Departamento de Comunicaciones

Universitat Politècnica de València

Especificación e Implementación de un Sistema de Red Definida por Software con Funciones Virtuales Adaptadas a Despliegues de Internet de las Cosas

Tesis Doctoral

Programa de Doctorado en Telecomunicación

Autor:

Jara Suárez de Puga García

Director:

Carlos Enrique Palau Salvador

Valencia, Octubre 2021

Para todos aquellos que me han ayudado a llegar donde estoy.
Y, en especial, a mis padres.

Agradecimientos

En primer lugar, debo dar mi más sincero agradecimiento a mi tutor Carlos Enrique Palau Salvador, gracias por esta oportunidad de participar en proyectos tan innovadores y emocionantes, y por apoyarme a lo largo de este trabajo.

Me gustaría agradecer también a mis compañeros del grupo de investigación SATRD, por el maravilloso equipo que formamos y por hacer del día a día una nueva oportunidad de aprender. Al equipo de los proyectos INTER-IoT y 5GENESIS, en el cual he tenido la oportunidad de conocer a gente brillante con ideas increíbles.

Con mucho cariño, quiero dedicar mi más profundo agradecimiento a mi familia. A mis padres Teresa y César, y a mis hermanos, por apoyarme, animarme y por darme esos buenos consejos en el momento adecuado.

También, a mis amigos, tanto los de siempre como los que han surgido a lo largo de esta etapa.

Y, por último, a mi compañero de batalla, Eneko, que ha luchado codo con codo a mi lado para que esto saliera adelante.

Resumen

La complejidad en la gestión de las redes de comunicación tradicionales, así como su poca escalabilidad y flexibilidad, supone un obstáculo para el desarrollo y consolidación de nuevas tecnologías emergentes como es el caso del Internet de las Cosas (*Internet of Things*), dónde la facilidad para el intercambio y manejo de grandes volúmenes de datos heterogéneos procedentes de sensores es un requisito clave para el correcto funcionamiento del sistema. El Internet de las Cosas se define cómo la interconexión digital de objetos cotidianos dotados de inteligencia (*Smart devices*) a través de redes de comunicación de datos ya sean públicas (*Internet*) o privadas. Sin embargo, el Internet de las Cosas no sólo está compuesto por estos dispositivos, toda la infraestructura, plataformas, aplicaciones y servicios que ayudan a los datos a viajar desde los dispositivos origen y hacia sus diferentes destinos, y la gestión de estos también forman parte del denominado Internet de las Cosas. El almacenamiento, análisis, procesado y gestión masiva de dichos datos es lo que se denomina *Big Data*, y está compuesto de grandes cantidades de datos (*massive data*) estructurados en diferentes formatos, modelos de datos y protocolos, lo que dificulta su tratamiento y su intercambio a través de las redes de datos convencionales. Ante esta problemática la implementación de redes virtuales definidas por software se presenta como una posible solución para dotar de flexibilidad,

escalabilidad y sencillez de gestión a las redes que interconectan estos dispositivos, plataformas y otros elementos IoT, permitiendo una visión global, una gestión centralizada y un desarrollo de servicios a nivel de red específicos para los entornos de Internet de las Cosas. Este proyecto se presenta como una aproximación de estas dos tecnologías y tendrá como objetivo el diseño de una solución donde probar las herramientas de control de redes definidas por software o programables (SDN) y las funciones virtuales de redes (NFV) aplicadas a despliegues de Internet de las Cosas (IoT) de forma que se puedan demostrar sus ventajas e implicaciones y se puedan descubrir nuevas líneas de desarrollo sobre esta base.

Resum

La complexitat en la gestió de les xarxes de comunicació tradicionals, així com la seua poca escalabilitat i flexibilitat, suposa un obstacle per al desenvolupament i consolidació de noves tecnologies emergents com és el cas de la Internet de les Coses (Internet of Things), on la facilitat per a l'intercanvi i maneig de grans volums de dades heterogènies procedents de sensors és un requisit clau per al correcte funcionament del sistema. La Internet de les Coses es defineix com la interconnexió digital d'objectes quotidians dotats d'intel·ligència (Smart devices) a través de xarxes de comunicació de dades ja siguen públiques (Internet) o privades. No obstant això, la Internet de les Coses no sols està compost per aquests dispositius, tota la infraestructura, plataformes, aplicacions i serveis que ajuden les dades a viatjar des dels dispositius d'origen i cap a les seues diferents destinacions, i la gestió d'aquests també formen part de la denominada Internet de les Coses. L'emmagatzematge, anàlisi, processament i gestió massiva d'aquestes dades és el que es denomina Big Data, i està compost de grans quantitats de dades (massive data) estructurats en diferents formats, models de dades i protocols, la qual cosa dificulta el seu tractament i el seu intercanvi a través de les xarxes de dades convencionals. Davant aquesta problemàtica la implementació de xarxes virtuals definides per software es presenta com una possible solució per a dotar de flexibilitat, escalabilitat i senzillesa de gestió a

les xarxes que interconnecten aquests dispositius, plataformes i altres elements IoT, permetent una visió global, una gestió centralitzada i un desenvolupament de serveis a nivell de xarxa específics per als entorns d'Internet de les Coses. Aquest projecte es presenta com una aproximació d'aquestes dues tecnologies i tindrà com a objectiu el disseny d'una solució on provar les eines de control de xarxes definides per software o programables (SDN) i les funcions virtuals de xarxes (NFV) aplicades a desplegaments d'Internet de les Coses (IoT) de manera que es puguen demostrar els seus avantatges i implicacions, i es puguen descobrir noves línies de desenvolupament sobre aquesta base.

Abstract

Nowadays, the complexity of traditional network administration, together with the lack of scalability and flexibility, has been a challenge for the proper development and integration of new emerging technologies which make use of this network. As an example, we have the so-called Internet of Things (IoT). The principal IoT network requirement that enables the growth of this paradigm is the need to facilitate high data volume exchange and administration, from very heterogeneous sources. The IoT concept is defined as the digital interconnection of daily objects endowed with more '*intelligence*' (*Smart devices*) through a data communication network either public (*Internet*) or private. However, this technological trend does not only depend on the '*smart devices*', but on the whole infrastructure, platforms, frameworks, services, and applications that helps data to travel from the source devices to their different destinations. Also, the handling of the massive volumes of data extracted from those smart devices, their storage, processing, and analysis, known as Big Data, is a key part of this paradigm. This data is gathered from very different sources, and hence, it has diverse data structures and formats. Moreover, it is exchanged using various network protocols (LoRa, CoAp, etc.) which hinder its management and communication through conventional networks, that were not created for such data traffic. Given this problem, several technological approaches

have emerged to solve it. Virtual software-defined networking is presented as a possible solution to provide flexibility, scalability, and simplicity of management to the networks that interconnect these devices, platforms, services, and other IoT elements. The virtualization of the network infrastructure, includes an extra layer of abstraction, thus providing a holistic vision of the network and centralizing the administration of its elements and the development of specific network services for IoT deployments. This project is presented as an approximation of these two technological paradigms and will have as the main objective the design of an architectural blueprint and testbed were testing the control tools of software-defined networks (SDN) and the virtualized network functions (NFV) applied to IoT deployments. Thereby, its advantages and implications can be evaluated, and new lines of development can be discovered on this base.

Tabla de contenidos

Lista de figuras

Lista de tablas

Acrónimos

3GPP	3rd Generation Partnership Project
5G	5th Generation
5G NR	5th Generation New Radio
6LowPAN	IPv6 over Low power Wireless Personal Area Networks
ACK	Acknowledgement
AIOTI	Alliance for the Internet of Things Innovation
API	Application Programming Interface
AR/VR	Augmented Reality/Virtual Reality
ASICs	Application-Apecific Integrated Circuits
BSS	Business Support System
BW	Bandwidth
CapEx	Capital expenditure
CEP	Context Event Processor
CLI	Command Line Interface
COAP	Constrained Application Protocol
COTS	Commercial Off-The-Shelf
DL	DownLink
DSCP	Differentiated Services Code Point
E2E	End to end
ELCM	Experiment Life Cycle Manager
EMBB	Enhanced Mobile BroadBand
eNB	Evolved NodeBase

EPC	Evolved Packet Core
ETSI	European Telecommunications Standards Institute
gNB	Next Generation Node Base
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
INTER-FW	INTER Framework
IoT	Internet of Things
IPSM	Inter Platform Semantic Mediator
ITU	International Telecommunication Union
JSON-RPC	Remote Procedure Call JSON Encoded
JVM	Java Virtual Machine
KPI	Key Performance Indicators
KVM	Kernel-based Virtual Machine
LAN	Local Area Network
LBO	Local Break-Out
LoRa	Long Range
LoS	Line-of-Sight
LPWAN	Low Power Wireless Area Network
LTE	Long Term Evolution
LTE-IoT	Long Term Evolution-Iot
M2M	Machine to Machine
MAN	Metropolitan Area Network
MANO	Management and Orchestration
MEC	Multi-access Edge Computing

MIMO	Multiple Input Multiple Output
mMTC	Massive Machine Type Communications
MPLS	Multiprotocol Label Switching
MQTT	Message Queuing Telemetry Transport
N2N	Network-to-Network
NAT	Network Address Translation
NB-IoT	Narrowband-Iot
NFV	Network Function Virtualization
NFV-MANO	Network Function Virtualization Management and Orchestration
NFVI	Network Function Virtualization Infrastructure
NOS	Network Operative System
NS	Network Slice
ODL	OpenDayLight
OF	OpenFlow
ONF	Open Network Fundation
OpEx	Operative expenditure
OSGI	Open Services Gateway initiative
OSI	Open Systems Interconnection
OSM	Open Source MANO
OSS	Operations Support System
OVS	OpenVSwitch
OVSDB	OpenvSwitch Database Management Protocol
PaaS	Platform as a Service
QoS	Quality of Service
RAN	Radio Access Network
RAT	Radio Access Technologies
REST	Representational State Transfer
RFID	Radio Frequency Identification
RRM	Radio Resource Management

RTT	Round Trip Time
SaaS	Software as a Service
SDK	Software Development Kit
SDN	Software Defined Networking
SDO	Standards Developing Organizations
SDR	Software Defined Radio
SF	Spreading Factor
SISO	Single Input Single Output
TCAM	Ternary-Content-Addressable Memory
TTM	Time To Market
UE	User Equipment
UHD	Ultra-High Definition
UL	UpLink
URLLC	Ultra Reliable and Low-Latency Communications
VIM	Virtual Infraestructure Manager
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNF	Virtual Network Function
VTEP	VXLAN Tunnel End Point
VXLAN	Virtual Extensible Local Area Network
WAN	Wireless Area Network
WSN	Wireless Sensor Network

Capítulo 1

Introducción

Con este capítulo se inicia la memoria del trabajo de investigación llevado a cabo en el departamento de comunicaciones de la Universidad Politécnica de Valencia. En esta sección se detalla el problema objeto de estudio, la motivación tras el desarrollo de este trabajo, sus objetivos, la metodología llevada a cabo y como todo ello se distribuye y presenta a través de este documento.

1.1. Introducción

La creciente popularidad del ya conocido paradigma del Internet de las Cosas (IoT) ha tenido un impacto directo en el rendimiento y funcionalidad de la infraestructura de red tradicional sobre la que opera, infraestructura que ha tenido que adaptarse de forma precipitada a nuevos tipos de arquitectura, nuevos perfiles en el tráfico de datos, nuevos protocolos y demás características. Esta popularidad viene generada por la necesidad de digitalización, entendida como la capacidad de habilitar, mejorar, o evolucionar, las funciones empresariales, operaciones comerciales, modelos de gestión de clientes, y/o procesos de comunicación, aprovechando las tecnologías digitales, que surge de pequeñas y grandes empresas, y que ha convertido al Internet de las Cosas en una herramienta clave y tecnología habilitadora para permitir y ayudar a esta *digitalización*. El problema surge cuando la integración y conexión de nuevos dispositivos, denominados inteligentes, de carácter heterogéneo a través de la red, y la creación de servicios telemáticos para hacer uso de esos dispositivos, colocado en el punto de mira de numerosas empresas y convirtiéndolo en una prioridad para la digitalización, no se presenta como una tarea sencilla de realizar, incluso para empresas con grandes recursos.

Hoy en día ya es posible encontrar todo tipo de dispositivos conectados a través de la red pública (Internet), desde contadores eléctricos (*Smart Meters*) a relojes (*E-health Bracelets*), y a través de redes de uso privado en empresas e industrias. Dispositivos que poseen nuevas características específicas; se comunican e identifican a través de diferentes protocolos (ya no se usa solo IPv4, sino IPv6, números de referencia particular, etc.) su estructura de datos es muy diversa, y los servicios que los gestionan son especializados, y que, sin embargo, están siendo integrados y utilizados usando tecnologías de red e infraestructura no evolucionada. Esta integración realizada a través de una infraestructura de red tradicional no está adaptada a los requisitos que demandan los nuevos dispositivos y servicios inteligentes. En concreto, la configuración, descubrimiento y gestión de los dispositivos, en la mayoría de

las ocasiones, no se basa en protocolos usados en la red tradicional que sigue el modelo OSI (DHCP; SNMP; ICMP; etc.), con los cuales se conectaban los nodos típicos de cualquier red de comunicación (Servidores, Clientes, Puntos de Acceso, *Routers*, *Switches*, etc.), sino que poseen protocolos IoT concretos y difíciles tanto de integrar como de traducir (CoAP, LoRa, MQTT, etc.).

Esta tendencia, que genera un problema de integración, parece que no solo no se va a reducir, sino que se está incrementando exponencialmente, pudiendo actualmente contar por billones el número de dispositivos inteligentes conectados a la red [6]. Como ya hemos apuntado, las redes tradicionales basadas en el modelo OSI, no fueron desarrolladas pensando en afrontar la comunicación entre estos dispositivos, incluyendo el aumento en su número y aumento del volumen de tráfico de datos correspondiente. Ya sea en la red pública como en redes privadas, las tecnologías anteriores han quedado algo obsoletas y hay que hacer paso a nuevas tecnologías y arquitecturas que sean más dinámicas y se adapten a los nuevos requisitos que el IoT demanda. Un ejemplo de ello son la virtualización y la programabilidad en redes.

Así pues, la flexibilidad y programabilidad ofrecida por las Redes Definidas por Software SDN se presenta como una propuesta con altas probabilidades de mejorar y facilitar el despliegue, la integración y la gestión de dispositivos en infraestructuras de red para el Internet de las Cosas. En este trabajo presentamos un estudio de la aproximación de estas tecnologías. Esto es, analizamos las potenciales ventajas, mejoras y facilidades que introduce el uso de redes definidas por software, o redes programables, así como funciones de red virtualizadas, en despliegues de red fundamentalmente basados en dispositivos inteligentes. Como veremos en la sección del marco teórico, existen numerosos estudios que ponen en valor, el uso de redes virtuales programables para el uso de comunicaciones de datos tradicionales. Numerosas empresas punteras en el sector de las TIC (Amazon, Google, Microsoft, etc.) hacen uso de las redes programables y de la virtualización en sus centros de datos desde hace ya varios años. Sin embargo, existe un campo de investigación que aún no ha sido totalmente analizado, y es el uso de estas redes en entornos donde los

datos provienen de dispositivos inteligentes (*Smart Devices*).

Es por ello por lo que este trabajo ha querido analizar la implementación de las tecnologías más populares usadas en las redes programables, en entornos de Internet de las Cosas, donde la mayoría de los datos provienen de dispositivos, sensores o actuadores, que en ocasiones necesitan de una baja latencia y de una flexibilidad en su integración y gestión no proporcionada hasta ahora por las redes de datos tradicionales. Este trabajo se ha llevado a cabo durante varios años, bajo el marco de varios proyectos europeos, principalmente el proyecto INTER-IoT y el proyecto 5GENESIS. En esta memoria se presentan las investigaciones previas, la metodología y procesos llevados a cabo, la propuesta de arquitectura, su validación, así como las conclusiones y resultados obtenidos bajo este estudio.

1.2. Motivación de la Tesis

Tanto la creciente aparición de dispositivos inteligentes conectados a la red, como la necesidad de creación ágil de servicios que manejen estos dispositivos, son los dos principales conductores en la creación e impulso de nuevas tecnologías a nivel de infraestructura, e implementación de herramientas como la virtualización no solo de aplicaciones (SaaS), sino también de plataformas (PaaS) y de la propia infraestructura (IaaS) en sí, incluyendo las redes. Estos paradigmas que ya llevan un largo tiempo siendo utilizados y que hacen un uso principal de la virtualización, dan un paso más allá e introducen la programabilidad a nivel de red para dotar de más flexibilidad y dinamismo a sus infraestructuras. Basándonos solo en la parte de red, sin entrar en almacenamiento, y otros recursos, ya podemos corroborar los enormes beneficios que la virtualización de sus funciones y la programabilidad añadida a las redes ofrece a los despliegues clásicos, dónde los componentes que se comunican son servidores o nodos de computación. Sin embargo, cuando a estos despliegues les añadimos dispositivos inteligentes todo te torna más

complejo. Ya se ha descrito la problemática que presenta el manejo y la gestión de dispositivos, sus datos y sus servicios a través de redes de comunicación tradicional. Estos dispositivos poseen características muy particulares y heterogéneas, no presentes en los elementos que tradicionalmente componían una red. Nuevos y diversos protocolos, fuentes de datos con un formato específico, necesidades y requisitos fuertemente ligados a su dominio, etc.

Requisitos que incrementan la dificultad de conexión entre elementos IoT, al no poder ser tratados como elementos genéricos de una red de comunicación, cómo por ejemplo un ordenador o un servidor. Además, muchos de los servicios y aplicaciones que hacen uso de estos dispositivos o de sus datos poseen también unos requisitos concretos, como pueden ser la demanda de una muy baja latencia para servicios en tiempo real o alarmas, mayor ancho de banda para servicios de realidad virtual o aumentada, una alta fiabilidad y seguridad en las conexiones, un tiempo de creación y mejora ágil, entre otros. La demanda de estos servicios es grande y los desarrolladores de aplicaciones y administradores de sistemas han de entregar software cada vez más específico y con más requisitos en un menor tiempo. La problemática es clara, y las nuevas tecnologías y paradigmas (virtualización, computación en la nube, DevOps, SDN/NFV, etc.) están presentes, el único obstáculo por superar es combinar de forma correcta dichas soluciones para poder brindar el mejor servicio y experiencia a los usuarios, adaptándolas a cada caso de uso y dominio donde sean necesarias.

Con esto en mente, el presente trabajo quiere colaborar en el acercamiento de dos paradigmas tecnológicos, que bien combinados podrían ayudar a mejorar la problemática descrita anteriormente. Como se ha mencionado, la virtualización en una tecnología que lleva años implementada en los sistemas informáticos y que ya forma parte de las empresas tecnológicas más punteras. Esta virtualización se ha llevado a cabo a nivel de aplicación y servicio, de plataforma e incluso de infraestructura (computación en la nube), entro de esta última uno de los elementos de la infraestructura que se virtualizan es la red, y los diferentes nodos que la componen. Extendiendo esta tecnología y

añadiéndole programabilidad se obtiene lo que se denominan redes definidas software, paradigma que como veremos, también lleva unos años siendo implementada en diferentes despliegues tecnológicos. Sin embargo, entre estos despliegues nos encontramos pocos de ellos enfocados al Internet de las Cosas, aun siendo otro de los paradigmas tecnológicos que ha ganado mucha fuerza en los últimos años.

De aquí parte la motivación de utilizar redes virtuales programables como base sobre la que desplegar un sistema de Internet de las Cosas de forma que aquellos problemas de integración y gestión de los dispositivos y sus servicios queden mitigados o de alguna forma se faciliten estas tareas, de forma que no solo los dispositivos se hagan más inteligentes, sino que también la infraestructura sobre la que operan se dote de mayor inteligencia a su vez.

1.3. Objetivos

El principal objetivo de este trabajo es desarrollar y analizar el uso de Redes Definidas por Software (SDN) en entornos de Internet de las Cosas (IoT), para mejorar las características de estos entornos dotándolos de más flexibilidad, escalabilidad y gestión dinámica. Para llevar a cabo este objetivo principal se han definido diferentes sub-objetivos tales como:

- *Estudiar y analizar las tecnologías y herramientas actuales relacionadas con las redes programables y la virtualización de los servicios de red:* esto incluye la creación de un marco teórico y un contexto para conocer el estado y avance de las diferentes tecnologías. En concreto, se analizarán herramientas de virtualización, principalmente infraestructuras que permitan la virtualización, y herramientas de gestión y control de servicios virtualizados, así como la arquitectura genérica que estas poseen. Además, en el ámbito de red se estudiarán los componentes principales de la red programable como los switches virtuales, y

los controladores, además de los servicios específicos o aplicaciones existentes cuyo fin es la gestión de recursos en la red.

- *Estudiar y analizar las tecnologías y elementos presentes en los despliegues IoT:* análisis de la arquitectura genérica presente en los sistemas IoT, sus componentes y las tecnologías que los implementan. En concreto, se analizarán en detalle aquellos componentes que sean gestionados y conectados a través de la red como las puertas de acceso (*gateway*), y los protocolos de acceso a la red más comunes que estás implementan, las plataformas IoT, y los protocolos de comunicación de datos IoT. Nótese que, en esta sección, el análisis de los dispositivos o sensores queda fuera de ámbito, pues el tipo de dispositivo en sí y su función no está relacionado con el comportamiento de la red. De igual forma, el vertical al que pertenece el sensor, así como las aplicaciones de alto nivel que utilizan los datos de los dispositivos, quedan fuera del ámbito de este análisis.

- *Diseño de una arquitectura que combine los principales elementos presentes en los despliegues IoT con las tecnologías y herramientas utilizadas en las redes virtuales programables:* Mediante el diseño de esta arquitectura se incluirán las herramientas más beneficiosas extraídas de las redes virtuales programables analizadas anteriormente y se definirán una lista de requisitos básicos que han de ser cumplimentados en estos despliegues.

- *Diseño de un caso de uso para validar la arquitectura propuesta y las tecnologías seleccionadas:* se diseñará un caso de uso práctico o escenario con uno o más objetivos particulares para la validación de la arquitectura y evaluación del sistema completo. Así mismo, se establecerán una serie de indicadores de prestaciones (KPI) en cada caso de uso para determinar su viabilidad y funcionalidad.

- *Instalación y evaluación de las diferentes herramientas que implementan el uso de redes definidas por software. Selección de las herramientas*

adecuadas para crear el caso de uso: Una vez conocidas y estudiadas las tecnologías relacionadas con las redes programables y el IoT, se procederá a seleccionar las herramientas más adecuadas para implementar en un entorno de pruebas realista la arquitectura diseñada. Se concretarán las tecnologías que implementaran cada elemento de la arquitectura, y se desarrollarán los elementos que falten para crear un entorno de pruebas concreto que posea todos los elementos básicos de un despliegue IoT, más el valor adicional que provee una red virtual programable, que conecte cada uno de estos elementos. Más aún, esta red se verá complementada con herramientas de monitorización y gestión de red, así como servicios de red virtuales (VNF) que doten de mayores ventajas al despliegue en su conjunto.

- *Modificación-extensión del controlador SDN seleccionado con el desarrollo de nuevos módulos para extender sus funcionalidades y adaptarlas a las necesidades de los despliegues IoT:* En relación con el anterior punto, uno de los principales módulos a desarrollar será la extensión de las capacidades del controlador SDN para que no solo gestione una red programable cualquiera sino que tenga en cuenta las características específicas de la red, es decir, el tipo de tráfico y los elementos que conecta, en este caso IoT. Es por ello por lo que se desarrollarán servicios sobre el controlador principal de la red, para que la toma de decisiones de este se vea influenciada por las características particulares de nuestro despliegue.

- *Colaboración con otros proyectos para la implementación de nuevas funcionalidades:* Además del desarrollo interno de este trabajo se ha tenido la oportunidad de llevar a cabo colaboraciones externas con terceros para incluir nuevas tecnologías y funcionalidades al sistema original. Se incluirá un apartado especificando estas colaboraciones, siendo la más destacada la llevada a cabo con el instituto de investigación

griego INFOLYSIS [1] para la integración de un sistema de traducción de protocolos de red en tiempo real.

- *Selección de utilidades de prueba para evaluar el comportamiento de la solución diseñada, obtención de datos y su análisis:* En concreto, con los KPIs previamente establecidos determinar la mejor forma de obtener y computar esos valores para posteriormente analizarlos. Entre los indicadores de funcionamiento seleccionados se encontrarán: facilidad de integración de nuevos elementos IoT al despliegue (escalabilidad), facilidad de configuración de nuevas reglas de enrutado de datos en el sistema (dinamicidad), facilidad de reestructuración de la topología de la red programable (flexibilidad), facilidad en el despliegue de nuevos servicios sobre el despliegue (*Service Creation Time*, Tiempo de Creación de servicios).

- *Validación de la hipótesis, redacción y exposición de documentos relacionados con la tesis. Exposición de conclusiones y definición de futuras líneas de investigación:* Una vez recogidos las diferentes métricas en los casos de usos, se lleva a cabo la computación de los indicadores de rendimientos y su evaluación, es decir, que representan estos indicadores y si estos valores están dentro de lo deseado. Con estos datos se lleva a cabo un análisis y discusión para mostrar si estos valores refutan o dan soporte a la hipótesis presentada. Si estos valores de rendimiento se encuentran dentro de lo esperado implicará que la integración de las redes programables en despliegues de Internet de las Cosas, son beneficiosas en primera instancia, y aunque aún haya más trabajo y análisis por hacer sienta una buena base de estudio y líneas de investigación para poder seguir trabajando sobre ella.

- *Difusión de resultados, participación en eventos de carácter técnico-científico e impulsar el uso de la combinación de estas tecnologías:* Como parte

[1] https://company.infolysis.gr

importante, aunque no técnica, de este trabajo se incluye la realización y participación de eventos, presentaciones, demostraciones, etc. en entornos especializados, así como la difusión de los procesos y avances del estudio en medios de carácter técnico, revistas de ciencia, etc. para impulsar el uso de las tecnologías utilizadas y expandir el conocimiento obtenido a la máxima audiencia posible.

1.4. Contribuciones Principales

La principal novedad de este trabajo radica en la integración de dos paradigmas tecnológicos muy actuales y de creciente popularidad. A pesar de que existe algún punto del trabajo fuertemente influenciado por el poco grado de maduración de las algunas tecnologías, se han llevado a cabo todos los objetivos propuestos en el punto anterior, teniendo que desarrollar en muchos casos algunas de las herramientas necesarias para su consecución al no estar éstas directamente disponibles en el mercado. Entrando más en detalle, se han realizado las siguientes contribuciones directas:

- Proporcionar un prototipo de arquitectura basada en una red definida por software, totalmente virtualizada, que integre elementos de los despliegues de internet de las cosas (dispositivos inteligentes, pasarelas, plataformas y aplicaciones).
- Extensión de un controlador SDN para ofrecer funcionalidades puramente orientadas al manejo de tráfico IoT por parte de los nodos virtuales que componen la red.
- Aplicación de esta arquitectura en diferentes escenarios:
    - Creación de una red virtual programable que dote de interoperabilidad a diferentes elementos de un despliegue IoT.

- Integración de protocolos y dispositivos IoT en una infraestructura 5G con *Core y Edge* basado en SDN/NFV.

Por otro lado, durante la ejecución de este proyecto, otras contribuciones indirectas o transversales han tenido lugar. Se denominan indirectas porque estas no han tenido un efecto directo en la consecución de los objetivos planteados, sin embargo, han sido útiles dentro del marco de la investigación y el aprendizaje. Estas son:

- Participación en congresos, charlas, y presentaciones para la distribución de resultados, discusión, e intercambio de información. Está colaboración con otros investigadores y proyectos ayuda en el crecimiento técnico personal y de todo el ecosistema de la investigación.
- Colaboración con la comunidad *Open Source*. Los módulos y herramientas que han sido desarrolladas a base de código escrito han sido publicadas estrictamente bajo licencia de software abierto. En concreto, todo el código fuente está público en la plataforma *Github* bajo licencia *GNU General Public License v2.0*.
- Colaboración en proyectos europeos. Este punto ha sido fundamental para la ejecución de este trabajo, puesto que sin la colaboración en proyectos de Investigación Europea no habría habido existido esta tesis. A través de la colaboración activa en los proyectos Europeos INTER-IoT y 5GENESIS se ha tenido como producto directo los resultados que mostramos en esta memoria.

1.5. Metodología y Retos Encontrados

Este proyecto de investigación se ha llevado a cabo siguiendo una metodología estándar en el desarrollo de investigaciones tecnológicas, con una fuerte investigación del estado del arte previo y análisis del marco teórico, así

de como de las tecnologías y soluciones disponibles actualmente, la proposición de un novedoso modelo de arquitectura combinado, la creación de un entorno de pruebas para la validación e implementación de esta arquitectura, y la recolección de observaciones y resultados que desembocan en una evaluación o conclusión final. En particular, este trabajo ha sido dividido en las etapas que podemos ver en la Figura 1.1.

Figura 1.1: Metodología de Investigación.

- *Concepción de la idea o problemática:* Mejora de las redes tradicionales para mejorar el rendimiento de los despliegues IoT.

- *Planteamiento del problema:* Se necesitan nuevos paradigmas de red que cumplan con las necesidades surgidas a raíz del aumento de tráfico y la distinta naturaleza de los datos debido al IoT.

- *Revisión de Literatura:* Creación de un estado del arte con todas las herramientas, proyectos y soluciones existentes hasta la fecha enmarcadas en este campo. Análisis de las características y funcionalidades de las herramientas SDN.

- *Propuesta de la solución:* Creación de una arquitectura que utilice elementos de las redes definidas por software para mejorar y complementar las características de los despliegues IoT en general, y los aquellos despliegues de tecnologías que ya hace uso de estos paradigmas y son de primera actualidad, como los despliegues IoT-5G.

- *Diseño de los Escenarios de Validación:* Implementación de un escenario de pruebas donde validar la viabilidad de la propuesta arquitectónica, utilizando herramientas SDN y los componentes básicos presentes en la mayoría de los despliegues IoT.

- *Ejecución de pruebas:* Definir indicadores de rendimiento para la validación, ejecutar diferentes pruebas sobre los escenarios.

- *Recolección de Datos y Observaciones:* Comprobación de la llegada de los datos y medición de parámetros de red.

- *Análisis e Interpretación de las Observaciones:* Análisis de los datos obtenidos.

- *Elaboración de Informe:* Elaboración de la presente memoria, junto con documentación y presentaciones de diferente índole para la exposición de los resultados.

Estas etapas principales comprenden o están divididas en una serie de subtareas más flexibles y pequeñas, no detalladas aquí, puesto que no siguen una metodología estándar definida. Aunque sí que cabe destacar que, para muchas de las tareas, principalmente de desarrollo e integraciones técnicas, se ha llevado a cabo una metodología Ágil, de iteración y mejora continuas sobre una misma tarea o módulo.

Por otro lado, las principales dificultades encontradas a la hora de llevar a cabo este trabajo de investigación se centran en dos puntos:

1. Dificultad para acceder a software específico, a causa de su precio y no disponibilidad al no ser software abierto. Así como la dificultad de acceder también a hardware específico debido a su alto precio y falta de instalaciones. Esto se ha subsanado buscando alternativas de código abierto (*Open Source*) para muchas de las herramientas utilizadas en la implementación de los escenarios, y colaborando con otras empresas y Universidades en proyectos de investigación que nos han permitido hacer uso de su infraestructura y herramientas durante el desarrollo del proyecto.

2. Dificultad de definir límites en el trabajo y uso de tecnologías. Puesto que las redes programables y virtualizadas y los entornos de Internet de las Cosas engloban una serie de herramientas, tecnologías, sistemas y servicios muy amplia y variada, en ocasiones se ha presentado la dificultad de no saber si incluir nuevas herramientas y funcionalidades a los Escenarios debido a que el alcance de estos se magnificaría y haría muy difícil centrar el objetivo principal del trabajo. Es por ello por lo que, aunque muchas otras herramientas y aplicaciones se han estudiado y probado, no han sido incluidas en las implementaciones finales de los escenarios para la validación de la arquitectura diseñada.

1.6. Alcance de la Tesis

El presente trabajo ha sido desarrollado en el marco de varios proyectos europeos, y forma parte de un trabajo de investigación mayor en colaboración con diferentes socios públicos (universidades) y privados (compañías). Pero como en toda investigación que desee ser realista, es necesario delimitar el alcance de este trabajo y aquello que ha sido proporcionado por terceros dentro de los proyectos en los que se ha colaborado. En un campo tan amplio como las redes y el IoT debemos enfocar nuestro trabajo en áreas concretas especialmente en aquellas dónde hay un margen de mejora y donde se puedan

integrar conceptos realmente innovadores.

De los proyectos en los que se ha participado durante está investigación, el primero de ellos fue INTER-IoT, cuyo principal objetivo se basaba en la creación de soluciones para integración de elementos en todas las capas de la pila de tecnologías del Internet de las Cosas. En este caso el trabajo realizado para este proyecto llevado a cabo bajo el marco de este estudio fue la creación de soluciones de integración a nivel de Red. Más concretamente la creación de una red programable para integrar e interoperar todos los elementos de red presentes en un despliegue IoT de forma virtual. Para ello, se llevó a cabo un detallado estudio de las tecnologías y protocolos más comunes a nivel de red en los despliegues IoT, el cual se muestra en detalle en los siguientes capítulos, se definió una arquitectura genérica y unos requisitos para llevar a cabo la interoperabilidad, se seleccionaron las herramientas adecuadas para implementar dicha arquitectura y se desarrollaron módulos claves para complementar las funcionalidades de la red programable. En concreto, habiendo seleccionado los componentes de la red definida por software, tanto enrutadores virtuales como controlador SDN, se extendieron las capacidades del controlador para darles un enfoque más práctico en un despliegue IoT. Se creó un módulo de enrutamiento de paquetes específico para tráfico IoT, con la posibilidad de priorizar tráfico según su tipo. Por último, se realizaron pruebas de validación sobre la red virtual construida. Todo el trabajo llevado a cabo dentro del proyecto INTER-IoT se corresponde a la primera parte de investigación y análisis, así creación el primer esbozo de la arquitectura y el primer Caso de Uso que se describe en la presente memoria.

El segundo proyecto, 5GENESIS, tenía como principal objetivo la creación de una plataforma para validación de los diferentes indicadores de rendimiento (KPIs) identificados para cumplimentar las especificaciones 5G. Esta plataforma estaba compuesta de cinco instalaciones geográficamente distribuidas a través de Europa: en Málaga, Surrey, Berlín, Limasol y Atenas. Cada una de las instalaciones estaba orientada en evaluar uno o varios de los principales casos de uso que se han identificado en los entornos 5G, como son:

comunicación masiva de dispositivos (*Massive Machine Type Communications (mMTC)*), mejora del ancho de banda móvil (*Enhanced Mobile BroadBand (EMBB)*) y comunicación ultra fiable de baja latencia (*Ultra Reliable and Low-Latency Communications (URLLC)*). En este caso en trabajo realizado se ha basado en la participación en la plataforma de Limasol, cuyo objetivo principal era la evaluación de escenarios de mMTC, es decir, aquellos que brindan conectividad a numerosos dispositivos IoT. El trabajo realizado en la plataforma incluye la creación de un escenario que integraba tecnologías IoT (LoRa) con la infraestructura 5G en cuyo borde (*Edge*) y en cuyo núcleo (*Core*) se desplegaban servicios IoT, sobre la red virtualizada. Estos servicios estaban encapsulados en contenedores virtuales para su fácil despliegue, y así limitar el tiempo de creación de los servicios. Todo el trabajo llevado a cabo dentro del proyecto 5GENESIS se corresponde con la segunda parte del proceso de investigación y análisis, mejora de la arquitectura y el segundo Caso de Uso descrito en la presente memoria. En ambos casos, la participación estaba relacionada con un despliegue IoT en el que se introducen en la red técnica de programación o virtualización. También, dentro de ambos proyectos se han llevado a cabo numerosas tareas de documentación, diseminación de resultados, y participación en congresos y eventos de carácter técnico. Estos dos proyectos europeos tienen un alcance mucho mayor al de esta tesis, por lo que únicamente las tareas relacionadas con esta tesis y que tengan relevancia serán descritas aquí a modo de contexto. El resto de las tareas realizadas dentro del marco de estos proyectos se dejan a modo de referencia para que el lector interesado pueda acceder a las fuentes. Aunque esta tesis corresponde a una parte limitada del total que componen ambos proyectos, esta parte ha sido de vital importancia para la consecución de los objetivos generales y de su finalización de forma exitosa.

1.7. Estructura de la Memoria

En este primer capítulo se presenta un resumen del trabajo de doctorado, sus objetivos principales, motivaciones y contribuciones. A lo largo de la memoria se desarrollarán de forma detallada las pinceladas de información que se han presentado en esta Introducción, en concreto encontramos:

Capítulo 2, se realiza un estudio profundo del estado del arte o marco teórico donde se encuadra este trabajo. En concreto, en la primera parte se describen las tecnologías que hacen posible la transición de una red tradicional a una red virtual programable. Principalmente, hablaremos de las capas que componen estas redes (capa de datos y capa de control) y el principal protocolo para comunicar sus elementos: *OpenFlow*. Por otro lado, nombraremos las herramientas utilizadas para implementar otros elementos de las redes virtuales como es el caso de *OpenVSwitch* para los enrutadores virtuales, o Open Source MANO (OSM) como orquestador principal de las funciones virtualizadas. Tras analizar las tecnologías y herramientas utilizadas a lo largo de este trabajo, se justifica el porqué de la elección de estas y como se han utilizado en el entorno de pruebas. Por otro lado, se analizarán proyectos con similar alcance y temática, y se describirá en qué se diferencian al presente trabajo.

Capítulo 3, en primer lugar, se muestra y explica la arquitectura genérica en la que se basa el paradigma SDN/NFV; que se utilizará como punto base para el desarrollo de nuestra propia arquitectura. Más adelante, se definen los requisitos que ha de cumplimentar el sistema, requisitos relacionados con parámetros como la latencia mínima, la seguridad, la escalabilidad, el tiempo de creación de servicios, entre otros. También, se presentan brevemente las funciones de red virtuales específicas que se han integrado en esta arquitectura. Por último, se describe una propuesta de arquitectura que incluye elementos de las redes programables, ya sean elementos abstractos como la división de plano de datos y plano de control, como tecnologías concretas, como el uso de *switches* virtuales y de funciones de redes también virtualizadas, y elementos característicos de los entornos IoT, como son los dispositivos inteligentes,

sensores y actuadores, las *gateways* o puertas de enlace, y los servicios.

Capítulo 4, se presenta la primera implementación de la arquitectura definida en el capítulo anterior en un escenario de pruebas, utilizando tecnologías concretas y realizando una serie de pruebas para analizar el comportamiento de la red. En este primer escenario se realiza un despliegue sencillo en un entorno virtualizado, dónde tanto los dispositivos como la red son emulados. Para la creación de este escenario se desarrollaron piezas clave, como la aplicación sobre el controlador SDN y se utilizaron otras herramientas pertenecientes al trabajo realizado en el laboratorio, como es el caso de la pasarela. Entre las pruebas que se presentan, las más importantes incluyen la priorización de tráfico y la traducción de protocolos.

Capítulo 5, en este apartado se describe el segundo escenario donde se implementa la arquitectura, en este caso un escenario 5G. Este segundo supuesto es más ambicioso que el anterior y cuenta con numerosos componentes y servicios extra, fuera del alcance de esta tesis, sin embargo, es necesario describirlos y conocerlos para entender el escenario en su conjunto. En este caso, el despliegue se lleva a cabo en un entorno real, no emulado, sobre una infraestructura de red móvil. Entre las pruebas que se detallan en esta memoria, las más importantes son las de medida de latencia y de tiempo de creación de servicios.

Capítulo 6, en este capítulo se definen los indicadores de rendimiento diseñados para cada escenario, y se muestran valores de dichos indicadores en cada caso. Esto es, las pruebas antes definidas son analizadas atendiendo a parámetros medibles y valores objetivos, en estos resultados se indica también la implicación que suponen.

Por último, el **Capítulo 7** muestra las conclusiones extraídas del trabajo realizado, no solo de los escenarios y sus indicadores de rendimiento, sino también de la información obtenida, las innovaciones llevadas a cabo y la participación y colaboración con otros proyectos de investigación y eventos. Además, se expone un subcapítulo de líneas futuras dónde se indica en qué

partes del trabajo se podría seguir trabajando, qué se puede mejorar y cuál serían los pasos que seguir de aquí en adelante.

Capítulo 2

Estudio Del Estado del Arte

Para poner en contexto este trabajo, primero debemos exponer las tecnologías y herramientas que son la base fundamental del estudio. En este capítulo, se muestra un análisis detallado de estas tecnologías. Además, se define, por un lado, qué son las redes definidas por software, de qué partes se componen y qué herramientas hardware y software las implementan, y por el otro, qué es el internet de las cosas, que partes lo componen y cuál es la problemática que supone para las redes tradicionales. Por último, se detalla el estado actual de estudios y proyectos que hacen uso de estos dos paradigmas y cómo este trabajo encaja dentro de este marco teórico.

2.1. Redes Definidas por Software (SDN)

El término Redes Definidas por *Software* en castellano no es un concepto del todo preciso. Esto se debe en parte a su traducción, puesto que este término viene del inglés *Software-Defined Networking*, dónde ese *networking* hace referencia no solo a las Redes en sí (*networks*) sino que engloba todos los procesos, arquitecturas y herramientas que están relacionados, o de alguna forma involucrados, en la comunicación de los datos a través de una red de computadoras. Por lo que el término anglosajón, incluye una terminología más amplia y, por tanto, certera respecto a lo que significan las siglas SDN [7, 8].

Es por ello por lo que cuando hablamos de SDN lo definimos como la aproximación, arquitectura o propuesta técnica hacia la gestión de redes de comunicación que implica el desacople lógico, en muchos casos también físico, del plano de datos y el plano de control. Esto permite la configuración de la red de una forma más dinámica, y lo que es más importante; programable. Gracias a este desacople del plano de control y el plano de datos, se permite una mejora en el rendimiento de la red y una facilidad en la monitorización de la infraestructura. Esta aproximación es fundamental en entornos como en la nube [9], por lo que este paradigma convierte la gestión de red tradicional en una gestión más parecida a la computación que se lleva a cabo en la nube.

Este paradigma surgió como respuesta a los problemas y limitaciones que se encontraban en las redes tradicionales. Debido al aumento de tráfico y su diversidad, así como los servicios y las aplicaciones que se ejecutan sobre la red, la complejidad en la configuración y gestión de redes ha ido aumentando significativamente con el tiempo. El desarrollo y la incorporación de nuevas tecnologías como la nube, las redes móviles o el Internet de las Cosas [10, 11], generó la necesidad de una red con un mayor ancho de banda y una mayor accesibilidad, también más manejable y flexible y, sobre todo, más dinámica y fácil de escalar [12]. Así pues, las redes programables surgen como una aproximación para abordar estas nuevas necesidades, permitiendo a la red un grado más de abstracción para poder manejar de forma independiente los

elementos lógicos de los físicos y romper con el modelo centrado en hardware que había anteriormente.

Este desacoplamiento implica que por un lado tengamos las funciones de control, o funciones inteligentes como pueden ser el enrutado de datos, el análisis de paquetes, etc. funciones que pueden ser ejecutadas en nodos de propósito general, y que no necesitan de hardware específico para funcionar, únicamente suficientes recursos de memoria y procesador. Y que, por el otro lado, tengamos las funciones de datos, es decir, aquellas que afectan a la infraestructura por donde viajan los datos (paquetes) y que no tienen por qué ser funciones elaboradas, sino que más bien se convierten en tareas repetitivas y básicas, que se realizan en un tiempo muy corto como es el reenvío de paquetes desde un punto al siguiente de la red, sin llevar a cabo ninguna tarea adicional. Gracias a ello, esta infraestructura de datos puede estar compuesta por elementos muy básicos: nodos o enrutadores, reales o virtuales, que no requieran grandes recursos ni una compleja gestión, que sean más baratos y fáciles de integrar, añadir o eliminar, pudiendo modificar así la topología de la red de una forma más sencilla y rápida, adaptándose al tipo de tráfico y las necesidades del momento.

Además, al desacoplar el plano de control del plano de reenvío de datos, se permite que todo el control de la red se lleve a cabo desde un punto único y sea directamente programable (controlador) [13], teniendo así una visión global de todo lo que pasa en la red, y que la infraestructura subyacente quede abstraída de las aplicaciones y servicios que se ejecutan sobre el controlador de la red. Como se puede leer en [14] existen actualmente numerosos casos de uso para los cuales las redes definidas por software han sido base fundamental para llevarlos a cabo. Para poder definir las ventajas que estas redes introduces, a continuación, vemos en detalle cuáles son los aspectos clave que poseen estas redes:

- Programable: el control de la red se lleva a cabo desde un punto centralizado a través de una aplicación, servicio o *framework*, que se

encarga de enviar comandos a los diferentes nodos para configurarlos. Por lo tanto, desde este punto central se puede indicar a modo de instrucciones o código, como configurar los diferentes nodos dependiendo del estado de la red [15].

- Ágil: sumado a lo anterior, abstrayendo el control del plano de datos permite a los administradores ajustar dinámicamente el flujo del tráfico en toda la red respondiendo a las necesidades cambiantes de las aplicaciones que se ejecutan encima [16].

- Gestión centralizada: toda la lógica e inteligencia de la red se concentra en un punto central, el controlador, que puede ser uno o varios servicios. Estos mantienen una visión global de la red, y se la muestra a las aplicaciones y los gestores de políticas ejecutándose sobre los mismos como una única unidad de enrutamiento lógico [17].

- Automatización y configuración de recursos: la arquitectura SDN permite a los gestores de red manejar, configurar, asegurar y optimizar los recursos de forma rápida y dinámica a través de automatizaciones. Normalmente estos son programas creados por los propios administradores, que ejecutan diferentes acciones tradicionalmente repetitivas. Más aún, estos programas se crean como scripts de automatización que se ejecutarán en la máquina del controlador y no suelen ser de software propietario [18].

- Estándares abiertos y proveedores neutrales: la arquitectura SDN se implementa mediante estándares abiertos, por lo que simplifica el diseño de la red y sus operaciones, ya que las instrucciones son producidas por los controladores de forma estandarizada, en vez de por múltiples dispositivos y protocolos de proveedores específicos [19].

Así pues, una vez definidas las características principales de las SDN y sus funciones, vamos a ver un esquema de su arquitectura y los elementos que componen una red de este tipo.

2.1.1. Arquitectura SDN

Este apartado será extendido en la sección segunda Capítulo 3 Diseño de una Arquitectura SD-IoT, donde se combina la arquitectura tradicional de un despliegue IoT, con la que se presenta en esta sección.

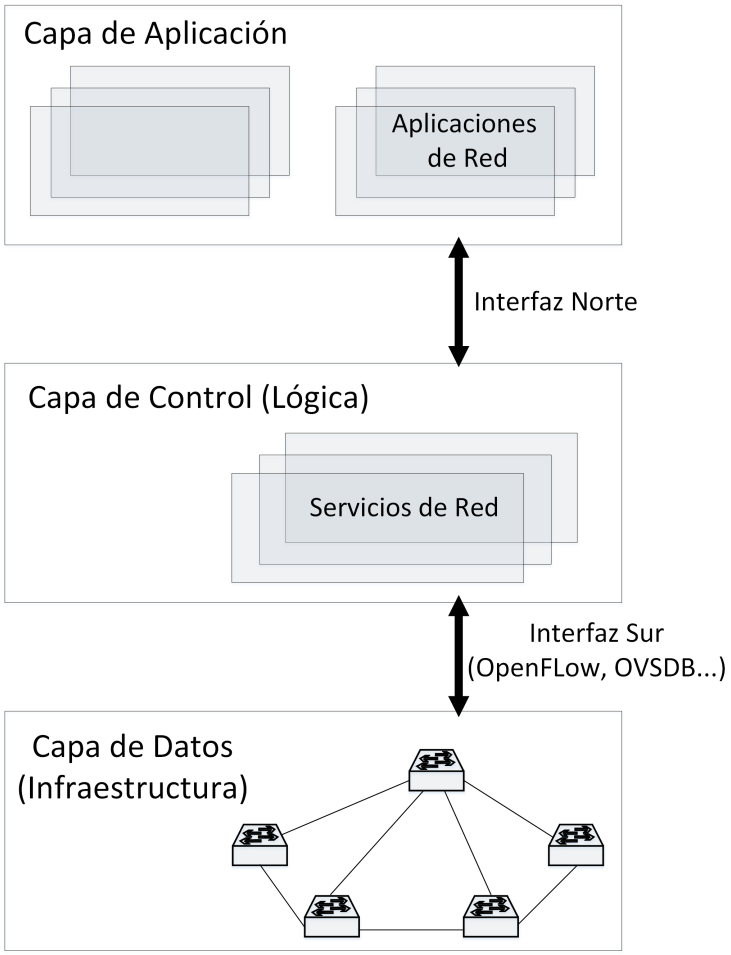

Figura 2.1: Diagrama de capas básico de una red definida por software.

La arquitectura de una red SDN está compuesta a grandes rasgos por las dos capas ya descritas, la capa de datos y la de control, más una capa extra que podríamos añadir denominada capa de aplicaciones [20]. Cada una de las capas se comunica entre sí mediante interfaces, la norte y la sur como

se puede observar en la Figura 2.1 En numerosas ocasiones la centralización del plano de control no recae únicamente en una sola aplicación o servicio, sino que son varias de ellas las que se conforman este plano. Para ello existen diferentes interfaces *west/east-bound*, que permiten la comunicación entre diferentes controladores, o módulos de un mismo controlador. Todos estos elementos serán más detallados y analizados en las siguientes subsecciones, y se mostrará también que tecnologías concretas pueden implementar cada uno de estos componentes.

Elementos

- Elementos del plano de datos: tradicionalmente los elementos que componen una red tradicional poseías tanto la parte de reenvío de datos como la lógica en un mismo nodo, estos nodos eran enrutadores, firewalls, IDS, etc. cada uno con una función diferente. En este tipo de arquitecturas los elementos del plano de datos se convierten en elementos pasivos, meros conmutadores de reenvío de paquetes cuya lógica está alojada en capas superiores.

- Elementos del plano de control: estos elementos son los que producen la lógica de la red, que está formada principalmente por los controladores, y todos los servicios y aplicaciones que se ejecutan sobre los mismos. Los controladores son el elemento central se una arquitectura SDN, son los que permiten centralizar la gestión y el control de la red, la automatización y el refuerzo de las políticas a través del entorno de red físico o virtual.

- Interfaz sur (*Southbound API*): intercambia información entre el controlador y cada uno los dispositivos que componen el plano de datos de la red, tales como puntos de acceso, enrutadores, conmutadores, firewalls, etc. aunque estos elementos posean ahora un comportamiento pasivo.

- Interfaz norte (*Northbound API*): intercambia información entre el controlador y las aplicaciones y motores de políticas, para las cuales toda la red SDN se presenta como un único nodo de red lógico.

> Una Red Definida por Software es una arquitectura o paradigma de red que permite dotar de flexibilidad y facilidad de gestión a las redes tradicionales. Para ello, se centraliza la gestión en un punto y se abstraen y separan las funciones de control (Plano lógico o de control) de las funciones de reenvío de datos (Plano de datos) de los dispositivos que forman la red.

2.1.2. Plano de Datos

Si hacemos una revisión de los trabajos hasta la fecha relacionados con las redes programables, fácilmente podemos observar que la mayoría de ellos se centran en el desarrollo y la innovación del plano de control, o plano programable, dónde ha habido un mayor número de avances en menor tiempo. Sin embargo, el plano de datos también es un área que ha sufrido cambios, y es importante describir sus funciones y la evolución que ha vivido a lo largo de los últimos años.

Como se ha mencionado previamente, las redes tradicionales utilizaban una aproximación para las funciones de la red centrada en hardware, es decir, existía un componente hardware diferente para cada una de las diferentes funciones de la red. Una red estaba constituida de enrutadores, puntos de acceso, y diferentes tipos de *middle-boxes* verticalmente integrados y diseñados con chips específicos o incluso con Application-Apecific Integrated Circuits (ASICs). Por lo que el mayor cambio que introdujeron las redes programables fue modificar está aproximación y enfocarla a una solución más centrada en software. El plano de datos, por tanto, pasa a ser un plano cuya principal y

casi única función es el reenvío de paquetes, una infraestructura creada por elementos tanto físicos como virtuales que llevan a cabo el transporte de flujos de datos a través de esta.

Infraestructura y composición del plano

En una infraestructura de red programable, podemos también encontrar diferentes equipos como enrutadores, conmutadores o *middle-boxes*, sin embargo, los elementos de reenvío son simples equipos con una lógica y un software sencillo y limitado. Esto es debido a que la inteligencia de la red y todo su estado se encuentra centralizado, la infraestructura subyacente es abstraída y virtualizada para ser presentada de manera global a las aplicaciones que se ejecutan por encima de la capa de control. En esta capa encontramos como elemento principal los dispositivos SDN de reenvío que pueden ser tanto físicos como virtuales, a los que comúnmente y a lo largo de toda la memoria llamaremos elementos de reenvío o *switches*. Estos elementos se comunican con la capa de control mediante un protocolo como OpenFlow, así se envían y reciben paquetes mediante un canal seguro, dónde la capa de control indica a los elementos de la capa de infraestructura que acción deben realizar.

- Switch OpenFlow, este tipo de switches implementan el protocolo OF para comunicarse con el controlador, poseen tablas con diferentes entradas donde se indica qué acción debe realizarse con cada paquete recibido. Este tipo de switches posee al menos tres partes principales: I) una tabla de flujo: las tablas están compuestas de entradas y son usadas para evaluar los paquetes recibidos y ejecutar la acción de reenvío, II) un canal de comunicación seguro con el controlador, normalmente basado en el protocolo TLS/SSL, y III) la implementación del protocolo OpenFlow para enviar y recibir paquetes en dicho formato así el controlador puede comunicarse con los switches. En la siguiente subsección de tecnologías,

hablaremos un poco más de este protocolo, cuál es su formato y como se utiliza.

- Switches Software, son aquellos enrutadores virtuales de código abierto que implementan las funciones de un enrutador básico, implementan las principales interfaces estándar de gestión y permiten extensiones programables y control de las funciones de reenvío. Hoy en día existen numerosos switches virtuales entre los que encontramos: OpenFaucet, Contrail-router, LINC, ofsoftswitch13, OpenFlow Reference, OpenFlow Click, OpenFlowJ, OpenWRT, Switch Light, XorPlus, y en nuestro caso, el que tiene mayor relevancia para este trabajo OpenVSwitch (OVS) ya que es uno de los enrutadores software utilizados para la implementación del primer Escenario de pruebas. Más adelante describiremos su funcionamiento y características.

- Switches Hardware, existen también numerosos enrutadores físicos comerciales que implementan los diferentes protocolos de gestión utilizados por las redes programables como OpenFlow, algunos de los más comunes han sido desarrollados por empresas punteras en desarrollo de herramientas de red como Juniper (EX9200 Ethernet), Extreme Networks (BlackDiamond X8) o HP (8200 zl and 5400 zl). Sin embargo, hemos de mencionar que estos productos normalmente se venden para despliegues particulares y su precio es muy elevado, por lo que cabe destacar dos opciones de bajo coste que implementan protocolos SDN a un precio moderado como son el Pica83920 (Pica8) o la ZodiacFX (Northbound Networks [1]). En concreto, este último ha sido utilizado en varias pruebas del Escenario 1 de este trabajo, intercambiando el switch virtual de OVS por este switch físico.

Interfaz Sur (*Southbound Interface*)

[1] https://northboundnetworks.com/pages/zodiac-fx-faq

Como ya se ha mencionado, la interfaz sur es aquella que conecta cada uno de los elementos que componen la infraestructura del plano de datos de una red programable con su controlador en el plano lógico. Esta interfaz está compuesta por un conjunto de API que permiten dicha comunicación entre planos. Como ya hemos dicho, el protocolo OpenFlow es el más implementado y conocido en la interfaz sur para comunicar controladores y los nodos de reenvío, siendo esta una de las tecnologías principales de las redes definidas por software de forma generalizada. Se proporciona una especificación para migrar el control y la lógica desde los nodos de la infraestructura al controlador. Por tanto, tanto el controlador como los conmutadores deberían tener implementado en protocolo OpenFlow, en su misma versión o en versiones compatibles.

Las arquitecturas basadas en OF tienen capacidades específicas que pueden ser explotadas por los investigadores para experimentar con nuevas ideas y probar nuevas aplicaciones. Estas características incluyen el análisis de tráfico basado en software, control centralizado, actualización dinámica de reglas de reenvío y abstracción de los flujos de datos. Numerosas aplicaciones basadas en OF se han propuesto para facilitar la configuración de la red, y simplificar así la gestión de esta, añadiendo seguridad a las redes virtualizadas y los centros de datos. Los autores de [21] presentan un estudio de todas las tecnologías relacionadas con OpenFlow que han sido desarrolladas por investigadores, creadores de servicios de red y otros, todas ellas para facilitar el diseño, testeo, y despliegue de ideas innovadoras, como la que se presenta en este trabajo, mayormente en entornos experimentales, pero también en producción, para acelerar la investigación en tecnologías y entornos de red. Sin embargo, como veremos en más detalle en nuestro capítulo de tecnologías y herramientas, OpenFlow no es la única interfaz de comunicación disponible en la parte sur. Existen otros protocolos implementables para esta interfaz. Algunos ejemplos como el PoF, protocolo de reenvío ajeno, OpenvSwitch Database Management Protocol (OVSDB) [22], protocolo de separación entre el reenvío y el elemento de control (ForCES) [23, 24], OpFlex protocolo de control [25], OpenFlow Config (OF-Config) [26], entre otros.

Durante la fase de investigación se llevó a cabo un análisis de las características de cada protocolo, cuáles eran los más utilizados y por qué. Aunque ha sido de utilidad estudiar todos ellos, en esta memoria no vamos a describir todos sino solo aquellos que fueron elegidos. Estos son OpenFlow y OVSDB. La elección de estos protocolos fue sencilla, puesto que ellos son los más desarrollados dentro del ámbito de la separación de red en planos, además de ser los más convenientes para la comunicación de datos entre planos y también relevantes para los Escenarios. Sin embargo, también se ha hecho uso en ocasiones concretas de otros protocolos, como es el caso de OF-Config, solo para pruebas concretas, sin llegar a ser un protocolo fundamental en el despliegue de la red.

Herramientas, Tecnologías y Protocolos

Openflow

OpenFlow es el protocolo principal sobre el que se desarrolla el paradigma de SDN. Este protocolo es el encargado de comunicar los comandos del plano de control al plano de datos a través de la interfaz sur. El protocolo surgió en la Universidad de Stanford, en el 2008 a raíz de un proyecto de investigación [27]. Es por ello por lo que su estándar es abierto e implementable por particulares y empresas. Gracias a la implementación de este protocolo en la interfaz sur de la mayoría de los controladores la red puede ser gestionada dese ese mismo punto central, las decisiones se pueden tomar teniendo una visión global del estado re la red, uno de los requisitos fundamentales y una de las principales ventajas de las redes programables. En un inicio OpenFlow se desarrolló para facilitar la movilidad entre máquinas virtuales (VM), sin embargo, su uso se extendió más allá de las máquinas virtuales dentro de un solo servidor, a toda una red virtualizada entre varios servidores de un mismo despliegue o centro de datos.

La idea principal de este protocolo es la de utilizar elementos comunes pertenecientes a la mayoría de los enrutadores ethernet, como son las tablas

de flujos (Flow tables) y modificar o gestionar estas para llevar a cabo el encaminamiento de datos y funciones de firewall, NAT y QoS, a través de las estadísticas recolectadas. Aunque cada fabricante implementa de forma diferente estas tablas de flujo existen características comunes en todos ellos, y la tendencia es a la homogeneización de este timo de estructuras de datos. Este protocolo ha estado desarrollándose desde el primer momento y aún en la actualidad está en pleno desarrollo, haremos un repaso por su sistema de versiones y qué características son inherentes a cada una de ellas (todas y cada una de las especificaciones se pueden encontrar en la página oficial de ONF [28]):

- *Versión 0.2:* primera versión prototipo publicada en marzo del 2008 como borrador.
- *Versión 1.0:* primera versión con soporte oficial. Esta ya soportaba el manejo de una única tabla de flujos con sus diferentes entradas, que consistían en: campos de cabecera, contadores y acciones.
- *Versión 1.1:* fue publicada en febrero de 2011, dónde se introdujo la capacidad de procesar varias tablas.
- *Versión 1.2:* publicada en febrero del 2012, después de la creación del consorcio de la Open Network Foundation (ONF), se añade el soporte para. Además, con esta versión se incluye la capacidad de conectar un switch a varios controladores a la vez de forma concurrente. Lo que permite por un lado el balanceo de carga en el plano de control, y la alta disponibilidad de controladores, y por otro aumenta la tolerancia a fallos y la resiliencia.
- *Versión 1.3:* proporciona más soporte para otros protocolos de red como son MPLS, además de nuevas cabeceras para IPv6. También mejora las capacidades de monitorización y contadores (e.g *per-Flow metering*).

- *Versión 1.4:* mejora la extensibilidad del protocolo en cuanto a emparejamiento, brindando capacidades de clasificación más flexibles al usuario para hacer coincidir los campos de encabezado del paquete y clasificar los flujos.

- *Versión 1.5:* publicada a finales del 2014 y última versión pública hasta la fecha. Incluye más flexibilidad en las estadísticas, y más cabeceras de TCP para coincidencia además de soporte para enrutadores ópticos.

- *Versión 1.6:* está disponible desde septiembre del 2016 pero únicamente para los miembros del consorcio de la ONF.

OVSDB

Después de OpenFlow otro de los protocolos más extensamente utilizados en la interfaz sur de los controladores es OVSDB [29]. Sin embargo, hemos de especificar que éste no es tanto un protocolo como un modelo de esquemas en una base de datos, la base de datos del servidor que lo implementa. En concreto, es el RFC7047 [30] el que especifica el protocolo basado en JSON-RPC que los clientes y servidores utilizan para comunicarse. Así pues, sus siglas se traducen literalmente como Base de Datos de Open vSwitch, y podría definirse mejor como un sistema accesible a través de la red para almacenamiento. Normalmente se suele referir a él como protocolo de gestión y fue creado por el equipo de Nicira para, más tarde, ser adquirido por VMware.

El principal uso de este protocolo es para configurar el conmutador virtual per se. A pesar de que en un principio parezca que OpenFlow se encarga de todas las configuraciones, eso no es cierto. OpenFlow se utiliza solo para programar las tablas y sus entradas, mientras que OVSDB configura el conmutador, creando, eliminando o modificando los puertos, puentes o interfaces de este. Un ejemplo de cómo funciona este protocolo lo podemos observar en la Figura 2.2.

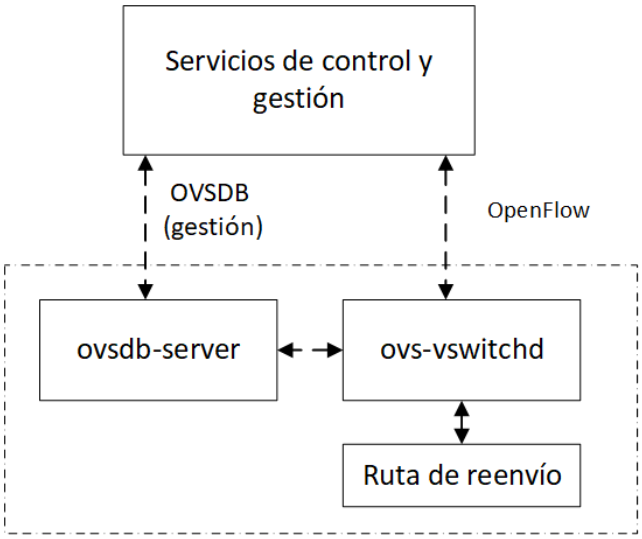

Figura 2.2: Diagrama de bloques sobre la relación entre los protocolos SDN y los diferentes componentes.

A través de sus esquemas se especifican las tablas en la base de datos y los tipos de sus columnas y, además, pueden incluir restricciones de integridad de datos, unicidad y referencias. Esta tecnología es utilizada para la sincronización de estados, ya que permite a cada cliente monitorear el contenido de la base de datos al completo, o un subconjunto de ella. De esta forma, siempre que la base de datos, o una parte, se modifica el servido le indica al cliente qué cambios de han realizado, agregación de tablas, modificación o eliminación. Así, los clientes pueden sincronizarse con el servidor y llevar un seguimiento de los contenidos más recientes de cualquier parte de la base de datos.

A pesar de su nombre, este protocolo es de uso general, y no está particularmente creado para ser usado por OVS. Sin embargo, es una de las características más útiles de este conmutador virtual. El uso principal que se hace de OVSDB es para configurar y monitorizar a través de la utilidad *ovs-vswitchd*, demonio que ejecuta el conmutador, utilizando el esquema que se encuentra almacenado en *ovs-vswitchd.conf.db*. Por otro lado, OVS incluye el esquema de "VTEP", que utilizan muchos conmutadores hardware de terceros, para permitir la configuración de VXLAN y, aunque OVS en sí

no utilice directamente dicho esquema permite la compatibilidad con otros conmutadores hardware (Juniper, Cisco, etc.). La implementación de este protocolo por parte de OVS incluye un servidor *ovsdb-server* y un cliente básico en línea de comandos *ovsdb-client*, así como bibliotecas cliente para C y para Python. Por otro lado, también se incluyen herramientas genéticas para trabajar con la base de datos que tienen esquemas específicos, *ovs-vsctl* trabaja con la base de datos de configuración del demonio, y vtep-ctl trabaja con la base de datos de VTEP. La herramienta *ovsdb-tool* trabaja con sus propios formatos de base de datos y muestra de forma sencilla a los usuarios las transacciones que han cambiado en la base de datos desde la última vez consultada.

Todas estas herramientas que se proporcionan dentro de OVS, serán utilizadas para la configuración del conmutador virtual cuando despleguemos la red programable, principalmente, en el primer escenario.

Switches Hardware

De un tiempo a esta parte, los principales fabricantes de equipamiento de red como son Cisco, Juniper, Aruba, etc. han sacado modelos compatibles con el protocolo OpenFlow, para implementar redes definidas por software en una red física. Estos equipos son de una calidad alta y enorme rendimiento. Sin embargo, debido a esta calidad su precio es muy elevado, lo que entra en contradicción con una de las principales ventajas aportadas por las redes definidas por software, la reducción del CapEx. Es por esto por lo que diversos fabricantes han proporcionado enrutadores compatibles con SDN a un precio mucho más accesible. Más aún, existen propuestas de enrutadores básicos compatibles con OpenFlow a un precio muy ajustado que permiten a investigadores y gente común, no empresas, poder montar una red virtual programable en casa o en el laboratorio y disfrutar de las ventajas que estas ofrecen o experimentar con ellas. Un ejemplo de esto es el switch de Pica8, que posee switches de unos 2.000$ en adelante, o incluso el caso de ZodiacFX con un precio de apenas 100$. En nuestro caso, seleccionamos este último

para llevar a cabo el despliegue del Escenario 1, debido a su accesibilidad y, ya que las características que posee eran suficientes para cumplir los requisitos de este primer escenario. A continuación, describiremos un poco más las características de este enrutador y su funcionamiento [31].

Zodiac FX

Considerado el enrutador SDN más pequeño y accesible del mundo en la actualidad. Este equipo está compuesto de 4 puertos Ethernet de velocidad 10/100 que son compatibles con OpenFlow hasta las últimas versiones como podemos observar en la Figura 2.3. Además, posee una sencilla interfaz de comandos para poder interactuar con el mismo, pudiéndose conectar al enrutador a través del puerto USB. Entre sus otras características técnicas encontramos:

- Procesador Amtel ATSAM4E Cortex M4.
- Soporte para versiones de OpenFlow 1.0, 1.3 & 1.4.
- Soporte para 512 entradas en las tablas de flujo.
- Buffer de trama de 64KB de almacenamiento y reenvío sin bloqueo.
- Soporte VLAN, protocolo 802.1q parar 64 grupos con 4096 IDs diferentes.
- Autenticación por puerto basada en el protocolo 802.1x.
- Implementation del protocol 802.1w Rapid Spanning Tree (RSTP).
- Implementación de prioridades en QoS / CoS con inserción de etiquetas 802.1q.
- LEDs de actividad en cada puerto.
- Interfaz de periféricos en serie (SPI) de alta velocidad.
- Puerto USB para gestión.

- Un tamaño ultra pequeño (10 cm x 8 cm) y ligero.

Por último, su firmware es de código abierto [2], por lo que se puede colaborar activamente en su desarrollo, mediante solicitud de funcionalidades a los desarrolladores o su implementación directa. Cualquiera puede descargar y modificar este software para su uso propio y hacer las personalizaciones necesarias.

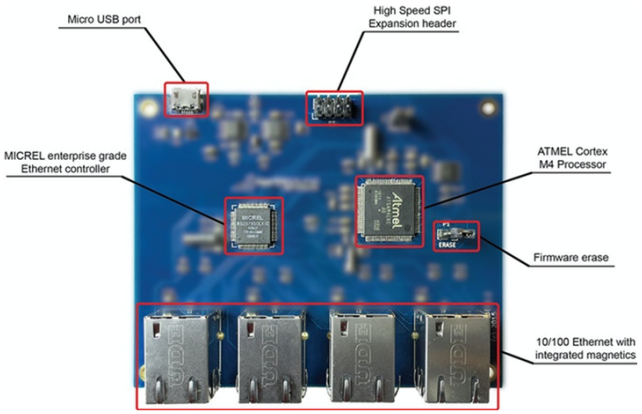

Figura 2.3: Vista en detalle de la placa principal que compone el enrutador ZodiacFX (Fuente [1]).

El código de este enrutador utiliza el software genérico de Atmel (ASF) para implementar la conectividad básica a través del puerto USB, puerto serie, etc. y encima de este código base se puede añadir código personalizado, como el mencionado anteriormente o incluso uno desarrollado por terceros. En este caso, se hizo uso del enrutador con el sistema operativo de microprocesadores FreeRTOS [3] que proporciona una fácil gestión de los recursos de memoria para los servicios principales de línea de comandos (CLI), enrutamiento (*switching*) e implementación del protocolo OpenFlow. En la Figura 2.4 se puede observar este diseño de este *firmware*, dónde el bloque superior corresponde al código que podemos encontrar de forma abierta en el repositorio antes indicado.

[2] https://github.com/NorthboundNetworks/ZodiacFX
[3] https://www.freertos.org

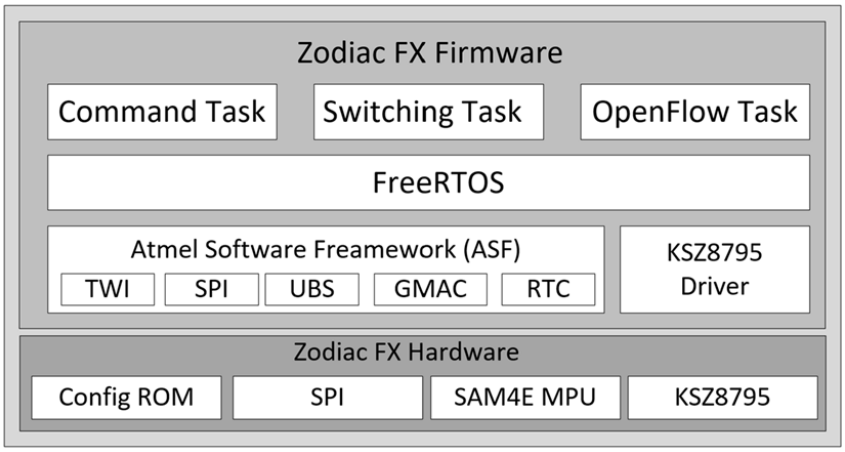

Figura 2.4: Diseño de bloques del Firmware de código abierto del enrutador ZodiacFX.

Switches Software

OpenVSwitch

Se trata de un conmutador virtual (Switch), uno de los más extendidos y usados en las redes virtuales de carácter abierto. Es multicapa, lo que permite la automatización de la red a través de extensiones programables, implementa las interfaces y protocolos de gestión estandarizadas más comunes en redes virtuales. Además, posee una API abierta para extender sus funcionalidades de forma programable. Este software puede ejecutarse en numerosos entornos virtuales como Xen, KVM, VirtualBox, etc. Exponiendo sus interfaces a la capa de red virtual, pero también puede ser ejecutado de forma distribuida sobre múltiples servidores físicos [32]. Entre las principales características que lo hacen tan popular y extensamente utilizado, encontramos:

- Seguridad: creación de redes virtuales aisladas (VLAN) con filtros para diferentes tipos de tráfico.
- Implementación de protocolos para monitorización, replicación de

puertos (*mirroring*[4]) y mejora de la visibilidad total de la red, como son NetFlow, sFlow, SPAN y RSPAN.

- Posee unas características básicas de implementación de Calidad de servicio, existe la posibilidad de crear colas en los puertos e implementar modelado de tráfico (traffic shaping[5]). Estas dos características son las que se han explotado en este trabajo para la utilización de técnicas de Calidad de servicio adaptadas a tráfico de sensores. Más adelante se describirá como son usadas estas funciones mediante un controlador.

- Una de las características más importantes es la implementación de protocolos denominados '*Southbound*', denominados así por los controladores del plano de datos, ya que son los protocolos utilizados para conectarse a ellos. Los protocolos más importantes y que implementa este enrutador virtual son: OpenFlow y OSVDB, ambos analizados en capítulos anteriores.

- Y la fácil integración con sistemas de gestión de infraestructuras virtuales y orquestadores como son OpenStack, OpenQRM, OpenNebula u oVirt.

Como se puede ir apreciando, estas características son muy útiles a la hora de desplegar una red virtual programable, si no clave. Asimismo, sumadas a estas funcionalidades, OpenvSwitch proporciona un complejo diseño, más allá de los conocidos puentes (bridges[6]) que son implementados por los conmutadores, aunque estos son ampliamente utilizados como base

[4]Técnica utilizada en redes para enviar una copia de los paquetes recibidos por un puerto (o una VLAN entera) por otro puerto dedicado a la monitorización, y así analizar dicho tráfico.

[5]Técnica utilizada en redes para limitar el ancho de banda o número de paquetes que son procesados por un puerto o un enrutador, atendiendo a un parámetro como tipo de tráfico, el origen o el destino del flujo de paquetes.

[6]Un puente de red es el método implementado por los conmutadores que permite la interconexión de elementos de red que operan en la capa 2 (nivel de enlace) del modelo OSI, interconectando segmentos de red formando una sola subred, o dividiéndola en subredes.

de su funcionamiento. Mientras que los puentes generalmente son ejecutados únicamente en el espacio del núcleo del equipo hospedador, este conmutador virtual usa no solo es espacio del núcleo, sino también el espacio de usuario para crear reglas de enrutado y procesamiento de paquetes más complejas. Si atendemos a su arquitectura, observable en la Figura 2.5 podemos encontrar los siguientes componentes software principales:

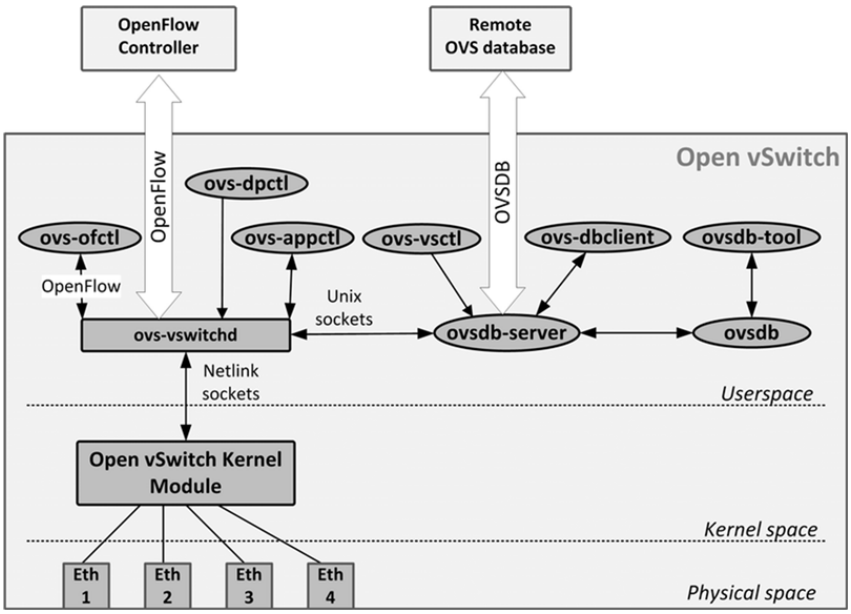

Figura 2.5: Componentes software principales del enrutador virtual OpenvSwitch.

- **ovs-vswitchd:** es el demonio (*daemon*[7]) principal que implementa el propio conmutador junto con una compilación del módulo del núcleo de Linux para llevar a cabo la conmutación basada en flujo.
- **ovsdb-server:** es un servidor con una base de datos ligera para almacenar y recuperar la configuración del conmutador.

[7]Un demonio es un programa o proceso que encontramos en sistemas operativos basados en Unix que se ejecuta junto con el mismo en un segundo plano, en lugar de estar bajo control directo del usuario, estos son activados por un evento o condición específica.

- **ovs-dpctl:** es una herramienta interactiva que permite configurar el módulo del núcleo.
- **ovs-brcompatd:** otro de los demonios que componen el conmutador, permite que el componente *ovs-vswitchd* actúe como sustituto de los puentes que genera Linux.
- **ovs-vsctl:** se trata del comando utilizado para poner en cola y actualizar la configuración de los demonios.
- **ovs-appctl:** es una utilidad que permite enviar comandos a los demonios del conmutador que se están ejecutando.

Aparte de estos componentes principales, existen algunas herramientas adicionales o extensiones que implementan funciones específicas como:

- **ovs-pki:** herramienta enfocada a la seguridad utilizada para crear y gestionar la infraestructura de clave pública de cualquier conmutador OpenFlow.
- **ovs-ofctl:** utilidad concreta que implementa el protocolo OpenFlow, en sus diferentes versiones. Protocolo que se utiliza para comunicarse con el controlador. Esta es una de las utilidades más importantes que utilizaremos de esta herramienta, puesto que nos permite modificar las tablas y entradas del conmutador, para así manejar el tráfico que viaja por nuestra red virtual.
- **testcontroller:** se trata de un controlador sencillo que viene por defecto incluido en el conmutador. En caso de que la red no posea un controlador SDN se puede utilizar para implementar enrutamientos de forma sencilla, normalmente es utilizado para testeo.
- **tcpdump-patch:** se trata de una utilidad para analizar mensajes del protocolo OpenFlow, que se funciona sobre el protocolo de transporte TCP.

De nuevo, teniendo como referencia la Figura 2.5 podemos observar cómo cada componente se relaciona con los demás ya sea en el espacio del núcleo o de usuario. Los dos protocolos principales que permiten al conmutador interactuar con componentes de gestión externo son OpenFlow y OSVBD, ambos descritos en subsecciones anteriores. Más adelante contemplaremos aspectos concretos de la Calidad de Servicio que proporcionan estas herramientas, y como han sido enfocadas a mejorar el tráfico y gestión de los despliegues basados en el Internet de las Cosas.

Otras tecnologías del Plano de Datos

Limitaciones de las SDN centradas en OpenFlow

Existen varias investigaciones que muestran las limitaciones que se presentan en los enrutadores que utilizan el protocolo OpenFlow, entre ellas: (1) Los conmutadores hardware actuales son muy rígidos, permitiendo la operación de "Emparejamiento-Acción" que se lleva a cabo mediante las entradas del conmutado solo en una serie de campos limitados, (2) La especificación de OpenFlow solo define un conjunto limitado de acciones para llevar a cabo el procesado de paquetes.

En [33] los autores proponen unas tablas con reconfiguración de emparejado (RMT, reconfigurable match tables) con una arquitectura de procesado inspirada en RSIC para los chips de los conmutadores, que permite modificar la configuración del plano de datos en tiempo de ejecución sin tener que modificar el hardware. Además, el RTM permite a los programadores modificar los campos de cabecera de forma más comprensiva a como se presentan en el protocolo OpenFlow.

Los autores en [34] defienden que se debe extender de forma detallada el paradigma SDN para controlar la programación de las rutas rápidas y el comportamiento de la puesta en cola por parte de los conmutadores. Argumentar que el camino a seguir requiere extender cuidadosamente SDN

para controlar la programación de ruta rápida y el comportamiento de puesta en cola de un conmutador. Para ello se propone añadir un elemento hardware programable como una FPGA al plano de datos, para permitir la programabilidad extendida.

En [35], los autores se centran en analizar el recorrido de los datos mediante la implementación de OpenFlow en equipos basados en Linux. De esta forma, comparan la capacidad de conmutación de OpenFlow (capa 2) en una red Ethernet, y el rendimiento del enrutado mediante IP en capa 3. En este caso se analiza el tiempo de procesado del reenvío y la latencia de los paquetes en condiciones de sobrecarga y baja carga, con diferentes patrones de tráfico. También la escalabilidad del sistema es analizada utilizando tablas de reenvío de diferentes tamaños, y la equidad en la distribución de recursos usada para ello.

Podemos observar en [36] como los huéspedes finales pueden coordinarse con la red para implementar un amplio rango de tareas de red. Esto se lleva a cabo embebiendo pequeños programas en los paquetes que se ejecutan directamente en el plano de datos. Su contribución principal es una interfaz programable entre todos los usuarios finales y las ASIC de los conmutadores que no sacrifica el rendimiento. Esta interfaz permite que las tareas de la red sean refactorizadas en dos componentes, un programa simple que se ejecuta en las ASICs y una tarea más costosa que se distribuye entre todos los huéspedes finales.

Por último, en [37] los autores muestran direcciones en la investigación que pueden reducir significativamente el Ternary-Content-Addressable Memory (TCAM) y los requisitos del plano de control mediante la compartición de la clasificación y él reúso de elementos existentes en la infraestructura. Se muestra como generalizar la arquitectura de procesado virtual para distribuir la carga entre dichos elementos, de forma que OpenFlow se puede extender a un procesado paralelo.

2.1.3. Plano de Control

Infraestructura y composición del plano

En las secciones anteriores hemos estudiado las particularidades del plano de datos y las tecnologías que lo implementan. En este apartado, vamos a definir más en detalle los componentes que forman el plano de control y las tecnologías más relevantes los que implementan. El plano de control, o plano lógico es aquel que posee todas las funciones de toma de decisiones y análisis de paquetes que se produce en una red programable. Cada capa lógica está compuesta como mínimo de dos elementos clave; las aplicaciones y el sistema operativo de red (Network Operative System (NOS)). En la parte de aplicación encontramos programas comunes de medición, monitoreo y análisis de red, mientras que el Sistema Operativo de Red es el componente que actúa como núcleo del controlador de la red [38].

Éste es el principal componente de este plano, el controlador, y sobre él, se ejecutan todos los servicios que llevan a cabo la monitorización y gestión de la red. El controlador actúa de capa intermediaria entre las aplicaciones, a través de la interfaz norte, y el plano de datos, a través de la interfaz sur. El papel principal del controlador es el de actuar como cerebro o ente pensante, y proporcionar una abstracción y una vista centralizada de la infraestructura de red subyacente. Cabe destacar, que la red puede tener más de un controlador, o que este se encuentre distribuido. Así cada controlador es responsable de una sección de red, y estos deben comunicarse entre sí. Para implementar esta implementación, un controlador a de actuar como controlador principal o maestro, y el resto actuarían como replicantes, reenviando toda la información relevante al controlador maestro.

Interfaz Norte (*Northbound Interface*)

Se trata del elemento de comunicación entre el controlador y las aplicaciones de red, dónde el primero presenta una vista abstraída de la red y lleva a cabo traducción directa del comportamiento de la red y sus requisitos. Comúnmente

esta interfaz se presenta en forma de interfaz de programación de aplicaciones (API), con diferentes llamadas para que las aplicaciones de control y gestión puedan recuperar el estado y los parámetros de la red, y hace uso de ellos.

Ambas interfaces, tanto la norte como la sur, son capas de abstracción clave para la arquitectura SDN. Hoy en día, aun no existiendo un estándar para las interfaces, todos los controladores implementan APIs comunes para ambas interfaces, en el caso de la interfaz norte, utilizando tecnologías si estandarizadas como REST. Sin embargo, en el caso de la interfaz sur, ya posee una propuesta ampliamente aceptada (OpenFlow) que se ha convertido en estándar de facto. Una de las ventajas que poseen estas redes, es que ambas interfaces se implementan de forma abierta, independientemente del proveedor o desarrollador. Es por ello por lo que los elementos de las redes programables son muy interoperables e intercambiables [39, 40, 41]. A la hora de elegir o diseñar un controlador, hay que ser cuidadosos y tener en cuenta los requisitos de la red a manejar, puesto que este componente es la pieza clave que afecta significativamente en el rendimiento general de la red. Debido a esto, existen muchas investigaciones para mejorar el diseño de los controladores, mejorar el rendimiento en diversos aspectos como:

- Coherencia y precisión del estado.
- Escalabilidad.
- Flexibilidad y modularidad.
- Disponibilidad.
- Seguridad.
- Ubicación y latencia.

Herramientas, Tecnologías y Protocolos

Controladores

En este apartado describiremos las funcionalidades y características de los controladores de redes definidas por software, y veremos un ejemplo de los controladores más utilizados. Independientemente del resumen que presentaremos de los controladores más populares, más adelante nos centraremos únicamente en dos controladores específicos, los que han sido utilizados para la implementación de los escenarios, estos son: RYU y OpenDayLight.

OpenDayLight (ODL)

Se trata de un proyecto de la fundación Linux de código abierto que incorpora, entre otras cosas, una plataforma con un controlador modular y flexible, además de otras aplicaciones de red complementarias. Este controlador software está implementado principalmente en Java e incorpora su propia máquina virtual (JVM). Como tal, puede desplegarse en cualquier plataforma hardware y sobre cualquier sistema operativo que soporte Java. Sin embargo, ODL se compone submódulos, entro los cuales también encontramos diferentes conectores con los protocolos más utilizados en las redes (NetCONF, SNMP, etc.) y, por supuesto, OpenFlow. Este controlador, a pesar de ser muy completo y potente, posee más dificultad en su instalación, más uso de recursos, y mayor complejidad en su extensión y adaptación a casos de uso particulares, por lo que, aun siendo posible utilizarlo directamente, y así se ha hecho en el segundo escenario, para el desarrollo de aplicaciones de red personalizada, en el caso del primer Escenario, se ha decidido utilizar un controlador más fácilmente extensible [42, 43].

RYU

RYU es un controlador SDN basado en componentes independientes. Es simple, modular y altamente diseñado para aumentar la agilidad de la red a través de su fácil gestión y versatilidad. RYU proporciona un componente principal (*ryu-manager*) que es el corazón del controlador y se encarga de proporcionar el entorno donde se ejecutarán los diferentes módulos y aplicaciones, y la comunicación entre estos módulos [44]. Además, RYU

define diferentes APIs para acceder a los componentes de software que facilitan a los futuros desarrolladores la creación de aplicaciones de control. Con todas estas ventajas en mente y teniendo en cuenta la posibilidad de personalizar el controlador atendiendo a las necesidades de INTER-IoT, se produjo un cambio de OpenDayLight a RYU tras la investigación y entrega de la versión anterior de este documento. Además, la simplicidad y la ligereza del controlador son características valiosas a la hora de tomar la decisión. Además, la compatibilidad con sistemas como OpenStack (tecnología que se explicará en siguientes apartados), fue la razón definitiva para realizar este cambio en la elección tecnológica. En la Figura 2.6 podemos observar la arquitectura de componentes de este controlador diferenciando las interfaces norte y sur, así como las capas de control y de aplicación que este implementa. Sin embargo, OpenDayLight no ha sido abandonado. Esta tecnología ha sido estudiada y analizada para obtener información y requisitos para futuras implementaciones o mejoras de la solución N2N [45].

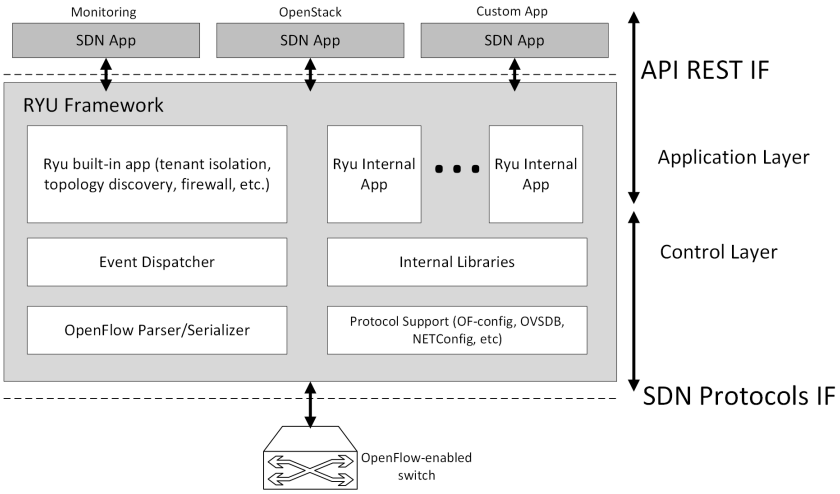

Figura 2.6: Componentes que forman el controlador SDN RYU.

Simuladores y otros frameworks

Mininet

La principal herramienta que hemos utilizado al inicio de la fase de pruebas

de este trabajo ha sido Mininet [46]. Se trata de una herramienta utilizada para la simulación de la red SDN virtual. Mininet permita la orquestación de redes emuladas. Su funcionamiento es sencillo, se trata de ejecutar una colección de hosts finales, conmutadores, enrutadores y, además, todos los enlaces que unen en un único núcleo de Linux. Utiliza una virtualización ligera para hacer que un solo sistema parezca una red completa, ejecutando el mismo núcleo, sistema y código de usuario. Otras opciones son OpenNet, un simulador de red de área local inalámbrica definida por software, EstiNet, una herramienta de software de renombre mundial para la planificación de redes y Ns-3, un simulador de redes de eventos discretos compatible con el entorno Openflow [47].

Depuradores y herramientas de prueba

- *Cbench*[8]: se trata de una herramienta de benchmarking para controladores. Su funcionamiento se basa en la emulación de switches que se conectan al controlador a evaluar para hacer pruebas de carga.

- *NICE*[9]: otra herramienta para evaluar controladores, en especial aplicaciones para controladores POX y NOX mediante diferentes modelos.

- *OFTest*[10]: se trata de un *framework* desarrollado en Python con una pila de pruebas o *unitest* para evaluar el funcionamiento del controlador, tanto a través de la interfaz sur, como de la norte.

- *ptf*[11]: similar al caso anterior, se trata de un bando de pruebas o *unitest*, desarrollados en Python, para evaluar el estado únicamente del plano de datos en este case.

[8] https://github.com/mininet/oflops/tree/master/cbench
[9] https://code.google.com/archive/p/nice-of/
[10] https://github.com/floodlight/oftest
[11] https://github.com/p4lang/ptf

- *OFLOPS*[12]: se trata de una extensión de la herramienta *cbench*, que no evalúa únicamente los controladores, sino que se centra más en probar los switches del plano de datos. Su objetivo es evaluar el número de operaciones OF que puede procesar un switch por segundo.

- Otras herramientas de prueba que han sido evaluadas, pero no se caracterizan por estar enfocadas a controladores o a redes definidas por software son aquellas como: *Iperf, PING, bmon, netstat*, etc.

Testbeds **(Entornos de pruebas)**

Existen numerosos entornos de prueba públicos y abiertos, para que los usuarios puedan llevar a cabo allí sus pruebas previamente diseñadas en un entorno controlado, sin embargo, estos entornos son limitados en el uso de tecnologías permitidas, es por ello por lo que nuestra intención en el proyecto fue de crear nuestro propio entorno de pruebas particular y personalizado. Además, estos entornos son pura y completamente virtuales, no pudiendo incluir elementos físicos conectados a la red, un requisito indispensable para nuestro caso al querer hacer pruebas con dispositivos inteligentes reales.

Algunos ejemplos de estos entornos de pruebas son: PlanetLab Europe (PLE) un entorno de red virtual que soporta características del protocolo OF a través de Silver-OVS (una versión modificada de OVS). Los usuarios pueden experimentar creando una capa de red que conecta diferentes nodos en el entorno (PLE *nodes*). También encontramos el entorno OFELIA una infraestructura europea que permite a los investigadores controlar dinámicamente y extender una red de prueba a través de OF, además posee un marco de control (OCF) que proporciona al usuario herramientas de verificación, acceso y creación de la red y de sus diferentes *slides*.

Otro entorno desarrollado entre Brasil y Europa como banco de experimentos es cuyo objetivo es desplegar y crear un banco de pruebas

[12] https://github.com/mininet/oflops

basado en OpenFlow que cree un espacio común entre Brasil y Europa para las futuras arquitecturas de Internet. Internet2 por otro lado admite servicios de red avanzados, como IPv6 y QoS, y proporciona bancos de pruebas de área amplia (WAN) para la comunidad de investigación de redes, incluido el apoyo a proyectos como PlanetLab, y GENI. Otra infraestructura de investigación para experimentos de red a gran escala es RISE: Se trata de otro proyecto cuyo objetivo es construir un banco de pruebas internacional de OpenFlow a gran escala basado en la red JGN-X en Japón. SURFnet es un entorno de pruebas OpenFlow que puede utilizarse para todo tipo de trabajos relacionados con SDN, como la obtención de experiencia práctica con conmutadores y controladores OpenFlow, el desarrollo de prototipos de SDN y/o la prueba de software SDN de terceros. El entorno de pruebas OpenFlow de California (COTN) creada para ayudar a los investigadores al desarrollo del Internet del mañana utilizando las redes SDN como base de pruebas para la innovación. Se conecta con otros bancos de pruebas OpenFlow dentro de las redes nacionales de investigación, como NLR e Internet2.

El entorno global para innovaciones de red (GENI) [48] en el que se basan otras plataformas como PlanetLab, Internet 2, Emulab, etc. para apoyar la investigación experimental en redes creando un enorme entorno de pruebas y que cuenta con el apoyo de la National Science Foundation. Por último, el entorno de acceso abierto para redes inalámbricas de próxima generación creado para ser utilizado en probar y evaluar protocolos innovadores en entornos del mundo real e incluye una red basada en OpenFlow.

> En el Plano de Datos encontramos aquellas herramientas y tráfico de datos destinado a los servicios (los conmutadores y el tráfico de la red), mientras que en el Plano de Control encontramos aquellas máquinas y tráfico orientado a la monitorización, gestión y mantenimiento de la propia red (controladores y tráfico interno de control).

2.1.4. Plano de Aplicaciones

La capa de aplicaciones es la capa más alta en la arquitectura de una red programable, por lo tanto, la que tiene mayor nivel de abstracción. Esta incluye todas las aplicaciones que explota la información proporcionada por el controlador para llevar a cabo tareas relacionadas con la red en diferentes ámbitos, entre los que encontramos:

- Ingeniería de tráfico.
- Gestión de red.
- Medición y monitoreo.
- Funciones de Middlebox.
- Seguridad.
- Virtualización.

Aplicaciones de Red (NetApps)

Las aplicaciones de red, denominadas por su término en inglés *NetApps*, son aquellas aplicaciones utilizan datos e información extraída de la propia red, proporcionando una funcionalidad que puede ir desde algo pasivo como la simple visualización de estos datos por parte del usuario, normalmente Administradores de Red, hasta algo más activo e incluso automatizado como el análisis de estos datos por parte de una IA para detectar anomalías o peligros en la red y actuar en consecuencia, modificando los parámetros de la red o su topología, creando filtros para determinado tipo de tráfico, o informando a las aplicaciones de una demanda mayor de tráfico para que estas actúen en consecuencia, por ejemplo, creando nuevas instancias de la misma para hacer frente a la demanda [49].

Es por ello por lo que estas aplicaciones de red son muy útiles para entornos dónde el tipo de tráfico no es siempre estable e igual, sino que tiene una naturaleza muy dinámica y cambiante. Entre estas aplicaciones de red encontramos principalmente aplicaciones orientadas a la detección de amenazas y monitoreo, balanceo de carga, análisis del tráfico, VPN, etc.

Numerosos de los actuales proyectos europeos basados en redes programables, se focalizan en el desarrollo y mejora de estas apps [50, 51].

2.1.5. Calidad de Servicio (*QoS*) en SDN

Como se ha expuesto anteriormente, el creciente sector del Internet de las Cosas es, desde el punto de vista de las redes, cada vez más exigente. A veces, miles de dispositivos IoT pueden estar interconectados, con varias aplicaciones o servicios que se ejecutan en estos dispositivos. Estas aplicaciones y servicios generan su propio tráfico de datos que se transmite a través de Internet. Hay que tener en cuenta el aspecto de la red de estas diferentes aplicaciones para que se entreguen con éxito a través de una red densa [52]. Algunas aplicaciones pueden requerir un tratamiento diferente. Por ejemplo, algunas aplicaciones, como el *streaming* de vídeo, requieren un determinado ancho de banda para sus flujos, mientras que otras aplicaciones, como el sistema de alarma, pueden ser más sensibles a los retrasos. Para responder a estos requisitos se necesitan mecanismos de calidad de servicio (QoS) implementados en la red. OpenFlow admite la obtención de un control de la QoS por flujo de una forma escalable, flexible y de granularidad fina.

En esta sección, repasamos las capacidades de QoS del protocolo OpenFlow y lo hacemos observando sus diferentes versiones y las de las plataformas de control SDN de código abierto.

Calidad de Servicio usando OpenFlow

En las últimas versiones de OF, proporcionar calidad de servicio a las rede ha sido uno de los objetivos claves a alcanzar, para ello, a partir de la versión

1.0 OpenFlow incorpora una serie de técnicas que ayudan en la gestión del tráfico dependiendo de su perfil y del modelado del tráfico que pasa a través de los *switches* OF. Pasemos a ver en detalle que técnicas son estas.

Colas (Queues)

Soportados por OpenFlow desde su versión 1.0, estas colas limitan o modelan el ratio de paquetes que entran y salen por los puertos del switch. Es la implementación más básica de calidad de servicio en este tipo de switches, y se encargan de modelar el tráfico, priorizando unos flujos sobre otros, o asegurando un ratio máximo o mínimo para los diferentes tipos de tráfico.

Reglas (Rules)

Son acciones para aplicar cuando se da una determinada situación. Es decir, cuando uno de los paquetes que llega al switch tiene un campo o una característica que coincide con la regla incluida en es mismo. SI este caso se da, esta regla determina cuál es el siguiente paso o acción para realizar con esos paquetes, que puede ser mandarlo al controlador, enviarlo por un puerto determinado, incluirle un tag de VLAN, etc.

Métricas (Meters)

Añadidos a partir de la v1.3 de OpenFlow, son *contadores* que complementan la funcionalidad de las colas, ya existentes en OpenFlow previamente, y que permiten la monitorización de la tasa de tráfico antes de la salida. Más concretamente, con estos contadores podemos supervisar la cantidad de tráfico en un puerto de entrada definida por un flujo. Los flujos pueden dirigir los paquetes a un contador utilizando la instrucción OpenFlow *goto-meter*, donde el contador puede realizar alguna operación basada en la tasa de recepción de paquetes como, por ejemplo, si la tasa sobrepasa un límite determinado incluir una regla en el switch que envíe el tráfico por otro puerto. Cada nueva especificación de OpenFlow (OF) ha introducido nuevas características y cambios relacionados con la QoS, incluyendo:

- OF1.0: un switch OF puede tener una o más colas para sus puertos.

También es posible leer/escribir cabeceras para la prioridad de la VLAN y el ToS de la IP.

- OF1.1: mejora de la correspondencia y el etiquetado de las etiquetas VLAN y MPLS y las clases de tráfico.
- OF1.2: Admite la consulta de todas las colas de un conmutador. Introducción del protocolo OF-CONFIG para reconfigurar las colas dentro del switch. La propiedad *Max-rate* puede establecerse en la cola. Los flujos también pueden asignarse a las colas adjuntas a los puertos.
- OF1.3: Esta versión introdujo los contadores (*meters*). Una entrada de contador consta de "Identificador de contador", "Bandas de contador" y "Contadores". Una banda de contadores, a su vez, consta de un "tipo de banda" (por ejemplo, drop o remark DSCP, etc.),"tasa" (por ejemplo, ráfaga de kb\s), "contadores" y "argumentos específicos del tipo" opcionales, como drop y remark DSCP. Los contadores pueden mantenerse por cola, por metro y por banda de metros, etc. Ayudan al controlador a recopilar estadísticas sobre la red. Puede haber una o más bandas de contadores por entrada en la tabla de contadores. Los medidores pueden combinarse con la acción opcional set_queue, que asocia un paquete a una cola por puerto para implementar complejos *framework* de QoS como *DiffServ*. Estos contadores complementan el *framework* de colas ya existente en OpenFlow permitiendo la monitorización de la tasa de tráfico antes de la salida. Más concretamente, con los contadores podemos supervisar la tasa de entrada del tráfico definida por una regla de flujo. Los paquetes pueden dirigirse a un contador específico utilizando la instrucción opcional meter (*meter_id*), donde el contador puede entonces realizar algunas operaciones basadas en la tasa que recibe los paquetes.
- OF1.4: En esta versión, un contador puede monitorizar otro contador, o más generalmente, las modificaciones realizadas en la tabla de flujo de los conmutadores predefinidos.

- OF1.5: Se pueden utilizar varios contadores.

> La Calidad de Servicio es la descripción o medida general del rendimiento de un Sistema, red o servicio, particularmente, desde el punto de vista de los usuarios. Para, cualitativamente, medir esta calidad, varios aspectos relacionados con la red deben ser considerados como: la tasa de pérdida de paquetes, el bit rate, la latencia, el ancho de banda, la disponibilidad del servicio, el jitter, etc. La QoS ha sido ampliamente utilizada en redes tradicionales, sin embargo, con la llegada del SDN su centralización, la visión global de la red y más oportunidades de gestión de flujos con mayor granularidad, estas características lo convierten en el mejor candidato para proporcionar QoS a las aplicaciones de forma más fácil y flexible en comparación con las arquitecturas de red tradicionales. La QoS puede beneficiarse del concepto de SDN: Mecanismos de enrutamiento de flujos multimedia y enrutamiento entre dominios, reserva de recursos, gestión y programación de colas, calidad de la experiencia (QoE), supervisión de la red y otros como el aprovisionamiento de QoS basado en la virtualización y la gestión de políticas, etc.

2.2. Funciones de Red Virtualizadas (NFV)

La Virtualización de funciones de red consiste en trasladar aquellos servicios de red que se ejecutaban tradicionalmente en un hardware dedicado (e.g *Firewall, Load Balancing, Routing*, etc.) a un contexto software o virtual. Es decir, estas funciones se ejecutan en máquinas de propósito general que

poseen una capa de virtualización sobre la que ejecutar diferentes funciones. Por lo tanto, en un único servidor físico poder proveer a toda una red de diferentes servicios, centralizando así su gestión y monitorizando el estatus general de la red en todo momento [53]. Normalmente, las funciones de red se ejecutan sobre un software de virtualización, ya sea un Hipervisor (e.g. VMware vSphere ESXi, Microsoft Hyper-V, Citrix XenServer, OpenStack, etc.) o, más recientemente sobre motores de virtualización como Docker.

2.2.1. Relación entre NFV y SDN

Si bien es cierto que, en una arquitectura de red con funciones virtualizadas, se hace uso de las redes programadas por software, estos dos términos, aunque fuertemente ligados, no se deben confundir. En una arquitectura NFV encontramos diferentes dispositivos virtualizados, que también podemos encontrar en un despliegue de red tradicional. Estos dispositivos incluyen tanto aquellos que afectan y son parte directa de la red, como enrutadores, conmutadores, puntos de acceso, así como firewalls, sistemas de gestión de dispositivos, etc. y otros dispositivos conectados a la propia red, como servidores de almacenamiento (NAS), bases de datos conectados a la red, etc. La virtualización de todos estos servicios es la misma que la que ya se ha venido aplicando desde hace un largo tiempo con servidores y máquinas, esto es, los recursos hardware pasan de ser dedicados a ser genéricos y son compartidos entre todos ellos. Teniendo este tipo de red, dónde todos los servicios son virtuales, la gestión de esta se lleva normalmente a cabo mediante controladores SDN, puesto que los componentes que forman la red son, efectivamente, virtuales [54].

Las redes definidas por software están compuestas pues de los programas de automatización de funciones, gestión de su ciclo de vida, entrega de servicios de forma ágil, y monitorización de parámetros de la red, para modificarlos en consecuencia, tanto el balanceo de carga como el tráfico, según sus necesidades. Por último, siempre que tengamos una red definida por software, podremos

tener o no funciones de red virtualizadas, aunque normalmente si se encuentren.

> Mientras que las redes definidas por software se centran en desacoplar la lógica en el plano de control, de las acciones en el plano de datos, para llevar a cabo la gestión de la red de forma automática y programable, la virtualización de funciones de red se centra en trasladar servicios de red ejecutados tradicionalmente en máquinas dedicadas, a una infraestructura virtual de propósito general.

2.3. El Internet de las Cosas (IoT)

2.3.1. Problemática: Interoperabilidad, escalabilidad y seguridad

El concepto de Internet de las Cosas se ha ido popularizando año tras año hasta llegar a incluir un amplio espectro de dispositivos, protocolos, tecnologías, servicios y datos que se presentan con una gran heterogeneidad. Como consecuencia de esta naturaleza heterogénea y debido a la falta de unos estándares globales de IoT, en lugar de poseer una facilidad de integración de nuevos elementos, proliferan los silos verticales, provocando sistemas cerrados y herméticos [55].

La interconexión entre sistemas, su falta de compatibilidad e interoperabilidad son dos de los aspectos más críticos que encontramos en los despliegues de Internet de las Cosas [56, 57].

2.4. Situación de redes definidas por software en entornos de Internet de las cosas

Como se ha mencionado en la introducción de esta memoria, uno de los mayores retos a los que se enfrentarán las redes tradicionales en los próximos años el crecimiento descontrolado de la cantidad de tráfico que generarán los dispositivos IoT. En este contexto, la capacidad de las redes SDN para ajustar los flujos y servicios de la red de forma dinámica y rentable se presenta como una solución prometedora.

Otro de los principales problemas que se añaden a este tipo de redes IoT, como podemos leer en [58] está relacionado con la seguridad. Los dispositivos con limitados recursos como sensores o etiquetas RFID normalmente operan con poca energía, potencia de cómputo, memoria y almacenamiento muy limitados, mientras que, las redes tradicionales operan con dispositivos cono suficientes recursos como ordenadores y portátiles, centros de datos, servidores, etc. Por lo tanto, sin ninguna limitación en términos de recursos, las redes tradicionales pueden implementar protocolos de seguridad complejos, mientras que las redes IoT carecen de suficientes recursos. En este contexto, se requiere un equilibrio entre la seguridad y la capacidad de recursos en los sistemas IoT, lo que exige algoritmos y protocolos de seguridad muy ligeros, que en ocasiones no cumplen con los requisitos mínimos en términos de esta. Tenemos el ejemplo de protocolos inalámbricos menos seguros, como ZigBee, 802.15.4e, SigFox, LoRa y 802.11x, que son ampliamente utilizados en despliegues IoT para conectarse con las pasarelas o con Internet [59], pero que provocan muchas fugas de datos y problemas respecto a la privacidad.

Más aún, encontramos diferencias en los sistemas operativos o firmwares entre nodos de ambas redes, dónde los formatos de datos son idénticos en los dispositivos de red tradicionales, mientras que, debido a las diferentes funcionalidades de las aplicaciones o al vertical al que pertenezcan y a la falta de un sistema operativo estándar, el ecosistema IoT se enfrenta a diversos

formatos y estructuras de datos muy diversas. Debido a esta diversidad, todavía no se ha desarrollado un protocolo de seguridad. Debido a estos problemas de seguridad y privacidad, y ya que la mayor parte del IoT genera datos personales y sofisticados, es un requisito inviable enviar todos los datos a los centros de datos para su procesamiento.

Transferir los datos a estos centros puede sobrecargar la infraestructura de comunicaciones, por lo que técnicas como la de computación de borde (*Edge computing*) ofrecen una alternativa usando el pre-procesado de datos en el borde que alivia en gran medida la sobrecarga de tráfico y trata de paliar los problemas de privacidad. Sin embargo, como describen en [60] debido al rápido y desproporcionado aumento de la complejidad en las infraestructuras IoT, se incrementa la necesidad de técnicas de gestión inteligente, ya que los dispositivos IoT en sí no pueden programarse para gestionar reglas complejas y reenvíos de tráfico personalizados debido, de nuevo, a sus limitaciones de memoria y recursos. De nuevo demostrando que la tecnología de red tradicional no puede ofrecer soluciones viables para gestionar las necesidades específicas de las aplicaciones IoT. Por otro lado, si nos adentramos en el área más específica de despliegues IoT para entornos 5G, como es el caso que veremos en nuestro Escenario 2. También encontramos nuevas problemáticas como las descritas en [61] donde se explica que la eficacia y la viabilidad de muchas aplicaciones emergentes del IoT, como la conducción autónoma y la cirugía robótica, requieren una fiabilidad muy alta y una latencia demasiado reducida, por lo que los entornos 5G se introducen como candidatos adecuados para proveer una infraestructura de red que cumple dichos requisitos.

Dichos requisitos que incluyen: (I) en futuras aplicaciones de IoT, como la RV/AR, un mayor flujo de datos, aproximadamente 25 Mbps, (II) las aplicaciones 5G-IoT como el Internet táctil, el diagnóstico médico, etc. requieren una baja latencia en torno a 1 ms, (III) mecanismos de seguridad potente para la protección de los servicios generales de la red y las aplicaciones de pago (IV) mejorar la vida útil de los dispositivos IoT utilizando un menor consumo de batería y, por supuesto, (V) 5G-IoT debe soportar un gran

número de comunicaciones de dispositivos que tengan en cuenta el factor de movilidad (*Roaming*). Como se puede ver en este trabajo, la problemática de los despliegues IoT sigue siendo la misma, pero ahora la necesidad de nuevas arquitecturas que puedan acomodar tecnologías como SDN y NFV han llevado a los entornos 5G a adquirir estas tecnologías como habilitadoras para cumplir con estos requisitos.

Capítulo 3

Diseño de una Arquitectura SD-IOT

En este capítulo presentamos finalmente el diseño de una arquitectura que combina el paradigma y las herramientas de las redes con funciones virtualizadas y definidas por software, con los componentes básicos que encontramos en cualquier despliegue de dispositivos inteligentes. Tras el detallado análisis realizado en capítulos anteriores hemos aislado aquellos componentes que consideramos necesarios para la creación de esta arquitectura, los hemos definido, y hemos diseñado su integración y comunicación con los demás componentes.

3.1. Introducción

3.2. Arquitectura por Capas Del IoT

En el año 2013, de la mano de la Unión Europea nació el proyecto IoT-A (Internet of Things Architecture) en un intento de diseñar una arquitectura y modelo de referencia estandarizada común para definir los despliegues IoT existentes y poder desarrollar nuevos sistemas que siguieran un mismo estándar. Este proyecto dio como resultado un modelo de referencia único [2], dónde se diferenciaban los diferentes módulos (o grupos funcionales como se denomina en el documento oficial, Functional Groups, FG) que debían estar inequívocamente presentes en los despliegues de IoT. Sin embargo, este modelo de referencia se presenta a un nivel más conceptual o funcional, haciendo referencia la función que tiene cada módulo en un sistema IoT, y no tanto diferenciando una arquitectura física por partes dónde localizar cada uno de los componentes del sistema.

En este modelo de referencia encontramos el grupo funcional de dispositivos (1) donde se encontrarían los sensores y actuadores propiamente dichos o todo aquello que genere información, (2) la comunicación, donde encontraríamos aquellos elementos que permitan la interacción o intercambio de datos entre elementos, (3) los servicios IoT, contiene funcionalidades específicas de los dispositivos, como el descubrimiento, la búsqueda y asignación o la resolución de nombres de dispositivos y otros servicios. En Organización de Servicios (4) es un módulo utilizado como eje central para la organización de los otros módulos, tanto para componer y orquestar servicios como especificar el ciclo de vida de estos, está estrechamente relacionado con el grupo funcional de (6) IoT Business process management, que se encarga de la integración conceptual de la inteligencia empresarial y los procesos de gestión del sistema. Por otro lado, las (5) Entidades Virtuales son un nivel de abstracción de los dispositivos reales que los representan como entidades digitales (Digital Twins) para que las (7) Aplicaciones puedan actuar rápidamente y de forma segura con estas

entidades, en lugar de tener que acceder directamente a los dispositivos físicos. Por último, tanto el módulo de (8) Management, cómo el de (9) Seguridad son transversales, y se pueden encontrar en diferentes lugares del despliegue real, en ellos recaen las funciones de gestión de los recursos y atención y recuperación ante fallos, así como la privacidad y anonimidad de los datos. Este modelo de referencia y sus componentes se pueden observar en la Figura 3.1

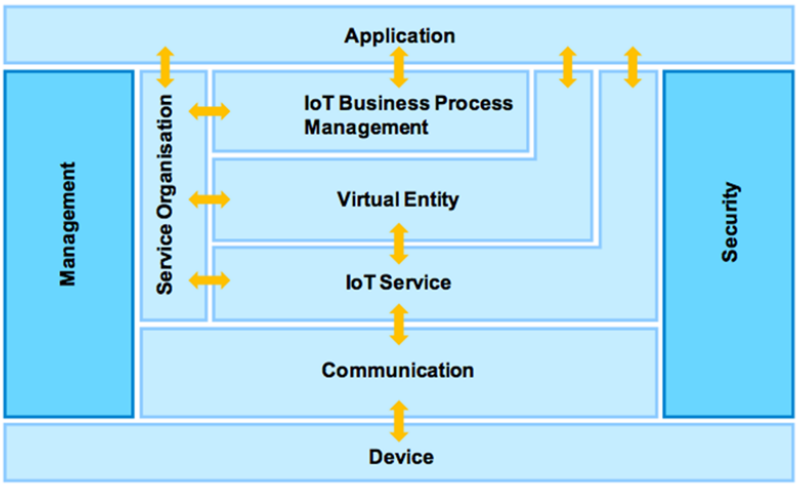

Figura 3.1: IoTA arquitectura modular de referencia (Fuente [2])

Como ya se ha dicho, cada uno de estos módulos se define con la funcionalidad más que con los elementos que lo componen, por tanto, una Gateway IoT podría encontrarse establecida entre los módulos de comunicación, servicios IoT, seguridad y gestión, ya que las funciones que esta realiza pertenecen a distintos módulos. Este modelo de referencia, aun siendo útil, se aleja de la visión pragmática y más concreta que deseábamos diseñar para nuestro proyecto, es por eso por lo que nos basamos en ella para la distribución de funciones en nuestra arquitectura, pero continuamos consultando otras arquitecturas IoT que casaran más con las necesidades de nuestro proyecto.

Algo más tarde, en el año 2014, Cisco, IBM e Intel, presentaron

conjuntamente un modelo de referencia para arquitecturas IoT en el IoT World Forum en Chicago [3]. Desde entonces, este modelo ha sido ampliamente usado en numerosos proyectos. El modelo se realizó conjuntamente por la colaboración de 28 miembros del Grupo de Trabajo de Arquitectura, Administración y Análisis de Foto Mundial de IoT. Este modelo propone un enfoque abierto y versátil conde cada cama no tiene por qué corresponder a un único elemento en el despliegue, sino que un mismo elemento, por ejemplo, un Middleware, puede actuar de almacenamiento de los datos y de procesador o agregador de los mismos a la vez.

En este modelo, mostrado en la Figura 3.2, encontramos diferenciadas 7 capas que corresponden a la capa de **dispositivos físicos y controladores** (1) que envían o reciben datos interactuando con **la capa de red** (2) dónde los datos son transmitidos, filtrados y normalizados haciendo uso de la **computación Edge** (3), una capa lógica que se encontraría físicamente localizada dentro de las dos primeras capas.

Más tarde los datos acabarían llegando a la capa de **almacenamiento de datos**(4), dónde también encontramos la capa lógica de **abstracción de los datos** (5), es decir la semántica y las ontologías que abstraen los datos recogidos para ponerlos en contexto. Por último, encontramos la **capa de Aplicaciones IoT** (6) que accederían a estos datos para utilizarlos con diferentes fines y, por último, otra capa de carácter abstracto que sería la **capa de colaboración y procesos** (7) dónde se encuentra la inteligencia empresarial (*Business Intelligence*) y el valor extraído de esos datos que es útil para la gente y los diferentes verticales.

En este fórum se volvieron a recalcar que las arquitecturas tradicionales de red, computación, aplicaciones y gestión de datos no soportan las necesidades críticas en volumen de datos y alta conectividad necesarias en el IoT. Es por ello por lo que este modelo trataba de unir el modelo estático de capas tradicional de las tecnologías de la información y un modelo dinámico de operaciones donde los datos en constante cambio son tratados, procesados e integrados,

desde el borde hasta los centros de datos (*Edge-to-Center, Cloud Continuum*). Este fue un primer paso para la estandarización del concepto y la terminología que rodea al mundo del IoT, sin embargo, no fue el único.

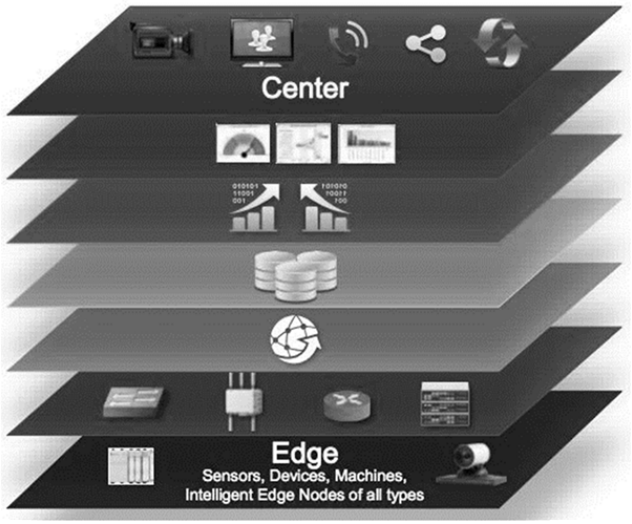

Figura 3.2: Arquitectura 7 capas IoT de referencia de Intel, IBM y Cisco presentada en el World Forum en Chicago.(Fuente [3])

A medida que se avanza en el diseño y la estandarización de una arquitectura IoT genérica, observamos que se van simplificando las capas o módulos que podemos encontrar en estos despliegues. En los últimos años, la aproximación más utilizada y realista, en el sentido de que está relacionada con la disposición física de los componentes en el despliegue, es la de la división en 4 secciones bien diferenciadas que podemos ver en la Figura 3.3 y se compone de: la capa de recolección de datos, la de red o comunicación, de procesamiento y almacenaje de datos y la de aplicaciones o utilización de datos. Más aún se distinguen localizaciones físicas: Dispositivos, Edge, y Data Center o Cloud.

- **Dispositivos y pasarelas:** pertenecen a el área de recolección de datos y los encontraríamos directamente en los dispositivos o como punto de

conexión primera antes de entrar al plano de comunicación y primer preprocesado de los datos.

- **Edge y comunicación:** en este punto podemos encontrar también algunas pasarelas, ya que parte del filtrado de datos y primeras clasificaciones o complementación de la información se lleva a cabo en ellas. Sin embargo, esta capa hace referencia a los primeros nodos de procesamiento, aquellos que aún se encuentran muy cerca, geográficamente hablando, del lugar donde se adquieren los datos y un paso previo a su envío a los centros de datos o la nube. También hablamos de los nodos de comunicación principales aquellos que transmitirán esta información hasta que lleguen a dichos centros.

- **Centros de Datos y la nube:** se trata del último estrato de localización de los datos. Aquí se almacenan los datos a largo plazo para su acceso futuro, se analizan en masa con una capacidad de procesado mayor, o se entregan a las diferentes aplicaciones y servicios, normalmente alojados también en la nube, para su visualización y tratamiento. En este último estadio encontramos los sistemas de soporte al negocio o el valor añadido sobre estos datos recogidos, cuyo uso puede darse en cientos de verticales o mercados.

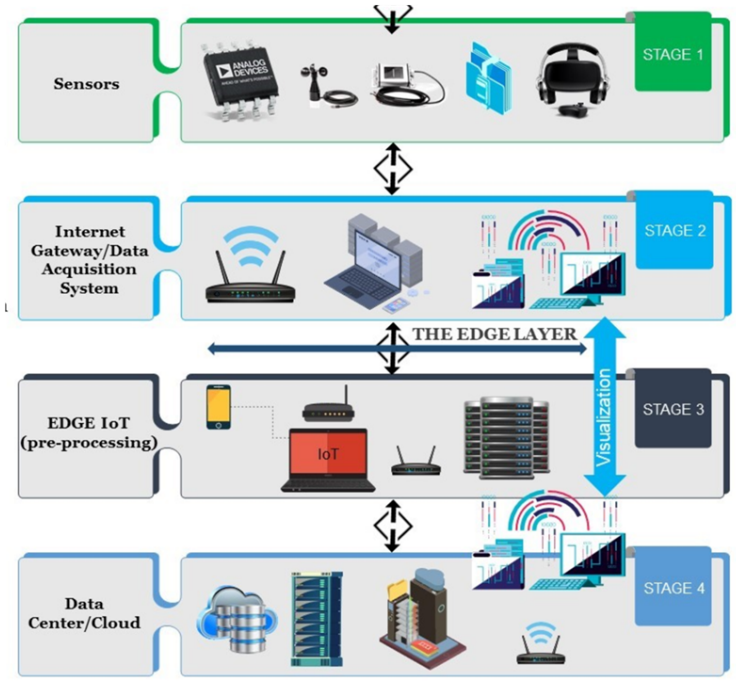

Figura 3.3: Arquitectura más reciente de 4 capas IoT de referencia.

Con todo esto en mente, en el 2016, se inició un exitoso proyecto europeo, INTER-IoT [56], encargado de diseñar soluciones de interoperabilidad en cada una de las capas que componen un sistema IoT. Para ello, se remodeló una nueva arquitectura genérica, basándose en las previamente diseñadas. Obteniendo una arquitectura sencilla pero funcional, que incorpora todos los elementos necesarios, tanto físicos como abstractos, que deben estar presentes en cualquier despliegue de IoT. Esta arquitectura está compuesta de las siguientes capas, orientadas desde el punto de vista de su interoperabilidad [62, 63].

3.2.1. Dispositivos ('Sensing layer')

En esta capa se encuentran tanto los sensores como los actuadores inteligentes, que envían la información recolectada en el mundo real al mundo

virtual. Hoy en día, las aplicaciones y plataformas IoT están estrechamente enlazadas de forma que la interacción de un dispositivo inteligente con varias aplicaciones o plataformas se torna un trabajo imposible. Cada dispositivo es diseñado para comunicarse con un tipo específico de aplicación o plataforma, y estas no contienen todos los servicios importantes que un dispositivo inteligente puede necesitar (e.g. descubrimiento de dispositivos o resolución de nombres). Estas dificultades imposibilitan la interoperabilidad entre dispositivos, por lo que en esta capa se presenta como solución de interoperabilidad las llamadas pasarelas o *gateways*. Estas pasarelas permiten la recolección de datos provenientes de sensores heterogéneos a través de sus diferentes tecnologías de acceso [64, 65].

La pasarela implementa los protocolos más comunes de conexión con dispositivos a bajo nivel (e.g Bluetooth Low Energy, LoRa, ANT+, etc.), para obtener los datos de éstos y llevar a cabo el reenvío de los datos a las capas superiores. Esto dota de flexibilidad a la capa de dispositivos pudiendo interactuar con sensores y actuadores de distintas características [66, 67]. Además, en las pasarelas se encuentra el primer punto de filtrado de datos y pueden implementarse varios servicios para casos de usos cuya baja latencia sea un requisito fundamental. En este caso, las pasarelas físicas, que encontramos lo más cerca de los dispositivos, no poseen recursos suficientes para ejecutar servicios complejos. Sin embargo, en la arquitectura presentada por INTER-IoT, estas pasarelas presentan un punto de innovación importante y es el desacople de las tradicionales pasarelas físicas en dos secciones, la ya conocida parte física y una contra parte virtual, normalmente localizada en el *edge*, aunque más adelante veremos sus ventajas cuando la trasladamos al Core de una infraestructura, como es en el caso de las infraestructuras de red móvil. Entonces, esta parte virtual está dotada de más recursos y poder computacional, para llevar a cabo funciones más complejas, con los datos extraídos de los dispositivos. Estas funciones incluyen, pero no se limitan a, el filtrado de datos, su agregación, análisis y procesado de eventos, generación de eventos o alertas [68].

3.2.2. Red y comunicación ('Communication layer and edge computing')

Una de las problemáticas más resaltables de esta capa es la dificultad en el manejo de la inmensa cantidad de flujos de tráfico generados por los dispositivos inteligentes. Problemática de la que parte este trabajo. Además, en esta capa encontramos escasa escalabilidad en los sistemas IoT, debido a la complejidad de integrar nuevos nodos de forma sencilla y de interconectar componentes de diferente naturaleza como dispositivos, pasarelas y plataformas [69, 70].

La solución propuesta por el proyecto INTER-IoT en su denominada capa de red tiene como objetivo proporcionar una infraestructura dinámica y segura para la movilidad de los objetos inteligentes y el enrutamiento de su información, facilitando la interconexión de pasarelas y plataformas a través de la red. La aproximación que proponía INTER-IoT era usar paradigmas como las SDN y NFV, logrando la interoperabilidad a través de la creación de una red virtual.

3.2.3. Plataforma o Middleware ('Data accumulation or storage layer)'

A nivel de *middleware* o plataforma, la solución propuesta por INTER-IoT consistía en un módulo de descubrimiento y otro de gestión de gestión de recursos para los dispositivos IoT en plataformas heterogéneas [71]. La interoperabilidad a este nivel se basaba en el establecimiento de una capa de abstracción que hacía de interconexión con las diferentes plataformas a modo de puente. La creación de habilitadores o puentes para la vinculación de las plataformas IoT a dicha capa de abstracción era necesaria para interconectar las mismas. Además, en esta capa encontramos la representación virtual o digital de los dispositivos físicos. Los diferentes módulos incluidos en este nivel proporcionan servicios para gestionar esta representación virtual de

objetos, de forma que se puedan definir todas sus características a través de los metadatos y acceder a ellas, además le la información que los dispositivos recogen [72, 73].

Por último, en esta capa se almacena toda la información de contexto de estos dispositivos (control) y los datos (datos) que estos recogen. Ambos tipos de datos se presentan a las capas superiores, como la de aplicación para que se lleven a cado diferentes operaciones con los mismos.

3.2.4. Aplicaciones y Servicios ('Application and Services layer')

En esta capa encontramos las aplicaciones y servicios finales, donde los datos extraídos por los sensores van a parar. Cada uno de estos servicios pertenece a un vertical o área de negocio diferente, y utiliza los datos ya procesados y filtrados para realizar diferentes acciones, como son su representación, establecer patrones, predicciones, etc. La interoperabilidad propuesta por INTER-IoT permite el acceso y uso de servicios que pertenecen a plataformas de IoT de una forma centralizada, normalmente, haciendo uso de sus APIs [74, 75]. Este enfoque permite descubrir, catalogar y componer servicios de diferentes plataformas y además proporciona una API homogeneizada como herramienta de integración para facilitar el desarrollo de nuevas aplicaciones que integren servicios IoT ya existentes [76, 77].

Esta capa es altamente relevante en el contexto de este trabajo ya que aquí también encontramos herramientas finales de monitorización con las que los datos sobre la propia red pueden ser representados y visualizados por los usuarios o administradores que se encargan de la infraestructura. Entre estas herramientas cabe destacar NODE-Red, como herramienta principal de composición y orquestación de aplicaciones y servicios. NODE-Red presenta un panel de composición de servicios sencillo donde a través de la programación podemos conectar desde dispositivos hardware, hasta servicios completos a

través de una API, o incluso servicios en línea (*streaming*).

A través de su panel principal se pueden conectar cada uno de estos "nodos" que representan un servicio único, de forma que podamos crear cadenas de servicios compuestos y definir el flujo por el cual los datos pasan de un servicio a otro. Esta herramienta ha sido utilizada para analizar tanto los datos de control como la información que viaja a través de nuestra red definida por software.

3.2.5. Datos y semántica ('Data abstraction Layer')

Por último, la capa de datos y semántica se suele ver representada en las diferentes arquitecturas de referencia como la capa final de la misma localizada sobre el resto. Si bien es cierto que los datos en sí, su codificación y estructura, los encontramos en cada una de las capas, la semántica, es decir, aquello que da sentido y significado a los datos y que los interpreta, es un componente que se da en las últimas etapas de comunicación de los datos, y suele utilizarse para su representación y contextualización dentro de un vertical específico. La solución de interoperabilidad propuesta por INTER-IoT se basa en la traducción semántica en tiempo real de las ontologías de las diferentes plataformas IoT hacia/desde una ontología modular común [78]. El componente Inter Platform Semantic Mediator (IPSM) se encarga de realizar las traducciones de ontología a ontología de la información utilizando alineaciones ontológicas [79]. Para ello, se definió una semántica explícita demarcada por OWL para cada artefacto IoT que se deseaba interoperar [80].

Los procesos llevados a cabo en esta capa no están relacionados con la red y su infraestructura, por lo que, aunque nombremos esta capa en el estado del arte para contextualizar, queda totalmente fuera del alcance de este trabajo.

3.3. Definición de la virtualización de funciones de red (NFV)

La virtualización de funciones de red (NFV) es un concepto de arquitectura de red que surge en los últimos años de la mano de las mejoras realizadas la relación con las técnicas tradicionales de virtualización. Esta virtualización de las funciones de red aparece como un aspecto emergente en el ámbito de las redes que tiene el potencial de redefinir radicalmente la esencia de lo que se denomina "infraestructura de red". De esta forma, NFV está basado, pero difiere de las técnicas tradicionales de virtualización en servidores, puesto que esta virtualización se hace no solo a nivel hardware, sino también a nivel de servicio. En concreto, esta tecnología hace uso de dichas técnicas para virtualizar funciones completas de un nodo de red, como si fueran bloques que pueden interconectarse entre sí, para crear sistemas completos de comunicación [81].

Una función de red virtualizada (VNF) consiste en un servicio encapsulado en contenedor o una máquina virtual, y que se puede ejecutar en servidores de propósito general, no en hardware especializado. Incluso, estas funciones pueden ejecutarse en infraestructuras en la nube, sobre una capa ya existente de virtualización que las desacopla otros recursos como son almacenaje, memoria o potencia de procesado. Los ejemplos más comunes de funciones de red virtualizada son:

- Balanceadores de carga
- Firewalls
- Dispositivos de detección de intrusos (IDS)
- Aceleradores WAN

NFV se define pues como el conjunto de dichas funciones virtualizadas más las herramientas para gestionarlas, la infraestructura para ejecutarlas, la

gestión y el manejo de su ciclo de vida, sus interfaces y como se interactúan los diferentes componentes entre sí. Aunque, son muchas las ventajas que esta tecnología presenta (reducción de CapEx/OpEx, rápido aprovisionamiento, reducción del TTM para nuevas funciones, entre otras) y hoy en día existen ya varios proveedores que ofrecen dispositivos virtualizados y *middleboxes* como productos comerciales, la NFV presenta varios retos críticos cuando se trata del despliegue automatizado y a gran escala de dispositivos virtualizados dentro de una infraestructura operativa. Mientras que las plataformas de gestión en la nube IaaS de última generación han demostrado ser muy eficaces en el despliegue de máquinas virtuales (VM) para alojar aplicaciones de usuario, el despliegue automatizado de dispositivos de red virtualizados es, en cambio, una tarea mucho más difícil, ya que implica la gestión conjunta de recursos de TI y de red dentro de la misma infraestructura, con el fin de para acoplar los servicios de conectividad de red existentes con las funciones de red desplegadas.

Aunque esta tecnología se confunde en ocasiones con lo que las SDN permiten, si bien son conceptos que están estrechamente unidos, no son lo mismo. Esta tecnología se desarrolló tiempo después de la mano de varias teleoperadoras, para dotar de mayor flexibilidad sus despliegues, basándose en las SDN como infraestructura [82].

3.3.1. Arquitectura y Componentes NFV

En la Figura 3.4 se muestra la arquitectura original diseñada por el Grupo de Especificación Industrial del Instituto Europeo de Normas de Telecomunicaciones para la Virtualización de las Funciones de Red (ETSI ISG NFV) que desarrolló tecnologías NFV y creó una arquitectura de referencia común.

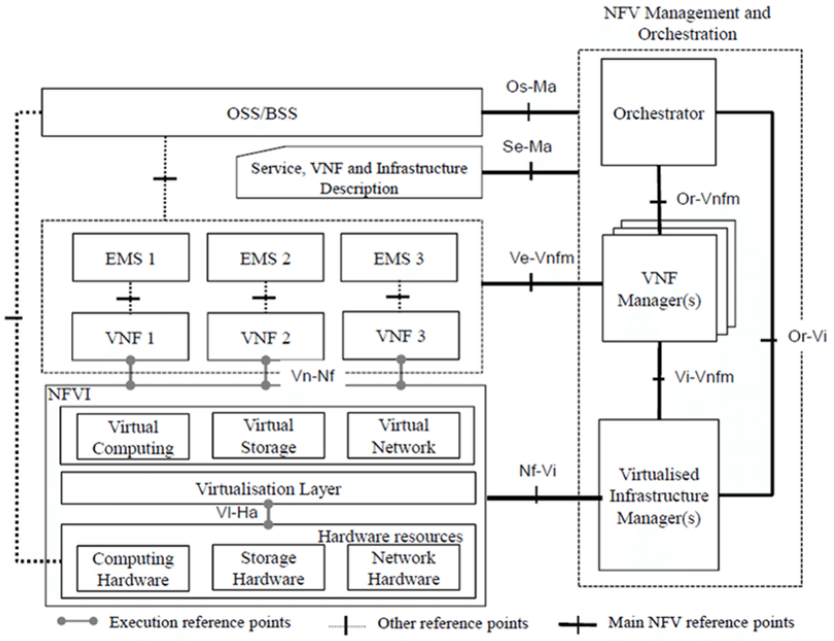

Figura 3.4: Arquitectura de componentes y sus interacciones NFV (Fuente: [4]).

Pasaremos a describir y explicar en concreto cada uno de los elementos que componen esta arquitectura.

VNF

Como se ha descrito anteriormente, una función de red virtualizada (VNF) es un servicio o aplicación software, encapsulado en un contenedor o máquina virtual con los recursos mínimos y necesarios para ejecutarse, así como unas interfaces de entrada y salida definidas, que puede ser desplegada en una infraestructura virtualizada preparada para ello.

NFVI

La infraestructura de virtualización de funciones de red está compuesta por aquellos componentes tanto hardware como software que forman parte del entorno dónde las VNF son desplegadas y ejecutadas. Esta infraestructura puede estar compuesto de un único nodo o varios, tanto físicos como

virtuales, y la red que proporciona la conectividad entre estos nodos también es considerada parte de la infraestructura. De esta forma podemos tener varias máquinas físicas que comparten sus recursos a través de una capa de abstracción y los expone como una única base para que se ejecuten las funciones, esta capa sería la NFVI.

NFV-MANO

Una vez tenemos la base donde ejecutar las funciones, y las funciones en sí, aún queda un componente para manejar el ciclo de vida y los recursos de esta. Esto sería componente de gestión y orquestación de virtualización de funciones de red (NFV-MANO *Architectural Framework*) que es el conjunto de todos los bloques funcionales, repositorios de datos, puntos de referencia e interfaces a través de los cuales estos bloques intercambian información con el propósito de administrar y orquestar tanto la infraestructura como las funciones virtualizadas [83]. Esto es, si una función necesita aumentar sus recursos este componente se encarga de asignar dichos recursos, si estás disponibles en la infraestructura, a dicha función. También, gestionan el ciclo de vida de las funciones virtuales, creando nuevas instancias de estas si es necesario, eliminando las que ya no sean utilizadas o limitando el tráfico de entrada y salida de estas . [84]

Tanto el componente de la infraestructura (NFVI) como el de gestión (MANO) pueden encontrarse en ocasiones definido como un solo componente lógico denominado *plataforma NFV*, y puede ser implementado en muchas ocasiones por una sola aplicación o software. Como es el ejemplo de OpenStack que proporciona tanto el software de virtualización de la infraestructura como parte de los componentes de gestión y orquestación, además de otros muchos componentes para diferentes propósitos [85].

3.4. Requisitos del sistema

Antes de definir la arquitectura que combina los elementos que previamente hemos analizado debemos definir que requisitos son necesarios para el sistema y su funcionamiento. Estos requisitos son funcionalidades básicas que el sistema ha de proporcionar para ser considerado productivo. Primero, definiremos requisitos a alto nivel de carácter genérico y estrictamente necesarios.

Requisitos genéricos básicos

Código	**Descripción**
RB1	Topología de red escalable
RB2	Soporte para la asignación de recursos de la red inteligente en redes de sensores inalámbricos heterogéneos
RB3	Selección automática y dinámica de protocolo de comunicación
RB4	Robustez, resiliencia y disponibilidad
RB5	Capacidades básicas de Qos, Prioridad de enrutamiento y procesamiento de mensajes críticos sobre datos de sensores de baja prioridad
RB6	La interacción entre los puntos finales de IoT puede seguir el concepto M2M
RB7	Alineación con otras arquitecturas de IoT, especialmente con AIOTI
RB8	Comunicación con protocolos eficientes en cuanto a tamaño de mensaje
RB9	Conectividad no basada en identificadores HW
RB10	Movilidad (*Roaming*) a través de redes
RB11	Soporte de red dinámica
RB12	Utilización de protocolos IoT muy o bastante utilizados extendidos. Se deben admitir los protocolos de comunicación comunes de IoT

Tabla 3.1: Requisitos básicos para validar la arquitectura SDN-IoT

Requisitos específicos del Escenario 1

Código	Descripción
RE1.1	Todos los elementos software que componen la parte de gestión y control de la red (Plano de Control) poseerán un API abierta de conexión, para futuras integraciones con otras aplicaciones. Open API, uso de tecnologías y APIs abiertas
RE1.2	La red donde se ejecutan los escenarios poseerá al menos un controlador totalmente programable y adaptado a las necesidades de dicho escenario
RE1.3	Todos los elementos del plano de Datos que componen los Escenarios estarán conectados a la capa de Gestión mediante protocolos SDN (e.g OpenFlow o OVSDB, etc.) y serán configurables hasta cierto punto
RE1.4	En cada escenario se debe utilizar una o más redes de acceso/protocolos considerados como IoT, entre ellas encontramos Lora, Wifi, BLE, PanStamp, NB-IoT, LTE-M, etc.
RE1.5	Orquestación o gestión de las aplicaciones, se han de utilizar por lo menos una herramienta de orquestación (Docker-compose, K8, OSM, etc.) la más adecuada para cada escenario
RE1.6	Los elementos que forman parte del plano de Control y Gestión (controladores, orquestador, VNFs etc.) han de estar en contenedores, para su fácil despliegue y gestión de su ciclo de vida, incluidos si es posible los propios controladores y elementos de gestión

Tabla 3.2: Requisitos básicos para validar la arquitectura SDN-IoT en el Escenario 1 parte primera

Código	Descripción
RE1.7	Las aplicaciones que se ejecuten sobre el controlador o en el plano de Aplicaciones y Servicios (e.g plataformas o *brokers* IoT) han de estar también en contenedores virtuales.
RE1.8	Validación y evaluación, se deberán definir KPI para validar y experimentar los escenarios con anterioridad a su implementación, así como proveer un reporte de los experimentos
RE1.9	Aplicación de monitorización, se deberá tener una aplicación con Interfaz de Usuario Gráfica (*user-friendly GUI*) que nos ayude a configurar la red y a monitorizar su estado (e.g. INTER-FW, ELK, etc.) en tiempo real
RE1.10	Seguridad, la seguridad estará involucrada en todos los pasos de implementación de los Escenarios, llevando a cabo una metodología de desarrollo e integración DevSecOps, teniendo especial cuidado en el cifrado de los datos tanto del plano de usuario como del de control. Además, La comunicación de los datos desde los sensores a las plataformas IoT ha de evitar pérdidas de datos (*data losses*)
RE1.11	Modelo de datos común: los datos procedentes de los diversos sensores se han de traducir a un protocolo y a un modelo de datos común, compatible con la mayoría de las plataformas IoT

Tabla 3.3: Requisitos básicos para validar la arquitectura SDN-IoT en el Escenario 1 parte segunda

Requisitos específicos del Escenario 2

Código	Descripción
RE2.1	Integración del acceso a la red (Internet) mediante protocolos de acceso de 5G new radio y los protocolos de acceso IoT
RE2.2	Plano de usuario y plano de control multiservicio en el estrato de acceso, esto es, se espera que más allá de las redes 5G se intente compartir el espectro milimétrico y de microondas tanto como sea posible, mientras se prestan diferentes servicios
RE2.3	Integrar diferentes tipos de red de transporte (*backhaul*). Para comprobar la capacidad de adaptación de la red en diferentes escenarios, se pretende implementar una red heterogénea que posea diferentes tipos nodos, y una topología compleja
RE2.4	Orquestación y gestión para conectividad múltiple: en el contexto de la conectividad múltiple, la optimización del rendimiento de un extremo a otro requiere considerar aspectos como la mejor tecnología disponible y la mejor selección de ruta para los datos, la mejor selección de protocolos y la programación óptima de paquetes

Tabla 3.4: Requisitos básicos para validar la arquitectura SDN-IoT en el Escenario 2

3.5. Visión general de la arquitectura combinada y tecnologías

Una vez hemos analizado las diferentes arquitecturas presentes en la literatura a lo largo de los últimos años que componen un despliegue del Internet de las Cosas, y hemos detallado el modelo de arquitectura y componentes estandarizados de la ETSI para una red con funciones virtuales, pasaremos a definir nuestra propia arquitectura conjunta que integra elementos de ambos despliegues para formar un modelo de sistema en

el que se priorizan los flujos de datos de sensores y cuyo objetivo principal es mejorar el tratamiento y procesado de dichos datos. En la Figura 3.5 podemos observar un diagrama de bloques con cada uno de los componentes de este modelo. Comenzando por la capa de abajo, o la capa sur, donde encontramos los dispositivos, que se comunican de modo inalámbrico o conectado físicamente a las pasarelas, mediante diferentes protocolos IoT. En el siguiente estrato encontramos las pasarelas, que se encargan de interpretar y traducir todos esos datos enviados por los dispositivos inteligentes y traducirlos a un protocolo de uso común. Dichas pasarelas físicas, o partes físicas de la pasarela se encuentran lo más cerca de los dispositivos como se es posible, lo que denominamos el Edge. Y es tras estas pasarelas físicas, tras las cuales nos encontramos con nuestra red virtual. Las pasarelas están conectadas a la capa de datos o infraestructura de nuestra red virtual. Dicha capa encamina los datos pre-procesados en las pasarelas físicas y los conduce hasta su destino determinado, que se trata de la contraparte virtual de dichas pasarelas. En esta parte virtual se terminarán de procesar y agregar los datos de los dispositivos virtuales, así como proveer de mayor contexto (metadatos) y prepararlos para su envío a las capas superiores de la arquitectura. En este punto podemos dividir el flujo de información en dos tipos: (I) flujo de datos real, o aquella información que proviene de los dispositivos inteligentes (II) flujo de datos de control, aquella información extraída de cada uno de los componentes que conforman la arquitectura, y que sirve para monitorizar y gestionar la misma, como veremos más adelante. En las capas superiores del flujo de datos real encontramos tanto las plataformas de datos IoT, como las diferentes aplicaciones orientadas a un vertical o negocio en concreto.

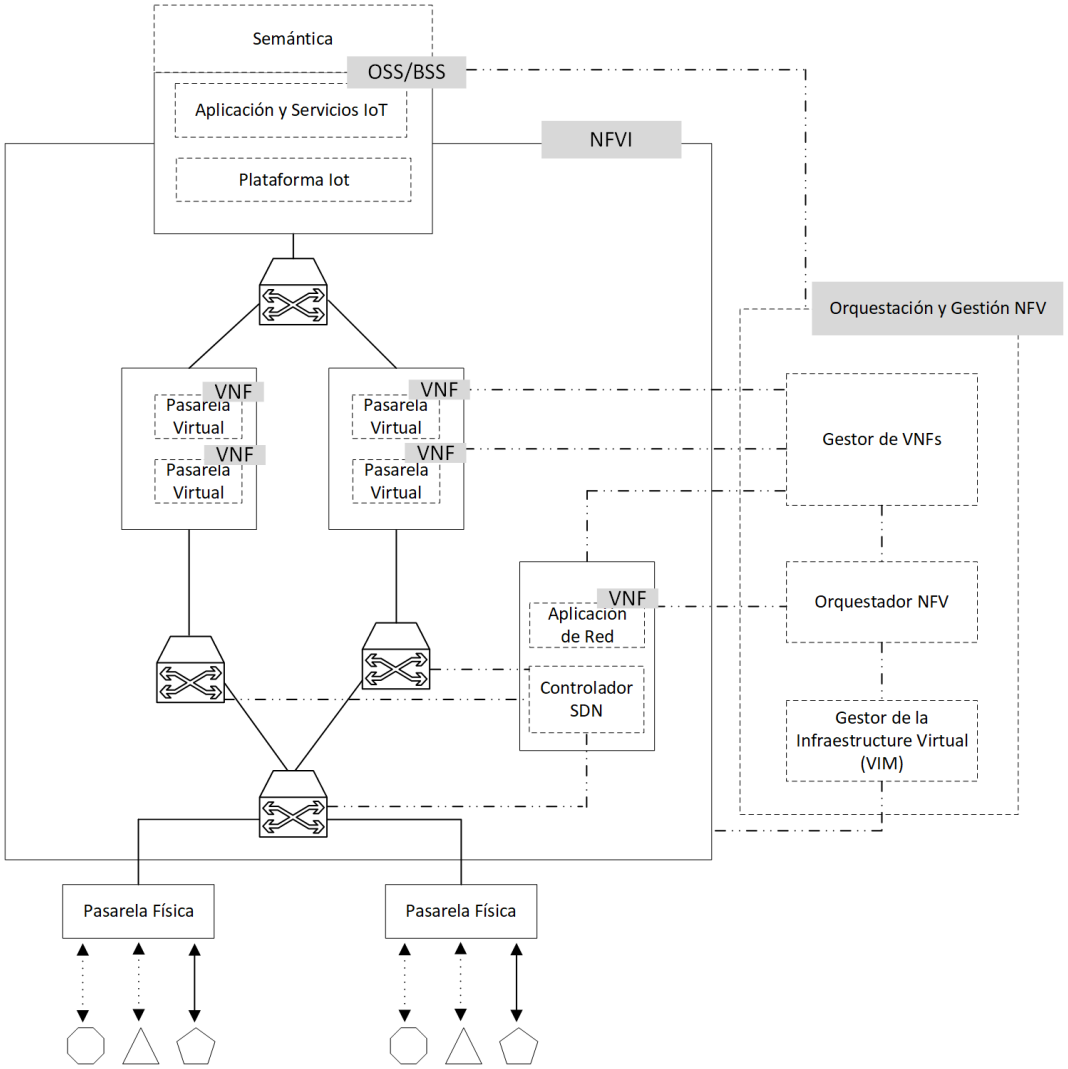

Figura 3.5: Arquitectura conjunta SD-IoT.

Mientras que en las capas siguientes del flujo de datos de control nos encontramos todos los componentes de orquestación y gestión del ciclo de vida de las funciones virtuales, y recursos virtualizados de la infraestructura. Así como aquellas aplicaciones de red que no están orientadas a ningún vertical, ni hacen uso de los datos provenientes de los sensores, sino que están orientadas a la gestión de la red misma. Esta diferenciación se puede hacer además a nivel

de usuario final o persona que hace uso de dichas aplicaciones y datos. Mientras que el usuario o consumidor final del flujo de la capa de datos real es un ente que interpretará los datos de los dispositivos ya actuará en consecuencia, el usuario o consumidor final del flujo de datos de control es un gestor o administrador de red, o cualquier aplicación que se encargue de la gestión de la misma, que interpretará los parámetros de la infraestructura y actuará en consecuencia. Para comprender mejor la figura anterior, en la Figura 3.6 se muestra una leyenda con cada una de las diferentes conexiones entre los módulos de la arquitectura, así como cada uno de los componentes de esta.

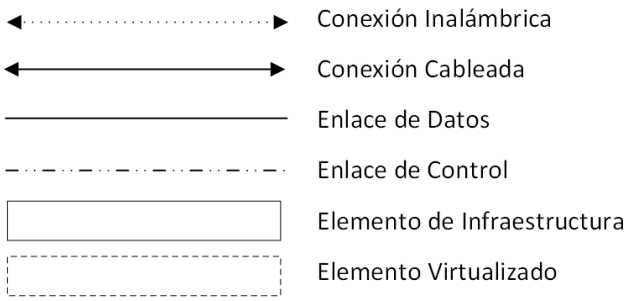

Figura 3.6: Leyenda de conexiones y bloques para la arquitectura.

Por otro lado, para entender donde se encuentran localizados cada uno de los componentes que componen esta arquitectura, la Figura 3.7 muestra tanto la superposición de cada una de las capas identificadas transversalmente en todos los modelos de arquitectura IoT analizados en el estado del arte, así como su localización en un despliegue de *cloud continuum*. Donde, como ya hemos mencionado, los dispositivos inteligentes se encuentran cerca del valor que se quiere medir, mientras que a medida que avanzamos en la complejidad de procesado y vamos a las capas superiores, nos aproximamos a la computación en la nube, sacrificando proximidad por capacidad de cómputo y procesado.

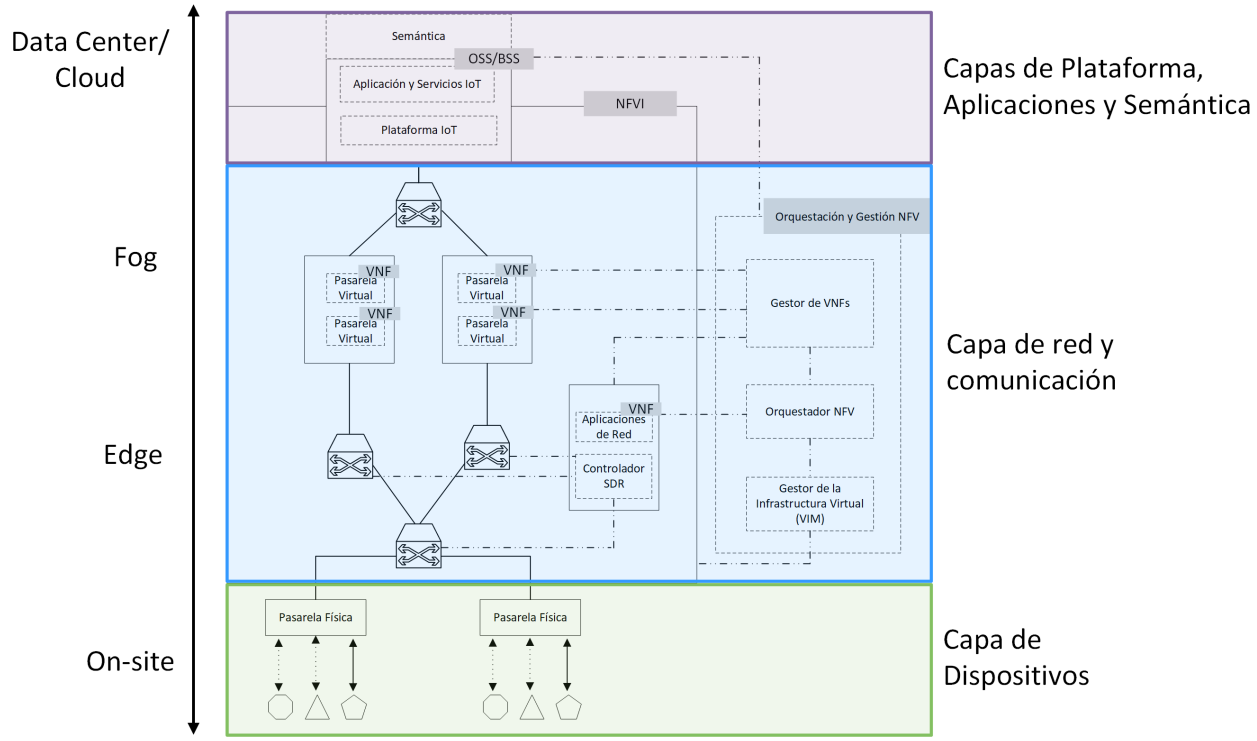

Figura 3.7: Capas de la Arquitectura IoT superpuestas en la arquitectura conjunta y sus localizaciones.

Por último, en la Figura 3.8 podemos observar cuál sería el flujo de datos de sensores o flujo real, que hemos mencionado anteriormente. Mientras que en la Figura 3.9 observamos cuál sería el flujo de los datos de control, que va de cada uno de los elementos de la infraestructura de red y componentes del despliegue IoT hasta las herramientas de gestión y orquestación que se encargan de monitorizar su comportamiento y adecuar la infraestructura a sus necesidades.

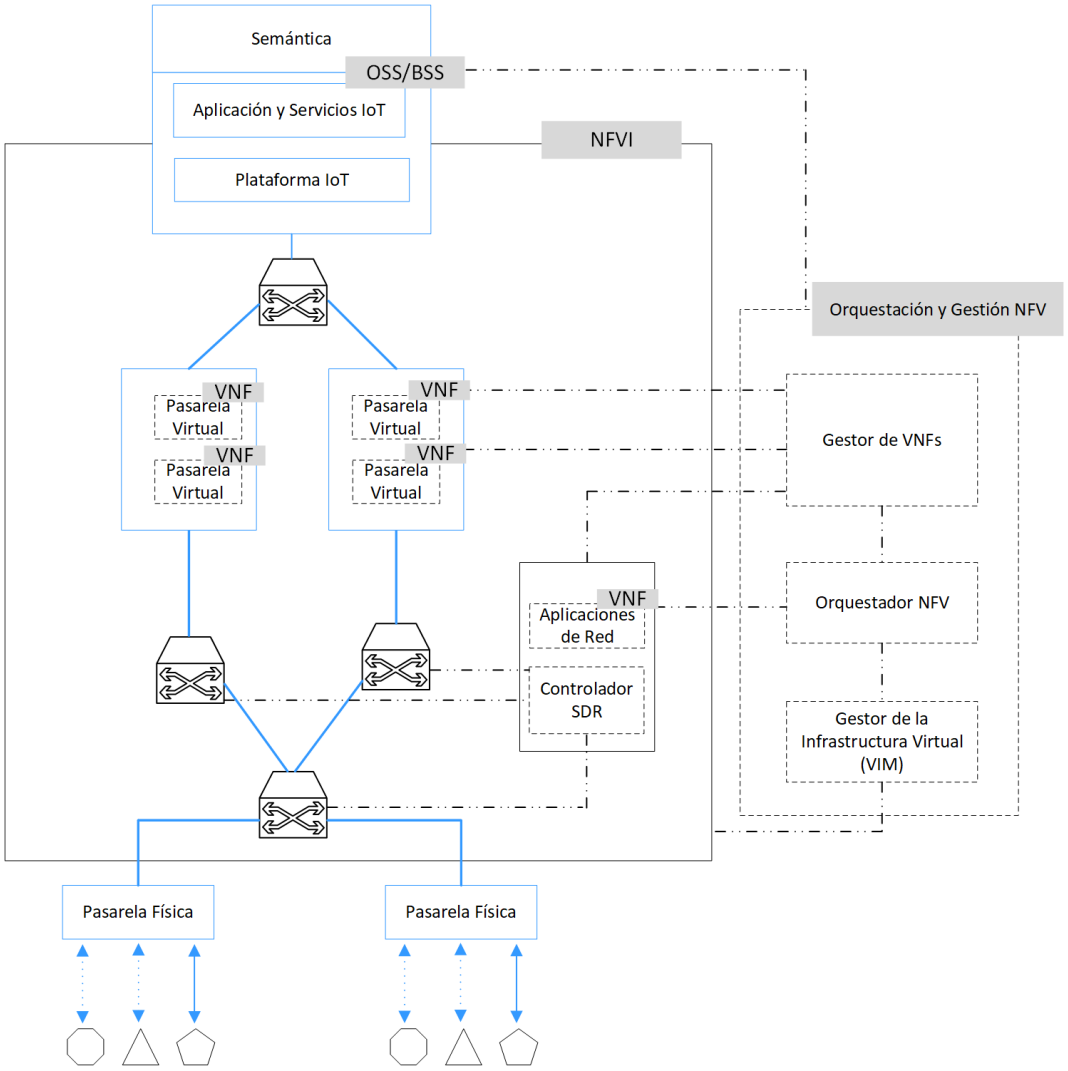

Figura 3.8: Flujo de paquetes de Datos en la arquitectura conjunta.

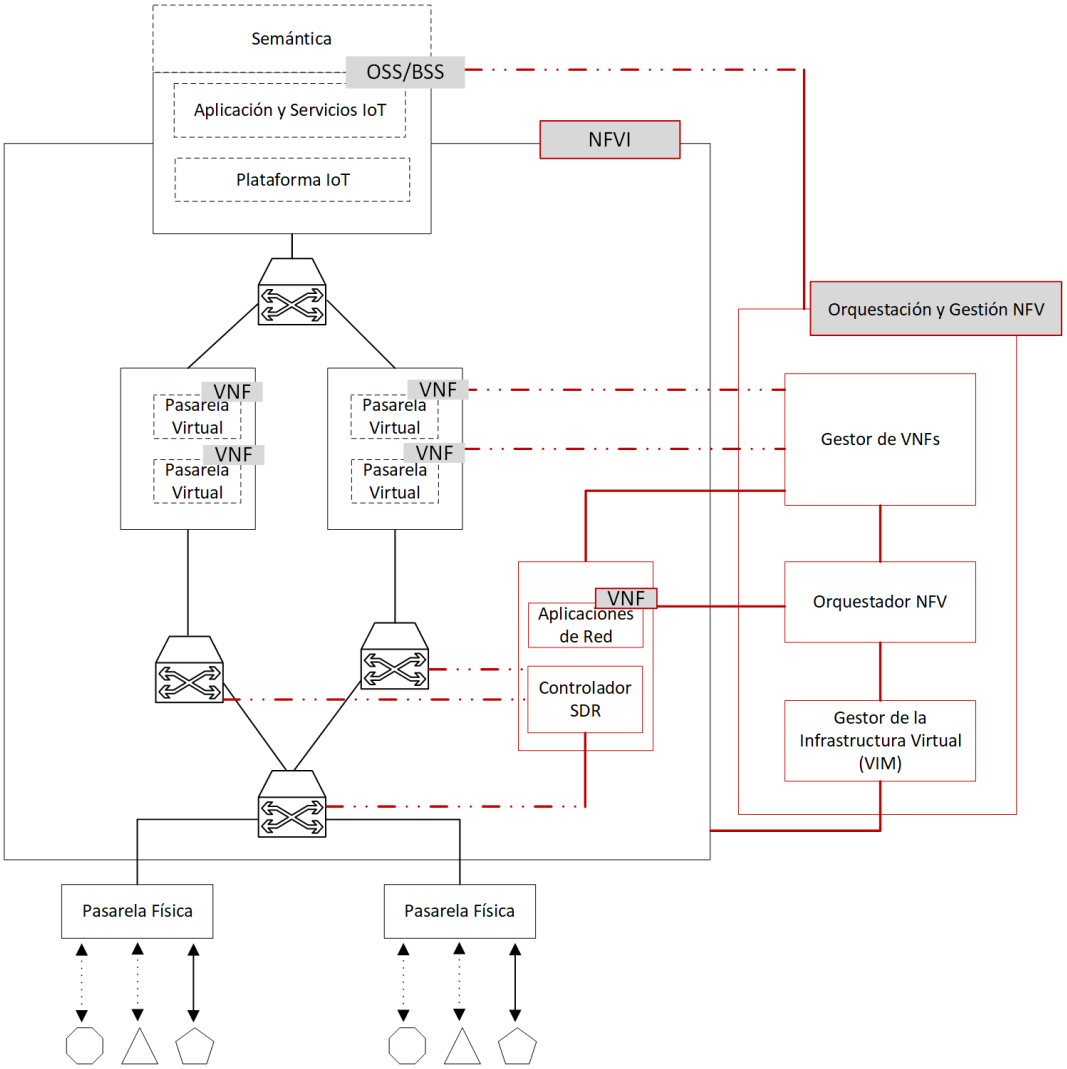

Figura 3.9: Flujo de paquetes de Control en la arquitectura conjunta.

3.6. Funciones Virtuales específicas para Internet de las cosas (IoT-VNF)

Una vez hemos definido la arquitectura genérica que combina la infraestructura de red virtual y la plataforma de orquestación de esta, junto

con los bloques que componen un despliegue de IoT, así como los requisitos que deseamos que nuestro sistema cumplimente, pasaremos a detallar las funciones de red específicas que serán instanciadas en esta arquitectura. Estas funciones de red han sido desarrolladas a la medida del tráfico esperado en un despliegue IoT, es decir, el tráfico heterogéneo, irregular, etc. y tanto su código como sus dependencias ha sido empaquetadas en un contenedor.

Virtual Gateway

Este es uno de los principales componentes de la arquitectura y más que una única función son varias, implementadas de forma modular pero encapsuladas en un único paquete. Está desarrollada puramente en Java, siguiendo el protocolo OSGI, y los diferentes módulos pueden ser cargados y ejecutados en diferentes configuraciones dependiendo de la necesidad del caso de uso [86, 87].

Priorización de tráfico

Esta función se encarga de evaluar el tipo de tráfico que viaja por la red, e informar al controlador SDN de los cambios que ha de implementar en la misma para facilitar la llegada de ciertos paquetes de forma más rápida y segura. En concreto, esta función analiza una de las cabeceras presentes en las tramas IP, el campo DSCP, que indica el tipo de prioridad que tiene el tráfico que está siendo enviado. De esta forma, si la prioridad es alta estas tramas serán procesadas con antelación por los conmutadores, en contraposición con otro tipo de tramas que podrán quedarse en las colas de entrada de los puertos, para ser procesadas más adelante [88].

Métricas de monitorización

Esta función se encarga de extraer métricas de la red para ser enviadas a la aplicación principal de monitorización. De esta forma, en la aplicación principal, que se ejecuta en otro componente diferente posee una visión en conjunto del estado de la red y puede implementar alertas basándose en diferentes condiciones o umbrales. Esto es, la función puede extraer

información sobre el número de paquetes procesados por minuto por un conmutador concreto, esta información se envía a la aplicación principal de monitorización donde se implementa una regla, por la cual si el número de paquetes procesados alcanza cierto umbral envía una alerta a otro componente, en este caso el controlador SDN, de forma que este puede desviar el tráfico en la red para descargar el volumen de paquetes procesador por dicho conmutador [89, 90].

Mapeo de protocolos

Esta función tiene como entrada un tipo de tráfico en un protocolo determinado y se encarga de des-encapsular los datos y reencapsularlos usando otro tipo de protocolo. En concreto, esta función obtiene datos mediante MQTT, HTTP o CoaP y lo traduce a tramas básicas UDP [91, 92].

Más información sobre estas funciones virtuales será desarrollada en la sección Capítulo 4

Capítulo 4

Implementación del Escenario 1

En este capítulo se describe la implementación de la arquitectura diseñada y previamente descrita, en un escenario real. Este primer escenario se ha llevado a cabo bajo el marco del primer proyecto (INTRER-IoT) y hace uso de las herramientas SDN y NFV para contribuir a la interoperabilidad entre elementos heterogéneos de un despliegue IoT. Además, sobre esta arquitectura se han desarrollado funciones virtuales específicas para estos despliegues, como son funciones de priorización de tráfico y de traducción de protocolos.

4.1. Escenario 1: Redes virtuales definidas por software para interoperabilidad en el Internet de las cosas (INTER-IoT)

4.1.1. Introducción

En este primer caso de uso vamos a implementar un entorno con las tecnologías previamente definidas y siguiendo la arquitectura propuesta en el Capítulo tercero. Este entorno será, en el primer caso, totalmente virtualizado y, más tarde en un segundo experimento, implementado parcialmente con un hardware físico real y hacer una comparativa del rendimiento en cada caso. Además, sobre este entorno se llevarán a cabo varios casos de uso o pruebas específicas para analizar el comportamiento y la utilidad de la red, entre los que encontramos: priorización de tráfico y conversión de protocolos de red en tiempo real. El objetivo principal de este escenario es centrarse además en la interoperabilidad entre protocolos de bajo nivel, tanto a nivel de dispositivos como a nivel de red. Mediante el uso de una pasarela dual, con parte física y parte virtual, se recogen los datos de varios dispositivos utilizando diferentes redes de acceso y tecnologías. En concreto la pasarela implementa tecnologías de acceso inalámbricas como LoRa, BLE y PanStamp, y protocolos de conexión no inalámbricos como Modbus. A este nivel se produce pues una solución de interoperabilidad, sin embargo, esta parte de la pasarela queda fuera del contexto del presente trabajo. Lo que nos importa es cuando los datos son enviados desde la pasarela física a su contrapartida virtual, que se encuentra en contenedores virtuales conectados mediante la red programable.

Es mediante esta red que se comunican las diferentes partes virtuales de la pasarela y estas además se comunican con otras plataformas IoT, como el caso de FIWARE Orion, plataforma utilizada por el primer caso de uso. Los datos recogidos por la parte física de la pasarela se envían a través de la red hacia sus contrapartidas virtuales que se encuentran virtualizadas en servidores

de propósito general. Para ello lo primero de todo es crear contenedores o máquinas virtuales dónde ejecutar el código de la parte virtual de la pasarela. Más tarde se llevará a cabo el despliegue de la red, incluyendo la ejecución de los enrutadores virtual y el controlador, su conexión y configuración y por último, el despliegue de los nodos virtuales que poseen la pasarela. Todo ello se describirá más en detalle en los siguientes apartados, pero antes, para contextualizar, explicaremos como se coloca este trabajo dentro del proyecto INTER-IoT.

4.1.2. Proyecto INTER-IoT

El proyecto INTER-IoT, cuyo objetivo era el de alcanzar la interoperabilidad en cada una de las capas definidas para la arquitectura IoT, define la interoperabilidad a nivel de red como la capacidad de interactuar y comunicarse a todos los elementos que forman parte de la infraestructura de la red. En concreto, si echamos un vistazo al modelo OSI tradicional de siete capas para las redes informáticas, la capa de red se encuentra convencionalmente dentro del tercer nivel, situándose directamente sobre la capa de enlace de datos (capa 2) y respondiendo a la capa de transporte (capa 4).

Sin embargo, se trata de un modelo de referencia que, con el nacimiento de las nuevas tecnologías de acceso y de los protocolos del IoT, ha tenido que adaptarse para ajustarse a la arquitectura de la capa de referencia del IoT con el fin de involucrar a una gama muy heterogénea de dispositivos y a sus protocolos, que ya no se basan en el modelo de red tradicional. Por lo tanto, la capa de red en una arquitectura IoT ha adoptado un sentido más amplio.

INTER-IoT entiende la capa de red de un despliegue de IoT como los protocolos, sistemas y dispositivos que trabajan en las capas 2 y 3, e incluso 4 en algunos casos, del modelo OSI tradicional. En este nivel hay que contemplar aspectos de las redes *pervasivas* (generales o presentes en todos las partes) y ubicuas que no suelen intervenir en las redes tradicionales. La particularidad

de las redes IoT es la integración de diferentes tipos de protocolos. Estos protocolos se adaptan a las necesidades de los dispositivos, que suelen ser de muy restrictivas, y a los tipos de datos que comunican la información del dispositivo. Además, la integración de un nuevo dispositivo en un despliegue ya operativo se convierte en una tarea difícil. Asimismo, las operaciones en entornos muy restringidos son también una restricción importante a tener en cuenta.

4.1.3. Intraoperabilidad a Nivel de Red (Network-to-Network Interoperability)

Como ya se ha nombrado en anteriores capítulos, los grandes retos a los que debe enfrentarse la interoperabilidad en la capa de red se deben a los siguientes problemas:

- Dificultad para gestionar la gran cantidad de flujos de tráfico generados por los dispositivos inteligentes.
- Escasa escalabilidad del sistema, con dificultades para integrar nuevos dispositivos.
- Problemas en la interconexión de pasarelas y plataformas a través de redes utilizadas por diferentes proveedores.
- Varios dispositivos con red de acceso radio totalmente diferentes tienen que ser accedidos desde una única pasarela como punto único de acceso.
- Gestión de la movilidad de los dispositivos a través de diferentes puntos de acceso.

Por lo tanto, uno de los principales enfoques para hacer frente a estos problemas es la virtualización de la capa de red, proporcionando un nivel extra de abstracción que facilite la gestión, la escalabilidad, el apoyo a la movilidad

de los objetos inteligentes (*roaming*) y el enrutamiento de la información. La solución de interoperabilidad de INTER-IoT se basa pues en la virtualización (NFV) y la programabilidad (SDN) [86, 93, 94].

Las capacidades implementadas dentro de la capa de red para lograr esta interoperabilidad se resumen en: (1) Desacoplamiento del plano de datos del plano lógico mediante el estudiado protocolo OpenFlow, (2) Virtualización de los servicios de red en la parte superior de la arquitectura, e (3) Implementación de técnicas de ingeniería de tráfico para manejar diferentes flujos de datos generados por los sensores en función de su prioridad.

- Protocolos de comunicación entre la capa de Datos y la de Control: OpenFLow (v1.3) y OVSDB.
- *Switches* de la capa de Datos: OpenVSwitch para la parte virtual y ZodiacFX para la real.
- Controladores: RYU y en concepto de prueba OpenDayLight.
- Funciones Virtuales de Red: Parte Virtual de la Gateway, módulo de priorización de tráfico y módulo de traducción de protocolos.
- Aplicaciones de gestión y de monitoreo: NODE-RED y INTER-FW

Este primer escenario de implementación de la arquitectura se ha centrado principalmente en interoperabilidad, con la traducción de diferentes protocolos IoT (HTTP, MQTT y COAP) a un protocolo de formato común (UDP), y en la implementación de la Calidad de Servicio mediante la priorización de tráfico IoT sobre tráfico de video en la misma red virtualizada. Cabe destacar también que en esta instanciación de la arquitectura no es completa, ya que no existen algunos de los componentes de Gestión y orquestación que hemos visto en el módulo de Gestión y Orquestación NFV.

Red SDN

Este es el principal módulo de todo es escenario, sirve de infraestructura para toda la comunicación entre los demás componentes. Como ya hemos visto en la Sección 2.1 Redes definidas por software (SDN), estas redes realizan el desacople del plano de datos y el plano de control. En este Escenario en particular, la red se ha virtualizado utilizando las tecnologías descritas en la Sección 2.1.3 Herramientas, Tecnologías y Protocolos, en concreto:

- Incluyendo como principal protocolo OpenFlow (versión 1.3), y OVSDB, ambos para conectarse a los switches virtuales y tener la capacidad de añadir/borrar/modificar cada entrada de flujo y recopilar estadísticas para los conmutadores, proporcionando información y estadísticas sobre los flujos, los puertos o la tabla de flujos en sí misma.
- Estos switches virtualizados han sido implementados con la herramienta OpenVSwitch y fueron conectados con el controlador RYU, extendido con aplicaciones personalizadas en el plano de control.
- Por otro lado, la capa N2N también proporciona una herramienta de visualización (GUI) para poder visualizar todo el despliegue y configurarlo a nivel de usuario, así como una herramienta de monitorización para entender su estado, ambas funcionalidades combinadas aseguran una visualización y control completo sobre red virtual y han sido implementadas a través del portal INTER-Framework.
- Por último, la capa de red proporciona una API de QoS para satisfacer los posibles requisitos de QoS del despliegue. Al utilizar la API de QoS de INTER-IoT, el desarrollador puede añadir/borrar/monitorear reglas, colas y medidores. Las reglas determinan si el tráfico especificado se asigna a una determinada cola o contador. Las colas están diseñadas para proporcionar una garantía en la tasa de flujo de los paquetes colocados en la cola. Se pueden utilizar diferentes colas con diferentes tasas para priorizar un tráfico específico.

El objetivo principal de esta solución SDN es de proporcionar una integración perfecta entre los elementos virtuales dentro del despliegue de INTER-IoT. La extensión de esta implementación incluye además la creación de esta red definida por software, un controlador adaptado para los despliegues de IoT Este proceso se llevó a cabo a través de las siguientes etapas:

- **Etapa 1** - Instalación y configuración de switches virtuales con tecnología Open vSwitch dentro de un servidor Linux. Esto aporta la posibilidad de crear topologías personalizadas (creadas con fines de prueba) con las características específicas de cada despliegue incluyendo la redundancia de enlaces o la interconexión de diferentes entornos.
- **Etapa 2** - Desarrollo de módulos dentro del controlador, la creación de una aplicación de gestión de switches para insertar reglas en los estos atendiendo a los flujos y a los parámetros de QoS definidos, la personalización del módulo de topología para mostrar los switches, enlaces y hosts conectados por la red de una manera más amigable para el usuario y la personalización del API REST para obtener información sobre los conmutadores.
- **Etapa 3** - Desarrollo de una interfaz de línea de comandos (CLI) para introducir comandos directos al controlador.
- **Etapa 4** - Interconexión de las pasarelas físicas y virtuales a través en esta red. Realizando pruebas de creación de rutas directas entre las pasarelas virtuales automáticamente a través del controlador. También, la conexión de la pasarela virtual con una plataforma IoT (en este caso FIWARE Orion [1]).
- **Etapa 5** - Desarrollo de scripts de automatización para ejecutar y detener toda la solución, incluyendo ejecución de los módulos, creación de la red, etc.

[1] https://github.com/FIWARE/context.Orion-LD

- **Etapa 6** - Definición de la API REST en *Swagger* para ser utilizada por los niveles superiores de INTER-IoT
- **Etapa 7** - Desarrollo de una GUI embebida en INTER-FW con módulos de Topología, Información y QoS para facilitar la visualización, control y configuración de la red.
- **Etapa 8** - Mejora de ambos módulos; información/estadística y QoS, para una mayor automatización y dinamismo a la hora de actuar sobre la red.

Extensibilidad

Una vez la red y sus componentes han sido implementados, debemos poner el punto de mira en aquellas características que implican una problemática seria en los despliegues IoT, como en este caso la escalabilidad y extensibilidad. La extensibilidad de la red SDN se puede obtener mediante dos formas principalmente:

- Incluyendo nuevos elementos o nodos en la red: normalmente nuevos switches virtuales que se conectan a los ya existentes y a nuevos nodos finales. Estos nuevos switches se pueden crear de forma dinámica dentro de la red virtual gracias a Open vSwitch mediante comandos o a través de la API, creando nuevas interfaces y conexiones con los nuevos elementos o conectando un switch físico a una interfaz física que tenga acceso a la red virtual.
- Incluyendo una red SDN completa gestionada por otro controlador que se comunique con el ya existente: en este caso se puede incluir toda otra red con su controlador independiente e interconectarlo con nuestra red. Esto se puede llevar a cabo a través de las interfaces Este/Oeste del controlador. Sin embargo, la conexión directa entre varios controladores no es algo óptimo, ya que se debería hacer uso de un hipervisor, que actúe como otra capa de abstracción de red para gestionar u orquestar todos los controladores que gestionan las redes.

API

Una vez se identificó que tecnologías se usarían para la red y los demás componentes de la arquitectura, se pasó a diseñar e implementar una API para acceder a la información y los recursos dentro de la SDN como parte de la solución de interoperabilidad. Para esta API, se tuvo en cuenta las posibilidades que nos ofrecía la interfaz REST ya proporcionada por RYU y se procedió a extender la misma a nivel de código. En Tabla 4.1 se recogen los recursos (*endpoints*) a los que se puede acceder a través de esta API y que son utilizados por otros componentes de INTER-IoT como INTER-FW. Además, la API a nivel de red, junto con otras API de las demás capas, se homogeneizaron siguiendo el mismo protocolo de diseño y se recogieron para estar disponibles a través del portal WSO2.

Recurso	**Métodos**	**Endpoint**
Switches	GET	/stats/switches
		/stats/desc/id
Flows	GET	/stats/flow/id
		/stats/flowentry/add
	POST	/stats/flowentry/modify
	POST	/stats/flowentry/delete
Ports	GET	/stats/port/id
		/stats/port/id/port
	POST	/stats/portdesc/modify
Tables	GET	/stats/table/id
		/stats/tablefeatures/id
Roles	POST	/stats/role
QoS Queues	GET	/qos/queue/status/id
		/qos/queue/id
	POST	/qos/queue/status/id
	POST	/qos/queue/id
	DELETE	/qos/queue/status/id
	DELETE	/qos/queue/id
QoS Rules	GET	/qos/rules/id
	POST	/qos/rules/id
	DELETE	/qos/rules/id
QoS Meters	GET	/qos/meter/id
		/qos/meter/id
	POST	/qos/meter/id
	DELETE	/qos/meter/id

Tabla 4.1: API *endpoints* para la solución de interoperabilidad a nivel de red.

En esta lista observamos los diferentes recursos de los cuales podemos extraer y modificar información a través de la interfaz programable. Para saber más en detalle a qué hacen referencia estos recursos los describimos a continuación:

- *Switches*; cada uno de los nodos (virtuales o físicos) de la infraestructura o capa de datos, las operaciones ejecutadas sobre los switches afectan a las configuraciones generales almacenadas en ellos. Se puede obtener información sobre el nombre e identificador del nodo, número de puertos, fabricante y versión, etc.

- *Flows*; cada una de las entradas en la tabla de flujo almacenadas en los *switches*, estas entradas siguen el estándar definido por OpenFlow. Como ya se ha explicado cuando hemos hablado de OF, estas entradas se componen de tres partes principales: campo coincidir, acción a realizar y estadísticas.

- *Tablas*; simplemente una colección de entradas OF. Por defecto, siempre existirá instalada en el switch una table con una entrada indicando que todos los paquetes que lleguen en un principio serán enviados al controlador.

- *Puertos*; Se trata de los puertos virtuales creados en el switch, que se pueden configurar a través de OVSDB por parte de las aplicaciones que se ejecutan sobre el controlador. Además, las características de estos puertos se pueden consultar y modificar directamente a través de la API.

- *Queues*; Estas colas son las que proporciona el protocolo OF, están siempre asociadas a un puerto y definen la prioridad en el procesado de los paquetes que llegan a los mismos, dependiendo de su configuración. Además, se puede definir un ratio de procesado de paquetes mínimo y máximo.

- *Reglas*: Son las diferentes configuraciones que se pueden implementar sobre una cola ya creada y asociada a un puerto, para definir los mencionados comportamientos.

- *Métricas*: Son elementos lógicos incluidos en el switch que proporcionan información sobre loa paquetes y controlan el ratio de ingreso de los mismos.

Código y documentación

El código desarrollado para la creación de la red virtual se puede dividir en dos partes principales: los módulos de *Backend* y los de *Frontend*. Los módulos *Backend* comprenden aquellos componentes relacionados con SDN y que actúan a nivel de red. Esto incluye principalmente el controlador RYU, pero también las API, y las aplicaciones de red personalizadas. Mientras que los módulos *Frontend* están diseñados para interactuar con el usuario y habilitar los módulos anteriores de forma abstracta. Estos módulos componen una interfaz gráfica de usuario (GUI) que permite la edición de la red SDN de una manera interactiva, así como la configuración de la calidad de servicio (QoS), con la creación de colar, reglas, etc. de forma sencilla.

Esta interfaz puede ejecutarse de forma independiente, pero, en el caso del proyecto INTER-IoT, se utilizó embebida dentro del portal INTER-FW. En algunos casos, el despliegue de esta interfaz no es posible, como en dispositivos con pocos recursos, por lo que también se desarrolló una CLI para configurar la red SDN y los parámetros de QoS en la terminal. La documentación describe con mayor precisión estos módulos, y a continuación se presenta un resumen. El código y la documentación completos se pueden encontrar en:

- **Código**
  - Controlador: `https://github.com/INTER-IoT/n2n-ryu`

  - Extensiones de la pasarela: https://github.com/INTER-IoT/gateway-extensions-physical and https://github.com/INTER-IoT/gateway-extensions-virtual

- **Documentación**

  - https://INTER-IoT.readthedocs.io/en/latest/#n2n
  - https://INTER-IoT.readthedocs.io/en/latest/#gateway

Módulos *Backend*

Para ejecutar los módulos de *backend* implementados para el Escenario 1, debemos descargar el código que se adjunta en la lista anterior. Ir al directorio principal del controlador y seleccionar las aplicaciones que se desean instalar cobre el mismo, y más tarde ejecutarlo. Las instrucciones de como llevar a cabo estas acciones se pueden encontrar en la documentación técnica, también adjunta en la tabla anterior.

En específico, se debe ejecutar la aplicación de INTER-IOT Simple Switch que se trata de una aplicación de *routing* desarrollada en específico para instalar automáticamente los flujos OF dentro de los switches. Esta aplicación se encuentra en la carpeta de aplicaciones (`ryu/ryu/app/...`). INTER-IOT Simple Switch también integra una API REST de QoS, y un *websocket* para la GUI. Ver la sección anterior para una descripción de la API. Para ejecutar los módulos de QoS, denominados *rest_qos* y *rest_conf_switch*, se han de llamar al iniciar el de *INTER-IOT Simple Switch*. También, desde esta parte del código también se puede llamar a una aplicación de interfaz gráfica ligera, diferente de la integrada en INTER-FW, que se ejecuta sobre el controlador directamente (`/app/gui_topology/gui_topology.py`)

Módulos *Frontend*

Por otro lado, los módulos de *frontend* se pueden dividir y ejecutar de la siguiente forma: interfaz por la línea de comandos (CLI) se encuentra en la carpeta*CLI*, desarrollado en Python 2.7, se puede ejecutar llamando a su script

de inicio *CLI.sh*. Esta interfaz de línea de comandos ejecutará y entenderá los comandos descritos en los archivos ubicados en la carpeta de comandos, y cuyo nombre comienza con *cmd_*. Los comandos integrados son:

- *commands:* Lista todos los comandos disponibles del controlador
- *mininet <params>:* Ejecuta mininet con los parámetros especificados en el comando
- *controller:* Inicia controladores SDN predefinidos (los scripts de inicio de los controladores SDN pueden añadirse en la carpeta commands/pkg_controller)
- *switch <params>:* Permite configurar los conmutadores, establecer las versiones de Openflow, obtener la configuración y las tablas de flujo del conmutador.
- *switch <params>:* permite configurar los conmutadores, establecer las versiones de Openflow, obtener la configuración y las tablas de flujo del conmutador.
- *qos:* Permite establecer reglas, establecer colas, obtener reglas y obtener colas de un conmutador QoS. Este comando también permite establecer la dirección OVSDB, y establecer el puerto y la dirección del controlador.
- *exit:* Finaliza la herramienta de interfaz de comandos.

Por otro lado, encontramos los módulos gráficos (GUI) y de monitoreo que se encuentran implementados en la parte de INTER-FW, y que serán explicados en más detalle en la siguiente sección.

Herramienta de monitorización (INTER-FW)

En este caso en específico no vamos a definir todas las características y funcionalidades de la herramienta INTER-FW [95], ni las tecnologías que la

implementan, puedo que esto cae fuera del alcance de la presente memoria. Sin embargo, mostraremos la parte relevante a este trabajo de doctorado que desarrollamos para visualizar, interactuar y monitorizar el estatus de la red y sus componentes [96, 97].

- **INTER-FW**
  - Código: (Private) https://git.INTER-IoT.eu/INTER-IoT/framework
  - Documentación: https://INTER-IoT.readthedocs.io/projects/framework/en/latest/
  - Imagen Docker: `(Private) docker.INTER-IoT.eu/inter-fw-web:1.0.0`
- **Gestor de APIs**
  - Código: (Private) https://git.INTER-IoT.eu/INTER-IoT/api_manager
  - Documentación: (Private) https://docs2.INTER-IoT.eu/docs/framework/latest/
  - Imagen Docker: `(Private) docker.INTER-IoT.eu/inter-api:2.0.0`

El portal de INTER-FW consiste en una interfaz gráfica de usuario a través de la cual estos pueden interactuar con los diferentes módulos de la red SDN. En la lista anterior se puede observar un ejemplo de la herramienta, y de cómo se muestran las entidades gestionadas por ella, que hacen uso de la API para extraer información de los nodos y presentarla en un formato más amigable al usuario. Desde este portal, ubicándonos en la sección de red podemos observar una topología completa de esta, así como modificar algunos de los parámetros configurables en los nodos que la conforman.

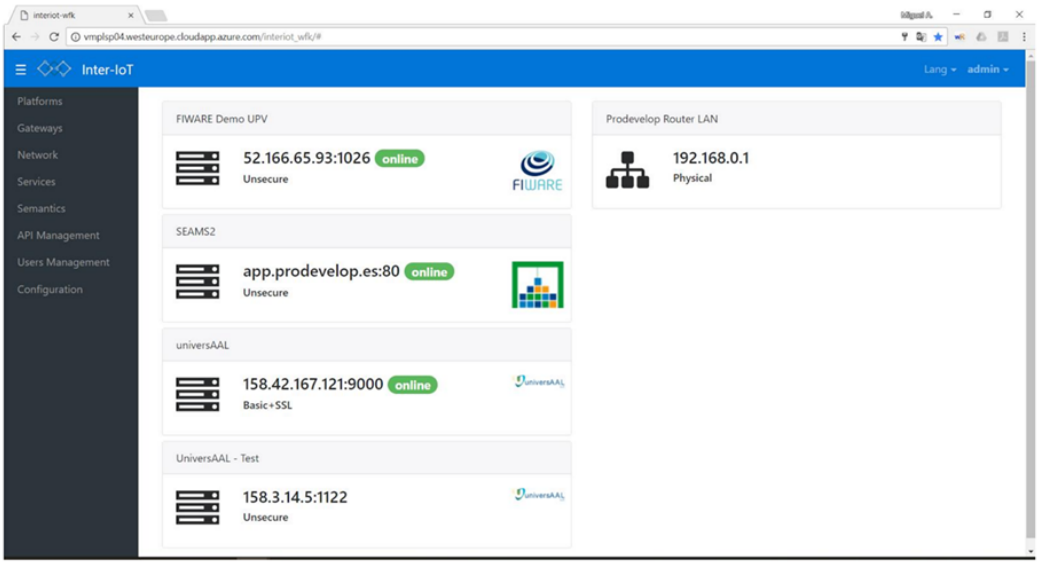

Figura 4.1: Captura de pantalla ejemplo del portal de INTER-FW.

4.1.4. Despliegue para la Interoperabilidad en IoT

Para este primer Escenario se llevaron a cabo diferentes pruebas sobre la arquitectura previamente implementada. Como la priorización de tráfico siguiendo conceptos de Calidad de Servicio del tráfico IoT sobre el video. Esta prueba o caso de uso fue presentado en la feria de la semana del IoT, en mayo de 2018 en Bilbao[2]https://iotweek.org/iot-week-2018-bilbao/. Donde la red SDN manejaba dos flujos principales de tráfico: tráfico de sensores IoT y tráfico de video, y mediante la configuración de reglas de QoS, se priorizaba el tráfico proveniente de sensores.

Otro de los casos de uso que se implementó basándose en esta arquitectura consistió en la configuración de un despliegue de dos pasarelas físicas ejecutándose sobre dos unidades Raspberry Pi, una de ellas conectada a

[2]\unskip\penalty\@M\vrulewidth\z@height\z@depth\dp

sensores de temperatura y humedad PanStamp, y la otra conectada a una placa Arduino que con LEDs como actuadores. Ambas pasarelas físicas estaban conectadas a su contraparte virtual, en forma de contenedor virtual (*dockerized*) en una máquina en la nube de Azure. Estas pasarelas virtuales se controlan y gestionan a través de la API de la red SDN e INTER-FW.

Finalmente, estas pasarelas virtuales se conectaban a través de esta red SDN al Middleware IoT (*Orion de FIWARE*) [98] con un Procesador de Eventos Complejos (CEP) configurado con reglas simples, de manera que, dependiendo de los valores obtenidos por el sensor de temperatura, se procedía al encendido de un tipo de LED u otro. La demostración consistió pues en esta actualización del estado de los actuadores (LED) en base a la lectura de los sensores, a la vez que se demostró cómo fluyen los datos a través de toda la red. En la Figura 4.2 se puede observar un ejemplo del despliegue realizado con cada uno de sus componentes.

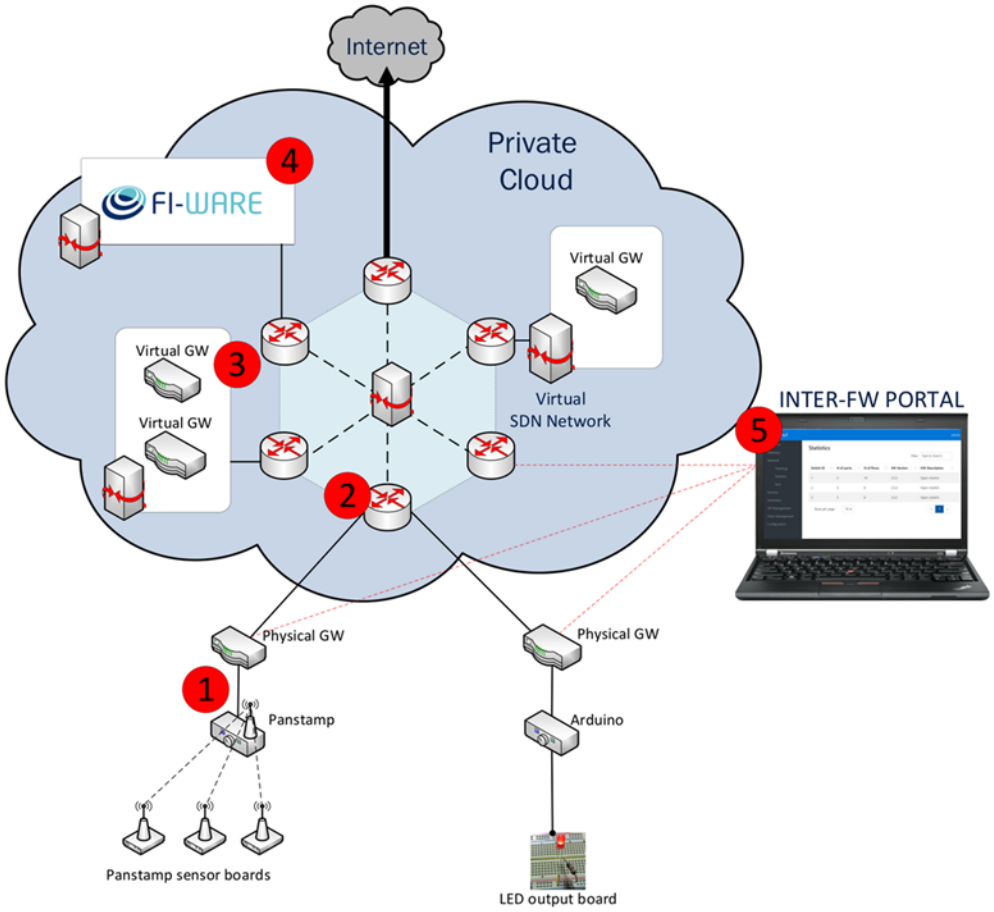

Figura 4.2: Despliegue del caso de uso de priorización de tráfico IoT sobre tráfico de video.

Este despliegue mostró la versatilidad de la pasarela y de los diferentes módulos de controladores de dispositivos que esta contenía para lograr la interoperabilidad a nivel de dispositivo y de red. También, tanto las pasarelas como la red a través de la API expuesta fueron monitorizados y configurados por la herramienta de INTER-FW.

En una segunda parte de este despliegue se ejecutó una tercera pasarela

virtual, pero esta vez conectada a la pasarela *RINICOM Prime-IoT*[3] (la primera implementación comercial de una pasarela física capaz de conectarse a la pasarela virtual de INTER-IoT) y los actuadores fueron controlados esta vez en base a las lecturas de la nueva pasarela y sus sensores, mostrando la posibilidad de intercambiar la pasarela física si se implementa el protocolo de comunicación de referencia correcto sin que afecte al despliegue en sí. Más información acerca de la creación de este despliegue puede encontrarse en [99, 100]

4.1.5. Integración de VNFs para la traducción de protocolos de Red

Como ya se ha expuesto, el principal objetivo del proyecto INTER-IoT, y de la capa de red en concreto, es conseguir la interoperabilidad entre tecnologías usadas sobre esta capa que tradicionalmente no pueden comunicarse entre sí. Uno de los grandes escoyos que encontramos a la hora de comunicar dos elementos pertenecientes a una misma red, es el uso de diferentes protocolos en algunos casos incompatibles entre ellos. En particular, en despliegues de dispositivos inteligentes, cada uno de ellos puede implementar un protocolo de alto nivel diferente, aunque todos ellos se basen en IP, tanto IPv4 como IPv6, al mirar en las capas superiores encontramos que el tráfico que se envía a través de la red es muy heterogéneo.

En numerosos casos las pasarelas o los mismos dispositivos utilizan protocolos como COAP, MQTT o incluso HTTP para enviar los datos. En numerosos proyectos, esta conversión de protocolo u homogeneización de estos se lleva a cabo a nivel de pasarela, es decir, la propia pasarela es la que decapsula el contenido de los paquetes, los reencapsula y los envía utilizando un protocolo común.

[3] https://rinicom.com/patient-telemetry/prime-iot/

Sin embargo, esta tarea requiere de bastantes recursos, ralentizando el funcionamiento de la pasarela, por lo que en esta aproximación se ha hecho uso de la infraestructura de red virtualizada y se han creado funciones de red virtualizadas para la traducción y conversión de protocolos en tiempo real, de los paquetes que viajan a través de la red. Para llevar a cabo este caso de uso particular sobre la red SD-IoT, se ha contado con la colaboración de empresas externas, como en este caso INFOLYSIS[4], con su proyecto *SOFOS: A software-defined end-to-end IoT gateway with virtualization.*

Esta colaboración parte de las mismas necesidades que el presente trabajo de investigación, la de reacondicionar la arquitectura de las redes futuras para adaptarse a los despliegues de IoT de extremo a extremo, ocupándose de: i) el aumento esperado en la generación de datos, ii) los problemas relacionados con la red IP de extremo a extremo de los dispositivos IoT con recursos limitados, iii) el desajuste de capacidad de recursos entre los dispositivos, y iv) la rápida interacción entre los servicios y la infraestructura.

Para este caso de uso se llevó a cabo el desarrollo de diferentes funciones de red cuyo objetivo era la traducción en tiempo real de diferentes protocolos. Se utilizó la infraestructura de red ya desplegada en el anterior caso de uso, haciendo uso de la solución propuesta por INTER-IoT. De esta forma se buscaba mejorar la interoperabilidad de la capa de red y ser capaz de gestionar desde una misma aplicación un diverso número de dispositivos inteligentes, cuyos protocolos eran incompatibles. Específicamente, se establecieron las siguientes etapas:

- Añadir automatización y verificación de componentes en la infraestructura IoT.
- Reacondicionar varias funciones IoT proporcionadas por HW específico, y trasladarlas a máquinas virtuales o contenedores (VNF)

[4] https://company.infolysis.gr/

- Mapear protocolos utilizados sobre IP para mejorar la interoperabilidad del Sistema, en concreto, COAP, MQTT y HTTP.
- Crear una cadena de funciones IoT conectando las diferentes VNFs.

En la Figura 4.3 se puede ver en detalle el fragmento de la arquitectura SD-IoT que integra las diferentes funciones virtuales de traducción de protocolo. A la izquierda encontramos los nodos IoT, que son aquellos sensores que recogen los datos de campo para enviarlos a través de la red, como ya hemos visto en el caso anterior estos datos se envían a través de las diferentes interfaces de comunicación de IoT hacia la parte física de la *gateway*, y es aquí, en la parte física donde la información en un protocolo específico se entrega a la función específica de traducción de protocolos, para llevar a cabo está conversión y enviar los datos a la red en un protocolo de red común y básico (UDP).

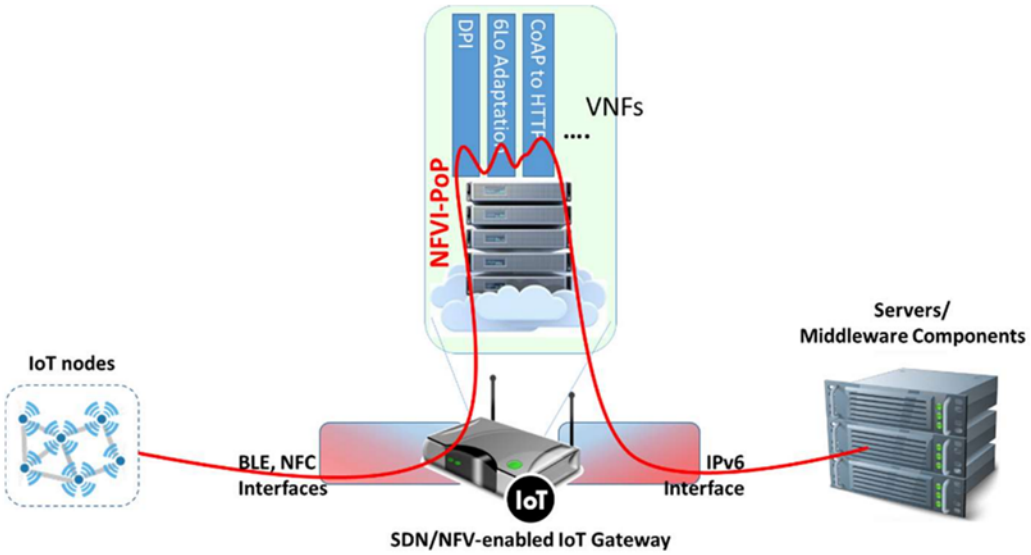

Figura 4.3: Pasarela IoT habilitada con capacidades SDN/NFV.

Estas funciones no se encuentran constantemente ejecutándose, sino que cuando la pasarela, que posee un controlador por cada protocolo, detecta la

llegada de paquetes de ese protocolo concreto, envía una señal al controlador SDN, y este junto con los componentes de gestión de red y funciones virtuales, son los encargados de levantar sobre la infraestructura virtual, que puede estar desplegada en el borde o en el núcleo de la red, un contenedor con la función indicada.

Esta función ejecutará la traducción, reenviando los datos hacia la red y dejará de ejecutarse, parando y eliminando el contenedor, hasta que vuelva a necesitarse de nuevo. EL código y documentación de este caso de uso se encuentra públicamente disponible a través de:

- **Código**: https://github.com/Nomorekek/interiot-sofos
- **Documentación**: https://files.INTER-IoT.eu/deliverables/accepted/D6.3%20-%20Use%20case%20oriented%20pilots%20final%20version.zip

Las funciones de traducción fueron desarrolladas en Python v3 y pueden encontrarse en su respectivo repositorio de Github. Se utilizó la misma arquitectura desplegada para el anterior caso de uso, sin embargo, para llevar a cabo este despliegue se tuvo que hacer una modificación en las tecnologías utilizadas en la arquitectura. En este caso el controlador SDN previamente utilizado (RYU), se intercambió por un controlador diferente: OpenDayLight (ODL), debido a la mayor compatibilidad con las funciones desarrolladas y su fácil integración.

Este controlador fue utilizado como Gestor de Red, indicado en la Figura 4.4 como Network Manager, de igual forma que el anterior, este es un cambio mínimo, puesto que el elemento controlador de red sigue estando presente en la arquitectura y ambos hacer uso del protocolo de red OpenFlow (versión 1.3) para comunicarse con los elementos de la infraestructura. Así, el controlador, instala las entradas apropiadas de OF en los switches de la red para que su tráfico se reenvíe al mismo, y este pase a ser tratado por las funciones de red de traducción.

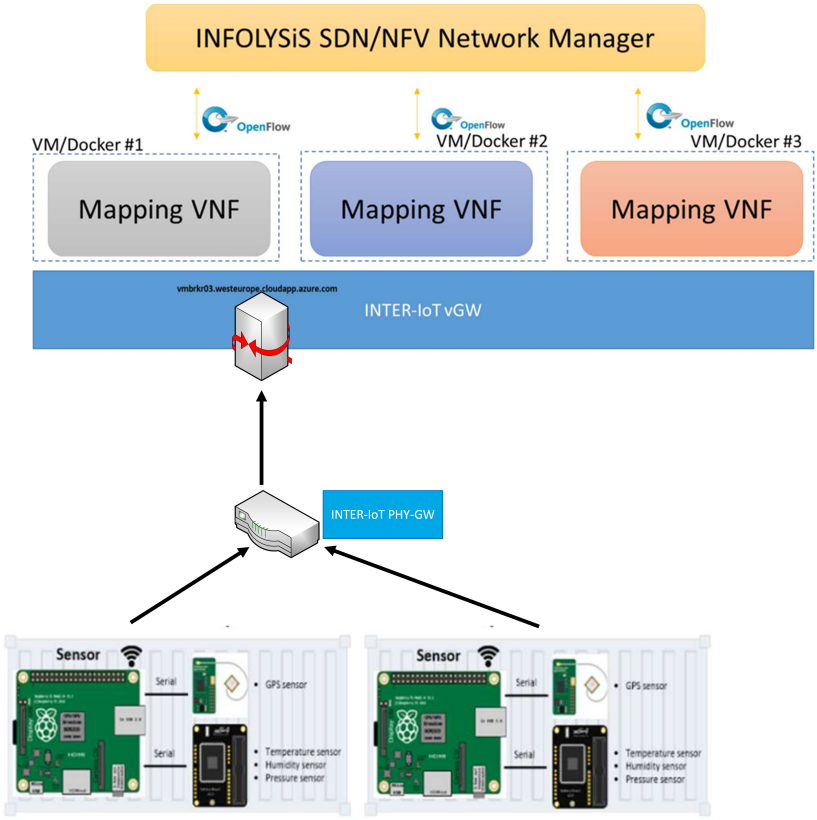

Figura 4.4: Visión general de la integración de funciones de SOFOS en la arquitectura de INTER-IoT.

En concreto, para la ejecución de pruebas en este escenario, los datos recogidos por la pasarela física utilizando la red de acceso de PanStamp, se mapearon a un protocolo de uso común, HTTP, para poder ser utilizado por diferentes aplicaciones. De nuevo, cada una de estas funciones virtuales de mapeo fue encapsulada dentro de un contenedor Docker, para poder desplegarlas fácilmente sobre la infraestructura.

Capítulo 5

Implementación del Escenario 2

En este capítulo se describe la implementación de la arquitectura diseñada y previamente descrita en un segundo escenario real. Este escenario se ha llevado a cabo bajo el marco del proyecto (5GENESIS) y extiende el uso de la tecnología IoT a entornos de despliegues 5G, donde ya se hace uso del paradigma SDN/NFV. En este segundo escenario, el foco se centra en la integración de estos componentes IoT en la red 5G y además en un caso de uso concreto en la plataforma de Limasol, dónde se busca incrementar la cobertura de red, además de la velocidad de creación de servicios, llevados más cerca del usuario a través de elementos como la computación al borde de la red.

5.1. Escenario 2: Redes virtuales definidas por software para despliegues de Internet de las cosas en 5G (5GENESIS)

5.1.1. Introducción

En este segundo escenario, nuestra arquitectura combinada de red programable con elementos de despliegue IoT da un paso más allá. Se presenta un entorno realista, alejado de las emulaciones para dar paso a un despliegue en el mundo físico. Además, se integra una nueva tecnología, que hace uso de las redes programables como principal facilitador, esta es el 5G.

Si bien es cierto que, este escenario se centrará en la parte de infraestructura 5G y no tanto en la parte física (no se hace uso de la tecnología MIMO, ni de nuevas técnicas de modulación de señal, entre otras). Por lo tanto, no estamos ante un despliegue totalmente autónomo (*Standalone*), sino un despliegue particular del caso no-autónomo (*Non-Standalone*). Las diferencias entre estos dos despliegues se pueden apreciar en la Figura 5.1. Mientras que el despliegue autónomo posee una infraestructura LTE totalmente independiente a la de 5G, incluidas estaciones base y núcleo de la red. La no autónoma posee componentes de ambos despliegues que se complementan y se unen en un solo núcleo de red que maneja los datos que provienen de ambas estaciones base de acceso a la red [101].

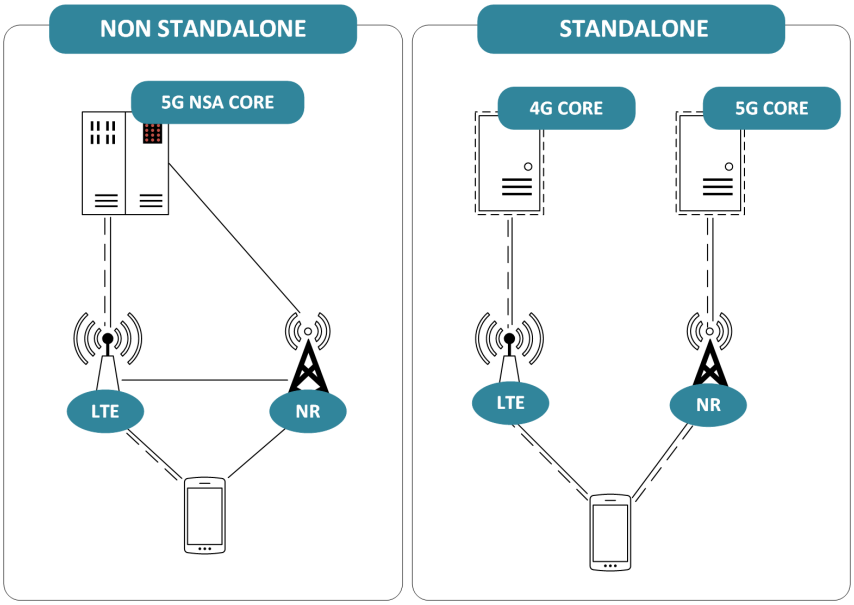

Figura 5.1: Despliegue autónomo y no autónomo de la infraestructura 5G.

En este capítulo, describiremos en un inicio brevemente las particularidades del 5G y, en concreto, como la virtualización de redes, y las funciones virtuales (SDN/NFV) han transformado en núcleo o *core* estos despliegues. De entre los tres principales casos de uso que se presentan en 5G, nos centraremos en definir el que tiene más relevancia para nosotros, y es aquel que está relacionado con el IoT, denominado en este ámbito: comunicación masiva de tipo máquina Massive Machine Type Communications (mMTC). Más adelante, se presentará con más detalle en que consiste el proyecto 5GENESIS, y dentro de éste, la plataforma de Limasol. En esta plataforma es donde se integran elementos de IoT (como es el caso de dispositivos inteligentes, nuestra pasarela físico-virtual, etc.) con la infraestructura 5G que está modelada sobre una red programable, con elementos de gestión y control como el Controlador SDN. Además, varios de los componentes de nuestro despliegue IoT, serán encapsulados en contenedores para la creación de funciones virtuales específicas dentro de este entorno.

Para ilustrar en más detalle los elementos innovadores de este escenario, y sus particularidades respecto al diseño de arquitectura genérica previo, los

cambios introducidos son los siguientes:

1. Integración de protocolos IoT en la parte física de 5G, la parte física de la pasarela ejecutará sus funciones de traducción de protocolos para recoger y enviar datos desde una red Lora, hacia la red 5G.

2. Se crean las funciones virtuales (VNFs) que incluyen la parte virtual de la pasarela y la aplicación final de recogida de datos IoT (*Broker* de *FIWARE Orion*) así como un nuevo tablero de visualización y gestión de los datos basado en Node-RED.

3. Se incluyen los orquestadores para gestionar los diferentes componentes del despliegue, en nuestro caso OSM, además de una infraestructura virtual, implementada con OpenStack.

4. Los indicadores de rendimiento medirán no slo el rendimiento del despliegue en general, sino la diferencia entre el despliegue de las funciones virtuales en el borde de la infraestructura y en su núcleo.

5.1.2. Principales servicios del 5G y el IoT

En un futuro inmediato se espera que la tecnología 5G proporcione a los usuarios una experiencia en la navegación que cumpla unos detallados requisitos de rendimiento, incluyendo un conjunto muy diverso de categorías de servicios que ofrecerán una mayor agregación, control de picos de tráfico en la red y en la tasa de datos, mejora de la eficiencia del espectro radio, una latencia muy reducida y una gran mejora en el soporte a la movilidad (*roaming*). Así mismo, entre las cualidades que se demandan en esta nueva generación móvil, se deben incluir el establecimiento de conectividad automática para un vasto rango de accesorios, máquinas y otros objetos inteligentes, sin que sea necesaria la intervención humana. Además de todo ello, considerando el mercado de las telecomunicaciones y las demandas de sus principales actores, la tecnología 5G deberá ser capaz de cumplimentar todos estos requisitos

haciendo un uso limitado del consumo de energía, reduciendo los costes en el material y equipos, así como los costes en el despliegue. Para alcanzar tales expectativas, definir claramente estos requisitos y dar una visión más homogénea de las especificaciones técnicas que esta tecnología necesita, las principales Organizaciones de Desarrollo de Estándares SDO, la ITU y el 3GPP, han categorizado los servicios proporcionados por el 5G en tres grandes grupos [102]:

- EMBB: del inglés *enhanced Mobile BroadBand*, mejora del ancho de banda móvil, se refiere a aquellos servicios que hace un uso intensivo del ancho de banda, en este grupo se incluyen la realidad virtual y aumentada (AR/VR), la reproducción o *streaming* de video en ultra definición (UHD), la difusión (*broadcasting*) de programas de TV, entre otros. Los requisitos en el ancho de banda para este tipo de servicios se encuentran entre los 100 Mbps/usuario, pudiendo alcanzar incluso el orden de Gbps, como es el caso del *broadcasting* que puede alcanzar los 10 Gpbs.

- URLLC: del inglés *Ultra-Reliable Low-Latency Communications*, comunicaciones de baja latencia ultra confiables, también llamadas comunicaciones críticas, hacen referencia a aquellos servicios que son extremadamente sensibles a la latencia, es decir, los requisitos de latencia son muy bajos y limitados. Estos servicios están enfocados a misiones críticas que incluyen seguridad pública, automatización industrial, comunicación en caso de desastre, control de drones, aplicaciones médicas, vehículos autónomos, etc. dónde la baja latencia es un factor clave para el correcto funcionamiento del servicio, ya que si falla puede poner en peligro a los usuarios. Los requisitos de latencia que se espera en este tipo de servicios se mueven en un rango de 1 ms a 2 ms de latencia máxima, incluyendo la interfaz de radio, y menos de 10 ms en el recorrido total, incluyendo todo el plano de datos [103].

- mMTC: del inglés *massive Machine-Type Communications*, comunicación

masiva de máquinas, también denominado Internet de las Cosas masivo, estos servicios extienden las funcionalidades ya presentadas en las especificaciones del LTE-IoT (NB-IoT) y hace referencia a aquellos servicios donde se comunican un gran número de dispositivos de bajo coste y consumo, donde hay una extensión de la cobertura y se prioriza la extensión de la vida de las baterías. En esta categoría se incluyen todos los servicios relacionados con el IoT como los sensores de clima, *eHealth*, *wereables*, etc. [104]

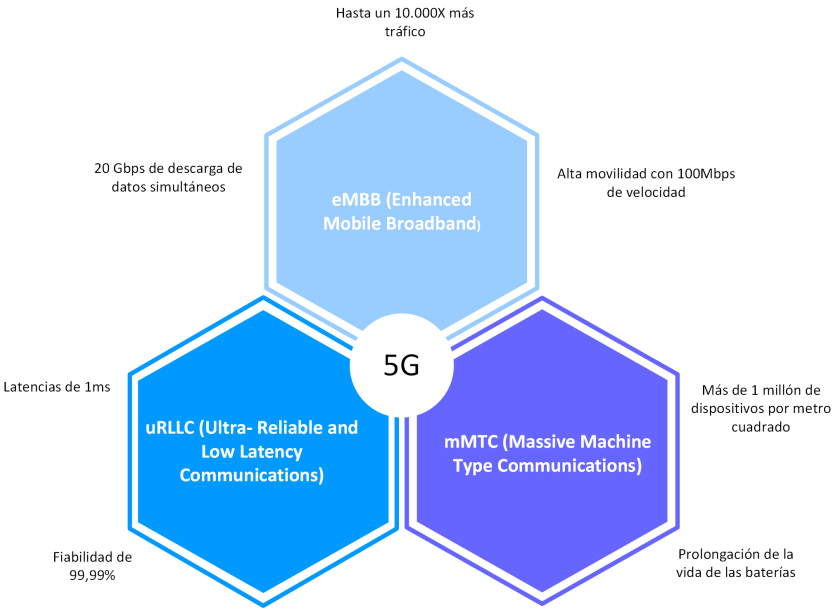

Figura 5.2: Principales Casos de Usos para 5G.

Aunque estas tres sean las principales categorías (Figura 5.2), cada servicio posee requisitos que pueden pertenecer a varias de esas categorías, o incluso a las tres. Por ejemplo, en el caso de un medidor de parámetros de salud que ha de enviar una alarma si alguno de los parámetros tiene un valor anormal, se necesitarán requisitos de baja latencia a la vez que una amplia cobertura y disminución del uso de baterías.

La tecnología 5G es un facilitador en el desarrollo y despliegue de sistemas de Internet de las Cosas, puesto que presenta una serie de requisitos útiles e incluso necesarios para la buena ejecución de los servicios IoT.

5.1.3. Proyecto 5GENESIS

Nuestro segundo Escenario se encuentra enmarcado dentro de las actividades llevadas a cabo en el proyecto 5GENESIS[1], proyecto de investigación financiado por la Unión Europea dentro del programa H2020. Este proyecto, denominado por su nombre completo como: 5ta generación de redes, experimentación, integración de sistemas y exhibiciones integrales, es un proyecto cuyo objetivo principal es la creación de una plataforma distribuida que implemente la tecnología 5G en diferentes escenarios para ser usados como lugar de implementación y ejecución de pruebas, y así validar los principales indicadores de rendimiento (KPIs) que han sido previamente definidos para las redes 5G por organismos oficiales de estandarización. En concreto, el proyecto está compuesto de cinco plataformas, distribuidas alrededor de Europa en las ciudades de: Málaga, Berlín, Atenas, Surrey y Limasol. Cada una de estas plataformas implementa las especificaciones descritas para despliegues de tecnología móvil 5G, sin embargo, cada una de ellas está focalizada en uno o varios de los tres servicios descritos previamente. Así pues, junto con los servicios asociados a cada plataforma, serán los indicadores de rendimiento especificados para esos servicios los que serán probados y demostrados que respectivas plataformas. Haciendo un resumen de las características y funciones principales de la infraestructura de 5GENESIS, encontramos:

- La implementación y verificación de la evolución de los estándares 5G

[1] https://5genesis.eu

mediante una integración iterativa y un proceso de pruebas.

- La utilización de una amplia diversidad de tecnologías e innovaciones que abarcan todos los dominios, logrando una cobertura completa del panorama 5G.
- La unificación de elementos heterogéneos tanto físicos como virtuales bajo un marco de referencia común de coordinación, abierto a los experimentadores de diferentes industrias que permita la experimentación automática y la separación o rebanado integral (*end-to-end slicing*) [105].
- Dar soporte a futuros proyectos de experimentación, en particular aquellos centrados en los mercados verticales.

Con estas como funciones principales, se llevó a cabo la creación de las cinco plataformas, con sus respectivos servicios y casos de uso, que son:

- *Plataforma de Atenas*: esta plataforma se centra en permitir la computación en el borde con un sistema de radio compartida, esto es, utilizando diferentes rangos de radiofrecuencia y superponiendo las áreas de cobertura, todo ello manejado desde un núcleo (*core*) habilitado por SDN/NFV. Su principal caso de uso se basa en aplicaciones de baja latencia y entrega segura de contenido en eventos multitudinarios.
- *Plataforma de Málaga*: basada en la orquestación automática y la gestión de diferentes secciones de red virtual sobre múltiples dominios, todo ello sobre una parte radio 5GNR (*new radio*) en la parte física, y un núcleo de red totalmente virtualizado. Sus principales casos de uso se centran en mostrar servicios de misión crítica en el laboratorio y en implementaciones al aire libre.
- *Plataforma de Surrey*: centrada en la implementación de numerosas interfaces radio para soportar la comunicación de máquinas de forma

masiva (mMTC), incluye 5GNR y NB-IoT combinados bajo una gestión de recursos radio (RRM) y una plataforma de compartición de espectro para mostrar servicios de IoT masivo.

- *Plataforma de Berlín*: se basa en el desplegado de varias redes para la cobertura de áreas ultradensas, incluyendo nodos interiores y exteriores que sean movibles o nómadas, coordinado a través de una tecnología avanzada de red de retorno (*backhauling*[2]). Su principal caso de uso se basa en aprovisionamiento de servicios inmersivos (AR/VR).

- *Plataforma de Limasol*: basada en múltiples interfaces de acceso radio con diferentes características, combinando comunicación satélite y terrestre. Su principal caso de uso está basado en la continuación de servicio, la cobertura y en el acceso ubicuo en ares remotas, como por ejemplo áreas rurales o marítimas.

En esta última plataforma es dónde se llevó a cabo el despliegue de la arquitectura diseñada y las pruebas pertinentes. Aunque durante la duración del trabajo de doctorado se participó en variar partes transversales del proyecto que incluyen tareas relativas a todas las plataformas, la plataforma de Limasol es la principal plataforma donde se ha contribuido, pues es en ella donde se llevaron a cabo los despliegues y pruebas de campo. A continuación, describiremos en detalle las características y componentes de la plataforma de Limasol y como se llevó a cabo la integración de los componentes diseñados durante este trabajo en la misma. Además, describiremos en detalle el caso de uso real que se diseñó y se llevó a cabo, así como sus pruebas y los indicadores de rendimiento que se midieron durante el proceso.

[2]La red de retorno es aquella porción de red que incluye los enlaces intermedios entre el núcleo de la red, y la red troncal, así como las pequeñas subredes que se encuentran en el borde (*Edge*) de la red.

La plataforma de Limasol

La Plataforma de Limasol es una de las cinco instalaciones que conforman el proyecto 5GENESIS. Esta plataforma integra diferentes infraestructuras en la ciudad de Limasol, en Chipre, para crear una instalación multi-radio, esto es, mediante diferentes interfaces de radio se podrán conectar los distintos dispositivos a esta plataforma, combinando comunicación satelital y terrestre con el objetivo final de extender la cobertura 5G a áreas remotas o de difícil acceso, como son las áreas rurales o mar adentro. Para conseguir este objetivo, la plataforma emplea tecnologías NFV y SDN, con el fin de manejar la comunicación satelital, así como la integración de los diferentes modos de acceso y manejar en la red troncal [5]. En concreto, partiendo de una arquitectura base 5G, las innovaciones que se presentan en esta plataforma son las siguientes:

i) Cobertura ubicua con reducción de latencia.

ii) Soporte para la división o rebanado (*slicing*) de múltiples accesos radio.

iii) Mejora del rendimiento (*throughput*) 5G mediante agregación de interfaces aéreas o modos de acceso, cuando sea necesario.

iv) Asignación dinámica de espectro entre las comunicaciones terrestre y satelital.

En la Figura 5.3, podemos observar la topología a muy alto nivel de la plataforma de Limasol con sus diferentes nodos principales geográficamente distribuidos en las localizaciones pertinentes.

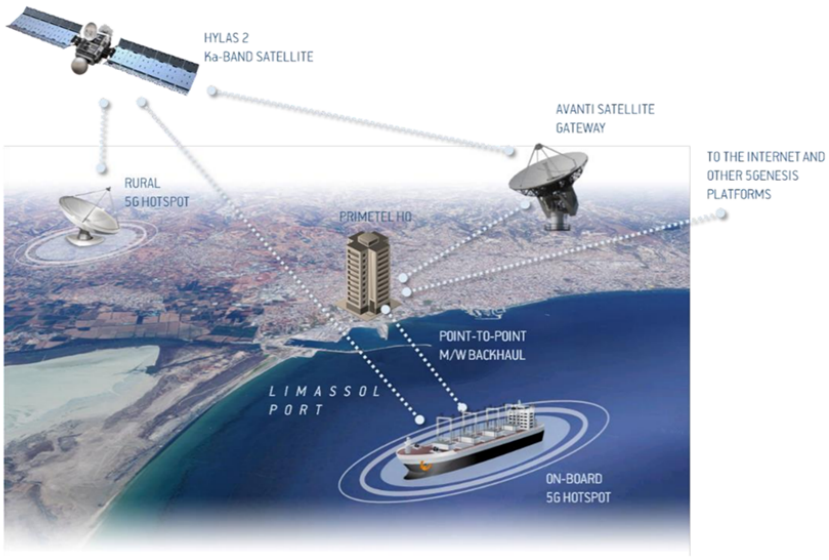

Figura 5.3: Ilustración a alto nivel de los diferentes componentes del despliegue completo de la plataforma 5G de Limasol.(Fuente: [5])

Entre los principales componentes encontramos, la pasarela satelital propiedad de Avanti, que será la conecta con el satélite (HYLAS-2) y ofrece servicios de SATCOM, también encontramos el principal punto o núcleo del despliegue que será el centro de datos localizado en las oficinas de Primetel. Es ahí donde se despliegan los servicios NFV de la plataforma alojados en el núcleo y desde donde se monitoriza y gestiona toda la red SDN.

También, desde este *testbed* se ofrece la interconexión entre la Pasarela Satelital y la red pública (Internet), así como la conexión con las demás plataformas dentro del proyecto 5GENESIS. Para el banco de pruebas realizadas en Limasol, el plan de trabajo de 5GENESIS incluyó todas las actividades necesarias para mejorar e integrar la infraestructura y los emplazamientos ya existentes con el fin de integrarlos entre sí y proveer las características 5G. En particular, se llevaron a cabo las siguientes actividades o pasos para el despliegue del banco de pruebas:

Interconexión de los sitios (*sites*)

Se llevó a cabo principalmente de interconectar el banco de pruebas de PrimeTel con: i) la pasarela Avanti y ii) con otros bancos de pruebas de 5GENESIS para realizar experimentos entre varios *sites* (Actividad del plano de la Infraestructura).

Despliegue de las funciones centrales de EPC y 5G NG

(PrimeTel site). Estas están centradas principalmente en el establecimiento de sesiones y la gestión de la movilidad. Este desarrollo se desarrolló totalmente alineado con las especificaciones del 3GPP (Actividad del plano de la Infraestructura).

Establecimiento del enlace terrestre de retorno (*backhaul*)

El enlace *backhaul* se estableció entre el edificio PrimeTel y la red remota (a bordo), basándose en tecnología de conectividad WiFi punto a punto y en la conectividad LTE tradicional (Actividad del plano de la Infraestructura).

Instalación e integración de los componentes 5G NR

Se implementaron en plataformas SDR tanto para el gNB como para el UE. Este desarrollo también está totalmente alineado con las últimas especificaciones posibles del 3GPP (Actividad del plano de la Infraestructura).

Habilitación de comunicaciones por satélite con capacidad NFV/SDN

Para integrar los componentes SATCOM (estación terrestre y terminales) sin problemas en el contexto 5G, fue necesario actualizarlos con capacidades SDN y NFV. Es decir: i) algunos elementos de la red en la cadena de suministro de SATCOM deben estar habilitados para SDN, ii) la pasarela de satélite debe mejorarse con una NFVI-PoP y iii) el terminal de satélite debe tener capacidades de SDN y NFV (Actividad del plano de la Infraestructura).

Adaptación e integración de la pila MANO (NFV) y MEC para alojar y gestionar VNFs en el núcleo y en el borde

Esto se llevó a cabo con la herramienta MANO de código abierto (OSM), integrada y adaptada para gestionar los VNFs específicos del satélite y la *gateway* IoT. La implementación siguió la familia de estándares ETSI ISG NFV y ETSI ISG MEC (Actividad del plano de Gestión y Orquestación).

Adaptación e integración del Marco de Seguridad

Aparte de la capa de gestión, se incluyó una mezcla de funciones de gestión de recursos centralizados y distribuidos y aprovechando las tecnologías y análisis de *Big Data* (Apache Spot, Apache Spark y Hadoop) para mejorar la seguridad de la red. (Actividad del plano de Aplicaciones y Servicios).

Adaptación e integración de los componentes de gestión del espectro

Basados en una mezcla de funciones de gestión de recursos centralizados y distribuidos y focalizados en el reparto dinámico del espectro entre la infraestructura terrestre y la satelital (Actividad del plano de la Infraestructura).

Adaptación e integración de las funciones virtuales de optimización de la WAN

Integración de funciones para la aceleración del rendimiento y procesado de paquetes en la red de área amplia (WAN) mediante la aceleración de TCP y la eliminación de la redundancia de datos, más la unión de enlaces entre la red de transporte terrestre y satelital. (Actividad del plano de la Infraestructura).

Establecimiento de la API norte para los experimentadores

Diseño de una API para futuras integraciones exponiendo los métodos y los parámetros relevantes de las características de la plataforma. Esta actividad se benefició de los componentes comunes desarrollados como parte de la actividad del plano de coordinación. (Actividad del plano de Aplicaciones y Servicios). Por último;

Adaptación e integración del servicio de interoperabilidad de IoT

En esta actividad es donde entra la integración de los dispositivos, nodos y servicios IoT. Como despliegue E2E que proporciona interoperabilidad de plataformas y sistemas de IoT cuyos componentes están distribuidos en las diferentes localizaciones de la plataforma de Limasol, incluyendo ubicaciones al borde y el núcleo de la red, facilitando la comunicación, el procesamiento y la homogeneización de los datos de los sensores (Actividad en el plano de la Infraestructura y también en el de Aplicaciones y Servicios).

En la Tabla 5.1 se puede observar de forma más estructurada, las diferentes mejoras respecto a la arquitectura base 5G realizadas en la plataforma, divididas por capas funcionales; la capa de infraestructura o capa más física, la capa de gestión y orquestación y la capa de coordinación.

Estas tres capas se corresponden con las capas definidas en las arquitecturas de redes definidas por software, donde la capa de infraestructura sería la capa de datos, por donde pasa el tráfico y los datos de las aplicaciones, y los elementos que la componen son elementos físicos o virtuales que definen la topología del despliegue, la capa de gestión y orquestación se corresponde a la capa de control, en esta encontramos el tráfico de gestión propio de la red, con meta-información o información de control referente al estado del despliegue, y por último la Capa de coordinación corresponde a la capa de Aplicación, que utilizando las APIs expuestas por los componentes de las otras dos capas hace de interfaz con el usuario para mostrar los datos de forma sencilla, y levar a cabo la interacción con el usuario de la plataforma.

Mejoras en la plataforma	**Plano de Gestión y Orquestación**	**Plano de Infraestructura**	**Plano de Aplicaciones y Servicios**
Actividad en el plano de Coordinación			X
Marco de Seguridad			X
Optimización de la WAN y agregación a nivel de enlace	X		
NFV y MEC MANO	X		
Interconexión de los sitios		X	
Gestión del espectro		X	
Despliegue del EPC y del núcleo 5GNR		X	
Instalación de 5G NR		X	
Establecimiento de la red de transmisión terrestre (*terrestrial backhaul*)		X	
Comunicación con soporte NFV y SDN		X	
Servicio de Interoperabilidad para IoT		X	

Tabla 5.1: Resumen de las mejoras respecto a la arquitectura base 5G realizadas en la plataforma de Limasol.

5.2. Caso de Uso: Integración de IoT con 5G

Una vez la plataforma de Limasol fue desplegada, se pasaron a realizar diferentes pruebas sobre ella. En concreto, se definió un caso de uso realista para la integración de componentes IoT en el despliegue 5G cuyo objetivo era el de proporcionar cobertura en áreas remotas o rurales donde las redes móviles tradicionales tienen un alcance limitado, por lo que tecnologías IoT de largo alcance como LoRa pueden integrarse a este despliegue para aumentar dicho alcance. La selección de LoRa como principal protocolo IoT a integrar viene dado que esta es una de las tecnologías más populares de largo alcance que podemos encontrar en los despliegues IoT, además de que podemos basarnos en previos estudios donde se ha llevado a cabo la integración de LoRa con redes virtuales como en [106]

Además, otro de los indicadores de rendimiento a medir en este caso de uso, es el de la creación de servicios dinámica, de forma rápida. Eso implica que, si nuevos sensores son añadidos al despliegue, la parte virtuales de las pasarelas y demás funciones virtuales se desplieguen o puedan ser escaladas de forma automática y veloz, para proveer del servicio lo más rápido posible. En la Figura 5.4 se describen cada uno de los componentes físicos que forman el caso de uso y las conexiones entre ellos.

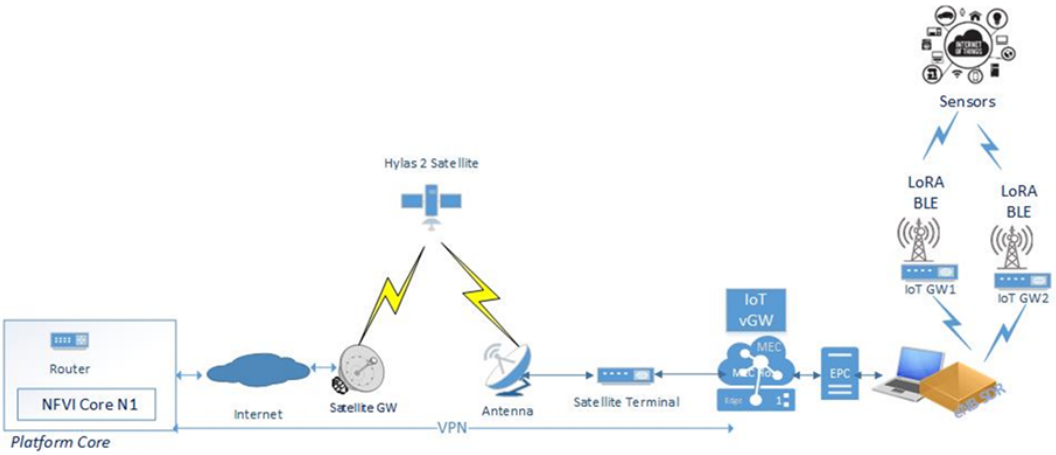

Figura 5.4: Despliegue de componentes en el caso de uso de la plataforma 5G de Limasol.

Si miramos el gráfico de despliegue podemos observar de derecha a izquierda la parte de dispositivos inteligentes, conectados a través de la red de LoRa a la parte física de nuestra pasarela. Una muestra de como se realizó este despliegue en el escenario físico puede observarse en la Figura 5.5, donde se aprecian los nodos o dispositivos, así como la parte física de la pasarela con el módulo de LoRa en ella. En esta parte física se lleva a cabo la traducción de protocolos, desencapsulando la información recibida por la red LoRa y transformándola un protocolo de red común para ser enviada a través de la red 5G.

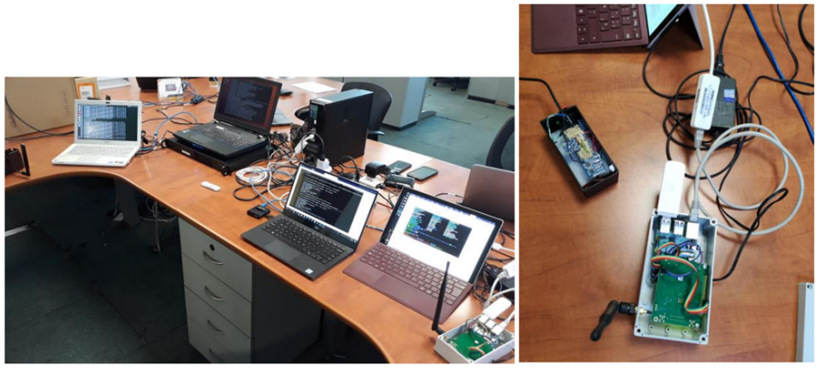

Figura 5.5: Dispositivos IoT y configuración para los experimentos en la plataforma de Limasol.

La parte física de la pasarela se conecta pues con el gNB y de ahí pasa al EPC que se encuentra conectado al nodo de procesado de datos de 5G en el borde, también denominado MEC. Este nodo MEC acerca los recursos tecnológicos al usuario final, procesando los datos y almacenando datos y servicios cerca del borde de la red, en contraposición de enviar todos los datos directamente a la nube, lo que, entre otras cosas, reduce significativamente la latencia. En este nodo se pueden implementar directamente alguna de las funciones virtuales ya descritas en el capítulo Sección 2.2 Funciones de Red Virtualizadas (NFV). En concreto, en este caso de uso se ejecutó la parte virtual de la pasarela en este nodo y se midieron los indicadores de rendimiento tanto con la parte virtual de la pasarela ejecutándose en este componente, como ejecutándose directamente en la nube o el núcleo evolucionado EPC.

Más adelante tenemos la sección de conectividad por satélite. El SATCOM también suele utilizarse para extender la conectividad de datos a zonas que no están bien cubiertas por la infraestructura celular terrestre (zonas rurales o subdesarrolladas, etc.). El apoyo a la red troncal móvil y a las redes de sensores remotos son escenarios de aplicación bastante habituales en las conexiones satelitales. Sin embargo, la mayoría de los servicios se limitan actualmente a la simple conectividad a Internet y son independientes de los servicios 3G o 4G.

En este caso, la red troncal móvil se implementa tanto de forma tradicional como a través de conexión satelital, con una estrecha integración en el despliegue 5G que permite la gestión de sesiones sobre esta red, la prestación dinámica de servicios virtualizados (VNFs) tanto en el núcleo y como en el borde, y la creación de redes definidas por software a través de SATCOM. Por último, en la parte final del despliegue encontramos el núcleo de la red, normalmente alojado en la nube o centros de datos centrales del operador, donde se implementa el PC de 5G y las diferentes funciones que este lleva a cabo como: la gestión de usuarios y accesos, la gestión de movilidad y las sesiones, etc. En este caso de uso en concreto, también se desplegará la parte virtual de la pasarela en dicho núcleo, para evaluar los indicadores de rendimiento del despliegue cuando este componente se encuentra ejecutándose allí. En la Figura 5.6 podemos ver como está dicho núcleo implementado en el despliegue real que se realizó en Limasol.

Figura 5.6: Equipos que implementan el núcleo de red de la plataforma de Limasol.

Es claro que la atención de este caso de uso se centra en las comunicaciones mMTC, aunque se emplean todos los componentes tradicionales de un despliegue 5G estándar. Por otro lado, este caso de uso también implica características como:

1. El reparto dinámico del espectro entre la red satelital y la terrestre, que se basa en una mezcla de funciones de gestión de recursos centralizadas

y distribuidas.

2. El despliegue de servicios virtuales en el núcleo y en el borde para la gestión eficiente del tráfico IoT, implementando a través de la pasarela y de sus funciones como la de conversión de protocolos, además de otros componentes como el nodo final de recepción de datos IoT que proporciona un acceso uniforme y centralizado a los datos que provienen de diferentes pasarelas y sensores IoT heterogéneo.

3. El análisis detallado del tráfico para detectar y clasificar los incidentes de seguridad, basados en tecnologías de *Machine Learning* y en la contextualización y traducción semántica de los datos.

Capítulo 6

Validación de los Escenarios y Discusión de Resultados

Tras haber descrito cada uno de los dos Escenarios utilizados para implementar la arquitectura combinada propuesta, este capítulo muestra la validación y las diferentes pruebas que se ejecutaron sobre cada uno de ellos. En primer lugar, se presentan los Indicadores de Rendimiento seleccionados para cada Escenario y, más adelante, los resultados de esos indicadores con una discusión de su implicación para este trabajo.

6.1. Indicadores de Rendimiento (KPIs) Escenario 1

6.1.1. Integración de controladores. Extensibilidad

Este indicador de rendimiento es más un indicador de flexibilidad, dónde la integración con numerosos controladores nos demuestra que, independientemente del controlador utilizado, estos hacen uso de protocolos de gestión de red abiertos (e.g *OpenFlow*), por lo tanto, el funcionamiento de la red y su rendimiento se mantienen constantes. Esta es una característica muy importante para la interoperabilidad, puesto que facilita la portabilidad de elementos entre diferentes sistemas y su comunicación. En concreto, este experimento estaba dirigido a integrar una serie específica de controladores diferentes, pero que pudieran ser intercambiables. Todos ellos debían ser compatibles con OpenFlow y OpenVSwitch (además de implementar, aunque fuera de forma parcial, otros protocolos como OVSDB).

Aunque, en caso específico del proyecto INTER-IoT se personalizó y extendió el controlador RYU para hacer uso de éste en los escenarios de validación, este ejercicio de intercambio se llevó a cabo para evaluar la flexibilidad de la arquitectura diseñada. Así el elemento controlador definido en la arquitectura puede ser sustituido para obtener los mismos objetivos que se pretende conseguir al introducir la capa de red programable en un despliegue IoT.

Test Case ID	Integration
Descripción general	Este experimento mide la capacidad de integración de diferentes tipos de controladores en la arquitectura diseñada. Su único objetivo es funcional para demostrar que independientemente de la herramienta que represente el elemento de control en la arquitectura, el funcionamiento general (*end-to-end*) es igual, manteniendo un rendimiento similar.
Objetivo	El experimento se llevó a cabo en un entorno emulado a través del software de Mininet, creando diferentes nodos virtuales dentro de una misma máquina, instalando el switch virtual de OpenVSwitch y los diferentes controladores y conectándolos entre sí.
Componentes involucrados	▪ Controladores para integrar: RYU, ODL, ONOS and POX/NOX ▪ Switch virtual de OpenVSwitch ▪ Infraestructura virtual proporcionada por Mininet

Tabla 6.1: Descripción de la prueba integración de controladores en la red virtual.

En concreto los controladores probados fueron: RYU[1], en la versión más actualizada en ese momento, OpenDayLight[2] la versión Boron en ese momento,

[1] https://github.com/faucetsdn/ryu
[2] https://github.com/opendaylight

ONOS[3] la versión Falcon, y POX/NOX[4], la versión más actual de ese momento en Github. Todos ellos fueron probados en un entorno virtualizado con Mininet [5], siendo el único criterio de éxito su correcta ejecución y conexión con el switch virtual de OpenVSwitch.

6.1.2. Latencia entre nodos (RTT)

La latencia o retardo se define como el tiempo que transcurre entre que un mensaje es enviado desde un punto concreto origen, hasta que es recibido en el punto de destino, por lo que es una consecuencia de la velocidad limitada con la que cualquier paquete de datos puede propagarse. La latencia es un indicador básico para determinar el funcionamiento correcto de una red, en trabajos como [107] podemos observar como la latencia se ha tratado de mejorar en este tipo de despliegues SDN-5G. Para cuantificar esta latencia, se suele registrar una marca de tiempo en el momento del envío del mensaje y otro en el momento de la entrega. En específico, en este experimento se midió el tiempo transcurrido entre el envío de un paquete de datos IP desde un nodo aleatorio de la red hasta su llegada a otro de los nodos de la red, así como el tiempo de retorno del mensaje de confirmación (ACK).

Para los despliegues de SDN en general podemos diferenciar dos tipos de latencia: aquella que se da en los paquetes entre los nodos del plano de datos y el controlador (plano de control) que los configura y aquella entre dos nodos que existen en el plano de datos. Este experimento se llevó a cabo utilizando diferentes herramientas de medición como *tcpdum, iperf, bmon y netstat* analizamos ambos tipos de latencia.

Plano de Control

[3] https://github.com/opennetworkinglab/onos
[4] https://github.com/noxrepo/pox
[5] https://github.com/mininet/mininet

Los resultados de este experimento sobre la comunicación entre el controlador y los nodos pueden ser observados en la Tabla 6.2

Test Case ID	Latencia_Plano_Control
Descripción general	Este experimento mide la velocidad de los datos desde un nodo cualquiera de la infraestructura hasta el controlador.
Objetivo	El experimento se llevó a cabo en un entorno emulado a través del software de Mininet, creando diferentes nodos virtuales dentro de una misma máquina, instalando el switch virtual de OpenVSwitch y los diferentes controladores y conectándolos entre sí. Estos nodos enviaron numerosos paquetes contra el controlador y se generó una media de la latencia medida. El canal creado entre los nodos y el controlador utiliza el protocolo OpenFlow sobre TCP.
Componentes involucrados	▪ Herramientas de medición de latencia: *tcpdum, iperf, bmon y netstat* ▪ Controlador RYU ▪ Switch virtual de OpenVSwitch
Estadísticas del experimento	Latency [resp/s]

Min	Max	Avg	Stdev
2532,0	2752,38	2639,45	53,22

Tabla 6.2: Descripción de la prueba integración de controladores en la red virtual.

Además, una muestra de la ejecución de los experimentos en la consola puede observarse en la Figura 6.1, dónde se utiliza la herramienta *cbench* para la medida de esta latencia entre nodos.

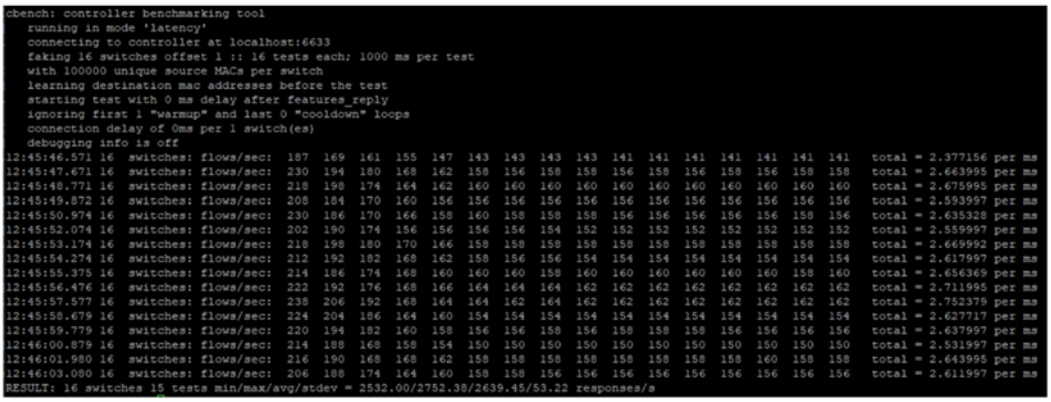

Figura 6.1: Ejemplo de prueba sobre latencia en el plano de control con la herramienta cbench.

Plano de Datos

En este caso realizamos pruebas entre los diferentes nodos del plano de datos. Estos nodos pueden comunicarse siguiendo diferentes protocolos de transporte (TCP, UDP, MPTCP, etc.). Utilizando una herramienta sencilla como el comando ping entre nodos podemos observar el tiempo que tarda un paquete ICMP en llegar de un host a otro y volver (RTT), y así su latencia. En una topología lineal simple de cuatro nodos conectados a través de tres conmutadores virtuales realizamos algunas pruebas con diferentes tráficos, ICMO, UDP y TCP. Los resultados de este experimento sobre la comunicación entre los nodos de nuestra red virtual pueden ser observados en la Tabla 6.3

<table>
<tr><td>Test Case ID</td><td colspan="4">Latencia_Plano_Datos</td></tr>
<tr><td>Descripción general</td><td colspan="4">Este experimento mide la velocidad de los datos desde un nodo cualquiera a otro dentro de la infraestructura de red.</td></tr>
<tr><td>Objetivo</td><td colspan="4">De igual forma, este experimento se llevó a cabo en un entorno emulado a través del software de Mininet, creando diferentes nodos virtuales dentro de la red. Los paquetes fueron enviados desde cada uno de los nodos a los tres restantes, con diferentes perfiles de tráfico y midiendo su latencia con herramientas determinadas, dependiendo del tipo de tráfico. En el caso de tráfico UPD y TCP, uno de los nodos actúa como servidor, mientras que el resto actúan como clientes, y son los que ejecutan la solicitud.</td></tr>
<tr><td>Componentes involucrados</td><td colspan="4">Herramientas de medición de latencia: tcpdum, iperf, PING y bmon</td></tr>
<tr><td rowspan="5">Estadísticas del experimento</td><td colspan="4">Latency [ms]</td></tr>
<tr><td colspan="4">ICMP</td></tr>
<tr><td>Min</td><td>Max</td><td>Avg</td><td>Stdev</td></tr>
<tr><td>0,029</td><td>0,222</td><td>0,95</td><td>0,04</td></tr>
<tr><td colspan="4">UDP</td></tr>
<tr><td></td><td>Min</td><td>Max</td><td>Avg</td><td>Stdev</td></tr>
<tr><td></td><td>0,14</td><td>1,51</td><td>0,73</td><td>0,34</td></tr>
</table>

Tabla 6.3: Descripción de la prueba de latencia entre nodos del plano de datos en la red virtual.

Además, una muestra de la ejecución de los experimentos en la consola puede observarse en la Figura 6.2, dónde se utiliza la herramienta *Ping* para la medida de esta latencia entre nodos.

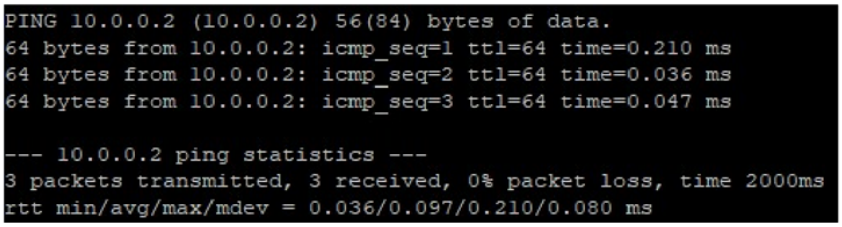

Figura 6.2: Ejemplo de prueba de medida de latencia en el plano de datos para tráfico ICMP con la herramienta ping.

Si comparamos la latencia del tráfico ICMP de la Figura 6.2 con el UDP usando iperf en la Figura 6.3 vemos un claro aumento ya que se involucra el procesado del paquete, al igual que lo veremos más adelante con el tráfico TCP.

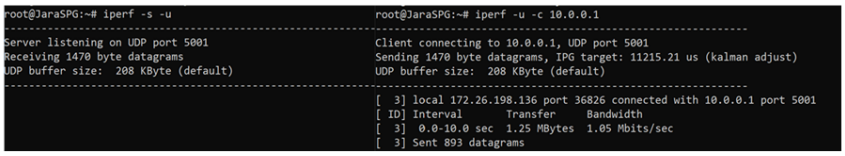

Figura 6.3: Ejemplo de prueba de medida de latencia en el plano de datos para tráfico UDP con la herramienta iperf.

Por último, para evaluar la latencia en el tráfico TCP el proceso fue algo más complicado. En este caso, ejecutamos un servidor HTTP en un nodo de la red (Nodo1) y realizamos una petición desde otros nodos (Nodo2 y 3) de la red. En la Figura 6.4 se puede ver un ejemplo de los datos intercambiados por red.

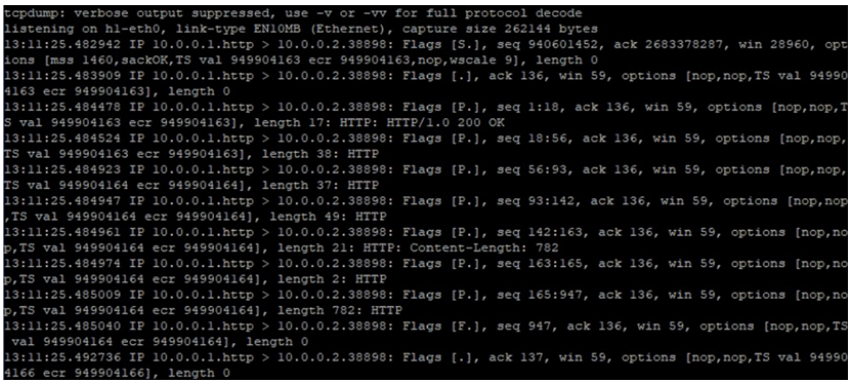

Figura 6.4: Ejemplo de prueba de medida de latencia en el plano de datos para tráfico TCP con la herramienta tcpdump.

Sin embargo, para medir la latencia completa de la petición TCP/HTTP desde que el mensaje de solicitud se envía, hasta que se recibe el contenido real de la misma en el destino, tenemos que tomar la marca de tiempo del primer y último paquete de la conexión de la petición HTTP obteniendo una latencia aproximada de:

$$(25{,}492736 - 25{,}482942) * 103 = 9,7ms$$

En este caso, esta es la latencia máxima que obtenemos después de realizar la pila de pruebas anteriormente descrita. En un despliegue clásico en el que dos o más máquinas con suficientes recursos se conectan a través de una red Gigabit Ethernet, el tiempo que se pasa en el *kernel* y en el espacio de usuario en la máquina de destino suele ser la mayor parte de la latencia total, alrededor del 70 % del tiempo.

Por lo tanto, el tiempo real empleado en viajar a través de la red SDN es menor que el tiempo de procesamiento empleado dentro de la máquina para responder a la solicitud recibida y devolver el paquete. Todavía en este caso caben algunas suposiciones. Entre ellas, el comportamiento variable de la latencia debido a que no es un valor constante y varía mucho con la aplicación, el protocolo, la plataforma, el tipo de operación, la regla de prioridad impuesta,

el grado de ocupación de la tabla de conmutación y las operaciones que está llevando a cabo el conmutador en ese momento.

6.1.3. Escalabilidad

La escalabilidad a nivel de red se define como la capacidad de conectar nuevos dispositivos a la red, sin que ello repercuta de forma alguna en el rendimiento y funcionamiento de esta.

Además, esta integración debe realizarse de forma sencilla y fluida, a pesar de tener que hacer reconfiguraciones en la red, estas deben minimizarse en lo posible, afectando al mínimo número de nodos. En nuestro caso, hablamos de nodos virtuales, gestionados por el controlador SDN. Por tanto, para considerar a un sistema escalable, los indicadores de rendimiento medidos antes de la inclusión de un nodo ($\widehat{0}$) deben ser lo más similares posible a los medidos tras la inclusión del nuevo nodo ($\widehat{1}$).

Para evaluar la escalabilidad de esta arquitectura, medimos en *msg/ms* el rendimiento (*throughput*) a nivel de nodo en la red. Cuanto mayor sea el rendimiento, más escalable será la red, ya que significa que podemos añadir muchos dispositivos sin sobrecargar los nodos de la red. El rendimiento se midió varias veces con la herramienta *iperf*, con dos hosts que intercambian paquetes TCP de 8KB a través del conmutador.

El valor del KPI del 107 % se obtuvo al medir la escalabilidad de una red virtual SDN virtual desplegada en la nube. En este entorno, se espera que el rendimiento sea muy bueno. Sin embargo, este valor puede disminuir si la red SDN desplegada no está alojada en un entorno puramente virtual, y contienen elementos físicos, como switches, o si la nube aloja los diferentes nodos de la red en diferentes ubicaciones, de forma distribuida.

6.2. Indicadores de Rendimiento (KPIs) Escenario 2

Como ya se ha mostrado en el Capítulo anterior, el Escenario número dos se llevó a cabo en la plataforma de Limasol (proyecto 5GENESIS), que está compuesta, por un lado, por un centro de datos principal virtualizado (a través de *OpenStack*) donde se alojará la parte virtual de la pasarela, así como una red de transmisión dual (*dual backhaul*) tanto terrestre como satelital. Y, por otro lado, por nodo de acceso a la red 5G (*gNodeB*) conectado a un nodo de procesado de datos en el borde (*Edge node*), a los cuales se conectará la parte física de la pasarela y que hará de traductor para los otros protocolos IoT, como en este caso LoRa, con los que se conectará a los dispositivos IoT.

El objetivo de estos Indicadores de Rendimiento es el de analizar la viabilidad del despliegue, y su comportamiento, de forma que nos puedan indicar si el uso de las tecnologías de virtualización en este supuesto ha incurrido en una mejora respecto a una red de comunicación de datos tradicional. En concreto:

- Verificar la correcta funcionalidad de la arquitectura implementada en este caso, así como de todos sus elementos en conjunto (*end-to-end*). Incluyendo la funcionalidad de la virtualización de las funciones de red referentes al IoT (pasarela IoT) tanto en el núcleo como en el borde de la infraestructura.
- Evaluar el rendimiento de la infraestructura en términos de Tasa de transferencia de datos
- Evaluar la latencia o retraso (*delay*) a la que se someten los datos que viajan a través de esta infraestructura de red.
- Evaluar la cobertura y alcance obtenido a través de la integración de tecnologías IoT en esta infraestructura.

6.2.1. Latencia entre nodos (RTT)

El objetivo de este indicador de rendimiento es evaluar el tiempo que tardan los datos de viajar de un punto a otro dentro de la infraestructura de red. En concreto, los experimentos realizados para la obtención de este indicador se basaron en la medida del denominado *Round Trip Time* a nivel de aplicación entre diferentes puntos de la red en la plataforma de Limasol, dónde se incluyen como nodos los dispositivos IoT y las aplicaciones que hace uso de los datos de estos dispositivos (*Node-RED*). Para llevar a cabo los experimentos se establecieron dos configuraciones diferentes:

1. RTT entre los dispositivos y las aplicaciones ejecutadas en el nodo *edge* de la infraestructura (*4G and EPC at the Edge*), y

2. RTT a través de todo el recorrido hasta el núcleo o *core* de la infraestructura, incluyendo la red de transporte (*backhaul*) tanto terrestre como satelital, partiendo de igual forma desde los dispositivos IoT y hasta la VNF desplegada en el *core*.

Es importante decir que todos los nodos de la infraestructura ejecutaban sistemas operativos basados en *Linux*; *Raspberry Pi: Raspbian* para la pasarela física, *Ubuntu* para el nodo *Edge* y *OpenStack* para el centro de datos.

Además, uno de los propósitos de este experimento es el de descubrir la mejora en la latencia, cuando las funciones virtuales (VNFs) se alojan en el borde de la infraestructura, en contraposición con el despliegue tradicional en el núcleo, justificando así la necesidad de virtualización e implementación de un MEC virtualizado, donde los servicios se ejecutan más cerca del usuario, lo que mejora los tiempos de respuesta y descarga de trabajo al resto de la infraestructura. En este experimento, se ha de tener en cuenta no solo la latencia de la red sino también la producida por el procesado de los datos en los puntos de destino.

Además, hay que tener en cuenta el encapsulamiento y des encapsulamiento de los datos, ya que estas mediciones se realizan a nivel de aplicación. El escenario desplegado está compuesto por los siguientes componentes:

- *Dominio IoT*: por un lado, para esta prueba hemos utilizado como sensor y emisor, un módulo creado por un dispositivo *Adafruit Feather M0* con transceptor de radio *LoRa RFM95 a 868 MHz*, y un SF (*Spreading Factor*) de 7, junto con un multi-sensor de temperatura, humedad, presión y duración de la batería *Adafruit BME280* y un submódulo RTC para la sincronización temporal. Este módulo se encargó de transmitir la información de los sensores a través del protocolo LoRa a la pasarela física implementada por una placa concentradora iMST iC880a integrada en un controlador implementado en una Raspberry Pi 3, en la que se han instalado los diferentes módulos de software capaces de decapsular, traducir y reencapsular de la red LoRa al dominio 4G/5G, y enviar esta información a través de un periférico USB Huawei E3372 que opera en la banda 7 de LTE. Por otro lado, utilizamos como receptor y actuador, una placa Arduino UNO R3 conectada a diferentes LEDs que recibe la información procesada y enviada por la pasarela virtual, des-encapsula la información y ejecuta una acción en respuesta.
- *Dominio de la infraestructura*: este dominio está compuesto por el nodo de borde, la red de transporte, y el núcleo de la infraestructura, donde se encuentran los componentes gestores de la parte programable de la infraestructura, como el controlador SDN y el orquestador. En ella, la pasarela virtual convertida en un contenedor se convertirá en un recurso que podremos instanciar de forma fácil en cualquier punto de la infraestructura. En el nodo del borde, este componente fue instanciado usando contenedores Docker, y en el núcleo sobre un gestor OpenStack, se creó una máquina virtual utilizando *Ubuntu 18.04.3 LTS Bionic Beaver*.

Los componentes del dominio IoT permanecieron inmutables durante toda

la ejecución de las diferentes iteraciones de la prueba. La única diferencia fue la ubicación de la instanciación de las pasarelas virtuales, servicios específicos para el caso de uso. El tráfico intercambiado durante la ejecución de las pruebas tiene un perfil TCP. En la Figura 6.5 podemos observar un boceto a alto nivel del despliegue realizado para esta prueba con cada uno de sus componentes y como se conectan entre ellos.

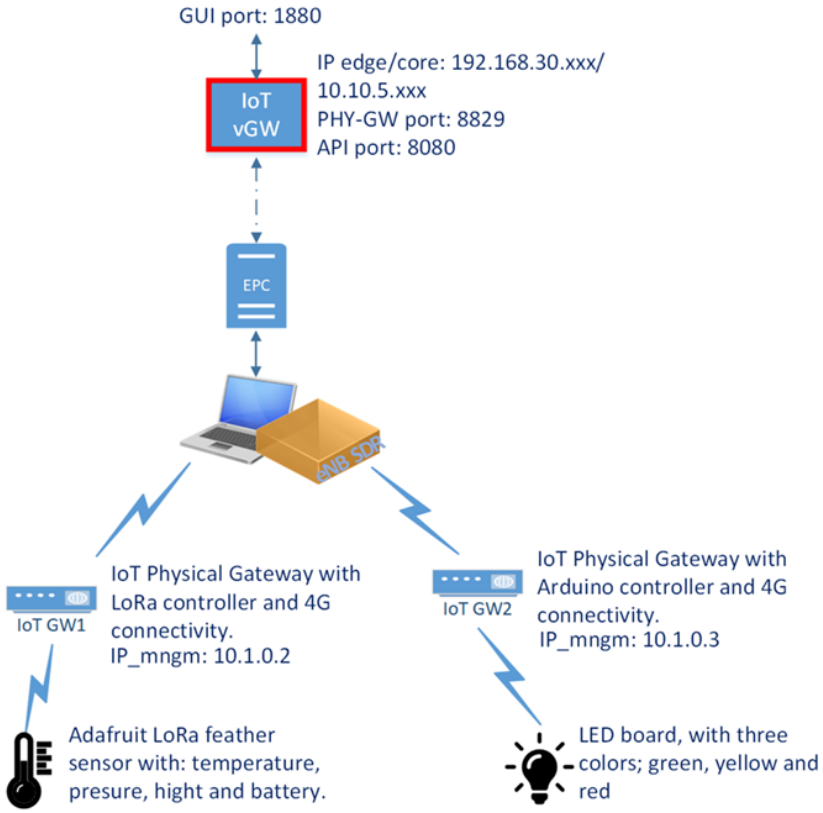

Figura 6.5: Escenario 2 despliegue para prueba de RTT.

Los valores de latencia obtenidos se presentan en las tablas siguientes:

Core

Test Case ID	RTT_Core
Descripción general	Este experimento mide la latencia sobre la infraestructura de red desplegada en la plataforma de Limasol. En concreto, entre los dispositivos conectados a la parte física de la pasarela IoT, y su aplicación final conectada a la contrapartida virtual de la pasarela desplegada como función virtual (VNF) en el núcleo de la infraestructura. El tráfico de datos atraviesa la pasarela, la red de acceso (RAN) 4G, el EPC y la red de retorno (*backhaul*) hasta llegar al *core*, donde se encuentra alojada la función virtual que recibe y des-encapsular los datos y los inyecta en la aplicación IoT. Para la obtención de estas medidas el proceso de envío de datos entre los dispositivos y la pasarela física, y entre la pasarela virtual y la aplicación IoT ha sido ignorado.
Objetivo	El experimento se llevó a cabo en un entorno interior controlado con una estación base (eNB) a una distancia de unos 5 m, con línea de visión directa (*ac:los*) y configuración de mono-antena (SISO). Operando en la banda LTE 7,5 MHz (25 PRBs). Loa paquetes de datos intercambiados procedían del tráfico IoT por lo tanto su tamaño era diferente, siendo el paquete de tamaño máximo de unos 32 bytes. La red de transmisión satélite fue configurada a 15 Mbps DL, 5 Mbps UL, con un ancho de banda especifico garantizado (*guaranteed BW*).

Componentes involucrados	▪ Dispositivos IoT (*Adafruit LoRa Feather* con sensor de humedad, temperatura y presión, y *Arduino Board* con LEDs integrados) ▪ Pasarela física (basada en Raspberry Pi), con conexión a infraestructura 5G a través de USB LTE dongle. ▪ Un nodo de acceso a la red en basado en SDR con el software OpenAirInterface (OAI de Eurecom) ▪ EPC proporcionado por Athonet ▪ Nodo Edge basado en Linux (Ubuntu 18.0) ▪ iDirect terminal satelital ▪ Núcleo con infraestructura NFVI implementada por OpenStack ▪ Controlador SDN (ODL) ▪ Orquestador (OSM) ▪ VNF con pasarela virtual y la aplicación Node-Red en contenedor virtual para visualizar la llegada de datos, sus valores y su marca de tiempo.

Estadísticas del experimento	Latency [ms]		
	Mean	95 % Conf. Interval	
		Lower bound	Upper bound
	347,7	342,1	353,3

Tabla 6.4: Descripción de la prueba de latencia entre nodos de la plataforma de Limasol en el núcleo.

Edge

Test Case ID	RTT_Edge
Descripción general	Este experimento mide la latencia sobre la infraestructura de red desplegada en la plataforma de Limasol. En concreto, entre los dispositivos conectados a la parte física de la pasarela IoT, y su aplicación final conectada a la contrapartida virtual de la pasarela desplegada como función virtual (VNF) en el núcleo de la infraestructura. El tráfico de datos atraviesa la pasarela, la red de acceso (RAN) 4G, el EPC y la red de retorno (*backhaul*) hasta llegar al *core*, donde se encuentra alojada la función virtual que recibe y des-encapsular los datos y los inyecta en la aplicación IoT. Para la obtención de estas medidas el proceso de envío de datos entre los dispositivos y la pasarela física, y entre la pasarela virtual y la aplicación IoT ha sido ignorado.
Objetivo	El experimento se llevó a cabo en un entorno interior controlado con una estación base (eNB) a una distancia de unos 5 m, con línea de visión directa (*direct* LoS) y configuración de mono-antena (SISO). Operando en la banda LTE 7, 5 MHz (25 PRBs). Loa paquetes de datos intercambiados procedían del tráfico IoT por lo tanto su tamaño era diferente, siendo el paquete de tamaño máximo de unos 32 bytes. La red de transmisión satélite fue configurada a 15 Mbps DL, 5 Mbps UL, con un ancho de banda especifico garantizado (*guaranteed BW*).

Componentes involucrados	▪ Dispositivos IoT (Adafruit LoRa feather con sensor de humedad, temperatura y presión, y Arduino Board con Leds integrados) ▪ Pasarela física (basada en Raspberry Pi), con conexión a infraestructura 5G a través de USB LTE dongle. ▪ Un nodo de acceso a la red en basado en SDR con el software OpenAirInterface (OAI de Eurecom) ▪ EPC proporcionado por Athonet ▪ Nodo Edge basado en Linux (Ubuntu 18.0) ▪ iDirect terminal satelital ▪ Núcleo con infraestructura NFVI implementada por OpenStack ▪ Controlador SDN (ODL) ▪ Orquestador (OSM) ▪ VNF con pasarela virtual y la aplicación Node-Red en contenedor virtual para visualizar la llegada de datos, sus valores y su marca de tiempo.

	Latency [ms]		
Estadísticas del experimento	Mean	95 % Conf. Interval	
		Lower bound	Upper bound
	69,6	67,8	71,4

Tabla 6.5: Descripción de la prueba de latencia entre nodos en la plataforma de Limasol en el borde.

Como se ve descrito en las tablas Tabla 6.4 y Tabla 6.5, podemos resumir los resultados de estos experimentos como sigue:

- Latencia del enlace ascendente (UL) en el segmento del borde: la media fue de 69,6 ms, mientras que los límites inferior y superior para el intervalo de confianza del 95 % fueron 67,8 y 71,4 ms respectivamente.
- Latencia del enlace ascendente (UL) para la ruta de extremo a extremo, núcleo: la media fue de 347,7 ms, mientras que los límites inferior y superior del intervalo de confianza del 95 % fueron de 342,1 y 353,3 ms, respectivamente.

Así, obtuvimos unos 70 ms para la red 5G en el borde y unos 350 ms para el E2E incluyendo el *backhaul* por satélite. Una vez más, la diferencia implica una gran reducción de la latencia percibida para el tráfico que se gestiona localmente en el borde en lugar de viajar hasta el nodo central o núcleo y pone de relieve el valor de acercar a los dispositivos las funcionalidades de la red (LBO) en el borde. Además, obtenemos una latencia máxima hasta el borde de 110 ms en un pico concreto, y un máximo de 635 ms hasta el núcleo, un valor mínimo de 321 ms y 50 ms, respectivamente.

6.2.2. Tasa de transferencia de datos efectiva (*Throughput*)

La tasa de transferencia de datos efectiva, conocida por su nombre en inglés como *throughput* es el volumen de datos o información neta, normalmente medida en bits, bytes, megabytes, etc. por segundo, que se viaja a través de un sistema o red. En este caso, es la cantidad de datos que se envían a través de nuestra infraestructura.

El objetivo de estos experimentos es determinar el volumen de datos que puede viajar de un punto a otro de la infraestructura tanto desde los dispositivos a las aplicaciones (UL), como a la inversa (DL). Este volumen de

datos se extrae midiendo el tráfico, en este caso tráfico TCP entre diferentes puntos de nuestra plataforma en Limasol.

De igual forma que en los experimentos de latencia, esta tasa de transmisión ha sido medida en dos supuestos:

1. Tasa de envío de datos entre los dispositivos y el segmento de infraestructura en el borde (*Edge*). En este caso se mide la tasa de trasmisión en LTE entre los dispositivos IoT y el nodo *edge* situado en el EPC.

2. Tasa de envío de datos a través de toda la infraestructura al completo (E2E), desde los dispositivos IoT hasta el mismo núcleo (*core*) de la infraestructura.

También, como en el caso del experimento anterior, éste se ha llevado a cabo en un entorno interior controlado, en un laboratorio, con dispositivos transmitiendo en la banda 4G, separados del nodo base unos 5 metros, línea directa de contacto (*direct* LoS) y configuración de entrada y salida única (SISO) La estación base (eNB) se ejecuta usando un software de implementación de SDR basado en *OpenAirInterface*, que opera en la Banda 7, con un ancho de banda máximo de 5MHz (25 PRBs). De igual forma, la pasarela IoT recoge los datos de los dispositivos utilizando el protocolo LoRa, operando en la banda europea de 868MHz, con un factor de propagación 7 (SF), y un ancho de banda de 125 kHZ, desencapsula estos datos y los envía a través de la red LTE mediante un componente propietario (COTS) con conexión USB a la pasarela.

En este caso, la tasa de transmisión será evaluada únicamente a partir de la pasarela y para todo el resto de la infraestructura, ya que la tasa de transmisión de los dispositivos a la pasarela está limitada por el protocolo LoRa, puesto que dada la configuración de este protocolo (SF 7, BW 125 kHz y 868MHZ band), se puede conseguir únicamente una tasa de envío máxima por canal de

4844,7 bps (*bits per second*) [108]. Por ello, la sección de la infraestructura que pertenece al protocolo LoRa queda fuera de la medida en este experimento, ya que nuestro interés es el de averiguar la tasa de transmisión de datos a través de nuestro despliegue de infraestructura programable 5G.

Para realizar este experimento se ha utilizado la herramienta de pruebas *iperf3*. Un par de instancias de esta herramienta se lanzaron en los dos nodos correspondientes al origen y destino de los datos. La instancia cliente se desplegó en el lado de la pasarela física, mientras que la instancia que hacía de servidor se desplegó en el borde (*Edge*) y en el núcleo (*Core*) para cada uno de los experimentos. Ambos tipos de tráfico; desde y hacia los dispositivos (UL and DL) eran tipo TCP con un tamaño de ventana por defecto, y llevando a cabo un total de 5 conexiones concurrentes, con 25 iteraciones por experimento. Para ejecutar los experimentos y extraer los resultados de estos se utilizaron pequeños scripts de automatización en Linux.

Core

Test Case ID	TP_Core
Descripción general	Medida de tasa de transmisión de datos entre los dispositivos y el Core.
Objetivo	Calcular la cantidad de datos que pueden viajar a través de los distintos segmentos de la infraestructura, usando un determinado tráfico de datos y una configuración específica.
Componentes involucrados	▪ Pasarela física (basada en Raspberry Pi), con conexión a infraestructura 5G a través de USB LTE dongle. ▪ Un nodo de acceso a la red en basado en SDR con el software OpenAirInterface (OAI de Eurecom) ▪ EPC proporcionado por Athonet ▪ Nodo Edge basado en Linux (Ubuntu 18.0) ▪ iDirect terminal satelital ▪ Núcleo con infraestructura NFVI implementada por OpenStack ▪ Controlador SDN (ODL) ▪ Orquestador (OSM) ▪ VNF con pasarela virtual ▪ Aplicación cliente y servidor *iperf* para simular el envío y recepción del tráfico.

	Throughput [Mbps] Downlink		
	Mean	95 % Conf. Interval	
		Lower bound	Upper bound
Estadísticas del experimento	9,81	9,48	10,15
	Throughput [Mbps] Uplink		
	Mean	95 % Conf. Interval	
		Lower bound	Upper bound
	4,41	4,33	4,48

Tabla 6.6: Descripción de la prueba de tasa de transmisión de datos en el núcleo de la red.

Edge

Test Case ID	TP_Edge
Descripción general	Medida de tasa de transmisión de datos entre los dispositivos y el Edge.
Objetivo	Calcular la cantidad de datos que pueden viajar a través de los distintos segmentos de la infraestructura, usando un determinado tráfico de datos y una configuración específica.
Componentes involucrados	▪ Pasarela física (basada en Raspberry Pi), con conexión a infraestructura 5G a través de USB LTE dongle. ▪ Un nodo de acceso a la red en basado en SDR con el software OpenAirInterface (OAI de Eurecom) ▪ EPC proporcionado por Athonet ▪ Nodo Edge basado en Linux (Ubuntu 18) ▪ iDirect terminal satelital ▪ Controlador SDN (ODL) ▪ Orquestador (OSM) ▪ VNF con pasarela virtual ▪ Aplicación cliente y servidor *iperf* para simular el envío y recepción del tráfico

	Throughput [Mbps] Downlink		
	Mean	95 % Conf. Interval	
		Lower bound	Upper bound
Estadísticas del experimento	9,09	8,74	9,44
	Throughput [Mbps] Uplink		
	Mean	95 % Conf. Interval	
		Lower bound	Upper bound
	4,09	3,92	4,26

Tabla 6.7: Descripción de la prueba de tasa de transmisión de datos en el borde de la red.

Los resultados de estos experimentos que se observan en la Tabla 6.6 y la Tabla 6.7 se pueden resumir de esta manera:

- *Throughput* TCP en el segmento *Core*
  - *Downlink*: la media fue de 9,81 Mbps, mientras que los límites inferior y superior del intervalo de confianza del 95 % fueron de 9,48 y 10,15 Mbps respectivamente.
  - *Uplink*: la media fue de 4,41 Mbps, mientras que los límites inferior y superior del intervalo de confianza del 95 % fueron de 4,33 y 4,48 Mbps respectivamente.
- *Throughput* TCP en el segmento *Edge*
  - *Downlink*: la media de transmisión fue de 9,09 Mbps, mientras que

los límites inferior y superior del intervalo de confianza del 95 % fueron de 8,74 y 9,44 Mbps respectivamente.

- *Uplink*: la media fue de 4,09 Mbps, mientras que los límites inferior y superior del intervalo de confianza del 95 % fueron de 3,92 y 4,26 Mbps respectivamente.

Los valores de transmisión obtenidos que se han presentado se acercan a los esperados y se corresponden con la capacidad nominal de los enlaces de *backhaul* y de acceso. Cabe destacar, que este despliegue poseía además una sección de red de transporte satelital, cuyo enlace fue configurado a 15 Mbps DL / 5 Mbps UL, y también el rendimiento de la RAN (a 5 MHz/25 PRBs) es algo inferior (10 Mbps DL). Es decir, los valores experimentales verifican las expectativas establecidas. A diferencia de las mediciones de latencia, en el caso del rendimiento no hay diferencia entre las mediciones con el borde y el núcleo. Esto se debe a que, en la configuración actual, el *backhaul* sea terrestre o por satélite no constituye un cuello de botella en la cadena total (E2E). En otras palabras, proveer las funciones de red más cerca de los dispositivos (LBO) no aporta un valor añadido para la tasa de transmisión.

6.2.3. Tiempo de Creación de Servicio (*Service Creation Time*)

El tiempo de creación de servicio se define, en este caso, como el tiempo transcurrido hasta que se produce la configuración de todos los elementos de la infraestructura virtual necesarios para el correcto funcionamiento del servicio (e.g. actualización de tablas de enrutado), la creación de la *slice* de red y el despliegue y ejecución de todos los componentes de la aplicación misma, en este caso, el despliegue de la VNF con la pasarela virtual y su correcta ejecución. Para este Indicador tendremos en cuenta el hecho de que el controlador de red, así como la infraestructura virtual ya han sido desplegadas, por lo tanto, solo se mide el tiempo de despliegue de los componentes específicos de la aplicación

y la configuración de la red para permitir el tráfico entre los dispositivos y la aplicación final.

Además, se ha de tener en cuenta que la parte física de este despliegue (dispositivos y pasarela física) deben estar también ejecutándose y transmitiendo en el momento de realizar la petición, para que los datos sean recogidos por el gNodeB y enviados a través de su correspondiente *slice*, una vez desplegada. De igual forma que en los otros Indicadores, la medición de la creación del servicio se llevará a cabo desplegando la VNF de la pasarela virtual en el Core de la infraestructura y en el Edge.

Core

Test Case ID	SCT_Core
Descripción general	Medida del tiempo de creación de servicio en el core.
Objetivo	Calcular el tiempo necesario para la creación de un segmento de red virtual donde implementar la VNF indicada en el núcleo del centro de datos.
Componentes involucrados	▪ Portal de gestión y ELMC ▪ Slice Manager (Katana) ▪ Controlador SDN (ODL) ▪ VNF (VIR-GW) ▪ Orquestador (OSM) ▪ Núcleo con infraestructura NFVI implementada por OpenStack
Estadísticas del experimento	Service Creation Time [s] Mean 45.198 +/- 2.064165

Tabla 6.8: Descripción de la prueba de medida de tiempo de creación de servicios en el núcleo de la red.

Esta prueba mide el tiempo de creación de una *slice* de red y el despliegue de un servicio que consiste en una VNF que contiene la pasarela virtual y que se localiza en el núcleo del despliegue, esto es, el centro de datos. Esta prueba se

inicia cuando desde al portal de la aplicación de gestión (denominada Portal) se ejecuta una petición para la creación de un nuevo servicio.

El Portal entonces envía esta petición al gestor del ciclo de vida del experimento (ELCM) que la recoge y le indica al gestor de secciones o *slices* (*Katana*[6]) que debe crear una nueva *slice* para alojar una nueva configuración de red y ejecutar la aplicación seleccionada. Este gestor entonces se comunica con el Orquestador de Servicios (OSM) para crear e inicializar la VNF, e indicar al controlador SDN que debe incluir reglas de enrutado en los switches virtuales para enviar el tráfico desde el origen (puerto de entrada) hasta la VNF. También, es el OSM el que se encarga de recoger los resultados del tiempo transcurrido entre que la petición es realizada hasta que la VNFs envía el primer signo de vida (*hearthbeat*) confirmando que está desplegada y ejecutándose. Por los datos obtenidos de las diferentes iteraciones de la prueba podemos ver que la media de tiempo de creación de servicio es alrededor de 45 segundos.

En la siguiente figura, Figura 6.6 se puede observar que este tiempo es similar en cada una de las iteraciones ejecutadas. Naturalmente, este valor varía dependiendo de la cantidad de recursos disponibles en ese momento por parte de la infraestructura (*OpenStack* clúster) y, lo que es más importante, el tamaño de la VNF a instanciar. En el gráfico se pueden observar el tiempo medio de cada una de las iteraciones (25), no pasando ninguna de ellas del minuto.

[6] https://github.com/medianetlab/katana-slice_manager

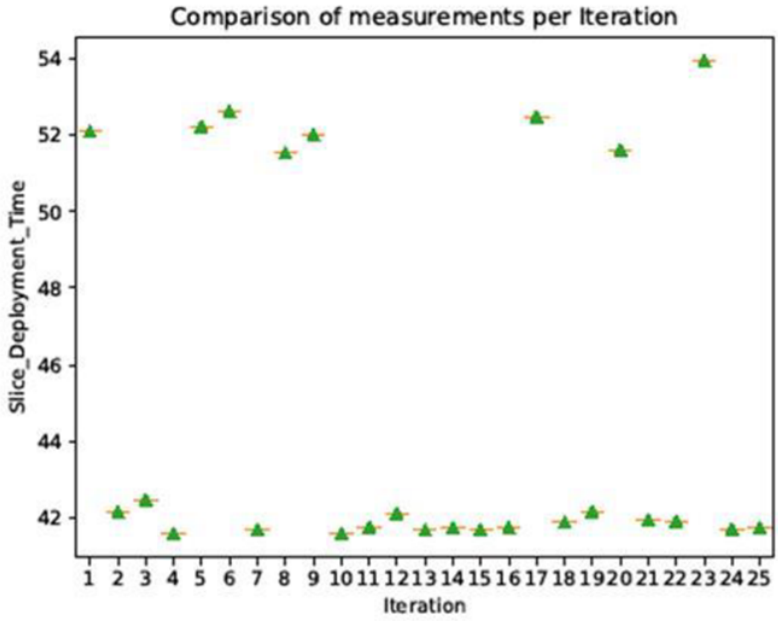

Figura 6.6: Comparación del tiempo de creación y despliegue de *slices* virtuales en el núcleo de la red.

Edge

Test Case ID	SCT_Edge
Descripción general	Medida del tiempo de creación de servicio en el edge.
Objetivo	Calcular el tiempo necesario para la creación de un segmento de red virtual donde instanciar la VNF indicada en el nodo del borde de la infraestructura.
Componentes involucrados	▪ Portal de gestión, Slice Manager (Katana) y ELMC ▪ Controlador SDN (ODL) ▪ VNF (VIR-GW) ▪ Orquestador (OSM) ▪ EPC proporcionado por Athonet ▪ Nodo Edge basado en Linux (Ubuntu 18) ▪ iDirect terminal satelital ▪ Núcleo con infraestructura NFVI implementada por OpenStack
Estadísticas del experimento	Service Creation Time [s] Mean 102.551 +/- 4.285408

Tabla 6.9: Descripción de la prueba de medida de tiempo de creación de servicios en el borde de la red.

En este segundo supuesto se midió el tiempo de creación del servicio de igual forma, sin embargo, la función virtual (VNF) se fue instanciada en el borde de la infraestructura (*edge*). Este experimento, por tanto, mide la creación del despliegue de la sección de red virtual y la instanciación y ejecución del servicio en el nodo que se encuentra más cerca de los dispositivos. De igual forma, este experimento se inicia cuando se envía la orden de creación de servicio desde el Portal al ELCM, y éste se encarga de transmitírselo al gestor de secciones (*slice manager*, Katana) que le indicará al controlador como configurar la nueva sección de red y al orquestador que cree y ejecute la función virtual (VNF) donde se encuentra la parte virtual de la pasarela.

Además, este orquestador es el que se encarga de recoger los datos de tiempo desde que recibe la petición hasta que esta se completa. En este caso, tanto el ELCM como el orquestador OSM, se encuentran en el núcleo o *Core* de la infraestructura, mientras que la función virtual a desplegar es creada en el *Edge*, por lo que no es casual que el tiempo de creación de servicio en este caso sea casi el doble, puesto que tiene en cuenta el tiempo de transmisión de datos a través de la red para levantar dicho servicio.

La medida de tiempo se lleva a cabo en 25 iteraciones, y en este caso el tiempo medio que obtenemos para la total creación del servicio es de 102 segundos. Independientemente de algunos casos concretos que se salen de la media, se puede observar un tiempo constante y que varía poco de una a otra iteración. En la siguiente figura, Figura 6.7, se puede ver detalladamente el valor del tiempo de creación para cada una de las iteraciones ejecutadas.

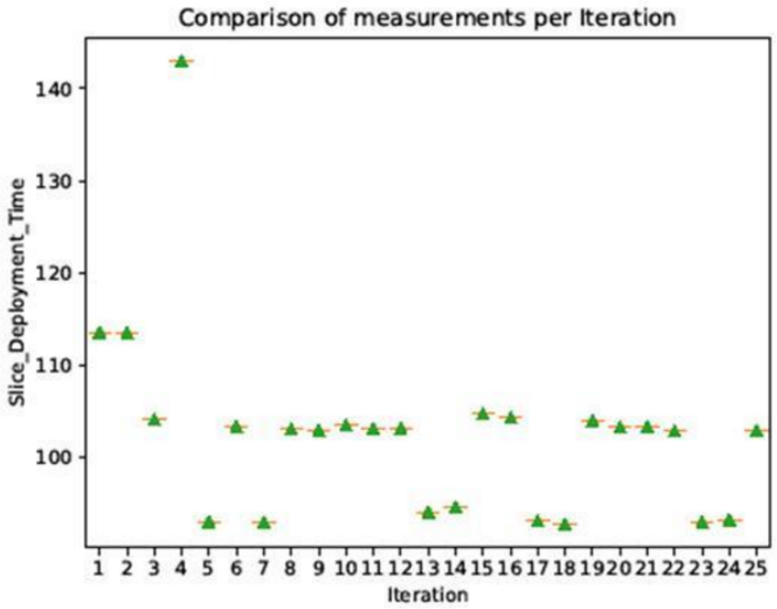

Figura 6.7: Comparación del tiempo de creación y despliegue de *slices* virtuales en el borde de la red.

De nuevo, cabe insistir que, obviamente, el tiempo medio de creación de servicio, en este caso, es casi el doble que en el primero. Sumado al retraso provocado en la comunicación de la solicitud hasta el nodo *Edge*, en comparación con el centro de datos donde se encuentra el núcleo de la infraestructura, este nodo *Edge* posee menos recursos en términos de potencia y memoria, por lo que instanciar y ejecutar la función virtual lleva incluida cierta demora. Sin embargo, observamos que el tiempo medio total no supera los 2 minutos, lo que es un periodo bastante aceptable para un despliegue de este tipo de servicios.

Capítulo 7

Conclusiones y Líneas de Trabajo Futuras

En este último capítulo de la memoria se realiza un resumen de todo lo presentado hasta ahora, analizando hasta que punto se han alcanzado los objetivos iniciales y presentando una resolución o conclusiones a la problemática existente de los despliegues de IoT. Por último, como posible continuidad a este trabajo, se presentan unas líneas futuras o temáticas dónde se podría hacer mayor hincapié y continuar extendiendo los estudios realizados hasta la fecha.

7.1. Conclusiones

La motivación de esta tesis venía dada por la problemática surgida en los últimos años alrededor de la gestión de redes en los despliegues cada vez más comunes de dispositivos inteligentes, o lo que comúnmente se denomina el IoT. Las redes tradicionales, incapaces de soportar y manejar el tráfico cambiante y enorme de datos de sensores que viajan a través de ella, han de reinventarse y mejorarse para ser capaces de cumplir los estrictos requisitos que demandan unas aplicaciones cada vez más extendidas. Una bajísima latencia para casos de uso críticos como coches autónomos u operaciones delicadas de forma remota, un amplio ancho de banda capaz de proveer del mejor video (UHD) y la mejor experiencia para la realidad virtual y aumentada, y una infraestructura de red capaz de soportar el tráfico de millones y millones de dispositivos heterogéneos comunicándose entre sí, son los principales desafíos a los que se enfrentan las redes del mañana.

Para esta reinvención, paradigmas como las redes virtuales brindan un enfoque nuevo en donde la abstracción de la complejidad inherente a la red facilita su gestión, y ayuda a cumplimentar estos requisitos.

La arquitectura propuesta quiere hacer uso de la virtualización de las redes para facilitar su gestión y es por ello, que hemos diseñado una nueva arquitectura que ha combinado los elementos más relevantes de esos paradigmas, por un lado los componentes presentes en cualquier despliegue IoT: dispositivos, *gateways*, plataformas y aplicaciones, y por el otro, las herramientas y perspectivas de gestión presentes en las redes SDN: switches virtuales, controladores de red en el plano lógico, aplicaciones enfocadas a la red y funciones virtuales. La arquitectura de ha puesto a prueba y validado en dos escenarios diferentes. A continuación, se resumen las conclusiones obtenidas en cada una de las etapas de esta investigación, desde la pura recopilación y análisis de las tecnologías, hasta los resultados de rendimiento obtenidos en cada uno de los escenarios.

7.1.1. Estado del arte

Durante esta etapa del trabajo se llevó a cabo un amplio análisis de la literatura disponible en relación con las arquitecturas más comunes definidas por organismos oficiales (*ITU, IEEE, IFTF*, etc.) para el Internes de las Cosas, así como los protocolos más usados en cuestión de comunicación entre dispositivos de recursos limitados (a nivel de comunicación y red como *IPV6, LoRa, ZigBee, PanStamp, BLE*, entre otros, y a nivel de aplicación y servicios COAP, MQTT, y en ocasiones HTTP). Por otro lado, se ha detallado las características de las redes definidas por software y las tecnologías que nos ayudan a implementarlas, así como la arquitectura principal definida por la ETSI para los entornos NFV.

Consideramos que toda esta revisión de la literatura ha sido muy útil y básicamente necesaria para llevar a cabo la selección de herramientas que más tarde implementarían la arquitectura de combinación en cada uno de los escenarios. En concreto, decisiones como qué controlador SDN utilizar o cómo desarrollar los módulos de la pasarela física y virtual para hacer un buen uso de los datos, han sido sustentadas en la información recogida durante esta fase, y en la comparativa con otros trabajos que se engloban en la misma temática.

7.1.2. Arquitectura

Tras el detallado análisis llevado a cabo sobre las diferentes arquitecturas del paradigma IoT y SDN, se llevó a cabo un proceso de diseño de una arquitectura de combinación presentada en el Capítulo 3. Esta arquitectura tiene en cuenta las necesidades y componentes básicos que hemos observado a través de todas y cada una de las arquitecturas IoT analizadas. De nuevo, dispositivos (capa de recolección de datos), pasarlas (*Edge*, capa de red), nodos de transporte (capa de red), plataformas (capa de procesado) y aplicaciones (capa de negocio), además de tener en cuenta en qué lugar cada una de estas capas ha de ser implementada, sea más cerca o más lejos del usuario (*Edge o*

Cloud).

Por otro lado, la arquitectura se adhiere perfectamente a los principios presentados por la arquitectura SDN/NFV, en la que tenemos los componentes físicos de la infraestructura, la capa de virtualización con todas las funciones de red ejecutándose sobre ellas y más adelante tanto los servicios y plataformas donde la información de los dispositivos queda almacenada, como la verdadera panacea de esta propuesta arquitectónica que es el módulo de orquestación y gestión de los recursos virtuales, donde encontramos tanto el controlador, como los demás componentes de orquestación (OSM, VIM y gestor de VNFs).

En un principio la arquitectura combinada posee todos los componentes necesarios para el funcionamiento óptimo de un despliegue IoT, y para su gestión dinámica y automatizada. Lo que quedará demostrado más adelante en los escenarios donde esta se ha validado.

7.1.3. Escenario 1

La mayoría de los despliegues actuales de la IoT se basan en conceptos de "bucle cerrado", centrándose en un propósito específico y aislándose del resto de posibilidades o casos de uso. La integración entre elementos heterogéneos dentro de un mismo despliegue es compleja y no suele tratar de hacerse a nivel de dispositivo o de red, sino que se limita a la homogeneización de los datos una vez recogidos.

En el proyecto INTER-IoT, donde se engloba este escenario se aspira a crear interoperabilidad en todas las capas de pila IoT, ya sea a nivel de dispositivos, redes, plataformas, servicios y/o aplicaciones, permitiendo una continuidad global de datos, infraestructuras y servicios que puedan servir para diferentes casos de uso IoT y no estar únicamente centrados en uno. Con esto en mente, el Escenario 2 implementa la arquitectura de combinación previamente diseñada en un entorno de interoperabilidad a nivel de red, donde se buscan como objetivos principales, la traducción de protocolos a nivel de red y la gestión

y priorización a través de parámetros de Calidad de Servicio de los flujos de datos que viajan a través de la red.

Observando más en detalle los indicadores de rendimiento que han sido evaluados en este segundo escenario, encontramos:

- Extensibilidad, o integración de nuevos controladores a la red. Una de las características a tener en cuenta al diseñar la arquitectura de combinación es la de la flexibilidad. Las tecnologías que la instancian esta arquitectura no son impuestas ni concretas, sino que simplemente deben cumplir unos requisitos, principalmente de interfaces y funciones, para que puedan ser utilizadas en los despliegues. Así pues, componentes como el controlador SDN no son estrictamente indicados, sino que dependiendo de cada escenario y la particularidad del caso de uso puede utilizarse uno u otro. En este caso, en un primer caso de uso se utilizó el controlador RYU, ya que su sencillez y modularidad permite el desarrollo de aplicaciones rápidas sobre el mismo, y el uso de APIs REST abiertas permite la gestión de los nodos de la red de una manera directa. Por otro lado, en el segundo caso de uso (así como en el segundo escenario) se optó por utilizar el controlador de OpenDayLight, ya que, su robustez y características eran más útiles para cada caso.

- Latencia entre los nodos de la red virtual. Este indicador es muy relevante en este caso, puesto que necesitamos demostrar que la separación entre plano de datos y lógico como principal aproximación en las redes SDN no supone ningún inconveniente, ni incorpora mayor retraso en el envío de paquetes que en el caso de una red tradicional.

- Por último, la escalabilidad, o la capacidad de añadir nuevos nodos al despliegue sin que el rendimiento del sistema se vea mermado o reducido de forma alguna. Para este indicador se evaluó la tasa de transferencia y la latencia con una cantidad incremental de nodos, analizando como afectaba en número de nodos total a estos valores. Como ya se indica en

el apartado este indicador demostró ser bastante estable, sin embargo, habría que probar otras configuraciones que no solo incorporaran nodos virtuales in la red, sino también nodos físicos, lo que podría suponer un cambio en los valores evaluados.

7.1.4. Escenario 2

Este escenario presenta los principales resultados de la implementación de la arquitectura combinada SDN-IoT en un entorno muy particular dónde encontramos la red 5G móvil. La plataforma de Limasol, centrada principalmente en al Caso de Uso de mMTC, fue desplegada con éxito y se pudieron llevar a cabo diferentes pruebas para evaluar su rendimiento.

Principalmente, los indicadores de rendimiento fueron los siguientes:

- Latencia entre nodos. En este indicador se llevó cabo una comparativa de latencia entre los servicios virtuales que se despliegan al borde de la red, y aquellos que se despliegan en el núcleo. Lo primero que cabe decir es que esta migración de servicios y la capacidad de llevar estos más cerca del usuario es imposible de conseguir si no se implementan a través de funciones virtuales sobre una red programable. Por lo tanto, es una ventaja directa de la sinergia entre estos dos paradigmas. Analizando más en detalle, observamos que sin ninguna duda la latencia se ve visiblemente reducida al acercar estos servicios al usuario. Mientras que esto puede sonar como algo directamente beneficioso, en nuestro caso de uso una baja latencia no implica directamente una mejora sustancial en el mismo, ya que en nuestro caso de uso en principal requisito a cubrir era el de proporcionar una cobertura decente en áreas remotas o rurales. Por lo que, dependiendo del caso de uso en particular se podría sacrificar un poco de esa latencia, para obtener una mejora en la cobertura.
- Transferencia de datos. En este caso, al igual que en el anterior, se hizo una evaluación de la tasa de transferencia de datos que puede viajar

a través de la red desde los dispositivos hasta los servicios, estando estos alojados en el borde o en el núcleo. De nuevo se demostró que la transferencia de datos era suficiente para que el funcionamiento del servicio fuera óptimo en ambos casos, pero cabe decir que aquí, de igual forma, se debe tener en cuenta el caso de uso particular que se esté implementando. La tasa de datos enviada por los dispositivos inteligentes suele ser muy baja, por lo que no hace falta un gran ancho de banda para su transmisión a no ser que se alcancen densidades muy elevadas de los mismos.

- Por último, el Tiempo de Creación de Servicio, indicador que consideramos el más relevante, puesto que nos indica la facilidad con la que un nuevo servicio puede ser desplegado y estar operativo, de nuevo en cualquier emplazamiento del despliegue. Esto es relevante, ya que, directamente tiene un impacto en la escalabilidad del sistema, al poder añadir una nueva pasarela física y crear su contra partida virtual en menos de 5 minutos, pudiendo así alojar más dispositivos, y procesar más datos en cuestión de minutos.

7.2. Líneas futuras de investigación

Una vez finalizado este trabajo, miramos atrás y aun sabiendo todos los resultados obtenidos y el aprendizaje y conclusiones que hemos extraído del mismo, también somos conscientes de la cantidad de posibilidades en investigación aún abiertas de cara al futuro. En concreto, queremos presentar varias líneas de trabajo en las que sería interesante ampliar el estudio realizado.

- Las tecnologías englobadas en los paradigmas principales; el IoT y las SDN, se encuentran aún en un estado embrionario y cambiante en el cual cada año se implementan mejoras en sus características

y funcionalidades o, directamente, nuevas herramientas aparecen en el mercado para sustituir a las anteriores. Es por ello por lo que la aunque la arquitectura propuesta sea relevante a lo largo del tiempo, las tecnologías que la implementan han de ser renovadas, probar nuevas herramientas y comparar unas con las anteriores para seguir mejorando el rendimiento de los despliegues.

- Dentro de las funciones de red virtual (VNF) nosotros hemos implementado aquellas que considerábamos relevantes para los escenarios. Sin embargo, estas funciones pueden ser extendidas y se deben desarrollar nuevas aplicaciones de red que hagan uso de todo el potencial que ofrece la integración de redes virtuales en despliegues de IoT.

- Dentro de los Escenarios propuestos de INTER-IoT y 5GENESIS, sería interesante analizar e incorporar otros casos de uso o aplicaciones para las arquitecturas allí implementadas, de forma que englobe un mayor número de aplicaciones de red.

- En el caso de protocolos IoT en el segundo Escenario, únicamente se pudo llevar a cabo la serie de pruebas con el protocolo LoRa. La incorporación de otros protocolos de sensores inalámbricos de baja potencia (LPWAN) como Wi-Fi, SigFox, BLE, entre otros, y su evaluación bajo los mismos indicadores de rendimiento sería muy interesante desde un punto de vista comparativo, o incluso para abarcar, como hemos dicho en el punto anterior, diferentes casos de uso para la misma plataforma.

Referencias

[1] Zodiac fx kickstarter project. https://www.kickstarter.com/projects/northboundnetworks/zodiac-fx-the-worlds-smallest-openflow-sdn-switch/. Accessed: 2020-09-30.

[2] Martin Bauer, Mathieu Boussard, Nicola Bui, Francois Carrez, Christine (SIEMENS, Jourik (ALUBE, Carsten (SAP, Stefan Meissner, Andreas IML, Alexis Olivereau, Matthias (SAP, Walewski Joachim, Julinda Stefa, and Alexander Salinas. Internet of things – architecture iot-a deliverable d1.5 – final architectural reference model for the iot v3.0, 07 2013.

[3] Cisco. Building the internet of things, 07 2014. URL: http://cdn.iotwf.com/resources/72/IoT_Reference_Model_04_June_2014.pdf.

[4] ETSI GS NFV-EVE. Network functions virtualisation (nfv); ecosystem; report on sdn usage in nfv architectural framework. Technical Specification ETSI NFV GS-EVE 005, European Telecommunications Standards Institute, 2015. URL: https://docbox.etsi.org/isg/nfv/open/Publications_pdf/Specs-Reports/NFV-EVE%20005v1.1.1%20-%20GS%20-%20SDN%20usage%20in%20NFV%20Report.pdf.

[5] D2.1 Requirements of the Facility section 4.3 Limassol Platform. 5GENESIS H2020 project, 2018. URL: https://5genesis.eu/research-papers/.

[6] Cisco annual internet report. White paper, Cisco, 2020. URL: https://www.cisco.com/c/en/us/solutions/collateral/executive-

perspectives / annual - internet - report / white - paper - c11 - 741490.pdf.

[7] Diego Kreutz, Fernando M. V. Ramos, Paulo Esteves Veríssimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015. doi:10.1109/JPROC.2014.2371999.

[8] Fei Hu, Qi Hao, and Ke Bao. A survey on software-defined network and openflow: From concept to implementation. *IEEE Communications Surveys Tutorials*, 16(4):2181–2206, 2014. doi:10.1109/COMST.2014.2326417.

[9] Raj Jain and Subharthi Paul. Network virtualization and software defined networking for cloud computing: a survey. *IEEE Communications Magazine*, 51(11):24–31, 2013. doi:10.1109/MCOM.2013.6658648.

[10] F. Sivrikaya and B. Yener. Time synchronization in sensor networks: a survey. *IEEE Network*, 18(4):45–50, 2004. doi:10.1109/MNET.2004.1316761.

[11] Li Da Xu, Wu He, and Shancang Li. Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics*, 10(4):2233–2243, 2014. doi:10.1109/TII.2014.2300753.

[12] Thavamani Shanmugam and B. Malarkodi. Analysis of campus network management challenges and solutions. In *2019 TEQIP III Sponsored International Conference on Microwave Integrated Circuits, Photonics and Wireless Networks (IMICPW)*, pages 312–316, 2019. doi:10.1109/IMICPW.2019.8933236.

[13] Yimeng Zhao, Luigi Iannone, and Michel Riguidel. On the performance of sdn controllers: A reality check. In *2015 IEEE Conference on Network*

Function Virtualization and Software Defined Network (NFV-SDN), pages 79–85, 2015. doi:10.1109/NFV-SDN.2015.7387410.

[14] Peterson, Cascone, O'Connor, Vachuska, and Davie. *Software-Defined Networks: A Systems Approach*. 2020. URL: https://github.com/SystemsApproach/SDN.

[15] Enio Kaljic, Almir Maric, Pamela Njemcevic, and Mesud Hadzialic. A survey on data plane flexibility and programmability in software-defined networking. *IEEE Access*, 7:47804–47840, 2019. doi:10.1109/ACCESS.2019.2910140.

[16] Taesang Choi, TaeYeon Kim, Wouter TaverNier, Aki Korvala, and Jussi Pajunpaa. Agile management of 5g core network based on sdn/nfv technology. In *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 840–844, 2017. doi:10.1109/ICTC.2017.8190795.

[17] Yudong Wang and Lirong Liu. Centralized network management in sdn-based communication network of substation. In *2019 IEEE International Conference on Industrial Internet (ICII)*, pages 87–91, 2019. doi:10.1109/ICII.2019.00027.

[18] Catello Di Martino, Anwar Walid, and Marina Thottan. A cloud-based platform enabling automation in resiliency and performance testing of sdn. In *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–2, 2018. doi:10.1109/NFV-SDN.2018.8725749.

[19] Bram Naudts, Wouter Tavernier, Sofie Verbrugge, Didier Colle, and Mario Pickavet. Deploying sdn and nfv at the speed of innovation: toward a new bond between standards development organizations, industry fora, and open-source software projects. *IEEE Communications Magazine*, 54(3):46–53, 2016. doi:10.1109/MCOM.2016.7432171.

[20] Mohamed Aslan and Ashraf Matrawy. On the impact of network state collection on the performance of sdn applications. *IEEE Communications Letters*, 20(1):5–8, 2016. doi:10.1109/LCOMM.2015.2496955.

[21] Kazuya Suzuki, Kentaro Sonoda, Nobuyuki Tomizawa, Yutaka Yakuwa, Terutaka Uchida, Yuta Higuchi, Toshio Tonouchi, and Hideyuki Shimonishi. A survey on openflow technologies. *IEICE Transactions on Communications*, E97.B:375–386, 02 2014. doi:10.1587/transcom.E97.B.375.

[22] Bruce Davie, Teemu Koponen, Justin Pettit, Ben Pfaff, Martin Casado, and et al. Gude. A database approach to sdn control plane design. *SIGCOMM Comput. Commun. Rev.*, 47(1):15–26, jan 2017. URL: https://doi.org/10.1145/3041027.3041030, doi:10.1145/3041027.3041030.

[23] Pedro Gonzalez Ramirez, Jaime Lloret, Susan Cordero, and Luis Arboleda. Design and implementation of forces protocol. *Network Protocols and Algorithms*, 9:1, 06 2017. doi:10.5296/npa.v9i1-2.10943.

[24] Evangelos Haleplidis, Jamal Hadi Salim, Joel M. Halpern, Susan Hares, Kostas Pentikousis, Kentaro Ogawa, Weiming Wang, Spyros Denazis, and Odysseas Koufopavlou. Network programmability with forces. *IEEE Communications Surveys Tutorials*, 17(3):1423–1440, 2015. doi:10.1109/COMST.2015.2439033.

[25] R. J. Adams, Mike Dvorkin, Vijoy Pandey, Nik Weidenbacher, Michael P. Smith, Pankaj Kumar Garg, and Youcef Laribi. Opflex control protocol. 2016.

[26] Rajarevanth Narisetty, Levent Dane, Anatoliy Malishevskiy, Deniz Gurkan, Stuart Bailey, Sandhya Narayan, and Shivaram Mysore. Openflow configuration (ofconfig) protocol: Implementation for the of management plane. pages 66–67, 03 2013. doi:10.1109/GREE.2013.21.

[27] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *Computer Communication Review*, 38:69–74, 04 2008. doi:10.1145/1355734.1355746.

[28] https://opennetworking.org. Open networking, 2021. URL: https://opennetworking.org/software-defined-standards/specifications.

[29] Cosmin Caba and José Soler. Apis for qos configuration in software defined networks. In *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*, pages 1–5, 2015. doi:10.1109/NETSOFT.2015.7116157.

[30] B. Pfaff and Ed. B. Davie. The open vswitch database management protocol. Rfc, RFC Editor, December 2013. URL: https://datatracker.ietf.org/doc/html/rfc7047.txt.

[31] Jihyun Lee, Myung-Ki Shin, Youngrak Kim, Sung-Hyuk Park, and Sungil Lim. Common hardware reference platform for smart networks: Network function board. In *2015 17th International Conference on Advanced Communication Technology (ICACT)*, pages 441–443, 2015. doi:10.1109/ICACT.2015.7224834.

[32] Hung-Wei Chiu and Shie-Yuan Wang. Boosting the openflow control-plane message exchange performance of openvswitch. In *2015 IEEE International Conference on Communications (ICC)*, pages 5284–5289, 2015. doi:10.1109/ICC.2015.7249163.

[33] Pat Bosshart, Glen Gibb, Hun-Seok Kim, George Varghese, Nick McKeown, Martin Izzard, Fernando Mujica, and Mark Horowitz. Forwarding metamorphosis: Fast programmable match-action processing in hardware for sdn. *SIGCOMM Comput. Commun. Rev.*, 43(4):99–110,

aug 2013. URL: https://doi.org/10.1145/2534169.2486011, doi:10.1145/2534169.2486011.

[34] Arman Pashamokhtari, Hassan Habibi Gharakheili, and Vijay Sivaraman. Progressive monitoring of iot networks using sdn and cost-effective traffic signatures. In *2020 Workshop on Emerging Technologies for Security in IoT (ETSecIoT)*, pages 1–6, 2020. doi:10.1109/ETSecIoT50046.2020.00005.

[35] A. Bianco, R. Birke, L. Giraudo, and M. Palacin. Openflow switching: Data plane performance. In *2010 IEEE International Conference on Communications*, pages 1–5, 2010. doi:10.1109/ICC.2010.5502016.

[36] Vimalkumar Jeyakumar, Mohammad Alizadeh, Yilong Geng, Changhoon Kim, and David Mazières. Millions of little minions: Using packets for low latency network programming and visibility. *SIGCOMM Comput. Commun. Rev.*, 44(4):3–14, aug 2014. URL: https://doi.org/10.1145/2740070.2626292, doi:10.1145/2740070.2626292.

[37] Kirill Kogan, Sergey Nikolenko, Ori Rottenstreich, William Culhane, and Patrick Eugster. Sax-pac (scalable and expressive packet classification). In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, page 15–26, New York, NY, USA, 2014. Association for Computing Machinery. URL: https://doi.org/10.1145/2619239.2626294, doi:10.1145/2619239.2626294.

[38] Qingyun Zuo, Ming Chen, Ke Ding, and Bo Xu. On generality of the data plane and scalability of the control plane in software-defined networking. *China Communications*, 11(2):55–64, 2014. doi:10.1109/CC.2014.6821737.

[39] Wei Zhou, Li Li, Min Luo, and Wu Chou. Rest api design patterns for sdn northbound api. In *2014 28th International Conference on Advanced*

Information Networking and Applications Workshops, pages 358–365, 2014. doi:10.1109/WAINA.2014.153.

[40] Yongni Li, Song Hu, Wei Tao, and Baiyi Li. Research on the industrial network architecture of northbound interface. In *2017 Chinese Automation Congress (CAC)*, pages 6505–6509, 2017. doi:10.1109/CAC.2017.8243949.

[41] Sean W. Pritchard, Reza Malekian, Gerhard P. Hancke, and Adnan M. Abu-Mahfouz. Improving northbound interface communication in sdwsn. In *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, pages 8361–8366, 2017. doi:10.1109/IECON.2017.8217468.

[42] Carlos D. Cajas and Dmitry O. Budanov. Sdn applications and plugins in the opendaylight controller. In *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 9–13, 2020. doi:10.1109/EIConRus49466.2020.9039054.

[43] Jan Medved, Robert Varga, Anton Tkacik, and Ken Gray. Opendaylight: Towards a model-driven sdn controller architecture. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pages 1–6, 2014. doi:10.1109/WoWMoM.2014.6918985.

[44] Saleh Asadollahi, Bhargavi Goswami, and Mohammed Sameer. Ryu controller's scalability experiment on software defined networks. In *2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC)*, pages 1–5, 2018. doi:10.1109/ICCTAC.2018.8370397.

[45] Rahamatullah Khondoker, Adel Zaalouk, Ronald Marx, and Kpatcha Bayarou. Feature-based comparison and selection of software defined networking (sdn) controllers. In *2014 World Congress on Computer*

Applications and Information Systems (WCCAIS), pages 1–7, 2014. doi:10.1109/WCCAIS.2014.6916572.

[46] Faris Keti and Shavan Askar. Emulation of software defined networks using mininet in different simulation environments. In *2015 6th International Conference on Intelligent Systems, Modelling and Simulation*, pages 205–210, 2015. doi:10.1109/ISMS.2015.46.

[47] Rogério Leão Santos de Oliveira, Christiane Marie Schweitzer, Ailton Akira Shinoda, and Ligia Rodrigues Prete. Using mininet for emulation and prototyping software-defined networks. In *2014 IEEE Colombian Conference on Communications and Computing (COLCOM)*, pages 1–6, 2014. doi:10.1109/ColComCon.2014.6860404.

[48] Mark Berman, Chip Elliott, and Lawrence Landweber. Geni: Large-scale distributed infrastructure for networking and distributed systems research. In *2014 IEEE Fifth International Conference on Communications and Electronics (ICCE)*, pages 156–161, 2014. doi:10.1109/CCE.2014.6916696.

[49] Cristian Patachia-Sultanoiu, Ion Bogdan, George Suciu, Alexandru Vulpe, Oana Badita, and Bogdan Rusti. Advanced 5g architectures for future netapps and verticals. In *2021 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, pages 1–6, 2021. doi:10.1109/BlackSeaCom52164.2021.9527889.

[50] Konstantinos Trichias, Giada Landi, Erin Seder, Johann Marquez-Barja, Ronan Frizzell, Marius Iordache, and Panagiotis Demestichas. Vital-5g: Innovative network applications (netapps) support over 5g connectivity for the transport amp; logistics vertical. In *2021 Joint European Conference on Networks and Communications 6G Summit (EuCNC/6G Summit)*, pages 437–442, 2021. doi:10.1109/EuCNC/6GSummit51104.2021.9482437.

[51] Konstantinos C. Apostolakis, George Margetis, Constantine Stephanidis, Jean-Michel Duquerrois, Laurent Drouglazet, Arthur Lallet, Serge Delmas, Luís Cordeiro, André Gomes, Marta Amor, Almudena Díaz Zayas, Christos Verikoukis, Kostas Ramantas, and Ioannis Markopoulos. Cloud-native 5g infrastructure and network applications (netapps) for public protection and disaster relief: The 5g-epicentre project. In *2021 Joint European Conference on Networks and Communications 6G Summit (EuCNC/6G Summit)*, pages 235–240, 2021. doi : 10 . 1109 / EuCNC/6GSummit51104.2021.9482425.

[52] Zhaogang Shu, Jiafu Wan, Jiaxiang Lin, Shiyong Wang, Di Li, Seungmin Rho, and Changcai Yang. Traffic engineering in software-defined networking: Measurement and management. *IEEE Access*, 4:3246–3256, 2016. doi:10.1109/ACCESS.2016.2582748.

[53] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys Tutorials*, 18(1):236–262, 2016. doi:10.1109/COMST.2015.2477041.

[54] Yong Li and Min Chen. Software-defined network function virtualization: A survey. *IEEE Access*, 3:2542–2553, 2015. doi : 10 . 1109 / ACCESS . 2015.2499271.

[55] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376, 2015. doi : 10 . 1109 / COMST . 2015.2444095.

[56] G. Fortino, C. Savaglio, C.E. Palau, J.S. de Puga, M. Ghanza, M. Paprzycki, M. Montesinos, A. Liotta, and M. Llop. Towards multi-layer interoperability of heterogeneous iot platforms: The inter-iot approach. *Internet of Things*, 0(9783319612997):199–232, 2018. URL:

https://www.scopus.com/inward/record.uri?eid=2-s2.0-85027051038&doi=10.1007%2f978-3-319-61300-0_10&partnerID=40&md5=94c2df7b8c2c571253637931831a4f01, doi:10.1007/978-3-319-61300-0_10.

[57] Regel Gonzalez-Usach, Diana Yacchirema, Matilde Julian, and Carlos E. Palau. Interoperability in IoT, 2019. ISBN: 9781522574323 Pages: 149-173 Publisher: IGI Global. URL: www.igi-global.com/chapter/interoperability-in-iot/224268, doi:10.4018/978-1-5225-7432-3.ch009.

[58] Waseem Iqbal, Haider Abbas, Mahmoud Daneshmand, Bilal Rauf, and Yawar Abbas Bangash. An in-depth analysis of iot security requirements, challenges, and their countermeasures via software-defined security. *IEEE Internet of Things Journal*, 7(10):10250–10276, 2020. doi:10.1109/JIOT.2020.2997651.

[59] H. Mroue, A. Nasser, S. Hamrioui, B. Parrein, E. Motta-Cruz, and G. Rouyer. Mac layer-based evaluation of iot technologies: Lora, sigfox and nb-iot. In *2018 IEEE Middle East and North Africa Communications Conference (MENACOMM)*, pages 1–5, 2018. doi:10.1109/MENACOMM.2018.8371016.

[60] Wajid Rafique, Lianyong Qi, Ibrar Yaqoob, Muhammad Imran, Raihan Ur Rasool, and Wanchun Dou. Complementing iot services through software defined networking and edge computing: A comprehensive survey. *IEEE Communications Surveys Tutorials*, 22(3):1761–1804, 2020. doi:10.1109/COMST.2020.2997475.

[61] Neha Gupta, Sachin Sharma, Pradeep Kumar Juneja, and Umang Garg. Sdnfv 5g-iot: A framework for the next generation 5g enabled iot. In *2020 International Conference on Advances in Computing, Communication Materials (ICACCM)*, pages 289–294, 2020. doi:10.1109/ICACCM50413.2020.9213047.

[62] Raffaele Gravina, Carlos E. Palau, Marco Manso, Antonio Liotta, and Giancarlo Fortino, editors. *Integration, Interconnection, and Interoperability of IoT Systems*. Internet of Things. Springer International Publishing, 2018. URL: https://www.springer.com/gp/book/9783319612997, doi:10.1007/978-3-319-61300-0.

[63] Giancarlo Fortino, Carlos Palau, Antonio Guerrieri, Nora Cuppens, Frederic Cuppens, Hakima Chaouchi, and Alban Gabillon, editors. *Interoperability, Safety and Security in IoT - Third International Conference, InterIoT 2017, and Fourth International Conference, SaSeIot 2017, Proceedings*, volume 242 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*. Springer, 2018. URL: https://doi.org/10.1007/978-3-319-93797-7, doi:10.1007/978-3-319-93797-7.

[64] Diana Cecilia Yacchirema Vargas and Carlos Enrique Palau Salvador. Smart IoT Gateway For Heterogeneous Devices Interoperability. *IEEE Latin America Transactions*, 14(8):3900–3906, August 2016. URL: http://ieeexplore.ieee.org/document/7786378/, doi:10.1109/TLA.2016.7786378.

[65] Diana Yacchirema, Andreu Belsa, Carlos Palau, and Manuel Esteve. Onem2m based-interworking architecture for heterogeneous devices interoperability in iot. In *2018 IEEE Conference on Standards for Communications and Networking (CSCN)*, pages 1–6, Paris, France, October 2018. IEEE. URL: https://ieeexplore.ieee.org/document/8581740/, doi:10.1109/CSCN.2018.8581740.

[66] ITU-T Study Group 20. Gateway functional architecture for internet of things applications. Recommendation ITU-T Y.4418, International Telecommunication Union - Standardization Sector, Geneva, CH, 2018. URL: http://handle.itu.int/11.1002/1000/13640.

[67] ITU-T Study Group 20. Common requirements and capabilities of a gateway for internet of things applications. Recommendation ITU-T Y.4101/Y.2067, International Telecommunication Union - Standardization Sector, Geneva, CH, 2017. URL: http://handle.itu.int/11.1002/1000/13384.

[68] Gianluca Aloi, Giuseppe Caliciuri, Giancarlo Fortino, Raffaele Gravina, Pasquale Pace, Wilma Russo, and Claudio Savaglio. Enabling iot interoperability through opportunistic smartphone-based mobile gateways. *J. Netw. Comput. Appl.*, 81:74–84, 2017. URL: https://doi.org/10.1016/j.jnca.2016.10.013, doi:10.1016/j.jnca.2016.10.013.

[69] ITU-T Study Group 20. Internet of things requirements for support of edge computing. Recommendation ITU-T Y.4208, International Telecommunication Union - Standardization Sector, Geneva, CH, 2020. URL: http://handle.itu.int/11.1002/1000/14162.

[70] Xiang Sun and Nirwan Ansari. Edgeiot: Mobile edge computing for the internet of things. *IEEE Communications Magazine*, 54(12):22–29, 2016. doi:10.1109/MCOM.2016.1600492CM.

[71] Andrei Palade, Christian Cabrera, Fan Li, Gary White, M. A. Razzaque, and Siobhán Clarke. Middleware for internet of things: an evaluation in a small-scale IoT environment. *Journal of Reliable Intelligent Environments*, 4(1):3–23, April 2018. URL: https://doi.org/10.1007/s40860-018-0055-4, doi:10.1007/s40860-018-0055-4.

[72] Giancarlo Fortino, Claudio Savaglio, Giandomenico Spezzano, and MengChu Zhou. Internet of things as system of systems: A review of methodologies, frameworks, platforms, and tools. *IEEE Transactions on Systems, Man and Cybernetics: Systems*, 51(1):223–236, 2021. URL: https://doi.org/10.1109/TSMC.2020.3042898, doi:10.1109/TSMC.2020.3042898.

[73] Giancarlo Fortino and Paolo Trunfio, editors. *Internet of Things Based on Smart Objects, Technology, Middleware and Applications*. Springer, 2014. URL: https://doi.org/10.1007/978-3-319-00491-4, doi:10.1007/978-3-319-00491-4.

[74] Robert Kleinfeld, Stephan Steglich, Lukasz Radziwonowicz, and Charalampos Doukas. glue.things – a mashup platform for wiring the internet of things with the internet of services. 10 2014. doi:10.13140/2.1.3039.9049.

[75] Angel Lagares Lemos, Florian Daniel, and Boualem Benatallah. Web service composition: A survey of techniques and tools. *ACM Comput. Surv.*, 48(3), December 2015. URL: https://doi.org/10.1145/2831270, doi:10.1145/2831270.

[76] Giancarlo Fortino, Wilma Russo, Claudio Savaglio, Mirko Viroli, and MengChu Zhou. Modeling Opportunistic IoT Services in Open IoT Ecosystems. page 6.

[77] Giancarlo Fortino, Claudio Savaglio, and Mengchu Zhou. Toward opportunistic services for the industrial internet of things. In *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pages 825–830, Xi'an, August 2017. IEEE. URL: http://ieeexplore.ieee.org/document/8256205/, doi:10.1109/COASE.2017.8256205.

[78] Maria Ganzha, Marcin Paprzycki, Wieslaw Pawlowski, Pawel Szmeja, Katarzyna Wasielewska, Bartlomiej Solarz-Niesluchowski, and Jara Suarez. Towards high throughput semantic translation. In Giancarlo Fortino, Carlos E. Palau, Antonio Guerrieri, Nora Cuppens, Frederic Cuppens, Hakima Chaouchi, and Alban Gabillon, editors, *Interoperability, Safety and Security in IoT*, pages 67–74, Cham, 2018. Springer International Publishing.

[79] M. Ganzha, M. Paprzycki, W. Pawlowski, P. Szmeja, and K. Wasielewska. Semantic technologies for the iot - an inter-iot

perspective. In *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pages 271–276, April 2016. doi:10.1109/IoTDI.2015.22.

[80] Salvatore F. Pileggi., Carlos E. Palau., and Manuel Esteve. Building semantic sensor web: Knowledge and interoperability. In *Proceedings of the International Workshop on Semantic Sensor Web - Volume 1: SSW, (IC3K 2010)*, pages 15–22. INSTICC, SciTePress, 2010. doi : 10.5220/0003112000150022.

[81] Juliver de Jesus Gil Herrera and Juan Felipe Botero Vega. Network functions virtualization: A survey. *IEEE Latin America Transactions*, 14(2):983–997, 2016. doi:10.1109/TLA.2016.7437249.

[82] Margaret Chiosi, Don Clarke, Peter Cablelabs, Chris Donley, Lane Centurylink, and et al. Bugenhagen. Network functions virtualisation (nfv) network operator perspectives on industry progress, 10 2013. doi:10.13140/RG.2.1.4110.2883.

[83] David Breitgand, Vadim Eisenberg, Nir Naaman, Nir Rozenbaum, and Avi Weit. Toward true cloud native nfv mano. In *2021 12th International Conference on Network of the Future (NoF)*, pages 1–5, 2021. doi : 10.1109/NoF52522.2021.9609908.

[84] Francesco Tusa, Stuart Clayman, Dario Valocchi, and Alex Galis. Multi-domain orchestration for the deployment and management of services on a slice enabled nfvi. In *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–5, 2018. doi:10.1109/NFV-SDN.2018.8725769.

[85] Hung-Li Chen and Fuchun Joseph Lin. Scalable iot/m2m platforms based on kubernetes-enabled nfv mano architecture. In *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber,*

Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pages 1106–1111, 2019. doi : 10 . 1109 / iThings / GreenCom/CPSCom/SmartData.2019.00188.

[86] D3.1 - Methods for Interoperability and Integration v.1. INTER-IoT H2020 project, 2016. URL: https://inter-iot.eu/deliverables.

[87] Matija Cankar, Eneko Olivares Gorriti, Matevž Markovič, and Flavio Fuart. Fog and Cloud in the Transportation, Marine and eHealth Domains. In Dora B. Heras, Luc Bougé, Gabriele Mencagli, Emmanuel Jeannot, Rizos Sakellariou, Rosa M. Badia, Jorge G. Barbosa, Laura Ricci, Stephen L. Scott, Stefan Lankes, and Josef Weidendorfer, editors, *Euro-Par 2017: Parallel Processing Workshops*, volume 10659, pages 292–303. Springer International Publishing, Cham, 2018. Series Title: Lecture Notes in Computer Science. URL: http : / / link . springer . com / 10 . 1007 / 978 - 3 - 319 - 75178 - 8 _ 24, doi:10.1007/978-3-319-75178-8_24.

[88] Burak Görkemli, Sinan Tatlıcıoğlu, Mustafa Kömürcüoğlu, Melih A. Karaman, Özgür Karakaya, and Aydın Ulaş. Qos control and prioritization with sdn. In *2015 23nd Signal Processing and Communications Applications Conference (SIU)*, pages 2613–2616, 2015. doi:10.1109/SIU.2015.7130421.

[89] Seongbok Baik, Younggil Lim, Jongwoo Kim, and Youngwoo Lee. Adaptive flow monitoring in sdn architecture. In *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 468–470, 2015. doi:10.1109/APNOMS.2015.7275368.

[90] Md. Faizul Bari, Arup Raton Roy, Shihabur Rahman Chowdhury, Qi Zhang, Mohamed Faten Zhani, Reaz Ahmed, and Raouf Boutaba. Dynamic controller provisioning in software defined networks. In *Proceedings of the 9th International Conference on Network and Service*

Management (CNSM 2013), pages 18–25, 2013. doi:10.1109/CNSM.2013.6727805.

[91] Ruihan Wen, Gang Feng, Wei Tan, Rui Ni, Wei Cao, and Shuang Qin. Protocol stack mapping of software defined protocol for next generation mobile networks. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6, 2016. doi:10.1109/ICC.2016.7511230.

[92] Vaios Koumaras, Andreas Foteas, Angeliki Papaioannou, Marianna Kapari, Christos Sakkas, and Harilaos Koumaras. 5g performance testing of mobile chatbot applications. In *2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–6, 2018. doi:10.1109/CAMAD.2018.8515004.

[93] D3.2 - Methods for Interoperability and Integration v.2. INTER-IoT H2020 project, 2017. URL: https://inter-iot.eu/deliverables.

[94] D3.3 - Methods for Interoperability and Integration Final Version. INTER-IoT H2020 project, 2018. URL: https://inter-iot.eu/deliverables.

[95] D4.3 - Interoperable IoT Framework Model and Engine v1. INTER-IoT H2020 project, 2017. URL: https://inter-iot.eu/deliverables.

[96] D4.5 - Interoperable IoT Framework API and Tools v1. INTER-IoT H2020 project, 2017. URL: https://inter-iot.eu/deliverables.

[97] D4.6 - Interoperable IoT Framework API and Tools v2. INTER-IoT H2020 project, 2018. URL: https://inter-iot.eu/deliverables.

[98] Maria Fazio, Antonio Celesti, Fermín Galán Márquez, Alex Glikson, and Massimo Villari. Exploiting the fiware cloud platform to develop a remote patient monitoring system. In *2015 IEEE Symposium on*

Computers and Communication (ISCC), pages 264–270, 2015. doi : 10.1109/ISCC.2015.7405526.

[99] D7.2 - Technical Evaluation and Assessment report. INTER-IoT H2020 project, 2018. URL: https://inter-iot.eu/deliverables.

[100] D7.3 - Final evaluation report. INTER-IoT H2020 project, 2018. URL: https://inter-iot.eu/deliverables.

[101] Qiang Ye, Junling Li, Kaige Qu, Weihua Zhuang, Xuemin Sherman Shen, and Xu Li. End-to-end quality of service in 5g networks: Examining the effectiveness of a network slicing framework. *IEEE Vehicular Technology Magazine*, 13(2):65–74, 2018. doi:10.1109/MVT.2018.2809473.

[102] Petar Popovski, Kasper Fløe Trillingsgaard, Osvaldo Simeone, and Giuseppe Durisi. 5g wireless network slicing for embb, urllc, and mmtc: A communication-theoretic view. *IEEE Access*, 6:55765–55779, 2018. doi:10.1109/ACCESS.2018.2872781.

[103] Mehdi Bennis, Mérouane Debbah, and H. Vincent Poor. Ultrareliable and low-latency wireless communication: Tail, risk, and scale. *Proceedings of the IEEE*, 106(10):1834–1853, 2018. doi : 10 . 1109 / JPROC.2018.2867029.

[104] Maria Rita Palattella, Mischa Dohler, Alfredo Grieco, Gianluca Rizzo, Johan Torsner, Thomas Engel, and Latif Ladid. Internet of things in the 5g era: Enablers, architecture, and business models. *IEEE Journal on Selected Areas in Communications*, 34(3):510–527, 2016. doi:10.1109/JSAC.2016.2525418.

[105] Alcardo Alex Barakabitze, Arslan Ahmad, Rashid Mijumbi, and Andrew Hines. 5g network slicing using sdn and nfv: A survey of taxonomy, architectures and future challenges. *Computer Networks*, 167:106984, 2020. URL: https://www.sciencedirect.com/science/article/

pii/S1389128619304773, doi:https://doi.org/10.1016/j.comnet.2019.106984.

[106] F. Holik, U. Roedig, and N. Race. Lora-sdn: Providing wireless iot edge network functions via sdn. In *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, pages 1795–1800, 2020. doi:10.23919/MIPRO48935.2020.9245378.

[107] M. Gharbaoui, C. Contoli, G. Davoli, G. Cuffaro, B. Martini, F. Paganelli, W. Cerroni, P. Cappanera, and P. Castoldi. Demonstration of latency-aware and self-adaptive service chaining in 5g/sdn/nfv infrastructures. In *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pages 1–2, 2018. doi:10.1109/NFV-SDN.2018.8725645.

[108] Juha Petajajarvi, Konstantin Mikhaylov, Marko Pettissalo, Janne Janhunen, and Jari Iinatti. Performance of a low-power wide-area network based on lora technology: Doppler robustness, scalability, and coverage. *International Journal of Distributed Sensor Networks*, Vol. 13:1–16, 03 2017. doi:10.1177/1550147717699412.