Camp, and C. Small, "Openflow vulnerability assessment," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 151–152.

[118] "IEEE Standard for Local and metropolitan area networks - Port-Based Network Access Control," *IEEE 802.1X-2010*, 2010.

[119] "eXtensible Access Control Markup Language (XACML) Version 3.0. 22 January 2013," *OASIS Standard*, 2013.

[120] P. Congdon, M. Sanchez, and B. Aboba, "RADIUS Attributes for Virtual LAN and Priority Support," RFC 4675 (Proposed Standard), Internet Engineering Task Force, Sep. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4675.txt

[121] ——, "RADIUS Filter Rule Attribute," RFC 4849 (Proposed Standard), Internet Engineering Task Force, Apr. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4849.txt

[122] A. K. Nayak, A. Reimers, N. Feamster, and R. Clark, “Resonance: dynamic access control for enterprise networks,” in *Proceedings of the 1st ACM workshop on Research on enterprise networking*. ACM, 2009, pp. 11–18.

[123] Y. Yamasaki, Y. Miyamoto, J. Yamato, H. Goto, and H. Sone, “Flexible access management system for campus VLAN based on OpenFlow,” in *Applications and the Internet (SAINT), 2011 IEEE/IPSJ 11th International Symposium on*. IEEE, 2011, pp. 347–351.

[124] S. Kinoshita, T. Watanabe, J. Yamato, H. Goto, and H. Sone, “Implementation and evaluation of an OpenFlow-based access control system for wireless LAN roaming,” in *Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual*. IEEE, 2012, pp. 82–87.

[125] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, and D. Spence, “AAA Authorization Framework,” RFC 2904 (Informational), Internet Engineering Task Force, Aug. 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2904.txt

[126] B. Aboba and M. Beadles, “The Network Access Identifier,” RFC 2486 (Proposed Standard), Internet Engineering Task Force, Jan. 1999, obsoleted by RFC 4282. [Online]. Available: http://www.ietf.org/rfc/rfc2486.txt

[127] D. Ferraiolo, J. Cugini, and D. R. Kuhn, “Role-based access control (RBAC): Features and motivations,” in *Proceedings of 11th annual computer security application conference*, 1995, pp. 241–48.

[128] E. Yuan and J. Tong, “Attributed based access control (ABAC) for web services,” in *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*. IEEE, 2005.

[129] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, “Nox: Towards an Operating System for Networks,” *SIGCOMM Comput.*

Commun. Rev., vol. 38, no. 3, pp. 105–110, Jul. 2008. [Online]. Available: http://doi.acm.org/10.1145/1384609.1384625

[130] C. Argyropoulos, D. Kalogeras, G. Androulidakis, and V. Maglaris, “PaFloMon – A Slice Aware Passive Flow Monitoring Framework for OpenFlow Enabled Experimental Facilities,” in *Software Defined Networking (EWSDN), 2012 European Workshop on*, 2012, pp. 97–102.

[131] “Spirent, How to test 10 Gigabit Ethernet performance (white paper),” 2012, http://www.spirent.com/~/media/White%20Papers/Broadband/PAB/How_to_Test_10G_Ethernet_WhitePaper.pdf.

[132] S. Bradner, “Benchmarking Terminology for Network Interconnection Devices,” RFC 1242 (Informational), Internet Engineering Task Force, Jul. 1991, updated by RFC 6201. [Online]. Available: http://www.ietf.org/rfc/rfc1242.txt

[133] “Intel Open Network Platform Server Release 1.5 Performance Test Report,” 2015, https://download.01.org/packet-processing/ONPS1.5/Intel_ONP_Server_Release_1.5_Performance_Test_Report_Rev1.2.pdf.

[134] M. H. Mazlan, S. H. S. Ariffin, M. Balfaqih, S. N. M. Hasnan, and S. Haseeb, “Latency evaluation of authentication protocols in centralized 802.11 architecture,” 2012.

[135] A. S. Tanenbaum and M. Van Steen, *Distributed Systems*. Prentice-Hall, 2007.

[136] “IEEE Standard for Local and metropolitan area networks - Media Access Control (MAC) Security,” *IEEE Std 802.1AE-2006*, 2006.

[137] T. C. Clancy, “Secure handover in enterprise WLANs: CAPWAP, HOKEY, and IEEE 802.11 r,” *Wireless Communications, IEEE*, vol. 15, no. 5, pp. 80–85, 2008.