Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación

PROPUESTA DE SOLUCIONES PARA LA INDUSTRIA 4.0 BASADAS EN REDES MÓVILES DE NUEVA GENERACIÓN SOPORTADAS POR ARQUITECTURAS NFV

TRABAJO FIN DE MÁSTER

Javier Andrés De Luca Batista

2019

Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación

Máster Universitario en
Ingeniería de Redes y Servicios Telemáticos

TRABAJO FIN DE MÁSTER

PROPUESTA DE SOLUCIONES PARA LA INDUSTRIA 4.0 BASADAS EN REDES MÓVILES DE NUEVA GENERACIÓN SOPORTADAS POR ARQUITECTURAS NFV

Autor
Javier Andrés De Luca Batista

Director
Luis Bellido Triana

Departamento de Ingeniería de Sistemas Telemáticos

2019

Resumen

Las redes industriales tradicionales, fueron pensadas para la automatización y la ejecución en tiempo real, tanto así que las primeras redes de automatización ni siquiera se consideraron como redes en absoluto, sino como buses serie. Con el tiempo esto ha ido cambiando y las redes industriales han ido migrando de los buses de serie a una integración con redes Ethernet tradicionales. En estos momentos la industria se encuentra en una etapa de transición de los procesos automatizados a la completa digitalización de los procesos de manufactura. Esta revolución fue bautizada con el nombre de Industria 4.0. Esta transición, se da de manera paralela a la implementación de la nueva generación de redes móviles, 5G, que con su arquitectura virtualizada provee de una gran flexibilidad y apunta a ser la red de redes. Esto presenta una gran oportunidad para las operadoras móviles, por lo cual es necesario identificar los escenarios en los cuales una solución 5G puede ayudar a una migración eficaz de los procesos de manufactura e inclusive a mejorarlos enormemente.

En este trabajo se analizará el estado del arte de la industria para hacer una evaluación de las tecnologías cableadas e inalámbricas existentes utilizadas en la fábrica conectada con el objetivo de comprender las brechas actuales de las soluciones existentes y proponer soluciones mediante el uso de tecnologías de virtualización y redes móviles de nueva generación.

Se realizará igualmente un estado del arte de redes nueva generación centrándonos en redes 5G y el concepto de Network Slicing para redes públicas.

Una vez realizado el estudio se propondrán casos de uso en línea con los requerimientos de la Industria 4.0 y los escenarios analizados. Por último, se propone un escenario de Servicios de Red, haciendo uso diferentes tecnologías de virtualización.

Palabras clave: 5G, Network Slicing, Virtualización, Industria 4.0, NFV, OSM, Docker.

Abstract

Traditional industrial networks were designed for automation and execution in real time, so much so that the first automation networks were not even considered as networks at all, but as serial buses. Over time this has been changing and industrial networks have been migrating from the serial buses to an integration with traditional Ethernet networks. Currently the industry is in a stage of transition from automated processes to the complete digitalization of manufacturing processes. This revolution was named Industry 4.0. This transition occurs in parallel with the implementation of the new generation of mobile networks, 5G, which with its virtualized architecture provides great flexibility and aims to be the network of networks. This presents a great opportunity for mobile operators, for which it is necessary to identify the scenarios in which a 5G solution can help execute an efficient migration of manufacturing processes and even greatly improve them.

This paper will analyze the state of the art of the industry to assess existing wired and wireless technologies used in the connected factory in order to understand the current gaps in existing solutions and propose solutions through the use of virtualization technologies and new generation mobile networks.

It will also be a state of the art of new generation networks focusing on 5G networks and the concept of Network Slicing for public networks.

Once the study is done, use cases will be proposed in line with the requirements of Industry 4.0 and the scenarios analyzed. Finally, a scenario of Network Services is proposed, making use of different virtualization technologies.

Key words: 5G, Network Slicing, Virtualization, Industry 4.0, NFV, OSM, Docker.

Agradecimientos

En estas líneas quiero agradecer a todas las personas e instituciones involucradas en la realización de este trabajo de fin de máster.

En primer lugar, a Huawei España, que mediante la cátedra Huawei-UPM, ha hecho posible la realización del trabajo presentado en esta memoria, por la ayuda económica brindada. Igualmente quiero agradecer al director de la cátedra, el Dr. José Manuel Riera, quien ha sido el enlace entre Huawei y mi persona en todas las comunicaciones necesarias en los últimos meses.

Agradezco a mi tutor, el Dr. Luis Bellido Triana, por haberme guiado en la realización de este trabajo y haber puesto a disposición tu tiempo y conocimiento cuando lo necesitaba.

Al Dr. David Fernández Cambronero, por haberme ayudado desde el inicio de esta investigación, por tu disponibilidad para conversar siempre que tuviera alguna consulta.

A Andrés Cárdenas Cordova, por haber compartido conmigo tus conocimientos y retroalimentación durante la investigación que estuvimos realizando así como tu experiencia previa en éste máster.

Al Dr. Carlos Mariano Lentisco Sánchez, por tus aportes brindados en las reuniones del grupo de investigación que mantuvimos durante la realización de este trabajo.

A Manuel Romero Ortega, por los ánimos que me has brindado para terminar este trabajo.

A mi padre, por tu apoyo incondicional en mi decisión de realizar este máster y por estar siempre a disposición para cualquier duda que tuviera.

A mi madre, por los ánimos que siempre me has dado y por estar siempre pendiente de mí.

Índice general

Resumen i
Abstract ii
Agradecimientos iii
Índice general iv
Índice de figuras vii
Índice de tablas ix
Siglas x
1 Introducción 1
1.1 Objetivo 1
1.1.1 Objetivos Específicos 1
1.2 Metodología 1
1.3 Estructura de la memoria 2
2 Estado del Arte de las Comunicaciones Industriales 3
2.1 Industrial Ethernet 3
2.1.1 Ethernet/IP 4
2.1.2 PROFINET 5
2.1.3 EtherCAT 6
2.2 Redes Inalámbricas Industriales 7
3 Estado del Arte en Redes de Nueva Generación 11
3.1 Máquinas Virtuales y Contenedores 11
3.2 Redes Definidas por Software (SDN) 12
3.3 Virtualización de Funciones de Red (NFV) 12
3.4 Edge Computing 14
3.4.1 Framework 15
3.5 5G 16
3.5.1 Ultra-Reliable and Low Latency communications (URLLC): 16
3.5.2 Enhanced Mobile BroadBand (eMBB): 17

3.5.3 Massive Machine Type Communication (mMTC): 18
3.6 Network Slicing 21
4 Industria 4.0 22
4.1 Tecnologías Habilitantes 23
4.1.1 Robots Autónomos 24
4.1.2 Internet de las Cosas (IoT) 24
4.1.3 Cloud Computing 28
4.1.4 Big Data 29
4.1.5 Realidad Aumentada/Realidad Virtual 30
4.1.6 Impresión 3D/ Manufactura Aditiva 31
4.2 Adopción de la Industria 4.0 32
5 Casos de Uso en la Industria 4.0 34
5.1.1 Realidad Aumentada 34
5.1.2 Aceleración de Video Inteligente 35
5.1.3 IoT Gateway 35
5.1.4 Automóviles Conectados 36
5.1.5 Realidad Aumentada 37
5.1.6 Redes 5G Privadas/Semi-Privadas 38
5.1.7 Network Slicing para la Industria 38
6 Escenario propuesto 41
6.1 Plataforma FIWARE 42
6.1.1 Prueba de Concepto de la plataforma 44
6.2 Herramientas Utilizadas 55
6.2.1 Docker 55
6.2.2 Open Source MANO (OSM) 57
6.2.3 Vim-emu 59
6.3 Diseño del Escenario 60
6.4 Prueba de Concepto 61
6.4.1 Generación de Imágenes 61
6.4.2 Orquestación en OSM 62
7 Conclusiones 71

7.1 Líneas Futuras 72
Anexo I: Pruebas con OSM 5.0 73
Anexo II: Pruebas con vim-emu 85
Bibliografía 86

Índice de figuras

Figura 1. Cuota de mercado de las redes industriales en 2018. 3
Figura 2. Vista OSI de DeviceNet y Ethernet/IP [3] 5
Figura 3. Canales de Comunicación PROFINET 5
Figura 4. Modelo de Referencia OSI EtherCAT 6
Figura 5. Redes según su alcance. 7
Figura 6. Rango de las tecnologías inalámbricas 10
Figura 7. Abstracción de las máquinas virtuales. 11
Figura 8. Abstracción de los contenedores. 12
Figura 9. Arquitectura NFV ETSI 13
Figura 10. Subdivisón Arquitectura NFV 13
Figura 11. Arquitectura MEC ETSI 15
Figura 12. Concepto de Network Slice 21
Figura 13. Evolución de la Industria [9] 22
Figura 14. Integración para la Industria 4.0 [10] 22
Figura 15. Análisis bibliográfico sobre la Industria 4.0 23
Figura 16. Tecnologías más relevantes según la IEEE 24
Figura 17. Tipos de conexiones IoT [15] 26
Figura 18. Hospedar, construir y consumir. 28
Figura 19. Control Virtualizado IoT 33
Figura 20. Despliegue MEC en un escenario de realidad aumentada 34
Figura 21. Despliegue MEC en un escenario de aceleración de video inteligente. 35
Figura 22. Despliegue MEC en un escenario IoT 35
Figura 23. Despliegue MEC en un escenario de Automóviles Conectados 36
Figura 24. Gafas de realidad aumentada. 37
Figura 25. Solución REMOTE AR 38
Figura 26. Industria Conectada Visión 5G-ACIA [27] 39
Figura 27. Network Slice para monitorización de la producción 39
Figura 28. Network Slice para manufactura inteligente. 40
Figura 29. Network Slice para mantenimiento remoto. 40
Figura 30. Despliegue de elementos 41
Figura 31. Información de Contexto 42
Figura 32. Bloques de la plataforma FIWARE 43
Figura 33. Arquitectura del escenario de prueba de sensores 45
Figura 34. Puesta en marcha del escenario 46
Figura 35. Monitor IoT 46
Figura 36. Petición curl 46
Figura 37. Cambio de estado y confirmación de tráfico 47
Figura 38. Detener servicios 47

Figura 39. Arquitectura general para Agente IoT 48
Figura 40. Arquitectura Agente IoT MQTT 48
Figura 41. Inicio de escenario MQTT 49
Figura 42. Monitor escenario MQTT 49
Figura 43. Visualización de mensajes MQTT 49
Figura 44. Envío de mensajes MQTT 49
Figura 45. Log broker MQTT 50
Figura 46. Estado agente IoT 50
Figura 47. Grupo de servicio 50
Figura 48. Aprovisionamiento de sensores 51
Figura 49. Recepción de mensajes MQTT 51
Figura 50. Registro Sensor MQTT 52
Figura 51. Envío de comandos mediante Orion usando MQTT 52
Figura 52. Mensajes MQTT en monitor web. 52
Figura 53. Listado de dispositivos registrados 53
Figura 54. Fin prueba MQTT. 53
Figura 55. Intercambio de mensajes FastRTPS. 54
Figura 56. Arquitectura OSM 57
Figura 57. Nivel de emulación de vim-emu [39] 59
Figura 58. Comunicación entre elementos 60
Figura 59. Comunicación entre contenedores 61
Figura 60. Diagramas VNFD 62
Figura 61. Validación YAML VNFD 65
Figura 62. Topología VNF 65
Figura 63. Diagrama NSD 66
Figura 64. Topologías del Servicio de Red 67
Figura 65. Orquestación del servicio de red. 67
Figura 66. Servicio de red orquestado 68
Figura 67. Lista de Datacenters. 68
Figura 68. Lista de nodos de computación 68
Figura 69. Utilización de recursos de contenedores 69
Figura 70. Líne de comandos de la VNF 69
Figura 71. Nueva instancia de Network Slice 70
Figura 72. Respuesta sobre incidencias con vim-emu 70

Índice de tablas

Tabla 1. Comparativa entre protocolos de Industrial Ethernet. 6
Tabla 2. Aplicaciones de URLLC 17
Tabla 3. Casos de uso 5G según los requerimientos IMT-2020..... 18
Tabla 4. Diseño VNF 63

Siglas

3GPP	3rd Generation Partnership Project
5G NR	5G New Radio
AMQP	Advanced Message Queuing Protocol
AR	Augmented Reality
BLE	Bluetooth Low Energy
CBIoT	Cellular Based IoT
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
CSMA/CA	Carrier-sense multiple access with collision avoidance
CSMA/CD	Carrier-sense multiple access with collision detection
D2D	Device to Device
DB	Database
DDS	Data Distribution Service
eMBB	Enhanced Mobile BroadBand
ERP	Enterprise Resource Planning
ETSI	European Telecommunications Standards Institute
FHSS	Frequency-hopping spread spectrum
GPS	Global Positioning System
HART	Highway Addressable Remote Transducer Protocol
HMI	Human Machine Interface
HNF	Hybrid Network Function
HVAC	Heating, ventilation, and air conditioning
IaaS	Infrastructure as a Service
ICRAN	Industrial Cellular Radio Access Networking
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IMT	International Mobile Telecommunications
IoT	Internet of Things
IP	Internet Protocol
IRT	Isochronous Real Time
ITU	International Telecommunication Union
LAN	Local Area Network
LoRa	Long Range
LPWAN	Low Power Wide Area Network
LTE	Long Term Evolution
LXC	Linux Containers
M2M	Machine to Machine
MAC	Medium Access Control
MANO	Management and Network Orchestration
MEC	Multiaccess Edge Computing
MIMO	Multiple Input-Multiple Output
mMTC	Massive Machine Type Communication
MQTT	Message Queuing Telemetry Transport
NB-IoT	Narrow Band IoT
NFV	Network Function Virtualization
NGMN	Next Generation Mobile Networks

NGSI	Next Generation Service Interface
NOMA	Non-Orthogonal Multiple Access
NRT	Non-Real Time
NR-U	New Radio Unlicensed
NSaaS	Network Slicing as a Service
NSD	Network Service Descriptor
NST	Network Slice Template
OPC	Open Platform Communications
OSM	Open Source Management and Orchestration
PaaS	Platform as a Service
PHY	Physical Layer
PLMN	Public Land Mobile Networks
PNF	Physical Network Function
POV	Point of View
PROFINET	Process Field Net
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
REST	Representational State Transfer
RFC	Remote Function Call
RFID	Radio-frequency identification
RT	Real Time
RTPS	Real Time Publish Subscribe protocol
SaaS	Software as a Service
SDN	Software-defined networking
SIM	subscriber identification module
TCP	Transmission Control Protocol
TDMA	Time-division multiple access
UDP	User Datagram Protocol
URL	Uniform Resource Locator
URLLC	Ultra-Reliable and Low Latency communications
UWB	Ultra-Wide Band
V2X	Vehicle-to-everything
VCA	VNF Configuration and Abstraction
VIM	Virtualized Infrastructure Manager
VM	Virtual Machine
VNF	Virtual Network Function
VNFD	VNF Descriptor
VR	Virtual Reality
WIM	WAN Infrastructure Manager
WLAN	Wireless Local Area Network
WMAN	Wireless Metropolitan Area Network
WSN	Wireless Sensor Networks
WWAN	Wireless Wide Area Network
XMPP	eXtensible Messaging and Presence Protocol
XRCE	eXtremely Resource Constrained Environments
YAML	YAML Ain't Markup Language
YANG	Yet Another Next Generation

1 Introducción

La Industria 4.0 es una revolución industrial que quiere cambiar el mundo como lo conocemos, una vez más, mediante la digitalización de los procesos tradicionales dentro de la industria.

La primera revolución industrial marca un punto de inflexión en nuestra historia, modificando todos los aspectos de la vida cotidiana. Con la aparición del primer telar mecánico se pasa de los procesos de manufactura manuales, a los procesos de mecánicos. Con esto cambió el mundo como lo conocíamos y tanto ésta revolución como las siguientes revoluciones han permitido que la tecnología avance de manera exponencial. Con estos avances, cada vez tenemos maneras más eficientes de realizar las tareas industriales y en nuestra época la siguiente revolución pasa por la digitalización.

La introducción de tecnologías como el Internet de las Cosas, Big Data y la Inteligencia Artificial ofrece grandes oportunidades en la industria, pero para lograr estos cambios, es necesario modificar la infraestructura industrial, ya que ésta no fue diseñada para estos escenarios. Las redes industriales están pensadas para trabajar en tiempo real y agregar la carga de información requerida para gestionar soluciones 4.0 como Inteligencia Artificial o Big Data, requiere de algunos cambios. Para esto es necesario la introducción de nuevas soluciones a nivel de redes, al no ser las redes industriales suficientes, creando una gran oportunidad para las operadoras de ofertar soluciones integrales, uniendo las necesidades de la industria a la hora de implementar soluciones 4.0 y las ventajas de las redes de nueva generación. En esta investigación analizaremos la situación actual de las redes industriales, las posibilidades que ofrecen las redes de nueva generación y cómo éstas pueden ayudar a resolver los problemas de adopción de las soluciones 4.0. Por último, haciendo uso los conceptos aprendidos, se propone un escenario didáctico de soluciones 4.0 desplegado en una arquitectura de nueva generación.

1.1 Objetivo

El objetivo principal de esta investigación es analizar el estado actual de la industria e identificar casos de uso de las redes móviles de nueva generación en la nueva Industria 4.0

1.1.1 Objetivos Específicos

- Realizar un estado del arte de las tecnologías cableadas e inalámbricas utilizadas en la industria actualmente.
- Identificar problemas que se presentan con las tecnologías Industriales.
- Identificar áreas de mejora a los problemas encontrados mediante el uso de redes móviles de nueva generación.
- Proponer un caso de uso práctico que haga uso de las tecnologías revisadas en la investigación.

1.2 Metodología

En este trabajo, realizamos una investigación que inicia con un estado del arte de las tecnologías involucradas en la Industria y las redes móviles, revisando información proveniente de distintas fuentes haciendo uso de agregadores de publicaciones científicas recientes como IEEE Xplore, Researchgate y ScienceDirect. Igualmente, para tener una

mejor perspectiva, accedimos a información provista tanto como por la Unión Europea, como por distintos gobiernos de las comunidades autónomas en España, para comprender la situación actual y los requerimientos que han sido identificados para la adopción de la Industria 4.0.
Por último, hemos ido en búsqueda del conocimiento de la comunidad open source a la hora de implementar el escenario propuesto, para aprender a utilizar estas herramientas y poder resolver los problemas que surgen en el camino a la hora de realizar las pruebas de concepto.

1.3 Estructura de la memoria

La memoria está dividida en 6 capítulos y 2 anexos detallados de la siguiente manera:

En el capítulo 1 se detallan los temas a tratar en la investigación, los objetivos y la metodología seguida.

Los capítulos 2 y 3 son estados del Arte de las redes industriales y de las redes móviles de nueva generación, en los cuales vemos a detalle las tecnologías más relevantes tanto en redes industriales como en despliegues de redes virtualizadas.

El capítulo 4 es una introducción al concepto de Industria 4.0, como se ha llegado a este concepto, las tecnologías necesarias para esta revolución y los inconvenientes a corto plazo para lograr su adopción.

En el capítulo 5, detallamos propuestas que hemos identificado durante la investigación que pueden ayudar a resolver los problemas de la adopción de la Industria 4.0 mediante el uso de redes móviles de 5ta generación y conceptos de virtualización.

En el capítulo 6, proponemos un escenario de pruebas, haciendo uso de propuestas que consideramos interesantes, de la información revisada en la memoria y de herramientas de virtualización. Además, hacemos introducción a la plataforma de gestión de datos FIWARE, realizamos algunas pruebas de concepto y desplegamos una solución de prueba con la herramienta OSM.

En el capítulo 7, contiene las conclusiones obtenidas en esta investigación y líneas futuras a seguir.

En los anexos, describimos con mayor detalle el uso de dos de la herramientas utilizadas en el capítulo 6: OSM y vim-emu.

2 Estado del Arte de las Comunicaciones Industriales

En entornos industriales las comunicaciones están enfocadas a sistemas de automatización y a un funcionamiento distribuido, por lo cual se requiere de comunicación en tiempo real, a diferencia de las redes empresariales tradicionales. Sumado a esto deben resistir un ambiente hostil en el cual tiende a haber ruido electromagnético y condiciones ambientales adversas como lo son altas/bajas temperaturas, humedad, vibración entre otras.

Estas redes inicialmente basadas en comunicaciones de bus de campo han ido evolucionando hasta una variación de las redes ethernet tradicionales, conocida como Industrial Ethernet. Igualmente podemos encontrar redes inalámbricas dentro de las comunicaciones industriales, pero en menor medida comparada con las redes de bus de campo y las redes Industrial Ethernet.

Según un estudio realizado por la empresa HMS Networks [1], en 2018 Industrial Ethernet superó por primera vez a los buses de campo en términos de nuevos nodos instalados, contando con el 52% de las nuevas instalaciones, a la vez que los nodos de bus se encuentran en declive, totalizando un 42% de nuevas instalaciones realizadas, en contraste con 2017 y 2016, siendo 48% y 58% respectivamente.

Cabe destacar, que, a pesar de tener un porcentaje minoritario en las redes industriales, las tecnologías inalámbricas, son las que mayor crecimiento han tenido, con un 32% con respecto al año 2017.

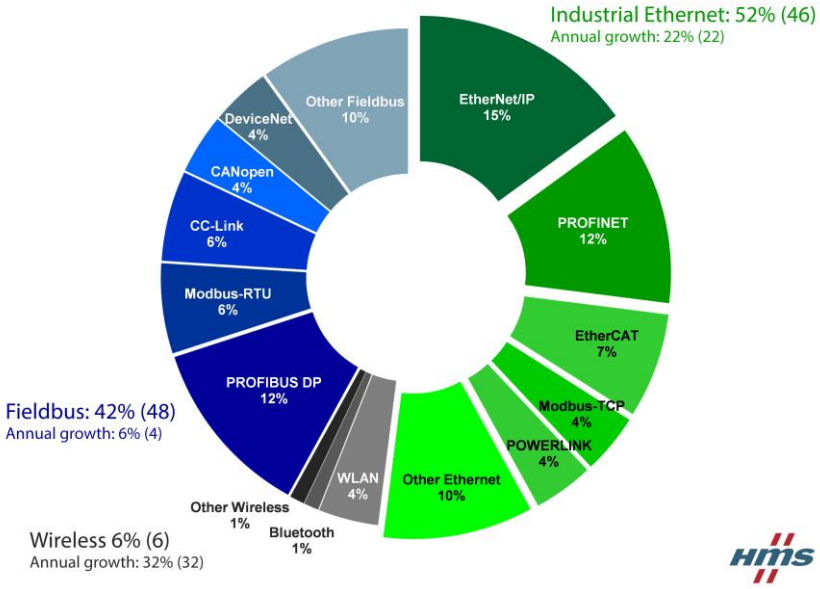


Figura 1. Cuota de mercado de las redes industriales en 2018.

2.1 Industrial Ethernet

Industrial Ethernet es el termino acuñado al uso de protocolos Ethernet en ambientes industriales mediante protocolos que proveen determinismo y respuesta en tiempo real. Los protocolos Ethernet tradicionales usan métodos de acceso al medio con detección, y, más recientemente, prevención de colisiones (CSMA/CA, CSMA/CD), lo cual agrega retardo a las comunicaciones. Esto es tolerable en redes Ethernet empresariales, pero no lo es ambientes industriales, ya que se requiere comunicación en tiempo real para los procesos de control. Además las redes industriales inicialmente se desarrollaban sobre

interfaces creadas por diferentes compañias y sin estandarizar, fragmentando el mercado.
Industrial Ethernet usa una capa de acceso al medio modificada, para lograr muy baja latencia y respuestas determínisticas [2]. A diferencia de los buses de campo, permite topologías de red más flexibles, mayor facilidad de expansión de la red, la utilización de múltiples protocolos de manera simultánea, alcanzar mayores velocidades y la integración con redes inalámbricas.
Existen diferentes propuestas de protocolos Industrial Ethernet, casi todos son algún tipo de adaptación de protocolos específicos de bus de campo a tecnologías Ethernet.

Estas adaptaciones pueden categorizarse en 3 grupos:

1. Standard Software/Standard Ethernet (S/S): Adaptación basada en TCP/IP y UDP/IP. Los mecanismos para garantizar respuesta en tiempo real son embebidos en la capa superior. Puede tener problemas de desempeño.
2. Open Software/Standard Ethernet (O/S): Esta adaptación sigue estándares, pero no garantiza la compatibilidad de que los dispositivos funcionen con otros fabricantes. Cada empresa sigue los estándares de la forma en que deseen y según sus necesidades. Este tipo de adaptación permite a las empresas no pagar regalias por los estándares, siempre y cuando realicen pruebas de compatibilidad con otros dispositivos, para verificar si pueden interactuar, sin la necesidad de garantizar la compatibilidad. Suelen necesitarse controladores propietarios para prevenir problemas de latencia.
3. Open Software/Modified Ethernet (O/M): Este grupo de adaptación usa nuevos protocolos abiertos, pero modifica el estándar Ethernet para lograr respuestas en tiempo real.

Hay 3 claros protocolos que acaparan las nuevas instalaciones industriales: Ehernet/IP, PROFINET y EtherCAT:

2.1.1 Ethernet/IP

Este protocolo es una adaptación S/S que extiende la familia de protocolos de automatizacion Common Industrial Protocol (CIP), utilizados en el protocolo de bus de campo DeviceNET, a las tecnologías Ethernet, para poder operar en tiempo real.
EtherNet/IP puede usar UDP, por lo que los mensajes deben pasar por el stack UDP/IP añadiendo algo de retraso en las comunicaciones, en comparación a otros protocolos utilizados en Industrial Ethernet.

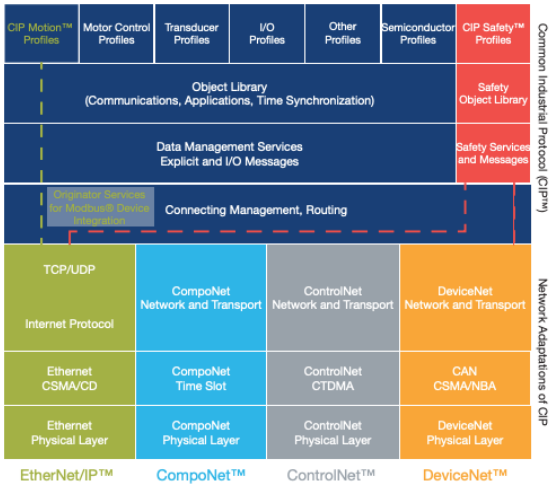


Figura 2. Vista OSI de DeviceNet y Ethernet/IP [3]

2.1.2 PROFINET

Es una adaptación O/M que extiende el protocolo de bus de campo PROFIBUS a las tecnologías Ethernet. Soporta tráfico estocástico y determinístico, por lo cual puede considerarse una tecnología intermedia entre los protocolos de bus de campo y los protocolos Ethernet tradicionales. Implementa calidad de servicio (QoS) para agregar determinismo. Se suelen utilizar protocolos TCP, UDP e IP para la transmisión de datos, pero no es suficiente para todos los casos de uso.

PROFINET usa tres canales para la comunicación, mostrados en la Figura 3, con diferentes clases de desempeño, según el uso [4]:

1. NRT (Non Real Time): PROFINET usa TCP/IP y UDP/IP para la transmisión de paquetes que no requieren una respuesta en tiempo real.
2. RT (Real Time): para un intercambio de datos optimizado y de alto desempeño, PROFINET descarta las capas de Red, Transporte y Session, ayudando a reducir latencia y jitter en la transmisión de datos.
3. IRT (Isochronous Real Time) para comunicación sincronizada: Para satisfacer respuestas inmediatas (<1ms) en aplicaciones de control de movimiento, PROFINET, descarta las mismas capas que en RT, pero utiliza una versión modificada del Standard 802.3 en la capa física.

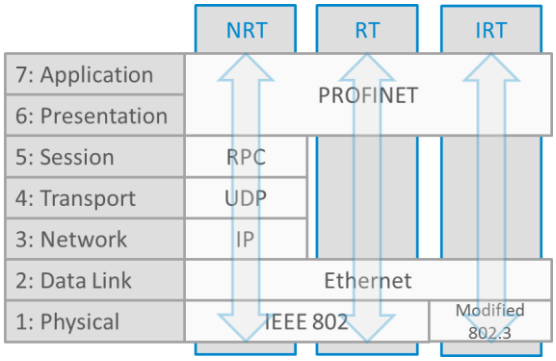


Figura 3. Canales de Comunicación PROFINET

2.1.3 EtherCAT

Es una adaptación O/M que ofrece una comunicación eficiente y en tiempo real.
Como desventajas tenemos que tiene un numero limitado de dispositivos soportados a la vez y que no utiliza el estándar TCP/IP, como podemos ver la Figura 4, para la transmisión de paquetes.

ISO/OSI Layer		EtherCAT	
Host layers	7. Application	HTTP*, FTP*	• Cyclic Data Exchange • Mailbox Acyclic Data Access
	6. Presentation	—	—
	5. Session	—	—
	4. Transport	TCP*	—
Media layers	3. Network	IP*	—
	2. Data link	• Mailbox/Buffer Handling • Process Data Mapping • Extreme Fast Auto-Forwarder	
		Ethernet MAC	
	1. Physical	100BASE-TX, 100BASE-FX	

Figura 4. Modelo de Referencia OSI EtherCAT

En general, estos protocolos tienen tiempos de respuesta similares, pero varían en cuanto a los tiempos de jitter, todos los protocolos Industrial Ethernet tienen una velocidad de datos máxima de 100Mbit/s, como podemos ver en la Tabla 1.
Se puede decir que a nivel de tiempos de respuesta y en general, EtherCAT es el gran vencedor: se distingue por tener una mejor relacion precio/rendimiento. Ofrece el mejor rendimiento, menos fluctuaciones, y el menor costo por incorporar el hardware propietario requerido para entregar comunicación en tiempo real, pero este protcolo no es compatible con el estándar TCP/IP, lo cual dificulta su integración con otros tipos de redes y esto se ha visto reflejado en las nuevas instalaciones industriales, en las cuales pasó de dominar el mercado Industrial Ethernet a tener un 7% de las nuevas instalaciones, según el último reporte de HMS.

Protocolo	Tiempo de Respuesta	Jitter	Velocidad
Ethernet/IP	1 ms	<1ms	100Mbit/s
Profinet IRT	1 ms	<1ms	100Mbit/s
EtherCAT	0.1 ms	<0.1ms	100Mbit/s

Tabla 1. Comparativa entre protocolos de Industrial Ethernet.

A pesar de que las tecnologías de bus de campo se encuentran dominan las instalaciones existentes, la adopción de Industrial Ethernet es cada vez mayor, siendo notorio en el número de nuevas instalaciones en los últimos años, ya que son una tecnología más flexible de implementar, mientras proveen dos de las grandes características requeridas en las redes industriales: desempeño en tiempo real y confianza.

2.2 Redes Inalámbricas Industriales

Las redes inalámbricas en la industria son atractivas debido a su flexibilidad, facilidad de uso, bajo coste y facilidad de despliegue. Estas son utilizadas tanto para despliegue de redes corporativas dentro de la Industria como para el despliegue de grandes redes de sensores y dispositivos industriales. A nivel empresarial, se siguen utilizando tecnologías tradicionales como WiFi o LTE sin mayores problemas, pero a nivel industrial, la situación es diferente, ya que estas redes de sensores son utilizadas para controlar y monitorizar diferentes tareas dentro de una fábrica, por lo que los requerimientos de alcance, energía y comunicación en tiempo real cambian drásticamente y no se logran cumplir con las tecnologías en las redes empresariales.

Tipos de Redes Inalámbricas según su alcance

Las redes inalámbricas utilizan ondas de radio y/o microondas para mantener las comunicaciones entre los dispositivos Al referirse al rango de cobertura de las redes inalámbricas en entornos industriales, tenemos hasta 5 categorías diferenets como podemos ver en la Figura 5:

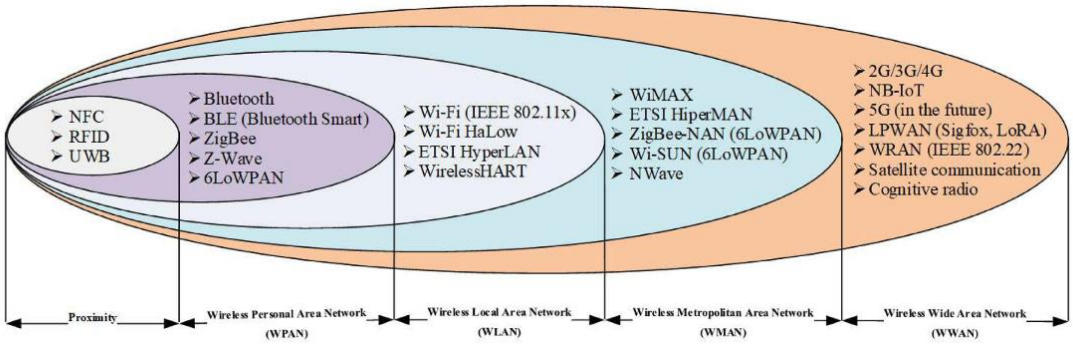


Figura 5. Redes según su alcance.

- De Proximidad:
  Un conjunto de pequeños dispositivos que realizan tareas muy simples como la detección, la computación, el almacenamiento y la activación. Protocolos: NFC, RFID, UWB.
- WPAN (Wireless Personal Area Network):
  Una red para conectar dispositivos dentro de un radio de aproximadamente una o dos habitaciones. Protocolos: Bluetooth, ZigBee, BLE, 6LoWPAN
- WLAN (Wireless Local Area Network):
  Una red que cubre el área de un edificio. Protocolos: IEEE 802.11 (WiFi), WirelesHART.
- WMAN (Wireless Metropolitan Area Network):
  Red para cubrir ciertas áreas metropolitanas, utilizando tecnologías de tranmisión por microondas.
  Protocolos: WiMax.
- WWAN (Wireless Wide Area Network):

Red que se despliega a gran escala geográfica y une pequeñas redes como WLAN o WMAN. Protocolos: Celular (LTE, 5G), LPWAN.

Descripción de los protocolos más comunes:

- Bluetooth:
  El estándar Bluetooth proporciona velocidades de datos de 800 Kb/s para la operación FHSS en la banda de 2.4 GHz. Debido a la potencia reducida del transmisor, las distancias son muy cortas. En general, los dispositivos Bluetooth se utilizan para transferir datos entre periféricos de computadoras personales y teléfonos móviles/celulares. Si bien algunos productos industriales Bluetooth están disponibles, este estándar no se usa comúnmente en la industria.

- ZigBee 802.15:
  ZigBee es un estándar muy reciente diseñado principalmente para sensores inalámbricos alimentados por baterías. Los dispositivos ZigBee pueden usar bandas ISM de 868/900 MHz o de 2,4 GHz y tienen una baja potencia de transmisor en dispositivos inalámbricos para adaptarse a baterías pequeñas e integradas.Una característica importante de este estándar es el despliegue en modo de malla inalámbrica mediante la cual los dispositivos pasan información de una unidad a otra. Los dispositivos ZigBee tienen gran popularidad en los sistemas de administración de edificios y calefacción, ventilación y aire acondicionado (HVAC) y se espera que se utilicen cada vez más en aplicaciones industriales.

- 6LoWPAN (IPv6 over low power wireless personal area networks):
  6LoWPAN es una especificación que define la transmisión de paquetes IPv6 en redes IEEE 802.15.4. Es descrito por el IETF en RFC49197 y RFC4944. La definición de 6LoWPAN se puede usar como una especificación independiente para redes de sensores inalámbricos(WSN).

- WirelessHART:
  WirelessHART fue desarrollado por HART Communications Foundation (HCF) y se lanzó en 2007. Es una extensión inalámbrica del protocolo HART (IEC 61158) que ya es popular. Utiliza tecnologías para radios 802.15.4, canales de salto, espectro de difusión y redes de malla. Permite la transmisión inalámbrica de mensajes HART y se basa en las capas PHY y MAC IEEE 802.15.4, aunque el MAC se ha modificado para permitir el salto de frecuencia. TDMA con salto de frecuencia se utiliza como un método de acceso al canal, y con una topología de

red de malla completa, WirelessHART ofrece comunicación multihop auto-configurable.

- ISA100.11a:
  Estándar desarrollado por una técnica sociedad de ingenieros, sin fines de lucro, llamada The International Society of Automation (ISA). Es un protocolo independiente y está diseñado para su implementación en toda la planta. Utiliza radios 802.15.4, canales de salto, amplio espectro, tecnologías de redes de malla y capacidades de interfaz para host multiprotocolo. En la especificación ISA100.11a, se utiliza TDMA con salto de frecuencia como mecanismo de acceso al canal. Es compatible con dispositivos de enrutamiento y no enrutamiento, por lo que las topologías de red pueden ser en estrella, en estrella o en malla completa, según la configuración y las capacidades de los dispositivos en la red. También utiliza el direccionamiento IPv6 al implementar 6LoWPAN en la capa de red. Una red ISA100.11a puede transportar múltiples protocolos de bus de campo, como Fieldbus y Profibus, así como HART. También hay soporte integrado para el tráfico y enrutamiento IPv6 en la capa de red.

- WiFi 802.11:
  El estándar 802.11 especifica los requisitos en las bandas ISM de 2,4 GHz y 5 MHz para dispositivos Ethernet inalámbricos que utilizan DSSS. Esta norma es conocida popularmente como WiFi. Desde finales de la década de 1990, los dispositivos WiFi han estado en el mercado, con varios estándares 802.11 emergentes, desde 802.11a a 802.11ac. La versión más utilizada es 802.11 ac, con velocidades de datos de Gigabits por segundo. En la actualidad se están realizando las primeras pruebas con la nueva versión del protocolo 802.11ax, conocido como WiFi 6, en entornos industriales [5]. Esta nueva versión, que se espera esté disponible al pública a mediados de 2019, promete mayor velocidad de descarga, menor latencia, un incremento en la capacidad de la red, un mejor desempeño en redes congestionadas y una mayor eficiencia energética [6]

- WiMax:
  WiMax es un estándar inalámbrico que opera en la banda de 11-66 GHz para servicios de datos de banda ancha fija. Se desarrolla con el grupo estándar de IEEE 802.16. En general, WiMax se entrega a través de redes de proveedores comerciales, aunque también se contemplan redes privadas.

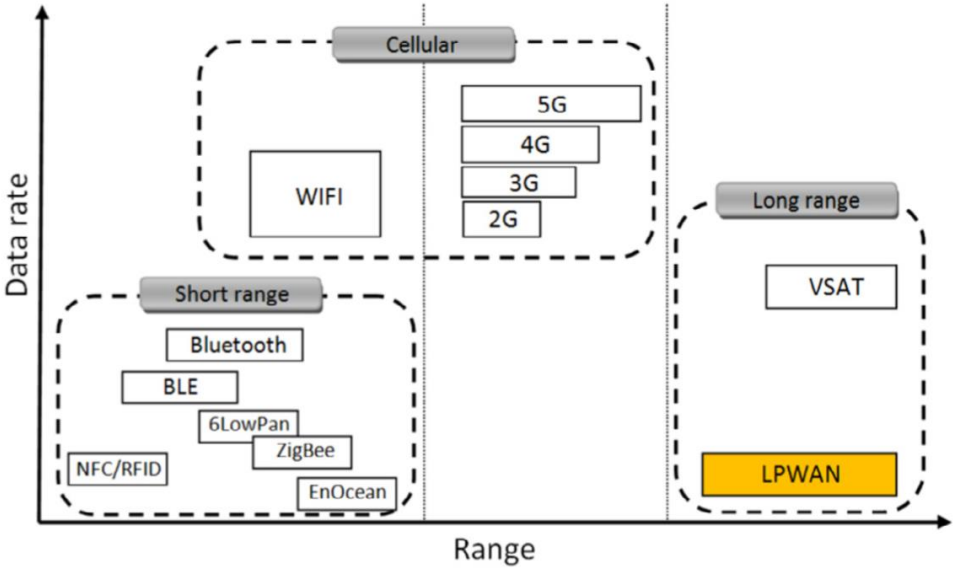


Figura 6. Rango de las tecnologías inalámbricas

Como podemos observar en la Figura 6, tanto WiFi como las tecnologías celulares proveen grandes anchos de banda, pero en la mayoría de los escenarios no logran suplir el alcance requerido

Según el alcance, tenemos que hay diferentes protocolos disponibles en entornos industriales. A nivel WLAN, el protocolo mayormente utilizado es WirelessHART, por encima de WiFi.

Existe un subgrupo de protocolos dentro del rango WWAN, conocido como Low Power Wide Area Network (LPWAN), que recoge protocolos de largo alcance, con el añadido de que requieren poca energía para funcionar, lo cual trae consigo el inconveniente de tener una velocidad de transmisión muy baja, en torno a los 0.3 y 50 kbits/s. Estos protocolos son útiles en entornos IoT que se manden mensajes muy pequeños, como intrucciones de control, pero las demandas en las redes IoT crecen cada vez más debido la integración con otro tipo de soluciones como Big Data o Business Intelligence, en las que se requiere envío de más información, por lo que la información a transmitir ya no es tan pequeña y se requieren velocidades de transmisión mayores, para poder funcionar en tiempo real.

En general las tecnologías inalámbricas actuales son insuficientes ya que no pueden satisfacer, a la vez, la amplia gama de requerimientos industriales como lo son: velocidad de transmisión de datos, latencia, confiabilidad, escalabilidad y eficiencia energética.

En algunos casos como en el control de vehiculos autónomos, en el que se necesita de una baja latencia y buena cobertura, para evitar colisiones entre los vehículos. Soluciones actuales como el uso de puntos de acceso WiFi o celdas LTE no suelen proveer de la cobertura suficiente causando pérdidas de conexión al movilizarse de una celda a otra o retraso en la comunicación.

3 Estado del Arte en Redes de Nueva Generación

3.1 Máquinas Virtuales y Contenedores

La propiedad de aislamiento de las máquinas virtuales proporciona a los usuarios un sistema independiente, es decir, una máquina completamente funcional, independientemente del hardware subyacente. La tecnología de las máquinas virtuales ofrece un control detallado para la creación de instancias y la finalización de tareas y procesos en cualquier momento sin afectar el hardware subyacente, lo que permite la flexibilidad en la provisión de recursos. Sin embargo, una máquina virtual, como podemos ver en la Figura 7, es una abstracción de una pila de hardware físico y requiere una imagen completa del SO huésped, binarios y bibliotecas adicionales para hospedar aplicaciones y servicios. A menos que una aplicación o servicio necesite un entorno de este tipo, una máquina virtual utiliza incorrectamente una cantidad significativa de recursos, además de su lento proceso de inicio debido al arranque de un sistema operativo completo.

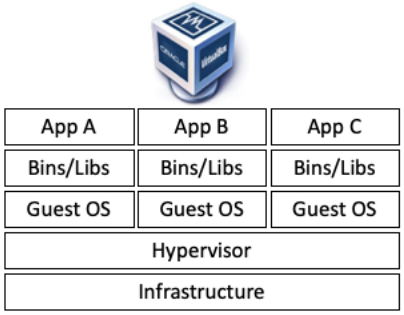


Figura 7. Abstracción de las máquinas virtuales.

En contraste, la abstracción en un contenedor, como podemos ver en la Figura 8, se lleva a cabo a nivel del sistema operativo, los programas y bibliotecas de soporte, y los recursos del sistema para ejecutar una aplicación o servicio específico. Los contenedores dividen los recursos de las máquinas físicas, creando múltiples instancias aisladas de espacio de usuario de un tamaño mucho más pequeño en comparación con las máquinas virtuales. Esto permite que se ejecuten múltiples contenedores dentro de un solo sistema operativo, lo que permite una implementación rápida con un rendimiento casi nativo en CPU, memoria, disco y redes. Por lo general, un contenedor ejecuta una aplicación o servicio, lo que brinda una fácil creación de instancias y beneficios de migración rápida debido a su naturaleza liviana, pero es menos seguro en comparación con las máquinas virtuales, que también podría ejecutar múltiples aplicaciones de manera más eficiente. En la práctica, se puede crear una instancia de un contenedor en la escala de milisegundos, mientras que la creación de instancias de una máquina virtual requeriría segundos o minutos, según las características del sistema operativo, el hardware físico y la carga del sistema.

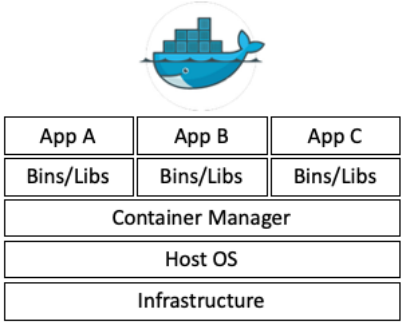


Figura 8. Abstracción de los contenedores.

3.2 Redes Definidas por Software (SDN)

Las redes definidas por software o SDN permiten un rápido despliegue de servicios innovadores, lo que permite la capacidad de programación de la red y el soporte multi-tenancy. Al desacoplar el plano de control del plano de datos y mediante el uso de API comunes, SDN introduce un control lógico centralizado, que puede arrancar y ofrecer fácilmente instancias de red virtual, al abstraer la infraestructura de red subyacente.. Por lo tanto, SDN puede admitir un encadenamiento de servicios flexible, ofreciendo un aprovisionamiento dinámico de servicios mediante la conexión de servicios VNF y MEC, que también pueden residir en el borde en un orden particular que cumpla con las demandas de rendimiento requeridas, a la vez que ayuda a la movilidad del servicio al permitir que los proveedores de aplicaciones y terceros guíen la infraestructura de red. SDN puede superar los desafíos actuales de enrutamiento asociados con la traducción de direcciones IP, el gran volumen de señalización de control, la sobrecarga de túneles y la administración dinámica de recursos.

3.3 Virtualización de Funciones de Red (NFV)

NFV es una tecnología que permite desacoplar las funciones y servicios de la red del hardware propietario, para que puedan ejecutarse como software en cualquier hardware de propósito múltiple. Esto permite una nueva arquitectura de diseño y despliegue de la red al consolidar las funciones y servicios de la red en plataformas en la nube, permitiendo ahorros de capital y gastos operativos. Igualmente, permite flexibilidad tanto en el plano de datos como en el plano de control, al ampliar y reducir los recursos asignados, lo que refleja la evolución de las demandas de servicio. NFV introduce funciones de red virtualizadas (VNF) que permiten la colocación de múltiples instancias en el mismo hardware o sobre el mismo entorno virtualizado, por ejemplo, máquinas virtuales. Las VNF pueden residir dentro de una única máquina virtual (VM)/contenedor o pueden distribuirse en múltiples VM/contenedores a través de una infraestructura en la nube. El Instituto de Estándares de Telecomuniciación (ETSI) mediante el grupo de desarrollo de las especificaciones de NFV (ETSI ISG NFV), se encarga de definir los requerimientos y detallar la arquitectura de esta tecnología, como podemos observar en la Figura 9.

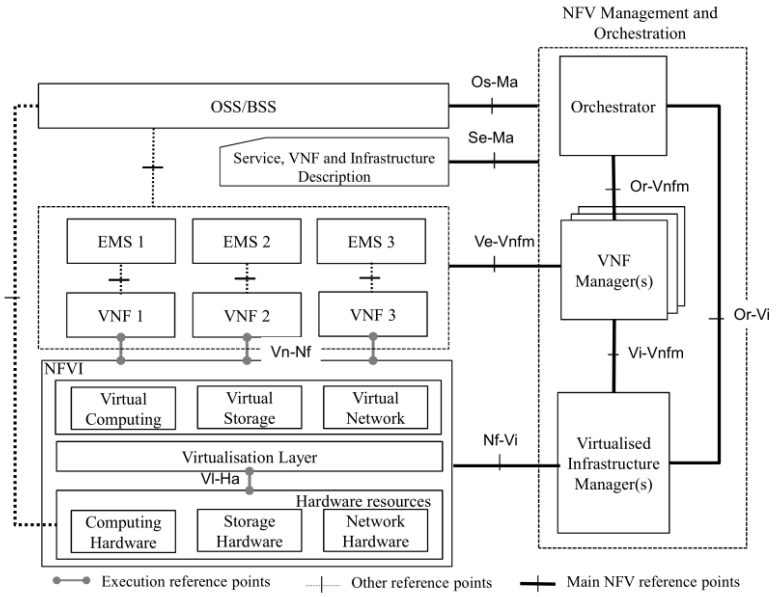


Figura 9. Arquitectura NFV ETSI

Esta arquitectura la podemos dividir en 3 grandes grupos, como podemos ver en la Figura 10 :

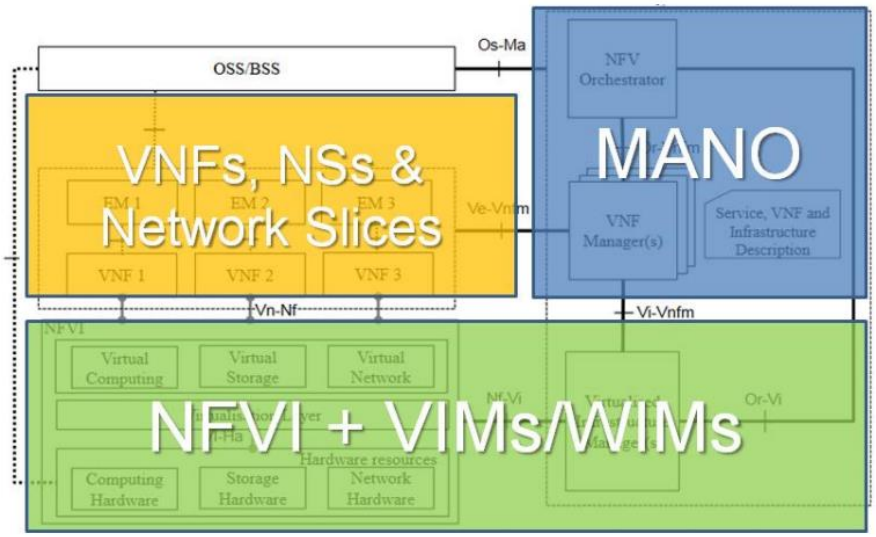


Figura 10. Subdivisón Arquitectura NFV

- NFVI + VIM/WIM:
  La primera parte (NFVI) es la Infraestructura NFV, la cual hospeda las maquinas virtuales o contenedores y los conecta mediante links virtuales.

- VNF, NS y Network Slices:
  Este grupo contiene la colección de funciones de red virtuales (VNF) que se interconectan entre si para formar servicios de red y estos a su vez se interconectan para formar networks slices.

- MANO:

  Este grupo es el que se encarga de manejar y orquestar los servicios (Management and Orchestration). Controla el ciclo de vida de las VNFs, NSs y Network Slices. Igualmente controla y mantiene la configuración y monitorea su salud y desempeño.

Uno de los aspectos más importantes para lograr un catálogo VNF unificado es tener una forma estándar de describir VNFs, por lo que las soluciones MANO deben dar la posibilidad de describir las VNFs mediante "descriptores" o VNFDs. Por ahora cada solución MANO tiene su manera de generar VNFDs, pero el objetivo de la industria es lograr la unificación a través de las diferentes plataformas.

3.4 Edge Computing

Edge Computing hace referencia a las aplicaciones, servicios y procesamiento llevado a cabo afuera de los centros de datos de los proveedores y más cerca del usuario final, en el borde. Dependiendo de los requerimientos de latencia y velocidad de acceso, la ubicación del borde puede variar.
Multi-Access Edge Computing (MEC) es una evolución del concepto de Edge Computing en el cual los operadores pueden desplegar soluciones Edge en su red, autorizando a terceros para asi integrar soluciones en la nube.
Es un estándar propuesto por ETSI con la participación de una gran parte de las empresas del mundo de las telecomunicaciones. Aparece inicialmente como una serie de propuestas propietarias y se encuentra en estos momentos en un proceso de estandarización para permitir a los operadores un despliegue más rápido y organizado de servicios.
MEC habilita soluciones modulares y abiertas, ofreciendo un ecosistema programable y transformando la experiencia de usuario.
Es una propuesta agnóstica al hardware, enfocada puramente en software, ya que está basada en tecnologías de virtualización.
El objetivo es definir una serie de APIs que permitan la creación de Funciones de Red Virtuales o VNFs que respondan a las necesidades en una red de comunicaciones. Debido a esto, puede proveer servicios de almacenamiento, computación, reconocimiento de ubicación y contexto, así como también agilidad y velocidad a las aplicaciones.
MEC habilita nuevos segmentos de negocios verticales y servicios para consumidores y clientes empresariales, como lo son:

- analítica de video
- servicios de localización
- internet de las cosas (IoT)
- realidad aumentada
- distribución optimizada de contenido local
- almacenamiento en caché de datos

3.4.1 Framework

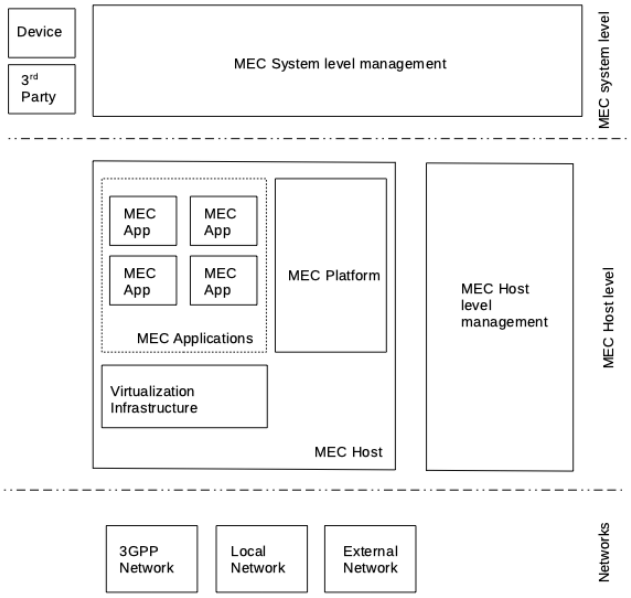


Figura 11. Arquitectura MEC ETSI

El framework (Figura 11) propuesto por ETSI muestra las entidades generales involucradas en la arquitectura del estándar. Éstas pueden ser agrupadas de la siguiente manera:

- Nivel de Sistema
- Nivel de Host
- Nivel de Red

Nivel de Host

A nivel de Host tenemos las siguientes entidades y subentidades:

- MEC host
  - MEC platform
  - MEC Application
- Infraestructura de Virtualización
- MEC host level management

Host MEC

El host MEC es una entidad que contiene la plataforma MEC, la infraestructura de virtualización, que provee los recursos de computación, almacenamiento y red para las aplicaciones MEC. La infraestructura de virtualización incluye el plano de datos que ejecuta las reglas de tráfico recibidas por la plataforma MEC y enruta el tráfico según aplicaciones, servicios, DNS o tipos de redes necesarios.

Plataforma MEC (MEC platform)

Es una colección de funcionalidades esenciales requeridas para ejecutar aplicaciones MEC en una infraestructura de virtualización particular y que las habilita para proveer y consumir servicios MEC. La plataforma MEC es responsable de las siguientes funciones:

- ofrecer un ambiente en el cual las aplicaciones MEC puedan descubrir, publicar, consumir y ofertar servicios MEC
- recibir reglas de tráfico del administrador de la plataforma MEC, aplicaciones o servicios e instruir al plano de datos de acuerdo con las instrucciones recibidas.
- recibir registros DNS del administrador de la plataforma MEC y en consecuencia configurar un servidor/proxy DNS
- alojar servicios MEC

- proveer acceso al almacenamiento persistente

Aplicación MEC (MEC application)

Las aplicaciones MEC corren como máquinas virtuales encima de la infraestructura de virtualización provista por el host MEC y pueden interactuar con la plataforma MEC para consumir y proveer servicios MEC. En algunos casos, también pueden interactuar con la plataforma MEC para realizar procedimientos de soporte relacionados con el ciclo de vida de la aplicación, como indicar disponibilidad o preparar una reubicación del estado del usuario.

MEC system level management

Las aplicaciones MEC pueden tener un cierto número de reglas y requisitos asociados, como los recursos requeridos, la latencia máxima, los servicios requeridos o útiles, etc. Estos requisitos son validados por MEC system level management y pueden asignarse los valores predeterminados si es necesario.

Orquestador MEC (MEC orchestrator)

El orquestador MEC es una función básica del MEC system level management. Es responsable de las siguientes funciones:

- mantener una visión general del sistema MEC basada en los hosts desplegados, servicios y recursos disponibles y la topología.

3.5 5G

La quinta generación de las redes comunicación móviles, conocida simplemente como 5G, trae consigo una nueva revolución en el mundo de las telecomunicaciones. El modelo de la red 5G es un modelo inclusivo que abarca todas las redes y está basado puramente en el protocolo IP.

Como con otras tecnologías, no hay una definición exacta de lo que es 5G, por ejemplo, 3GPP considera cualquier sistema que use su propuesta 5G New Radio (5G NR) como un sistema 5G, mientras que otros consideran que una tecnología es 5G si cumple con los requerimientos la International Telecomunicaciones Unión (ITU), el estándar IMT-2020 [7].

Aún con el inconveniente de tener diferentes definiciones de lo que es realmente 5G, esta nueva generación trata de resolver una serie de problemas que aparecen debido a los requerimientos de las nuevas tecnologías y que no pueden ser resueltos por generaciones anteriores:

- soporte para volúmenes de datos hasta 1000 veces mayores
- velocidades de hasta 10 Gb/s
- latencia ultra-baja (< 5ms)
- comunicación ubicua y conexiones masivas de más de 300k dispositivos por celda
- confiabilidad ultra-alta del 99.999% (disponibilidad de "los cinco nueves ")
- consumo de energía reducido hasta un 90%

En esta nueva generación se identifican 3 casos de uso principales:

3.5.1 Ultra-Reliable and Low Latency communications (URLLC):

Caso de uso que cumple con los requisitos estrictos de latencia y confiabilidad para comunicaciones de tipo crítico, como automatización de la factoría, conducción autónoma, cirugía remota e internet táctil. Estas aplicaciones requieren una latencia inferior a 1 ms con tasas de error inferiores a 1 pérdida de paquetes por cada 10^5

paquetes. Con LTE, la latencia se encuentra en el rango de 4 milisegundos bajo 3GPP versión 14. URLLC es parte del release 15 y tiene el objetivo de garantizar latencias de 1 milisegundo. En la tabla 2, podemos ver algunas aplicaciones en la industria de URLLC.

Industria Vertical	Aplicación
Factoría Conectada/ Automatización Industrial	Control industrial Control de robots Comunicación M2M Control de procesos
Salud	Diagnóstico remoto Respuesta a emergencias Cirugía remota
Entretenimiento	Juegos en línea Entretenimiento inmersivo
Transporte	Conducción asistida Conducción autónoma Control de tráfico
Manufactura	Control de movimiento Aplicaciones AR y VR
Energía	Smart Grid Smart Energy

Tabla 2. Aplicaciones de URLLC

3.5.2 Enhanced Mobile BroadBand (eMBB):

Ofrece banda ancha móvil que garantiza una alta velocidad y capacidad, similares a las que pueden ofrecer redes fijas como la fibra óptica. eMBB es una evolución natural de las tasas de datos que se ofrecen en LTE. El objetivo próximo de 5G es entregar hasta 10 Gb/s de rendimiento máximo, 1 Gb/s en situaciones de movilidad y hasta 10,000x de

tráfico de red total. Esto se puede lograr gracias a tecnologías en desarrollo como las ondas milimétricas y el MIMO masivo.

3.5.3 Massive Machine Type Communication (mMTC):

La comunicación tipo máquina (MTC) o comunicación máquina a máquina (M2M) se refiere la comunicación de datos automatizada entre dispositivos y la infraestructura de transporte de datos subyacente. Esta comunicación de datos puede ocurrir. entre un dispositivo MTC y un servidor, o directamente entre dos dispositivos MTC. 5G espera que este tipo de comunicación sea masiva y la contempla como uno de sus casos de uso principales. Ya se han logrado avances como parte de las tecnologías 3GPP de la versión 13/14 de Low Power Wide Area (LPWA), que incluye NB-IoT. Se espera que estos cumplan con la mayoría de los requisitos de 5G mMTC, mientras que otros que requieren más ancho de banda con baja latencia ultra confiable (URLLC completo) requerirán la implementación del núcleo 5G para una reducción total de la latencia de extremo a extremo. Las aplicaciones de misión crítica que son especialmente sensibles a la latencia también requerirán una amplia cobertura, lo cual es altamente improbable en las implementaciones iniciales de 5G, por lo que este desarrollo se realizará más adelante.

Capacidad	Objetivo 5G	Caso de Uso
Velocidad de datos máxima	20 Gbit/s	eMBB
Velocidad de datos experimentada por el usuario	1 Gbit/s	eMBB
	100 Mbit/s	eMBB
Latencia	1 ms	URLLC
Movilidad	500 km/h	eMBB/URLLC
Densidad de Conexión	10^6/km^2	MMTC
Eficiencia Energética	4G	eMBB
Capacidad de tráfico por área	1000 (Mbit/s)/m^2	eMBB
Eficiencia Espectral	3–4x 4G	eMBB

Tabla 3. Casos de uso 5G según los requerimientos IMT-2020

Para lograr cumplir con los requerimientos, mencionados den la Tabla 3, es necesario la introducción de nuevas técnicas y tecnologías de apoyo [8]:

- Massive MIMO:

  El concepto de MIMO ya existe en las redes actuales: red inalámbrica que permite la transmisión y recepción de más de una señal de datos simultáneamente a través del mismo canal de radio, generalmente haciendo uso de arreglos de 2x2 o 4x4, pero en 5G las estaciones base pueden admitir alrededor de cien puertos, lo que significa que muchas más antenas pueden caber en un solo arreglo, llegando hasta 128 antenas. Esa capacidad significa que una estación base podría enviar y recibir señales de muchos más usuarios a la vez, lo que aumenta la capacidad de las redes móviles, siendo de vital importancia para lograr despliegues mMTC.

- Ondas Milimétricas:

  Las ondas milimétricas se emiten en frecuencias entre 30 y 300 GigaHertz, en comparación con las bandas por debajo de 6 GHz que se utilizaron para dispositivos móviles en el pasado. Se llaman ondas milimétricas porque varían en longitud de 1 a 10 mm, en comparación con las ondas de radio que sirven a los dispositivos conectados de hoy en día, que miden decenas de centímetros de longitud.

- Small Cells:

  Se le conoce como Small Cell a las estaciones base portátiles en miniatura que requieren una potencia reducida para funcionar y tienen una cobertura más limitada con respecto a las macro celdas. Se utilizan principalmente para aumentar la capacidad en zonas con una alta demanda de los usuarios y para rellenar áreas que no están cubiertas por la red macro, tanto en exteriores como en interiores. También mejoran el rendimiento de la red y la calidad del servicio mediante la descarga de las macro celdas. Para evitar pérdidas de señal, los operadores pueden instalar miles de estas estaciones en una ciudad para formar una red densa que actúa como un equipo de retransmisión, recibiendo señales de otras estaciones base y enviando datos a los usuarios desde cualquier ubicación.

- Beamforming:

  Es un sistema de señalización de tráfico para estaciones base celulares que identifica la ruta de entrega de datos más eficiente para un usuario en particular y reduce la interferencia para los usuarios cercanos. Dependiendo de la situación y la tecnología, hay varias formas en que las redes 5G pueden implementarlo.

- WLAN Aggregation:

  Mediante la agregación de redes WLAN a las redes celulares es posible tener usuarios de la red celular que se conecten a redes WLAN desplegadas por el

operador celular, sin la intervención del usuario, manteniendo el acceso a los recursos de la red del operador. Esto ayuda a los operadores a aumentar la capacidad del sistema liberando recursos de la celular y permitiendo la reutilización del espectro.

- Non-Orthogonal Multiple Access (NOMA):
  Mediante el uso de multiplexación en dominio o energía, es posible dar acceso a la red a múltiples dispositivos al mismo tiempo y en la misma frecuencia. A este conjunto de técnicas de múltiplexación se les conoce como Non-Orthogonal Multiple Access.
- Time Sensitive Networking (TSN):
  Es un grupo de estándares IEEE 802.1 que permite la convergencia del control en tiempo real con transmisión de tiempo crítico y datos en masa en una sola red de comunicación. Reduce la complejidad y los costos a través de la convergencia de múltiples tipos de aplicaciones en una sola red. Protege el tráfico crítico contra los efectos del tráfico masivo convergente y no crítico y proporciona una latencia garantizada para todos los datos de control críticos.
- Uso de espectro sin licencia:
  También conocido como NR-U, consiste en el uso de las bandas de 5 y 6 GHz para el acceso a las redes 5G. Existen dos modos de NR-U: acceso asistido licenciado y acceso stand-alone. El primer modo permite a las operadoras mejorar el desempeño de sus redes mediante la agregación de espectro sin licenciar en sus redes. El segundo modo, no utiliza en ninguna parte del acceso a la red espectro licenciado y es el modo que permitiría la existencia de redes privada 5G para despliegues industriales.

Una manera de manejar la gran cantidad de conexiones que se esperan en las nuevas redes 5G, es mediante el despliegue de un gran número de celdas de diferentes tamaños y tecnologías en los nodos de acceso, conocido como HetNet (Heterogeneus Networks). Los operadores pueden proveer una mayor conexión mediante el uso de HetNets, integrando diferentes tecnologías de acceso, como WiFi, LTE, 5G, inclusive redes satelitales, así como también diferentes tamaños de celdas, desde las macro celdas en áreas abiertas, hasta femto celdas en entornos industriales, para cubrir posibles zonas sin cobertura, lo cual permite una mayor calidad de conexión.

3.6 Network Slicing

El concepto de Network Slicing hace referencia a la división de una red física en múltiples instancias virtuales, cada una de ellas diseñada y optimizada para un requisito específico y/o aplicación/servicio específico. La división de la red permite el despliegue de múltiples redes lógicas y auto contenidas en una infraestructura física común, lo que permite el aislamiento de recursos y una operación de red personalizada. La división de la red introduce un entorno multiusuario que admite el aprovisionamiento flexible de los recursos de la red, así como la asignación dinámica de las funciones de la red, los tipos de acceso de radio y las aplicaciones.

Para habilitar la personalización del servicio en la división de la red, es esencial una combinación de tecnologías NFV y SDN, que admita una coordinación estrecha para la asignación de VNF y el aprovisionamiento de servicios en la nube del borde, a la vez que permite un control de servicio flexible.

De acuerdo con NGMN en [9], una instancia de Network Slice (NSI) puede estar compuesta por ninguna, una o más subnets de NSI (NSSI), las cuales pueden ser compartidas por otra NSI. Igualmente, las NSSI, se conforman de una serie de funciones de red, ya sean VNF o PNF o HNF.Paralelamente, ETSI, considera que un Network Slice está compuesto por Network Subnets que forman parte de al menos un Network Service y a su vez los Servicios de Red están compuestos por Funciones de Red Virtuales (VNF), Funciones de Red Físicas (PNF) y/o Funciones de Red Hibridas (HNF).

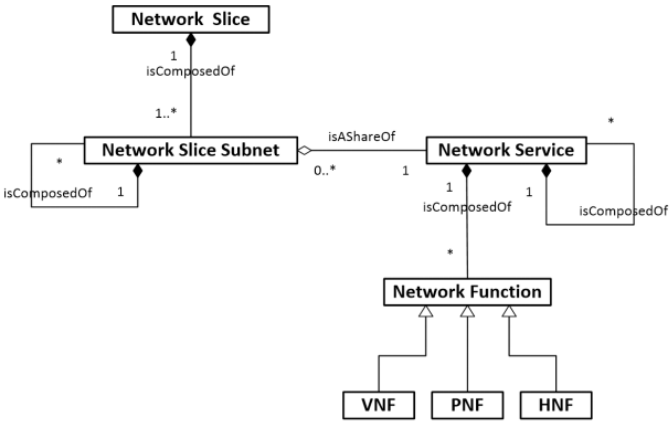


Figura 12. Concepto de Network Slice

Para simplificar la idea, podemos decir que las Network Slices operan como un agrupamiento de Servicios de Red.

Network Slicing es una de las tecnologías claves que ayudará a fomentar la adopción de 5G, debido a que permite desplegar sólo las funciones necesarias para proveer servicios a clientes y mercados específicos de manera expedita. El uso de TSN en Network Slicing 5G promete permitir soluciones end-to-end con latencias determinísticas para usos industriales.

4 Industria 4.0

La digitalización de la industria plantea retos y genera oportunidades para el sector industrial, que deberá adaptar sus procesos, productos y modelos de negocio.
El concepto de Industria 4.0, término originado gracias a la estrategia a largo plazo del gobierno alemán para promover la digitalización de la manufactura, se refiere a la cuarta revolución industrial y consiste en la introducción de las tecnologías digitales en la industria. El uso de estas nuevas tecnologías permite que dispositivos y sistemas colaboren entre sí, permitiendo mejorar los productos, los procesos y los modelos de negocio.

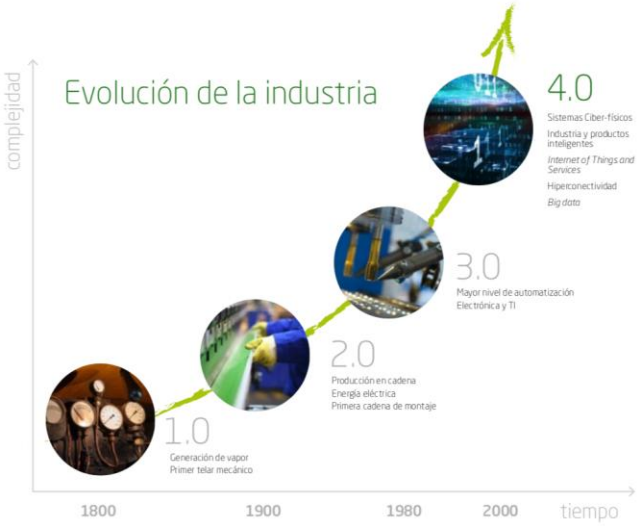


Figura 13. Evolución de la Industria [10]

La industria 4.0 es la siguiente fase en la evolución de los procesos de manufactura y esta se da debido a tres grandes disrupciones: la gran relevancia que comienzan a tener los datos, usados para inteligencia de negocio, el gran poder computacional y la hiperconectividad a la que tenemos acceso actualmente, y nuevas formas de interacción hombre-máquina.
Esta revolución implica a todos los niveles en la cadena de producción, siendo posible gracias la hibridación entre el mundo físico y el mundo digital, logrando que dispositivos y sistemas colaboren entre ellos y con otros sistemas creando una industria inteligente, como se muestra en la Figura 14.

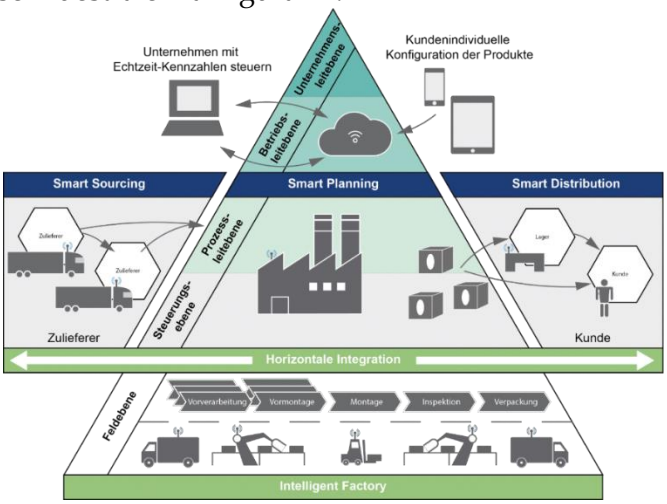


Figura 14. Integración para la Industria 4.0 [11]

De acuerdo con el parlamento europeo [12], para que una solución sea considerada como parte de la Industria 4.0 necesita cumplir con los siguientes requisitos:

- Interoperabilidad
- Virtualización
- Decentralización
- Capacidad en Tiempo Real
- Orientado a Servicios
- Modularidad

4.1 Tecnologías Habilitantes

Para lograr esta hibridación es necesario el apoyo de diferentes tecnologías, que se consideran como tecnologías habilitantes de la Industria 4.0. Al iniciar esta investigación en distintas plataformas científicas, se ha detectado que algunas de estas tecnologías tienen mayor relevancia que otras.

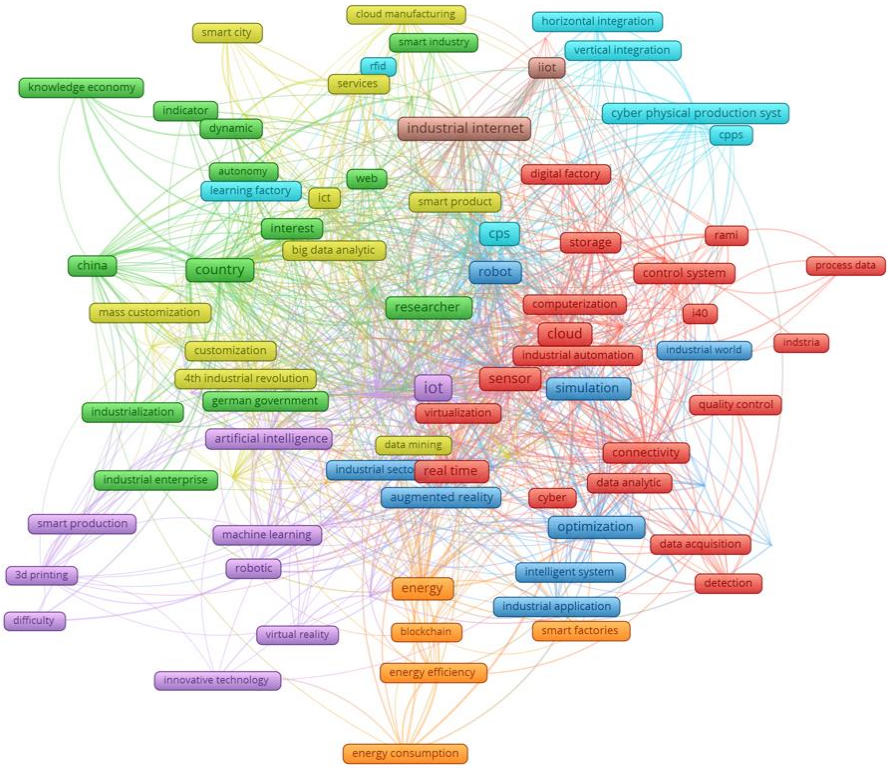


Figura 15. Análisis bibliográfico sobre la Industria 4.0

La IEEE ha detectado ciertas tendencias en las publicaciones que recoge referentes al tema de la Industria 4.0(mostradas en la Figura 16), de las cuales podemos identificar Internet de las cosas, Robots Autónomos, Big Data e Integración de Sistemas como los conceptos mayormente citados. S.S. Kamble et al [13], identifican conceptos similares relacionados con la Industria 4.0

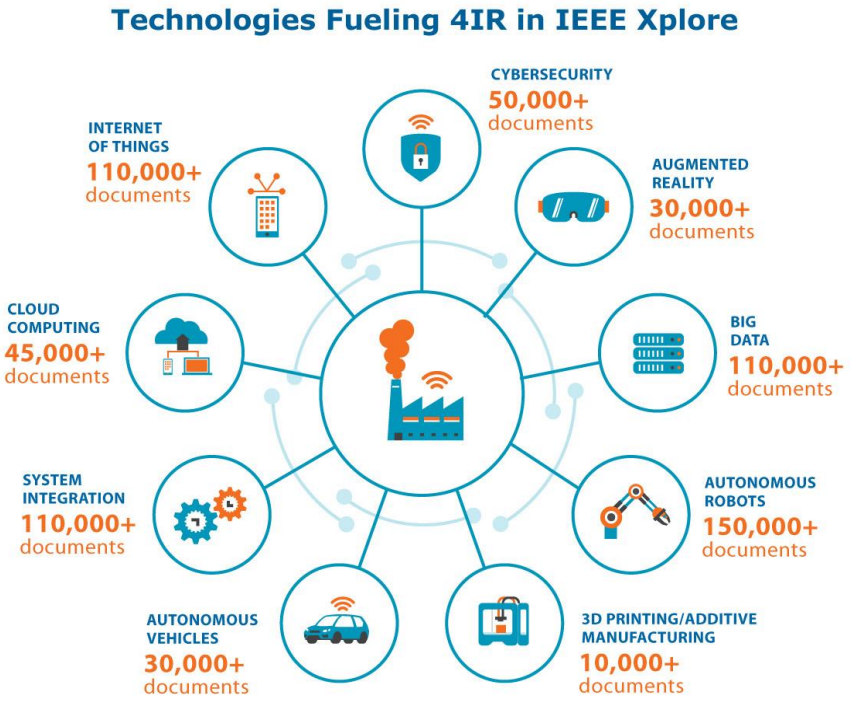


Figura 16. Tecnologías más relevantes según la IEEE

4.1.1 Robots Autónomos

En la actualidad está surgiendo una nueva generación de robots cada vez más útil. Son sistemas cada vez más autónomos, flexibles y capaces de trabajar de forma cooperativa. Se busca la interacción entre sistemas robóticos para trabajar de forma segura al lado de seres humanos, e incluso, aprender de ellos. El precio de estos robots se está reduciendo y abarcan nuevas capacidades aplicables en los entornos industriales. Estos robots están conectados de forma remota a los sistemas informáticos que proporcionan la inteligencia al robot móvil para que establezca correctamente la conexión con otros robots. Los robots móviles navegan de forma autónoma, actúan en enjambres y ofrecen una flexibilidad total para la fabricación industrial. Esto es especialmente importante para la logística interna.

Kuka Robotics [14], fabricante europeo de equipamiento robótico, ofrece sistemas autónomos robotizados que interactúan entre ellos. Estos robots se conectan de manera que puedan trabajar juntos y automáticamente coordinar sus acciones en la línea de producción. Equipados con sensores y unidades de control de alta prestaciones permiten una estrecha colaboración con las personas que participan el proceso productivo.

4.1.2 Internet de las Cosas (IoT)

La explosión prevista en el número de dispositivos conectados en los próximos años como lo indican varios estudios, entre ellos los de IoT Analytics [15] y Statista [16], promete mucho potencial para aplicaciones innovadoras a través de diferentes industrias. El precio a pagar por esta cantidad de aplicaciones se refleja en los desafíos para gestionar múltiples decenas de miles de millones de dispositivos conectados y los correspondientes redes, también llamada Internet de las cosas (IoT).

Kevin Ashton, mientras trabajaba en P&G, utilizó por primera vez el concepto de internet de las cosas, al usarlo de título en una presentación que hizo en 1999 para llamar

la atención de la alta gerencia, uniendo una nueva tecnología de la época, RFID, con el término de moda, Internet [17].
Es un concepto muy amplio, que abarca la interconexión de dispositivos, con cualquier otro equipo que este a su alrededor, en cualquier momento y en cualquier lugar por medio del internet. IoT ha evolucionado a partir de la convergencia de tecnologías inalámbricas, sistemas electromecánicos, microservicios e internet. La convergencia ha ayudado a derribar los silos entre la tecnología operacional y la tecnología de la información, permitiendo que los datos no estructurados generados por máquinas sean analizados para obtener información para impulsar mejoras. Cada año, son más los dispositivos que se conectan entre sí, por medio de redes inteligentes y se utilizan en entornos como salud, gestión de desastres, predicción meteorológica, agricultura, transporte, control del medio ambiente, domótica, sector financiero, ciudades inteligentes, gobierno, entre otras áreas. Como las redes IoT se utilizan en diversos sectores, con diferentes tecnologías e inclusive, con protocolos de comunicación diferentes, hace que sea una red heterogénea. A pesar de que el término IoT, se ha estado utilizando desde varios años, es una tecnología que no tiene un estándar definido. Se tienen muchos esfuerzos de normalización, intentando conseguir un estándar global, por esto se tienen diferentes arquitecturas. Por lo general, cuando se habla de IoT, se utilizan una serie de dispositivos para formar la red, estos pueden ser sensores, actuadores, microcontroladores, etiquetas RFID y dispositivos de comunicación inalámbrica conectados a Internet en un entorno inteligente.
Estos sensores pueden enviar pequeñas cantidades de datos, en ciertas ocasiones y sin prioridad y en otros casos pueden ser grandes cantidades de datos, los que serán enviados a través de la red, para ser almacenados o tratados con retardo o pérdidas de paquetes. Estos dispositivos, no necesariamente tienen que estar siempre conectados, dependiendo de la necesidad de la red, se puede ahorrar energía, cambiando su estado de forma dinámica, estando más tiempo desconectados (apagados) a la red. Esta opción se puede utilizar al adecuarlos a casos, donde los dispositivos no requieran ser alimentados con electricidad, si no con baterías y que su duración tenga que ser más larga. Uno de los más grandes retos que tiene la IoT, es que, al utilizar tantos dispositivos con características y protocolos diferentes al comunicarse, forman una red heterogénea.
Los numerosos elementos que componen un escenario IoT abarcan diferentes tipos de comunicación de datos con tecnologías bien conocidas como Bluetooth, ZigBee, WiFi, 6LoWPAN hasta redes móviles.
IoT genera mensajes adicionales en las redes de telecomunicaciones y requiere pasarelas para agregar los mensajes y garantizar una baja latencia y seguridad. Debido a la naturaleza de algunos de los dispositivos que se conectan, se requiere una capacidad de tiempo real y se necesita una agrupación de sensores y dispositivos para un servicio eficiente. Los dispositivos IoT a menudo tienen recursos limitados en términos de capacidad de procesador y memoria. Existe la necesidad de agregar varios mensajes de dispositivos de IoT conectados a través de la red de acceso de los dispositivos. Esto también proporciona una capacidad de procesamiento analítico y un tiempo de respuesta de baja latencia. Varios dispositivos se conectan a través de medios de acceso, como 3G, LTE, WiFi u otras tecnologías propietarias. En general, los mensajes son pequeños, están encriptados y vienen en diferentes tipos de protocolos. Existe la necesidad de un punto de agregación de baja latencia para administrar los diversos protocolos, la distribución de mensajes y para el procesamiento de los datos.

Tipos de Conexiones

Los sistemas IoT tienen una arquitectura de 3 niveles: dispositivos, puerta de enlace y sistemas de datos. La informacion fluye entre estos 3 niveles de 4 maneras diferentes como podemos ver en la Figura 17:

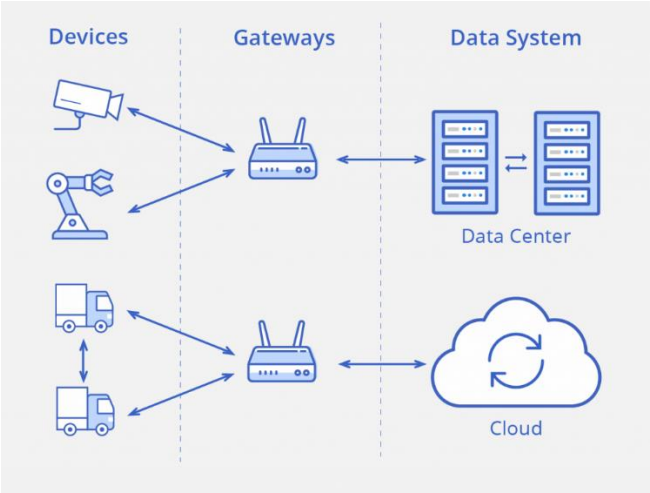


Figura 17. Tipos de conexiones IoT [18]

- Dispositivo a Dispositivo (D2D):
  Comunicación directa entre dos objetos inteligentes cuando comparten información instantáneamente sin intermediarios. Por ejemplo, los robots y sensores industriales están conectados entre sí directamente para coordinar sus acciones y realizar el ensamblaje de componentes de manera más eficiente. Este tipo de conexión no es muy común, porque la mayoría de los dispositivos no son capaces de manejar este tipo de procesos.

- Dispositivo a puerta de enlace:
  Comunicación entre sensores y nodos de pasarela. Las pasarelas son dispositivos informáticos más potentes que los sensores. Tienen dos funciones principales: consolidar los datos de los sensores y enviarlos al sistema de datos correspondiente; para analizar datos y, si se encuentran algunos problemas, devolverlos al dispositivo. Hay varios protocolos de puerta de enlace IoT que pueden adaptarse mejor a diferentes soluciones dependiendo de las capacidades de computación de la puerta de enlace, la capacidad y confiabilidad de la red, la frecuencia de generación de datos y su calidad.

- Puerta de enlace a sistemas de datos:
  Transmisión de datos desde una puerta de enlace al sistema de datos apropiado. Para determinar qué protocolo utilizar, debe analizar el tráfico de datos (frecuencia de ráfaga y congestión, requisitos de seguridad y cuántas conexiones paralelas se necesitan).

- Entre sistemas de datos:
  Transferencia de información dentro de centros de datos o nubes. Los protocolos para este tipo de conexión deben ser fáciles de implementar e integrar con las aplicaciones existentes, tener alta disponibilidad, capacidad y recuperación de desastres confiable.

Protocolos de comunicación:

- AMQP (Advance Message Queuing Protocol):
  Es uno de los protocolos más populares para los sistemas de cola de mensajes. Aunque el protocolo en sí es independiente del software y permite una gran flexibilidad arquitectónica, este enlace de transporte ha sido diseñado para funcionar con el agente RabbitMQ.

- XMPP (eXtensible Messaging and Presence Protocol)
  Es un protocolo de la capa de aplicación, para el intercambio de datos XML, de código abierto. Con el protocolo XMPP queda establecida una plataforma, para el intercambio de datos XML en tiempo real. Para el caso de IoT, el usuario puede recibir o enviar mensajes a máquinas u objetos.

- CoAP (Constrained Application Protocol):
  Es un protocolo de la capa de aplicación, para dispositivos y redes restringidas. protocolo cliente/servidor que copia el modelo HTTP, utilizando los requisitos como multidifusión, baja sobrecarga y simplicidad que son requerimientos importantes para IoT. Ya que los dispositivos implementados para IoT tienen menos capacidad de memoria y fuente de alimentación que los tradicionales. Utiliza UDP como protocol de transporte.

- MQTT (Message Queuing Telemetry Transport)
  Es un protocolo ligero de la capa de aplicación de tipo publicación-suscripción, diseñado para el envío de flujos de datos sencillos desde dispositivos de baja energía a capas superiores. Funciona sobre TCP/IP. Pertenece a la familia de protocolos máquina a máquina (M2M).

4.1.3 Cloud Computing

La computación en la nube reduce el costo de administrar software y hardware, a la vez que permite un alto poder de cómputo y alto rendimiento, accesibilidad conveniente y disponibilidad constante. Proporciona un conjunto de recursos compartidos, que se pueden escalar de manera dinámica, esto lo hace mediante una separación entre hardware, plataforma y aplicaciones.
La computación en la nube hace posible el aprovechamiento económico de la infraestructura, optimizar recursos según necesidades instantáneas, el ahorro en personal y un ecosistema de comunicaciones y cómputo, con mecanismos de competentes.

Figura 18. Hospedar, construir y consumir.

Se pueden identificar 3 modelos principales de computación en la nube (Figura 18):

Infrastructure as a Service (IaaS)

Consiste en la entrega recursos de procesamiento, almacenamiento, redes y otros recursos informáticos fundamentales donde el consumidor puede implementar y ejecutar software arbitrario, que puede incluir sistemas operativos y aplicaciones. El consumidor no administra ni controla la infraestructura subyacente, pero tiene control sobre los sistemas operativos, el almacenamiento y las aplicaciones implementadas y un control limitado de componentes de red seleccionados (por ejemplo, servidores de seguridad de host). El proveedor IaaS cobra por consumo de recursos utilizados: espacio en disco utilizado, tiempo de CPU, espacio para base de datos, transferencia de datos, etc. Las IaaS permiten desplazar al proveedor la mayor parte de los factores relacionados con la gestión de las máquinas con el ahorro de costos al pagar sólo por lo consumido y olvidarse de tratar con máquinas y su mantenimiento. Por otro lado, IaaS puede permitir una escalabilidad automática o semiautomática, de forma que se puedan contratar más recursos según se requieran.

Platform as a Service (PaaS)

Consiste en la entrega de una plataforma que contiene todo lo necesario para que el cliente pueda alojar las aplicaciones creadas o adquiridas por el consumidor mediante lenguajes de programación, bibliotecas, servicios y herramientas compatibles con el proveedor.
El proveedor PaaS es el encargado de escalar los recursos en caso de que la aplicación lo requiera, de que la plataforma tenga un rendimiento óptimo, de gestionar y garantizar la seguridad de acceso.
Con el uso de PaaS, se abstrae del hardware físico al cliente, lo cual es interesante para muchos desarrolladores web, ya que no se administra ni controla la infraestructura subyacente, incluida la red, servidores, sistemas operativos o almacenamiento, pero se

tiene control sobre las aplicaciones implementadas y posiblemente los ajustes de configuración para el entorno de alojamiento de aplicaciones.

Software as a Service (SaaS)

Consiste en la entrega de aplicaciones completas como un servicio. Se puede acceder a las aplicaciones desde varios dispositivos cliente a través de una interfaz de cliente ligero, como un navegador web o una interfaz de programa. El consumidor no administra ni controla la infraestructura subyacente, incluidas la red, los servidores, los sistemas operativos, el almacenamiento o incluso las capacidades de las aplicaciones individuales, con la posible excepción de los ajustes limitados de configuración de aplicaciones específicos del usuario. El proveedor SaaS es el encargado de operar y mantener estas aplicaciones que pueden ser desarrolladas por el proveedor o por un tercero.

4.1.4 Big Data

No existe un consenso sobre la definición de Big Data, inicialmente la mayoría de las definiciones giraban en torno a las 3 Vs [19]:

- Volumen:
  Se refiere a la cantidad de datos creados, representa el tamaño total del conjunto de datos debido a la agregación de un gran número de variables y un conjunto aún mayor de observaciones para cada variable.
- Variedad:
  Se refiere a la pluralidad de las fuentes de datos estructuradas o no estructuradas, como texto, video, redes, gráficas entre otros.
- Velocidad:
  Se refiere a qué tan rápido se recoge y se analiza la información, Ya sea en tiempo real o casi en tiempo real desde sensores, transacciones de ventas, publicaciones en redes sociales y datos de sentimientos.

Pero actualmente, se considera que son 5 Vs, las 3 mencionadas anteriormente y, además [20]:

- Veracidad:
  Se refiere al hecho de que la calidad y exactitud de los datos puede no ser de un alto nivel, es decir, menos uniformados, consistentes y controlable.
- Valor:
  Se refiere a la capacidad individual u organizativa de convertir los datos en valores reales, que incluye una capacidad de recopilación y luego aprovechar los datos para lograr objetivos específicos.

Cada vez generamos y podemos recolectar más datos por lo cual el análisis de Big Data se ha convertido en un importante campo de estudio tanto para investigadores como para profesionales, y a la vez genera una gran demanda de soluciones para empresas.

4.1.5 Realidad Aumentada/Realidad Virtual

La realidad aumentada (AR) es la combinación de una vista del entorno del mundo real y una entrada sensorial complementaria generada por computadora, como sonido, video, gráficos o datos de GPS. La realidad aumentada puede mejorar la experiencia de usuario mostrando información relacionada a lo que está viendo.
Los servicios de realidad aumentada requieren una aplicación para analizar la salida una cámara de un dispositivo y/o una ubicación precisa para complementar la experiencia del usuario cuando observa algún punto de interés o dispositivo específico al proporcionar información adicional al usuario sobre lo que está sucediendo actualmente. La aplicación debe ser consciente de la posición del usuario y la dirección hacia la que está orientada, ya sea a través de técnicas de posicionamiento o a través de la vista de la cámara, o ambas cosas. Después de analizar dicha información, la aplicación puede proporcionar información adicional en tiempo real al usuario.

P. Fraga-Lamas et al [21], identifican los siguientes casos de uso:

- Control de Calidad:
  Mediante el uso de cámaras 3D y técnicas de visión computarizada es posible monitorizar la calidad del producto final comparando imágenes del producto final, en tiempo real, con una base de datos previa del producto esperado y detectar posibles imperfecciones.
- Asistencia en los procesos de manufactura:
  Asistencia a operadores, mediante el uso de gafas de realidad aumentada, para la visualización de modelos 3D de referencia.
- Ubicación de productos y herramientas (administración de bodegas):
  Asistencia a operadores, mediante el uso de gafas de realidad aumentada, para la localización de productos o herramientas en las plantas de producción.
- Mantenimiento predictivo:
  Mediante la recolección de datos de las máquinas de producción para la predicción de daños, que luego son analizados y mostrados a ingenieros de planta mediante gafas de realidad aumentada, para que puedan tomar las decisiones pertinentes.
- Comunicación aumentada:
  Facilita la comunicación entre operadores y controladores/expertos a través de dispositivos de realidad aumentada portátiles como gafas o tablets. Permite la orientación o resolución de incidencias sobre el terreno, compartiendo el punto de vista (POV) del operador independientemente de ubicación, permitiendo la superposición de información sobre la imagen real vista por el operador, grabando anotaciones y/o comunicaciones de audio/video.

- Visualización de zonas ocultas:
  Consiste en la visualización de instalaciones ocultas detrás de mamparas o techos mediante la superposición de elementos virtuales 3D sobre el entorno real. Una solución ideal ayudaría a los operadores durante los procesos de montaje para revelar elementos internos que serían difíciles de ver de otra manera. Además, tal sistema facilitaría el mantenimiento o la reparación de fallas. Un ejemplo de escenario sería uno en el cual, después de la detección de una falla en un barco, el operador utiliza el sistema propuesto para inspeccionar las instalaciones que están a simple vista en la zona afectada, ayudando a decidir de una manera más rápida e informada.
- Operación remota de dispositivos:
  Un panel de control virtual se puede superponer a un producto inteligente para ser operado directamente usando una dispositivo de realidad aumentada, gestos con las manos, comandos de voz, o incluso a través de otro producto inteligente.

4.1.6 Impresión 3D/ Manufactura Aditiva

La fabricación aditiva se refiere a un proceso mediante el cual los datos de diseño 3D digital se utilizan para construir un componente en capas mediante el depósito de material. El término "impresión 3D" se usa cada vez más como sinónimo de Fabricación Aditiva. Sin embargo, este último es más preciso porque describe una técnica de producción profesional que se distingue claramente de los métodos convencionales de eliminación de material. En lugar de fresar una pieza de trabajo a partir de un bloque sólido, por ejemplo, la fabricación aditiva acumula componentes capa por capa utilizando materiales que están disponibles en forma de polvo fino. Se puede utilizar una gama de diferentes metales, plásticos y materiales compuestos. La impresión 3D es una tecnología de transformación que ofrece grandes posibilidades para cambiar las cadenas de suministro actuales ya que amplía el concepto de desarrollo de productos y permite a las personas no solo desarrollar productos sino también fabricarlos. Las herramientas de diseño 3D se han convertido en herramientas que para diseñar productos que se pueden imprimir de forma más accesible y esto tiene un gran impacto en el proceso de diseño de partes al agregar flexibilidad en la creación de prototipos.
Aún más sorprendente es su capacidad para pasar los límites en el diseño de productos, nadie hubiera creído posible hace unos años. Es posible crear nuevas formas e integrar partes en el producto general que con el uso de tecnologías sustractivas tendrían que producirse por separado y montarse en el producto original mediante el uso de pernos o tornillos.
Además, se adoptan materias primas adicionales para ser utilizadas en la impresión 3D. El tiempo en que solo fue posible utilizar polímeros ha pasado. Hoy en día, el acero, el aluminio y el titanio se están considerando para el desarrollo de materiales para la impresión 3D.

4.2 Adopción de la Industria 4.0

Las redes industriales han evolucionado desde tecnologías de buses hasta el Industrial Ethernet, pero todavía son un cuello de botella, que no logran satisfacer todas las demandas de las nuevas tecnologías requeridas por la Industria 4.0, ya que no fueron pensadas para cumplir estos requisitos.

Las redes 5G plantean una posible solución a estos problemas y una manera de acelerar la adopción de la Industria 4.0, algunas de las ventajas de estas redes que lo permiten son [22]:

- Virtualización de los recursos de red:
  La misma infraestructura física puede ser utilizada en múltiples casos de uso, no hay necesidad de construir una red física para cada nuevo caso de uso, al poder virtualizar los recursos de red y orquestar nuevos servicios y topologías según sea necesario.
- Mayor soporte de movilidad:
  Permite controlar dispositivos que puedan reubicarse de manera instantánea o que migren de una red a otra sin perder conectividad en el proceso.
- Mayor duración de batería:
  5G promete conexiones por hasta 10 años, 15 años en casos extremos, sin necesidad de cambiar las baterías de los dispositivos.
- Privacidad:
  Con 5G es posible instalar redes privadas de una manera eficiente, dentro de las fábricas, ayudando a disminuir el riesgo de fuga de datos sensitivos.
- Seguridad:
  El uso de tarjetas SIM proveen una manera segura de gestionar dispositivos y restringir el acceso a la red.
- Economía de escala:
  Debido a la gran demanda de conexiones los equipos de conexión a la red verán disminuido su costo, generando mayor oportunidades de negocio.

Pero esta no es la solución definitiva, al menos no de manera inmediata. 5G en su estado actual todavía no tiene definidas todas las tecnologías que podrían beneficiar grandemente el despliegue de soluciones 4.0, 3GPP en su Release 15 considera únicamente los 3 tipos de servicio básico eMBB, mMTC and URLLC. se puede utilizar como un medio para separar estos servicios de una manera óptima y conveniente: por ejemplo, se puede configurar un segmento separado para cada servicio. En términos de rendimiento (por ejemplo, con respecto a la latencia, rendimiento, disponibilidad, resistencia, confiabilidad), cada segmento podría cubrir los requisitos de los diferentes casos de uso.
No es sino hasta la Release 16, también llamada 5G Fase 2 y prevista para diciembre 2019 [23], que 3GPPP comienza a estudiar y definir nuevos casos de uso como la comunicación dispositivo a dispositivo(D2D), vehículo a cualquier cosa (V2X), soporte

satelital, IoT celular (CBIoT), soporte para redes LAN. Igualmente, los requerimientos de conectividad, movilidad, energía, ancho de banda siguen siendo muy diversos según el escenario industrial a implementar.
Teniendo esto en cuenta, cabe destacar que la solución a corto plazo para acelerar la adopción del concepto de Industria 4.0, pasa por la combinación de tecnologías actuales (Industrial Ethernet, Industrial Wireless Networks) y los nuevos casos de uso arropados bajo el estándar 5G (NB-IoT, CB-IoT, mMMTC, eMBB, URLLC, D2D, entre otros). Esto se puede lograr gracias a la visión de redes Heterogéneas y la flexibilidad que nos permite el uso de la virtualización en la arquitectura 5G.

A. Neumann, L. Wisniewski et al [24], prevén este tipo de inconvenientes y proponen 3 escenarios de integración, de los cuales queremos destacar la integración de varios tipos de sensores, sitios remotos y tecnologías de acceso. Esta integración es la más versátil y es la que tenemos en cuenta en el escenario propuesto en el capítulo 6, debido a la facilidad de introducir los componentes a la red de manera virtualizada sin tener que depender de equipos propietarios, más allá de los dispositivos Industriales a conectar que pueden utilizar protocolos y medios de acceso tanto abiertos como propietarios.

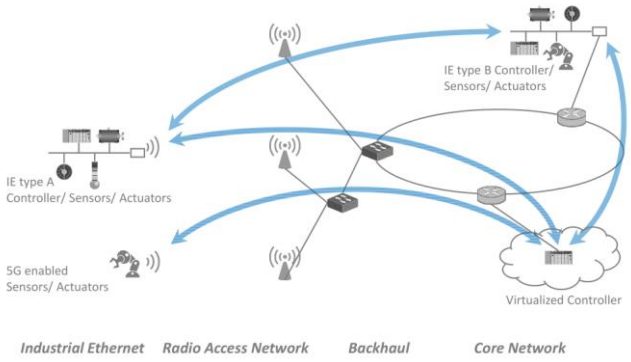


Figura 19. Control Virtualizado IoT

En este escenario, se virtualiza el controlador (Figura 19) en la red del operador y se pueden gestionar diferentes tipos de sensores independientemente del medio de acceso o su ubicación. Con esto, podemos integrar sensores 5G que usen NB-IoT o CBIoT o sensores con otros protocolos Industriales como LoRa, SigFox, que pueden ser gestionados en el borde de la red con una solución MEC o con un controlador en el core del operador.

Igualmente, se espera poder ofrecer servicios end-to-end con latencia determinística en 5G mediante el uso de Time Sensitive Networking (TSN), pero esta idea no está lista para un entorno de producción y se requiere de un mayor estudio para poder ser aplicado, según las conclusiones a las que llegó el grupo de estudio en cojunto de la IEEE 802 y la ITU-T en el workshop "Building Tomorrow's Networks" [25], lo cual nos da mayores razones para sostener la idea de que la transición hacia la Industria 4.0, todavía no es factible con una infraestructura 100% 5G. Las soluciones a mediano plazo pasan por la interconexión de redes industriales tradicionales y las redes de nueva generación hasta que los estándares hayan sido propiamente definidos.

5 Casos de Uso en la Industria 4.0

El consorcio 5GPPP identifica 5 familias de casos de uso para la Industria 4.0 en redes 5G [26]:

1. Optimización de procesos dentro de la factoría conectada
2. Comunicación dentro de la factoría conectada *non time-critical*
3. Control remoto de la factoría conectada
4. Comunicación Intra/Inter empresa
5. Productos Conectados (Interconexión de la cadena de Suministro)

Basado en los requerimientos que plantea la revolución de la Industria 4.0 y las familias que resume 5GPPP hemos encontrado diferentes propuestas que se revisarán más a fondo a continuación:

5.1.1 Realidad Aumentada

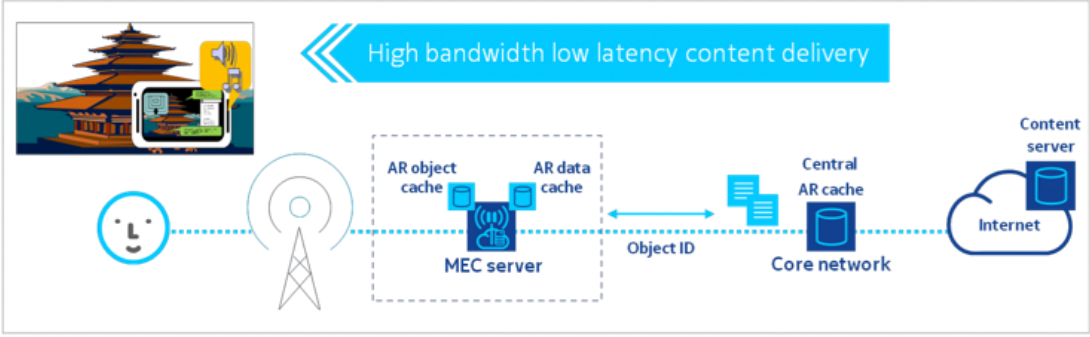


Figura 20. Despliegue MEC en un escenario de realidad aumentada [27]

Alojar el servicio de Realidad Aumentada en una plataforma MEC en lugar de en la nube es ventajoso, ya que la información complementaria relativa a un punto de interés es altamente localizada y, a menudo, es irrelevante más allá del punto de interés particular. Además, el procesamiento de la ubicación del usuario o la vista de la cámara se puede realizar en una plataforma MEC en lugar de en un servidor más centralizado. Puede ser necesario actualizar la información a una velocidad rápida, dependiendo de cómo se mueva el usuario y el contexto en el que se utiliza el servicio de realidad aumentada. Los datos de realidad aumentada requieren una latencia baja y una alta tasa de procesamiento de datos para proporcionar la información correcta al dispositivo del usuario dependiendo de la ubicación y orientación del usuario. Realizar dicho procesamiento de datos en la plataforma MEC también tiene la ventaja de recopilar métricas, metadatos anónimos, etc., con el fin de analizar el uso del servicio y ayudar a mejorar el servicio para proporcionar una mejor experiencia al usuario.

5.1.2 Aceleración de Video Inteligente

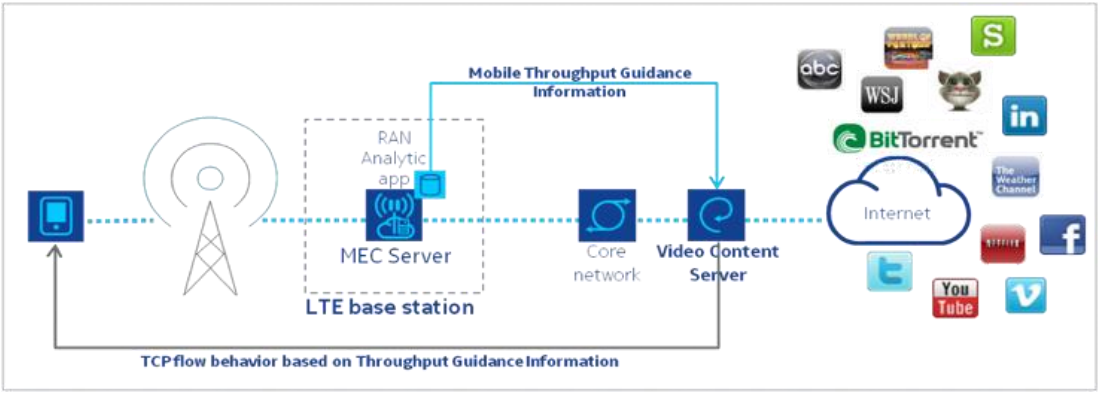


Figura 21. Despliegue MEC en un escenario de aceleración de video inteligente.

La calidad de experiencia (QoE) del usuario final y la utilización de los recursos de la red de radio se pueden mejorar a través de la aceleración de video inteligente. Los medios de Internet y la entrega de archivos normalmente se transmiten o descargan hoy utilizando el protocolo HTTP a través del protocolo TCP. La capacidad disponible puede variar en un orden de magnitud en segundos (como resultado de los cambios en las condiciones del canal de radio, los dispositivos que entran/salen de la red). Es posible que TCP no pueda adaptarse lo suficientemente rápido a las condiciones que varían rápidamente en la red de acceso de radio (RAN). Esto puede llevar a una subutilización de valiosos recursos de radio y a una experiencia de usuario subóptima.

Una aplicación de análisis de radio, que reside en un servidor MEC, proporciona al servidor de video una indicación del rendimiento estimado disponible en la interfaz del enlace descendente de radio. Esta información se puede usar para ayudar a las decisiones de control de congestión de TCP (por ejemplo, al seleccionar el tamaño de la ventana inicial, establecer el valor de la ventana de congestión durante la fase para evitar la congestión y ajustar el tamaño de la ventana de congestión cuando las condiciones en el enlace de radio se empiezan a deteriorar). La información también se puede utilizar para garantizar que la codificación a nivel de la aplicación coincida con la capacidad estimada en el enlace de bajada.

El servidor de video puede usar esta información para ayudar a las decisiones de control de congestión de TCP. El tiempo de inicio del contenido, así como los casos de bloqueo de video pueden reducirse, lo que permite mejorar la calidad y el rendimiento del video.

5.1.3 IoT Gateway

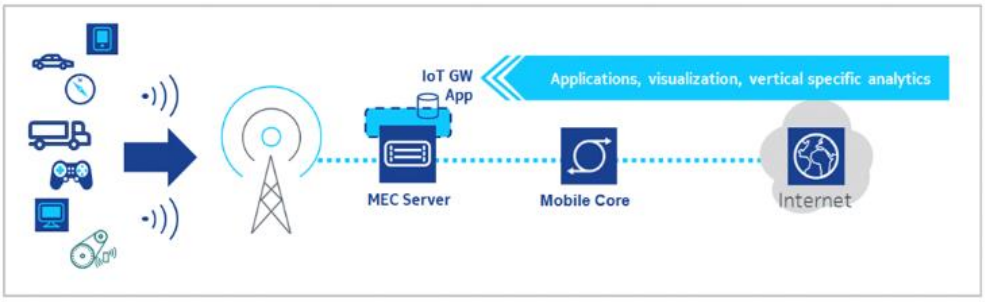


Figura 22. Despliegue MEC en un escenario IoT

MEC se puede utilizar para conectar y controlar dispositivos de forma remota, analizar y proporcionar aprovisionamiento y análisis en tiempo real. MEC permite la agregación y distribución de servicios de IoT en el entorno de estación base móvil altamente distribuida, y permite que las aplicaciones respondan en tiempo real. Esto puede reducir

el tiempo de ida y vuelta de los datos y habilitar una capa de abstracción tanto de la red central como de las aplicaciones en la nube. Las aplicaciones IoT pueden ejecutarse en servidores MEC que se despliegan en el sitio de la estación base LTE o 5G para proporcionar esta funcionalidad.
Algunas de las ventajas de desplegar una pasarela IoT:

- Permite la integración con sistemas de control en la nube
- Funciona como punto de agregación para diferentes tecnologías inalámbricas
- Los datos se transmiten mediante interfaces REST
- La información puede ser procesada de tal manera que solo alarmas sean enviadas mediante interfaces 5G

Este caso de uso se verá más a fondo en el caso escenario de pruebas propuesto en el capítulo 6.

5.1.4 Automóviles Conectados

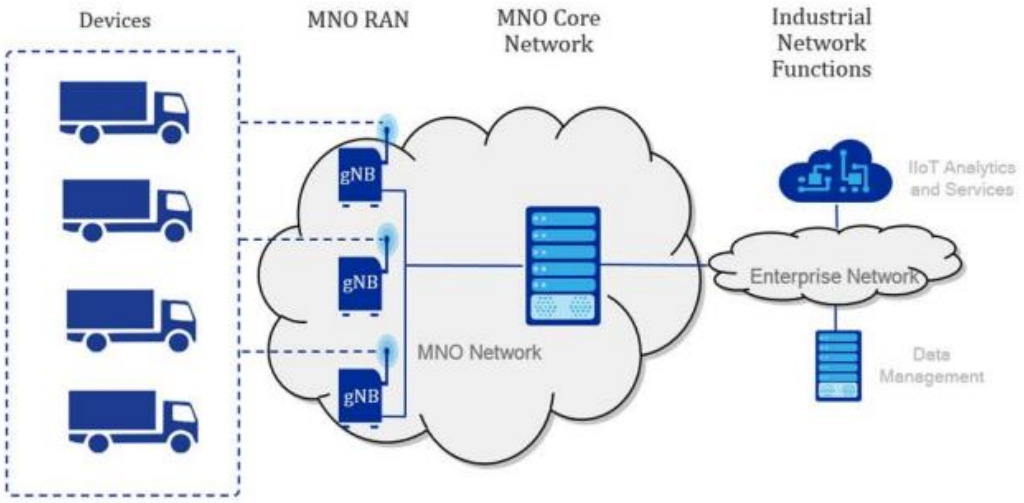


Figura 23. Despliegue MEC en un escenario de Automóviles Conectados

El número de vehículos conectados está creciendo rápidamente y continuará haciéndolo en los próximos años. La comunicación de los vehículos y los sensores de carretera con una unidad de carretera está destinada a aumentar la seguridad, la eficiencia y la comodidad del sistema de transporte, mediante el intercambio de datos críticos de seguridad y operacionales. La comunicación también se puede utilizar para proporcionar servicios de valor agregado, como buscador de automóviles, ubicación de estacionamiento y para soportar servicios de entretenimiento.
A medida que aumenta el número de vehículos conectados y los casos de uso evolucionan, el volumen de datos continuará aumentando junto con la necesidad de minimizar la latencia. Si bien los datos almacenados y procesados centralmente pueden ser adecuados para algunos casos de uso, pueden ser poco confiables y lentos para otros. MEC se puede utilizar para extender la nube del automóvil conectado al entorno de una estación base móvil, y permitir que los datos y las aplicaciones se alojen cerca de los vehículos. Esto puede reducir el tiempo de viaje de ida y vuelta de los datos y permitir una capa de abstracción tanto de la red central como de las aplicaciones proporcionadas a través de Internet. Las aplicaciones MEC se pueden ejecutar en los servidores MEC que se implementan en el sitio de la estación base para proporcionar la funcionalidad en la carretera. Las aplicaciones MEC pueden recibir mensajes locales directamente de las aplicaciones en los vehículos y los sensores de la carretera, analizarlos y luego propagar, con latencia extremadamente baja, advertencias de peligro y otros mensajes sensibles a la latencia a otros automóviles en el área. Esto permite que un automóvil cercano reciba

datos en cuestión de milisegundos, lo que permite que el conductor reaccione de inmediato.
La aplicación MEC en la autopista podrá informar a los servidores de computación de MEC adyacentes sobre el evento o eventos y, al hacerlo, habilitar estos servidores para propagar advertencias de peligro a los automóviles que están cerca del área afectada.
La aplicación en la autopista podrá enviar información local a las aplicaciones en la nube del automóvil conectado para un procesamiento e informes centralizados adicionales.

5.1.5 Realidad Aumentada

Localización de Productos

Figura 24. Gafas de realidad aumentada.

Mediante el uso de gafas de realidad aumentada conectadas a redes 5G, tags RFID y una plataforma Edge, para el procesamiento de la información, es posible identificar productos en una planta de producción, un depósito o un puerto.

Soporte Human-Machine Interaction (HMI)

La realidad aumentada puede ayudar a mejorar la percepción humana en diferentes escenarios mostrando información relevante en el campo de visión de los empleados. Mediante el uso de dispositivos móviles como tablets o celulares, o directamente gafas con conexión 5G, puede visualizar en tiempo real información de contexto sobre el estado actual de las máquinas y poder actuar apropiadamente.

Asistencia Remota

Soluciones de Realidad aumentada pueden ser desplegadas para proveer un segundo nivel de soporte, ayudando a técnicos remotos a instruir a personal en sitio con información precisa que puede ser mostrada en gafas AR, tablets o teléfonos móviles.
La empresa SCOPE AR [28] provee soluciones de este tipo (Figura 25) que pueden ser mejoradas con el despliegue de redes móviles dedicadas para tablets o gafas AR.

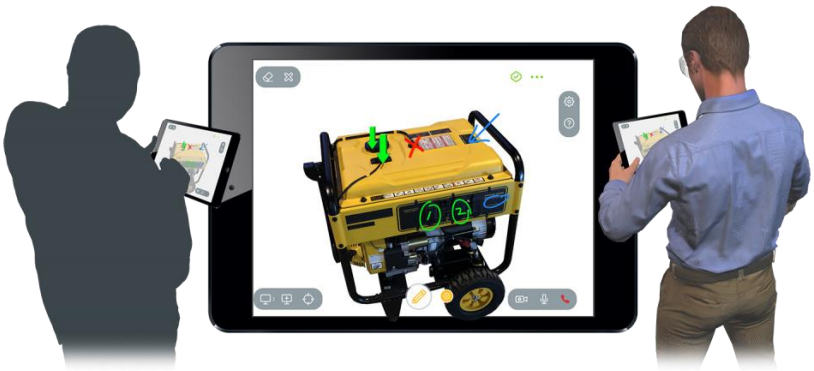


Figura 25. Solución REMOTE AR

5.1.6 Redes 5G Privadas/Semi-Privadas

Con el uso de redes 5G existe la posibilidad de implementar y operar redes privadas en áreas bien definidas como, por ejemplo, dentro de una fábrica. El espectro requerido podría adquirirse directamente de los reguladores del espectro a través de licencias regionales/locales o, en algunos casos, obtenerse de un operador de red móvil mediante subarrendamiento en un área geográfica específica.

Actualmente se discuten tres tipos diferentes de redes [29]:

- Public Land Mobile Networks (PLMN): redes clásicas 3GPP
- "Tipo A": red privada con seguridad 3GPP, en y roaming de salida a las PLMN, continuidad del servicio
- "Tipo B": red privada con seguridad alternativa, no roaming a/interacción con PLMNs.

Propuestas como ICRAN (Industrial Cellular Radio Access Networking) de Huawei [30] adaptan soluciones RAN a las necesidades industriales con el objetivo de proveer cobertura 5G a los dispositivos dentro de la fábrica. Puede ser desplegado de manera independiente o conectarse a una PLMN.

5.1.7 Network Slicing para la Industria

T. Taleb et al en [31] proponen un framework para para la creación y el manejo del ciclo de vida de network slices para la Industria 4.0. Dentro de éste ámbito es necesario soportar diferentes flujos de tráfico cada uno diferentes requerimientos como desempeño, confiabilidad y seguridad y las arquitecturas monolíticas no soportan estos requerimientos.

Este framework introduce el concepto de Slices Federados en el borde que van en línea con otras propuestas que hemos revisado durante la investigación, como por ejemplo la visión de la alianza 5G-ACIA de la Industria Conectada, la cual podemos ver en la Figura 26.

Los Slices federados no son más que la unión de Slices de diferentes proveedores en un único punto, en este caso en la red del cliente, lo cual permitiría interconectar slices para diferentes servicios dentro de una red industrial.

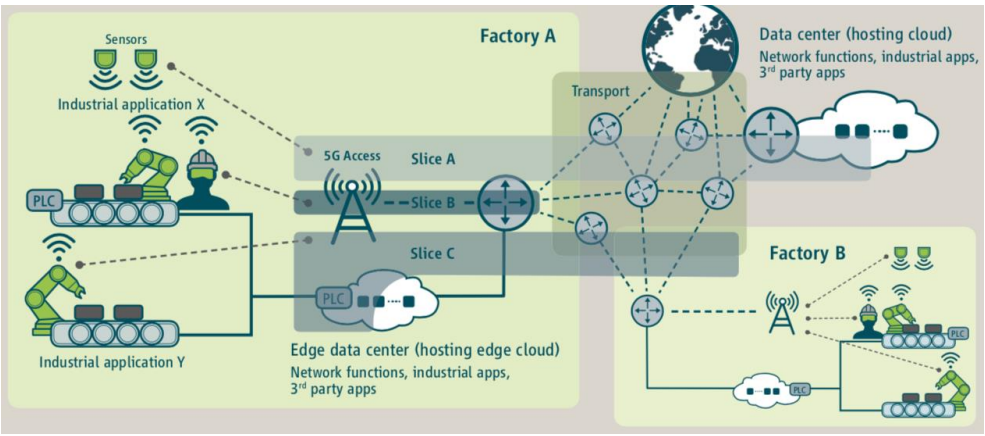


Figura 26. Industria Conectada Visión 5G-ACIA [32]

Estos Slices Federados posibilitan el manejo de la información más cerca de la fuente, para cada uno de los servicios ofrecidos, pudiendo así cumplir con los requisitos de ancho de banda y latencia que se tienen en la Industria 4.0.

Monitorización Remota de la Producción

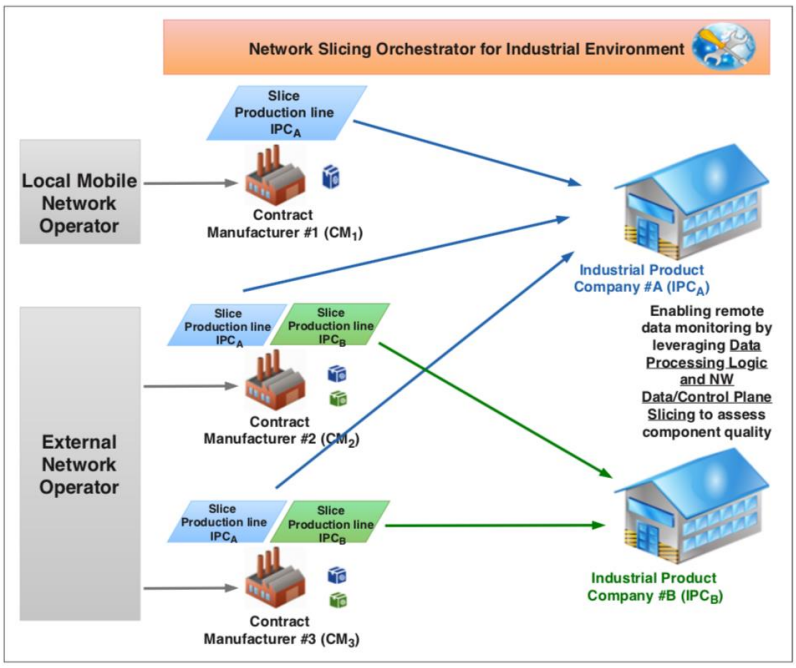


Figura 27. Network Slice para monitorización de la producción

En empresas multinacionales que tercerizan la producción de partes, es necesario monitorizar la calidad de las partes. Gracias a la programabilidad de la infraestructura en la Industria 4.0 y a soluciones network slicing es posible estar al tanto de la calidad del producto final en varias fábricas de manera simultánea y bajo una sola infraestructura de red y realizar modificaciones en tiempo real. Esto permite a las industrias una mayor flexibilidad en sus cadenas de producción y mayor control de sus distintas fábricas propias o subcontratadas a nivel mundial bajo una sola infraestructura.

Manufactura Inteligente

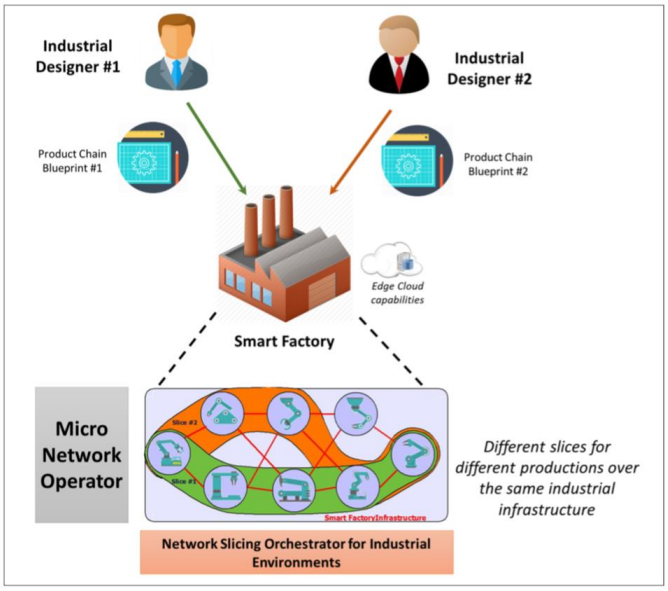


Figura 28. Network Slice para manufactura inteligente.

De la mano del caso anterior, las industrias pueden tercerizar el uso de su infraestructura, de manera similar a lo que hacen los proveedores cloud, permitiendo el uso bajo demanda de esta, mediante el uso de network slices.
Esto introduce un modelo conocido como Smart-Factory-as-a-Service, el cual permite rentar de manera total o parcial la infraestructura industrial en modo pago-por-uso.
Es posible utilizar la misma infraestructura de manera simultánea para la fabricación de diferentes partes solicitadas por diferentes clientes.

Mantenimiento Remoto

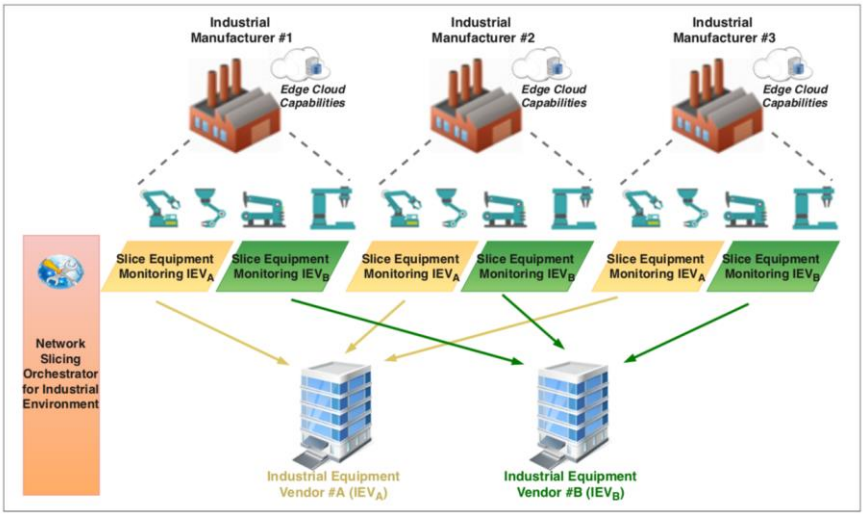


Figura 29. Network Slice para mantenimiento remoto.

Se pueden habilitar Slices para que el proveedor de equipos industriales A pueda monitorear sus equipos y proveer mantenimiento preventivo, a la vez que el proveedor B puede realizar sus tareas preventivas en otro Slice de manera independiente. Esta solución permite a los proveedores de equipos industriales estar en constante monitorización de los mismos, sin el temor de sus clientes a la exposición de información sensitiva. Para esto, la información de mantenimiento se procesa en el borde y se envía a los servidores del proveedor mediante un slice dedicado.

6 Escenario propuesto

Unificando diferentes propuestas de casos de uso en la Industria 4.0 en conjunto con las diferentes tecnologías que habilitan las redes móviles de nueva generación, se propone una arquitectura para el despliegue de una solución para el manejo de datos de dispositivos Industriales, como un servicio de red que a la vez puede ser comercializado por operadoras de telefonía como un producto Edge Computing, siendo desplegado como un servicio de red en el borde de la red, o de Network Slicing como un modelo Network Slicing as a Service (NSaaS), según el tipo de despliegue y las necesidades del cliente.

Esta solución puede ser utilizada en diferentes escenarios IoT, ya sea para administración y control de sensores como también para obtención de datos de los dispositivos IoT para análisis Big Data o Business Intelligence.

Al diseñar esta solución mediante un orquestador de funciones de red, es posible tener servicios personalizados y orquestarlos bajo demanda. Por ejemplo, se puede orquestar un servicio de red para un solo protocolo, con los servicios básicos de monitoreo o una solución más robusta, multiprotocolo, en conjunto con controladores virtualizados dedicados, integración con sistemas ERP para monitoreo y planificación.

Este escenario está pensado como parte de una etapa intermedia de adopción de la Industria 4.0, utilizando las facilidades de la virtualización de redes para poder desplegar de manera dinámica y bajo demanda diferentes servicios, mientras se logra una transición de tecnologías de acceso al medio. Por lo cual los dispositivos pueden conectarse tanto con protocolos propietarios industriales como con redes 5G.

Como se menciona en el capítulo 5 en la propuesta 5.1.3 IoT Gateway, se puede desplegar una pasarela IoT como una solución MEC para gestionar conexiones de diferentes dispositivos, con diferentes maneras de acceder al medio.

Esto lo hemos trasladado a una solución de pasarela de gestión de dispositivos industriales, que puede ser desplegada tanto en el borde como en a lo largo de la red de la operadora, como podemos ver en la Figura 30, dependiendo de los requerimientos.

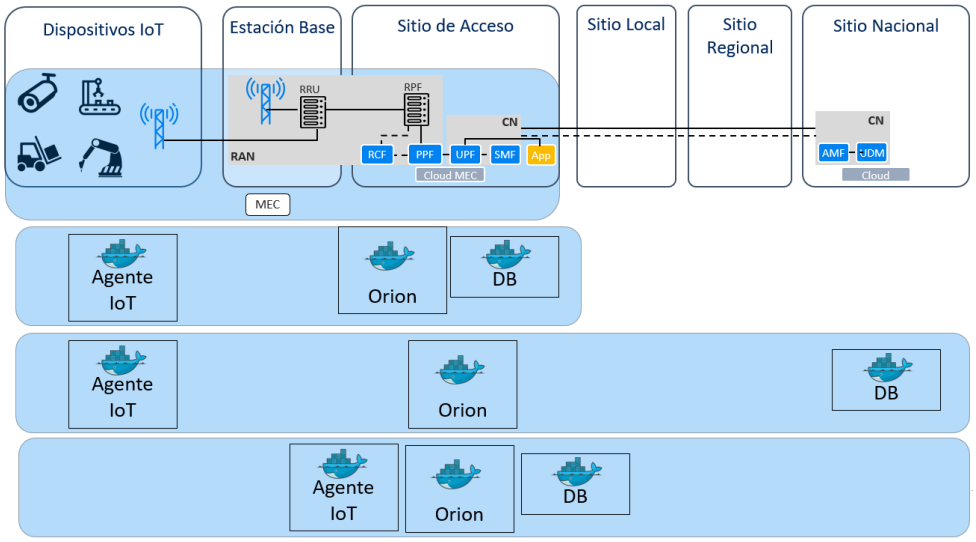


Figura 30. Despliegue de elementos

Para realizar una prueba de concepto hemos trasladado esta idea una plataforma de gestión y orquestación en la cual desplegamos microservicios como contenedores independientes para realizar la función de la pasarela de dispositivos industriales, manejando los datos con una plataforma opensource multiprotocolo, lo cual nos da la posibilidad de implementar esta solución en diferentes escenarios y poder desarrollar a

futuro nuevas implementaciones para nuevos protocolos de comunicación industriales de ser necesario. La idea es que esta solución sirva como herramienta intermedia entre el paso de protocolos propietarios a protocolos abiertos con la adopción de 5G como acceso al medio.

6.1 Plataforma FIWARE

FIWARE es una iniciativa open source que quiere definir un estándar para el manejo contextual de datos y así facilitar el desarrollo de Smart Solutions en diferentes ámbitos como: Smart Cities, Smart Factories, Smart Health. En el ámbito de IoT, esta iniciativa es permite capturar información procedente de redes de sensores, aunque se comunican usando diferentes protocolos y lenguajes, capaz de resolver la complejidad de tratar la información recogida por los sensores y traducirlos a un lenguaje común.
Describe cómo recopilar, gestionar y publicar información de contexto y adicionalmente aporta elementos que permiten explotar esta información una vez recopilada. A esto se le conoce como Context Broker [33], un elemento que maneja y comparte información en tiempo real, describiendo lo que está sucediendo en el momento a las diferentes partes involucradas.

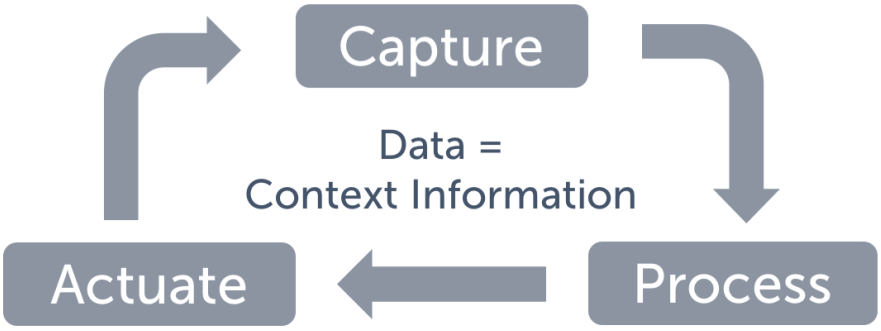


Figura 31. Información de Contexto

Para lograr esto propone el estándar FIWARE NGSI (Next Generation Service Interface) REST API. Orion es el Context Broker de la plataforma FIWARE implementando la API NGSI, con el cual es posible manejar diferentes elementos a través de peticiones y actualizaciones.

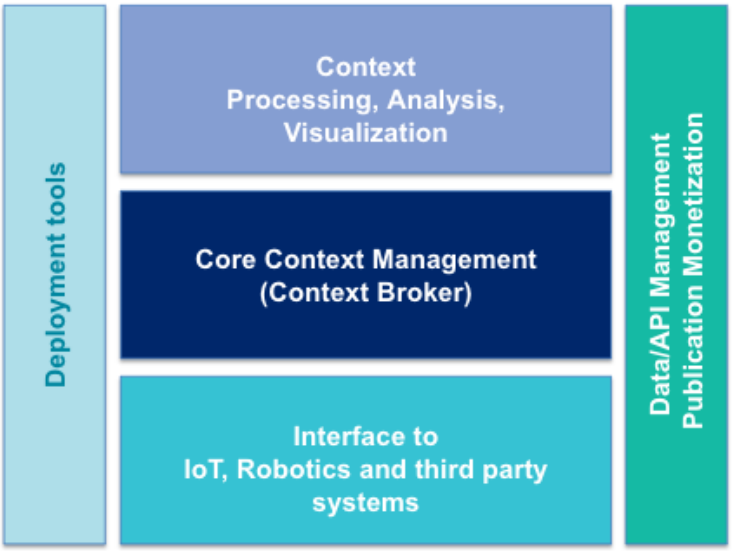


Figura 32. Bloques de la plataforma FIWARE

Otro de los bloques que forman parte de la plataforma FIWARE es el Agente IoT, el cual es la interfaz de comunicación entre el Context Broker y los dispositivos IoT. Estos agentes interactúan cada uno con un protocolo diferente, según el dispositivo con el que van a comunicar como, por ejemplo:

- IoT Agent para JSON: puente entre los protocolos HTTP/MQTT (con payload JSON) y NGSI
- IoT Agent para LWM2M: puente entre el protocolo Lightweight M2M y NGSI
- IoT Agent para Ultralight: puente entre los protocolos HTTP/MQTT (con payload UL2.0) y NGSI
- IoT Agent para LoRaWAN: puente entre el protocolo LoRaWAN y NGSI
- IoT Agent para OPC-UA: puente entre el protocolo OPC Unified Architecture y NGSI
- IoT Agent para Sigfox: puente entre el protocolo Sigfox y NGSI

Igualmente es posible crear un nuevo agente para trabajar con nuevos protocolos [34].

Dentro de este bloque, existe una incubadora de proyectos para la creación de nuevos agentes que soporten nuevos protocolos, en su mayoría dedicados a la comunicación con sistemas robóticos:

- Proyecto Fast RTPS: Ha sido adoptado como la solución middleware para sistemas ROS2(Robot Operating System).
- Micro XRCE-DDS: Solución middleware basada en un modelo publish-subscribe.
- OpenMTC: Implementación Open Source del standard OneM2M, para comunicación máquina a máquina. standard. Una interfaz northbound con Orion es implementada como parte del producto.

- Domibus: Proyecto que ayuda a usuarios en el intercambio de información electrónica y documentos de una manera confiable y segura.

El último de los bloques es el bloque de procesamiento, análisis y visualización de los datos. En este bloque hay diversos proyectos soportados oficialmente que se pueden implementar en una solución FIWARE como, por ejemplo:

- Wirecloud: plataforma web para generar dashboards personalizables. Generic Enabler brings a powerful web mashup platform making it easier to develop operational dashboards which are highly customizable by end users.
- Knowage: ponderosa plataforma de Business Intelligence para realizer tareas de analítica de negocio sobre sistemas tradicionales y fuentes big data.
- Kurento: procesamiento en tiempo real de flujos de datos, que soporta la transformación de cámaras de videos en sensors. También permite la aplicacion de diversas funciones como comunicacion audiovisual integrada, realidad aumentada y reproducción/grabación flexible de material multimedia.
- Cosmos: plataforma para realizar análisis Big Data en context. Se integra con las plataformas Big Data más populares.
- FogFlow: Framework de ejecución distribuida que soporta el procesamiento dinámico de flujos de datos tanto en la nube como en el borde de la red.

6.1.1 Prueba de Concepto de la plataforma

Para realizar pruebas con esta plataforma, familiarizarnos con sus diferentes componentes e investigar las posibilidades que presenta, seguimos algunos de los tutoriales que ofrecen su página web [35].
Implementar estos tutoriales es bastante sencillo, solo es necesario contar con Docker, como infraestructura básica y Docker Compose, para la ejecución de aplicaciones multicontenedor, instalados en una máquina corriendo cualquiera de los sistemas operativos más populares, Xubuntu en nuestro caso, por temas de consumo de recursos.
Estos tutoriales ofrecen información muy detallada y además los archivos necesarios para su ejecución se encuentran disponibles tanto en Docker Hub, para las imágenes, como en GitHub para instrucciones y scripts de implementación.

Pruebas con sensores:

FIWARE propone una serie de escenarios de prueba para interactuar con sensores como puertas automáticas, timbres, sensores de movimiento o lámparas inteligentes mediante peticiones REST usando curl o Postman. A pesar de que los escenarios están basados en sensores IoT más del mundo de la domótica, este concepto puede ser trasladado a otro tipo de dispositivos IoT, por lo cual son válidos en nuestra investigación.
Mediante la API Southbound se puede interactuar con estos sensores enviándole comandos para encender o apagarlos de manera remota.
De igual manera mediante la API Northbound es posible obtener mediciones de los sensores y saber su estado actual.

En este tutorial solo se simulan un conjunto de dispositivos IoT en un contenedor Docker y se utilizan comandos curl para simular un agente IoT.

El contenedor Docker envía las peticiones mediante al puerto 3001 y escucha las peticiones mediante el puerto 7896, que es un puerto común para Ultralight via HTTP.

```
tutorial:
    image: fiware/tutorials.context-provider
    hostname: iot-sensors
    container_name: fiware-tutorial
    networks:
        - default
    expose:
        - "3000"
        - "3001"
    ports:
        - "3000:3000"
        - "3001:3001"
    environment:
        - "DEBUG=tutorial:*"
        - "PORT=3000"
        - "IOTA_HTTP_HOST=iot-agent"
        - "IOTA_HTTP_PORT=7896"
        - "DUMMY_DEVICES_PORT=3001"
        - "DUMMY_DEVICES_API_KEY=4jggokgpepnvsb2uv4s40d59ov"
```

Plantilla Dockerfile Context Provider

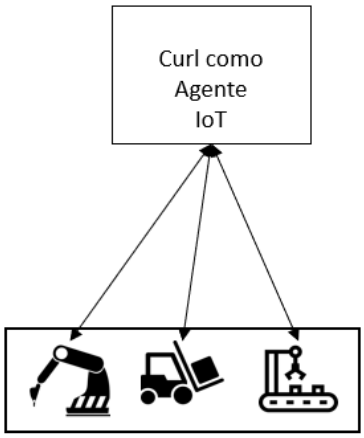


Figura 33. Arquitectura del escenario de prueba de sensores

Primero descargamos los archivos necesarios del git *tutorials.IoT-Sensors* [36], luego nos movemos a la carpeta que se descargó y ejecutamos los scripts necesarios (Figura 34).

```
openstack@openstack:~$ cd tutorials.IoT-Sensors
openstack@openstack:~/tutorials.IoT-Sensors$ ./services create; ./services start
;
Pulling Docker images
Pulling mongo-db  ... done
Pulling orion     ... done
Pulling tutorial  ... done
Pulling iot-agent ... done
Stopping containers
Starting one container Tutorial
- Tutorial acts as a series of dummy IoT Sensors over HTTP

Creating db-mongo        ... done
Creating fiware-tutorial ... done
Creating fiware-iot-agent ... done
Creating fiware-orion     ... done
⌛ Waiting for Orion to be available

⌛ Loading context data  done

Now open http://localhost:3000/device/monitor
```

Figura 34. Puesta en marcha del escenario

Luego de esto, tendremos disponible un portal web para visualizar el estado los dispositivos conectados, y las interacciones que se realizan.

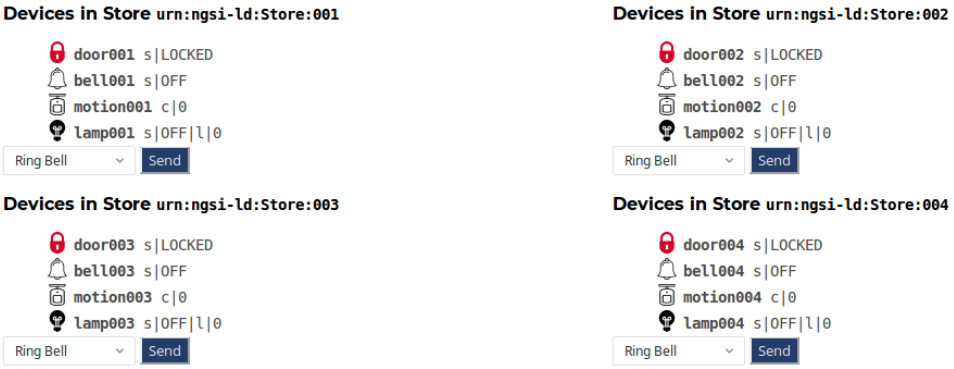


Figura 35. Monitor IoT

Ahora podemos interactuar con los sensores disponibles mediante peticiones REST utilizando comandos curl. Si quisiéramos encender la lámpara 001, enviamos la siguiente petición de encendido:

```
openstack@openstack:~/tutorials.IoT-Sensors$ curl -iX POST \
>   --url 'http://localhost:3001/iot/lamp001' \
>   --data urn:ngsi-ld:Lamp:001@on
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
Content-Length: 30
ETag: W/"1e-8jk2SMCcIha4GLXGj6Gs3YCzRDM"
Date: Fri, 24 May 2019 14:01:35 GMT
Connection: keep-alive

urn:ngsi-ld:Lamp:001@on| on OKopenstack@openstack:~/tutorials.IoT-Sensors$
```

Figura 36. Petición curl

Podemos confirmar en el monitor web, el estado de la lampara 001 y también el tráfico Northbound para confirmar que la petición fue realizada con éxito.

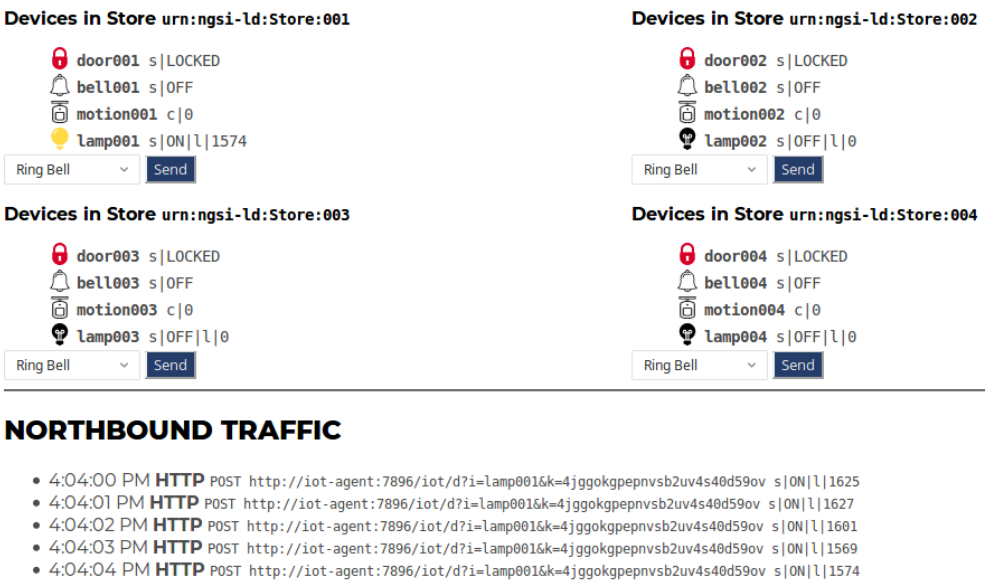


Figura 37. Cambio de estado y confirmación de tráfico

Con este tutoriales podemos ver la interacción vía API REST para interactuar con diferentes sensores IoT, esto puede ser llevado a diferentes escenarios, en nuestro caso dispositivos industriales, mediante el uso de los protocolos adecuados.

Una vez finalizadas las pruebas, terminamos los contenedores, para evitar problemas en las siguientes pruebas, como direcciones ip o puertos que quedan en uso.

```
Stopping containers
Stopping fiware-orion     ... done
Stopping fiware-iot-agent ... done
Stopping fiware-tutorial  ... done
Stopping db-mongo         ... done
Removing fiware-orion     ... done
Removing fiware-iot-agent ... done
Removing fiware-tutorial  ... done
Removing db-mongo         ... done
openstack@openstack:~/tutorials.IoT-Sensors$
```

Figura 38. Detener servicios

Pruebas con Agentes IoT:

Como se mencionó anteriormente, el Agente IoT es el componente que permite a los dispositivos IoT enviar sus datos y ser manejados por Orion, usando sus protocolos de comunicación nativos.

Para realizar estas pruebas, utilizaremos un agente para 3 protocolos diferentes: UltraLight, Mosquito y RTPS, con el fin de poder implementar una solución multiprotocolo IoT.

El ecosistema FIWARE tiene disponible una amplia librería de agentes para diversos protocolos, así como también las herramientas para desarrollar nuevos agentes para otros protocolos, de ser necesario.

Agente UltraLight 2.0 (UL 2.0)

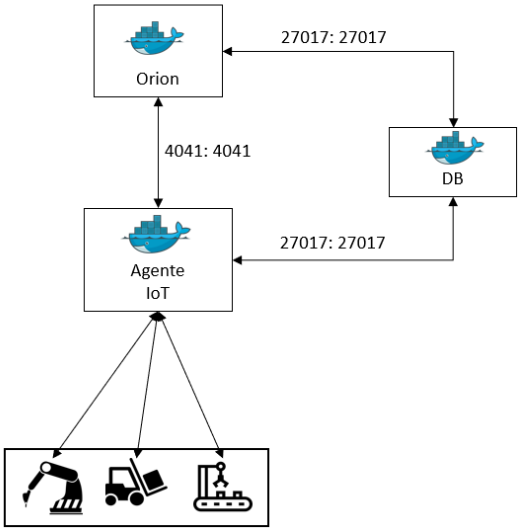


Figura 39. Arquitectura general para Agente IoT

Ultralight 2.0 es un protocolo ligero basado en texto dirigido a dispositivos y comunicaciones donde el ancho de banda y la memoria del dispositivo pueden ser recursos limitados [37]. Este agente de IoT puede ser utilizado con diferentes enlaces de protocolos de transporte para el mismo protocolo como HTTP o MQTT.
Este agente puede ser desplegado desde 0 o ejecutando un contenedor Docker, previamente configurado y disponible en el repositorio oficial de FIWARE [38]. Este es el agente IoT base que utilizaremos en las pruebas con protocolos de transporte pensados para comunicaciones industriales, en nuestro caso MQTT.

Agente UL 2.0 con MQTT

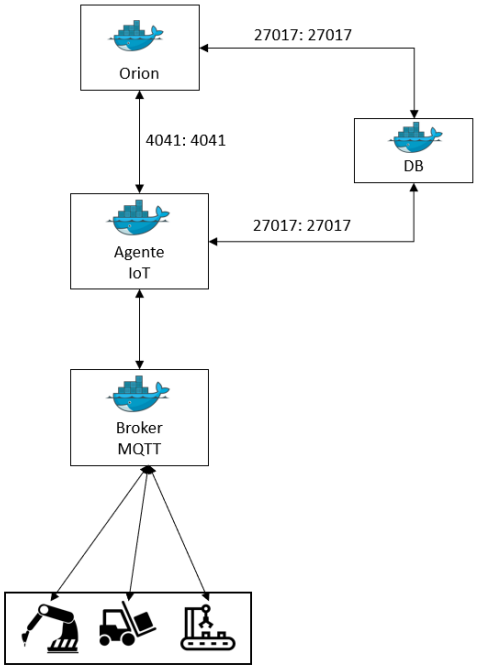


Figura 40. Arquitectura Agente IoT MQTT

Al utilizar un modelo publish-suscribe, es necesario realizar una suscripción entre los dispositivos involucrados, por lo cual se introduce un nuevo elemento en el escenario que proponemos, para gestionar estas suscripciones.
Eclipse Mosquitto es un bróker open source bajo licencia EPL/EDL, desarrollado por la fundación ECLIPSE, que implementa el protocolo de transporte MQTT en sus versiones 5.0, 3.1.1 y 3.1 [39].

Para iniciar las pruebas, obtenemos los archivos necesarios de github [40] y ejecutamos el script de inicio.

```
openstack@openstack:~/tutorials.IoT-over-MQTT$ ./services start
Force removal of mqtt-subscriber
mqtt-subscriber not running
Stopping containers
Starting five containers Orion, IoT-Agent, Tutorial, Mosquitto MQTT broker and a MongoDB database.
- Orion is the context broker
- IoT-Agent is configured for the UltraLight Protocol
- Tutorial acts as a series of dummy IoT Sensors over MQTT
- Mosquitto acts as an MQTT Message Broker

Creating db-mongo  ... done
Creating mosquitto    ... done
Creating fiware-orion     ... done
Creating fiware-iot-agent ... done
Creating fiware-tutorial  ... done
⌛ Waiting for Orion to be available

⌛ Loading context data ......... done


NAMES               STATUS                            PORTS
fiware-tutorial     Up 27 seconds (health: starting)  0.0.0.0:3000-3001->3000-3001/tcp
fiware-iot-agent    Up 29 seconds (health: starting)  0.0.0.0:4041->4041/tcp
fiware-orion        Up 31 seconds (healthy)           0.0.0.0:1026->1026/tcp
mosquitto           Up 33 seconds                     0.0.0.0:1883->1883/tcp, 0.0.0.0:9001->9001/tcp

Now open http://localhost:3000/device/monitor
```

Figura 41. Inicio de escenario MQTT

En este escenario tenemos nuevamente una plataforma para monitorear los dispositivos IoT Simulados y ver en tiempo real la transmisión de los mensajes mediante el protocolo MQTT.

Figura 42. Monitor escenario MQTT

Luego de esto, realizamos una prueba para confirmar si elemento bróker MQTT está funcionando correctamente. Para esto abrimos dos consolas, una la utilizaremos para recibir y visualizar los mensajes y la otra para enviar los mensajes.

```
openstack@openstack:~$ docker run -it --rm --name mqtt-subscriber \
>   --network fiware_default efrecon/mqtt-client sub -h mosquitto -t "/#"
HELLO WORLD
Prueba Escenario MQTT
```

Figura 43. Visualización de mensajes MQTT

```
openstack@openstack:~$ docker run -it --rm --name mqtt-publisher \
>   --network fiware_default efrecon/mqtt-client pub -h mosquitto -m "HELLO WORLD" -t "/test"
openstack@openstack:~$ docker run -it --rm --name mqtt-publisher   --network fiware_default efrecon/mqtt-client pub -h
 mosquitto -m "Prueba Escenario MQTT" -t "/test"
```

Figura 44. Envío de mensajes MQTT

Una vez finalizada esta prueba detenemos el contendor. Adicionalmente, podemos revisar el log del contenedor para ver el historial de comunicación

```
openstack@openstack:~$ docker logs --tail 10 mosquitto
1558873442: New client connected from 172.20.1.6 as mqttjs_cac296ef (c1, k60).
1558873810: New connection from 172.20.1.7 on port 1883.
1558873810: New client connected from 172.20.1.7 as mosqsub|1-30e8c87b1f1d (c1, k60).
1558873826: New connection from 172.20.1.8 on port 1883.
1558873826: New client connected from 172.20.1.8 as mosqpub|1-086a6caadc1a (c1, k60).
1558873826: Client mosqpub|1-086a6caadc1a disconnected.
1558873915: New connection from 172.20.1.8 on port 1883.
1558873915: New client connected from 172.20.1.8 as mosqpub|1-10a17f9477d6 (c1, k60).
1558873915: Client mosqpub|1-10a17f9477d6 disconnected.
1558874150: Socket error on client mosqsub|1-30e8c87b1f1d, disconnecting.
```

Figura 45. Log broker MQTT

Por último, revisamos el estado del Agente IoT mediante un GET:

```
openstack@openstack:~$ curl -X GET \
>   'http://localhost:4041/iot/about'
{"libVersion":"2.7.0","port":"4041","baseRoot":"/","version":"1.7.0"}
```

Figura 46. Estado agente IoT

El siguiente paso es realizar la conexión con los dispositivos IoT, en este caso al usar el protocolo MQTT hay que suscribir los dispositivos IoT a un canal de suscripción o topic como se conoce en el contexto del protocolo. Para esto se crea un grupo de servicio indicando el canal de suscripción con el cual se comunican los dispositivos.

Para realizar esta suscripción se envía un POST a **http://localhost:4041/iot/services** detallando el tipo de servicio y grupos de dispotivos con los headers `fiware-service` y `fiware-servicepath`. Igualmente se detalla el canal de suscripción en el campo apikey y la dirección del bróker en el campo `cbroker`.

```
openstack@openstack:~/tutorials.IoT-Sensors$ curl -iX POST \
>   'http://localhost:4041/iot/services' \
>   -H 'Content-Type: application/json' \
>   -H 'fiware-service: openiot' \
>   -H 'fiware-servicepath: /' \
>   -d '{
>  "services": [
>    {
>      "apikey":      "4jggokgpepnvsb2uv4s40d59ov",
>      "cbroker":     "http://orion:1026",
>      "entity_type": "Thing",
>      "resource":    ""
>    }
>  ]
> }'
HTTP/1.1 201 Created
X-Powered-By: Express
Fiware-Correlator: ba28c8cd-6f89-4f50-936c-71ece04572b7
Content-Type: application/json; charset=utf-8
Content-Length: 2
Date: Sun, 26 May 2019 13:10:41 GMT
Connection: keep-alive
```

Figura 47. Grupo de servicio

Luego de esto se crea un sensor dentro del catálogo mediante un POST a **http://localhost:4041/iot/devices**, en el cual se detallan id, nombre de entidad, protocolo IoT y protocolo de transporte, que en este caso es MQTT.

```
{}openstack@openstack:~/tutorials.IoT-Sensors$ curl -iX POST \
>   'http://localhost:4041/iot/devices' \
>   -H 'Content-Type: application/json' \
>   -H 'fiware-service: openiot' \
>   -H 'fiware-servicepath: /' \
>   -d '{
>  "devices": [
>    {
>      "device_id":   "motion001",
>      "entity_name": "urn:ngsi-ld:Motion:001",
>      "entity_type": "Motion",
>      "protocol":    "PDI-IoTA-UltraLight",
>      "transport":   "MQTT",
>      "timezone":    "Europe/Berlin",
>      "attributes": [
>        { "object_id": "c", "name": "count", "type": "Integer" }
>      ],
>      "static_attributes": [
>        { "name":"refStore", "type": "Relationship", "value": "urn:ngsi-ld:Store:001"}
>      ]
>    }
>  ]
> }
> '
HTTP/1.1 201 Created
X-Powered-By: Express
Fiware-Correlator: 0ef87472-fb7b-407b-a181-aa652388f96f
Content-Type: application/json; charset=utf-8
Content-Length: 2
Date: Sun, 26 May 2019 13:11:34 GMT
Connection: keep-alive
```

Figura 48. Aprovisionamiento de sensores

Ahora es posible enviar mensajes desde el Agente y podemos ver en el monitor web que se ha recibido el mensaje.

Figura 49. Recepción de mensajes MQTT

Una vez conectados el agente IoT y los dispositivos IoT es necesario conectar el broker para poder pasar los comandos desde capas superiores a través de Orion.
Para esto se realiza un POST a **http://localhost:1026/v2/registrations** con los comandos a registrar y el dispositivo al que corresponden en el cambo `attrs`:

```
openstack@openstack:~/tutorials.IoT-Sensors$ curl -iX POST \
>   'http://localhost:1026/v2/registrations' \
>   -H 'Content-Type: application/json' \
>   -H 'fiware-service: openiot' \
>   -H 'fiware-servicepath: /' \
>   -d '{
>   "description": "Door Commands",
>   "dataProvided": {
>     "entities": [
>       {
>         "id": "urn:ngsi-ld:Door:001", "type": "Door"
>       }
>     ],
>     "attrs": [ "lock", "unlock", "open", "close"]
>   },
>   "provider": {
>     "http": {"url": "http://orion:1026/v1"},
>     "legacyForwarding": true
>   }
> }'
HTTP/1.1 201 Created
Connection: Keep-Alive
Content-Length: 0
Location: /v2/registrations/5cea95fa1d8245497037c7b1
Fiware-Correlator: 0956a926-7fbb-11e9-b310-0242ac140104
Date: Sun, 26 May 2019 13:34:50 GMT
```

Figura 50. Registro Sensor MQTT

Una vez registrado podemos enviar los comando mediante un GET desde Orion y confirmar en el monitor web que se recibió el comando:

```
openstack@openstack:~/tutorials.IoT-Sensors$ curl -iX PATCH \
>   'http://localhost:1026/v2/entities/urn:ngsi-ld:Door:001/attrs' \
>   -H 'Content-Type: application/json' \
>   -H 'fiware-service: openiot' \
>   -H 'fiware-servicepath: /' \
>   -d '{
>   "open": {
>       "type" : "command",
>       "value" : ""
>   }
> }'
HTTP/1.1 204 No Content
Connection: Keep-Alive
Content-Length: 0
Fiware-Correlator: dbb33d44-7fbb-11e9-92dc-0242ac140104
Date: Sun, 26 May 2019 13:40:43 GMT

openstack@openstack:~/tutorials.IoT-Sensors$
```

Figura 51. Envío de comandos mediante Orion usando MQTT.

MQTT MESSAGES

- 3:41:54 PM **MQTT** mqtt://mosquitto/4jggokgpepnvsb2uv4s40d59ov/motion001/attrs c|1
- 3:41:54 PM **MQTT** mqtt://mosquitto/4jggokgpepnvsb2uv4s40d59ov/door001/attrs s|CLOSED
- 3:41:55 PM **MQTT** mqtt://mosquitto/4jggokgpepnvsb2uv4s40d59ov/motion001/attrs c|0
- 3:41:59 PM **MQTT** mqtt://mosquitto/4jggokgpepnvsb2uv4s40d59ov/door001/attrs s|OPEN
- 3:42:00 PM **MQTT** mqtt://mosquitto/4jggokgpepnvsb2uv4s40d59ov/motion001/attrs c|1

Figura 52. Mensajes MQTT en monitor web.

Es posible obtener los datos de todos los dispositivos registrados mediante un GET a **http://localhost:4041/iot/devices**

```
{}openstack@openstack:~/tutorials.IoT-Sensors$ curl -X GET \
>   'http://localhost:4041/iot/devices' \
>   -H 'fiware-service: openiot' \
>   -H 'fiware-servicepath: /'
{"count":4,"devices":[{"device_id":"motion001","service":"openiot","service_path":"/","entity_name":"urn:ngsi-ld:Motio
n:001","entity_type":"Motion","transport":"MQTT","attributes":[{"object_id":"c","name":"count","type":"Integer"}],"laz
y":[],"commands":[],"static_attributes":[{"value":"urn:ngsi-ld:Store:001","type":"Relationship","name":"refStore"}],"p
rotocol":"PDI-IoTA-UltraLight"},{"device_id":"bell001","service":"openiot","service_path":"/","entity_name":"urn:ngsi-
ld:Bell:001","entity_type":"Bell","transport":"MQTT","attributes":[],"lazy":[],"commands":[{"object_id":"ring","name":
"ring","type":"command"}],"static_attributes":[{"value":"urn:ngsi-ld:Store:001","type":"Relationship","name":"refStore
"}],"protocol":"PDI-IoTA-UltraLight"},{"device_id":"door001","service":"openiot","service_path":"/","entity_name":"urn
:ngsi-ld:Door:001","entity_type":"Door","transport":"MQTT","attributes":[{"object_id":"s","name":"state","type":"Text"
}],"lazy":[],"commands":[{"object_id":"unlock","name":"unlock","type":"command"},{"object_id":"open","name":"open","ty
pe":"command"},{"object_id":"close","name":"close","type":"command"},{"object_id":"lock","name":"lock","type":"command
"}],"static_attributes":[{"value":"urn:ngsi-ld:Store:001","type":"Relationship","name":"refStore"}],"protocol":"PDI-Io
TA-UltraLight"},{"device_id":"lamp001","service":"openiot","service_path":"/","entity_name":"urn:ngsi-ld:Lamp:001","en
tity_type":"Lamp","transport":"MQTT","attributes":[{"object_id":"s","name":"state","type":"Text"},{"object_id":"l","na
me":"luminosity","type":"Integer"}],"lazy":[],"commands":[{"object_id":"on","name":"on","type":"command"},{"object_id"
:"off","name":"off","type":"command"}],"static_attributes":[{"value":"urn:ngsi-ld:Store:001","type":"Relationship","na
me":"refStore"}],"protocol":"PDI-IoTA-UltraLight"}]}openstack@openstack:~/tutorials.IoT-Sensors$
```

Figura 53. Listado de dispositivos registrados.

Al finalizar la prueba detenemos los contenedores:

```
openstack@openstack:~/tutorials.IoT-over-MQTT$ ./services stop
Force removal of mqtt-subscriber
mqtt-subscriber not running
Stopping containers
Stopping fiware-tutorial  ... done
Stopping fiware-iot-agent ... done
Stopping fiware-orion     ... done
Stopping mosquitto        ... done
Stopping db-mongo         ... done
Removing fiware-tutorial  ... done
Removing fiware-iot-agent ... done
Removing fiware-orion     ... done
Removing mosquitto        ... done
Removing db-mongo         ... done
```

Figura 54. Fin prueba MQTT.

Agente RTPS

Para escenarios industriales la comunicación en tiempo real es primordial, por lo cual uno de los protocolos que decidimos utilizar para las pruebas es RTPS (Real Time Publish Subscribe) y FIWARE tiene documentación para realizar pruebas con dos implementaciones de este protocolo: Fast-RTPS y Micro-RTPS [41].

Fast-RTPS es una implementación de RTPS que provee comunicación publish-subscribe sobre protocolos de transporte como UDP. Es utilizado también para interoperar con el estándar DDS (Data Distribution Service).

Los contenedores permiten la suscripción y desuscripción para el envío y recepción de mensajes mediante el protocolo RTPS.

```
Starting
Publisher matched
Message: HelloWorld with index: 1 SENT
Message: HelloWorld with index: 2 SENT
Message: HelloWorld with index: 3 SENT
Message: HelloWorld with index: 4 SENT
Message: HelloWorld with index: 5 SENT
Message: HelloWorld with index: 6 SENT
Message: HelloWorld with index: 7 SENT
Message: HelloWorld with index: 8 SENT
Message: HelloWorld with index: 9 SENT
Message: HelloWorld with index: 10 SENT
```

```
Subscriber matched
Message HelloWorld 1 RECEIVED
Message HelloWorld 2 RECEIVED
Message HelloWorld 3 RECEIVED
Message HelloWorld 4 RECEIVED
Message HelloWorld 5 RECEIVED
Message HelloWorld 6 RECEIVED
Message HelloWorld 7 RECEIVED
Message HelloWorld 8 RECEIVED
Message HelloWorld 9 RECEIVED
Message HelloWorld 10 RECEIVED
Subscriber unmatched
```

Figura 55. Intercambio de mensajes FastRTPS.

6.2 Herramientas Utilizadas

En esta sección se describen las herramientas utilizadas durante el diseño del escenario sobre el cual se realizó este proyecto.

6.2.1 Docker

Docker es una plataforma de gestión de contenedores de código abierto creada en el año 2013 por dotCloud que se ha convertido en un entorno que permite a desarrolladores la creación de aplicaciones y servicios que funcionan de manera autosuficiente.
A diferencia de LXC, que crea contenedores de sistema operativo, Docker gestiona principalmente solo contenedores de aplicación. Incluso si LXC y otras herramientas de gestión son soportadas por Docker gestión desde la versión 0.9, Docker introduce libcontainer, que, como LXC, es un *wrapper* de cgroups y namespaces Linux [42]. Inicialmente Docker era una extensión de una tecnología de la empresa dotCloud, para ejecutar su negocio en la nube en miles de servidores. Docker permite desarrollar, enviar y ejecutar aplicaciones, proporciona la facilidad de separar las aplicaciones de la infraestructura para así poder entregar software de una forma más rápida. provechando las metodologías de Docker para enviar, probar e implementar código rápidamente, se puede reducir significativamente el retraso entre código y ejecutarlo en un entorno de producción. Docker provee la capacidad de empaquetar y ejecutar una aplicación en un entorno aislado llamado contenedor. La seguridad y el aislamiento permiten ejecutar muchos contenedores simultáneamente en un host determinado. Por lo general los contenedores son ligeros y podrían no necesitar la carga adicional de un hipervisor como es el caso de las máquinas virtuales, en este caso se ejecuta directamente dentro del núcleo de la maquina host. Lo que significa que se puede ejecutar más contenedores en una combinación de hardware dada. Incluso se puede ejecutar contendores Docker dentro de máquinas host que en realidad son máquinas virtuales.

- Ligereza: Los procesos de Docker son más ligeros debido a que no se emula o virtualiza una máquina y su sistema operativo, como sucede en el caso de las máquinas virtuales. No es necesario el sistema de archivos completo, es por eso por lo que se puede aprovechar de mejor manera el hardware ya que aumenta la cantidad de servicios que se pueden tener en una maquina utilizando una fracción del espacio de almacenamiento.
- Portabilidad: Docker es manejado por imágenes que pueden ser desplegadas en cualquier entorno o sistema que soporte esta tecnología, como un contenedor. De esta manera no es necesario volver a instalar y configurar todas las aplicaciones que se utilizan.
- Seguridad: El aislamiento es fundamental en esta plataforma ya que las aplicaciones entre si no tiene contacto, de igual sucede con las infraestructuras subyacente. Docker proporciona aislamiento por defecto lo que significa una ventaja cuando se presenta un problema en alguna aplicación esto proporciona que se limite a un único contenedor en lugar de toda la máquina.
- Autosuficiencia: Los contenedores Docker son autosuficientes ya que solo se necesita de la imagen del contenedor para poder desplegar los servicios que este contiene.

Docker tiene 3 elementos principales:

- Imágenes: Una imagen Docker es un archivo binario que incluye todos los requisitos para ejecutar un único contenedor Docker, así como metadatos que describen sus necesidades y capacidades. Los contenedores Docker solo tienen acceso a los recursos definidos en la imagen, a menos que otorgue al contenedor acceso adicional al crearlo.

  Si se desea modificar el contenido de una imagen, la única opción que Docker permite es añadir otra capa con los nuevos cambios. La arquitectura de Docker aprovecha de una manera efectiva este concepto de capas para añadir perfectamente capacidades adicionales a las imágenes existentes para satisfacer las diferentes necesidades del negocio y aumentar la reutilización de imágenes.

  Lo que significa es que se pueden agregar capacidades a las imágenes existentes agregando capas adicionales encima de la imagen y derivando una nueva imagen. Docker maneja la relación padre e hijo. Se llama a una imagen base que no tiene ningún padre. Al tener en cuenta que siempre se empieza con una imagen base y cada adición que se realiza a la imagen base original se almacena en una capa separada como una imagen que hace referencia a otra imagen en este caso sería un padre.
- Contenedor: Un contenedor es una instancia ejecutable de una imagen, se puede crear, ejecutar, detener o eliminar un contenedor mediante un API o mediante CLI, además se pueden conectar un contenedor a una o más redes internas o externas, agregar almacenamiento o incluso crear una nueva imagen basada en su estado actual.
- Docker Hub: El repositorio oficial de imágenes, gestionado por Docker que se encarga de almacenar y velar por la integridad de las imágenes.

6.2.2 Open Source MANO (OSM)

OSM es un proyecto alojado por la ETSI para desarrollar una pila de software Open Source NFV que se encargue de la gestión y orquestación alineada con la especificación ETSI NFV. Este proyecto es desarrollado por una comunidad impulsada por la industria de las Telecomunicaciones con el fin de satisfacer los requisitos de orquestación de las redes NFV con soporte a los modelos de información. Los miembros fundadores de OSM incluyen 25 participantes entre los cuales están A10 Networks, Amdocs Ltd., Canonical USA Inc., Cisco Systems, EANTC, Ensemble (ADVA Optical Networking), Ericsson LM, F5 Networks, Fortinet, Fraunhofer FOKUS, Hewlett-Packard Enterprise, Huawei Technologies Co. Ltd, Ixia, Keynetic Technologies, Mahindra, Mirantis Netrounds, Openet, Palo Alto Networks, Red Hat Limited, RIFT.io, Telefónica.

El proyecto OSM cubre tanto la orquestación de recursos como la orquestación de servicios para permitir el despliegue automático y la interconexión de todos los componentes, tanto de los escenarios NFV como de la gestión de los ciclos de vida de los servicios de red.

Este proyecto no aparece de la nada y se basa en 3 proyectos preexistentes:

- OpenMANO, como orquestador de recursos base.
- Riftware, como orquestador de servicios base.
- Juju, como referencia externa para configuración y administración de VNF.

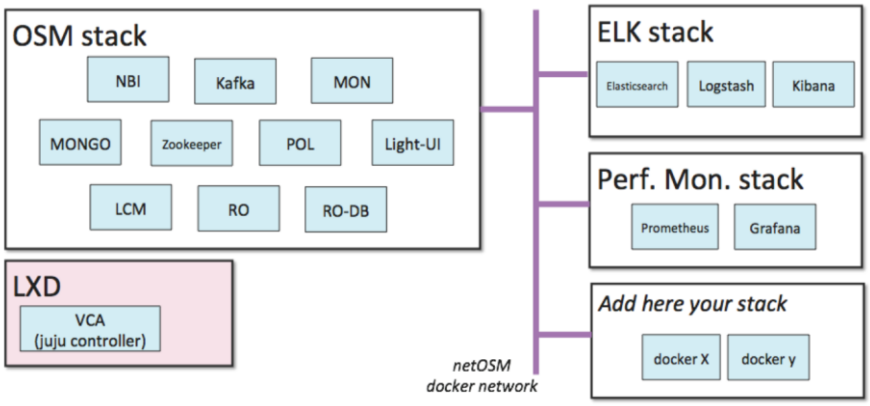


Figura 56. Arquitectura OSM

La arquitectura de OSM está compuesta de 2 componentes principales (Stack OSM y VCA) y 2 componentes opcionales (Stack ELK y Stack monitorización y desempeño), con la flexibilidad de agregar componentes adicionales según sea necesario como podemos ver en la Figura 56. Estos componentes se despliegan como contenedores Docker a excepción de VCA que se despliega como un contenedor LXD:

- Stack OSM:
    - NBI (Northbound Interface):

      Permite el funcionamiento completo del sistema y los servicios de red y segmentos de red bajo su control. El NBI de OSM ofrece el servicio de administración del ciclo de vida de los Servicios de red (NS) y las Instancias de segmentos de red (NSI), proporcionando como servicio

todas las abstracciones necesarias para permitir el control completo, la operación y la supervisión del ciclo de vida del NS/NSI mediante Sistemas cliente, evitando la exposición de detalles innecesarios de sus elementos constituyentes.

- LCM (Life Cycle Management):
  Gestión del ciclo de vida de las VNFs, servicios de red y Network Slice.
- RO (Resource Orchestrator):
  Este servicio lo desarrolla Telefónica, ésta capa es la responsable de procesar los requerimientos de localización de las instancias VNF y direccionar al VIM a una ubicación apropiada de computación, red y almacenamiento, para desplegar VNFs con sus interconexiones. RO conoce el nivel de uso, estado, estadísticas y resto de parámetros asociados a cada uno de los elementos. Los componentes SO y RO se pueden correlacionar juntamente con la entidad NFVO en la arquitectura ETSI MANO.
- MON:
  Recolecta métricas de las VNF, previamente definidas en los descriptores, mediante VCA, y métricas de los NFVI disponibles mediante los sistemas de telemetría del VIM.
- POL:
  Crea alarmas a través de MON.

- Stack ELK:
  - Elasticsearch:
    Motor de búsqueda escalable y base de datos de eventos.
  - Beats:
    Parte de los 'beats' de Elastic, que evolucionan el antiguo componente Logstash para proporcionar registros genéricos y recopilación de métricas, respectivamente.
  - Kibana:
    Herramienta gráfica para explorar todos los eventos recopilados y generar vistas y paneles personalizados.

- Stack Monitorización y Desempeño:
  - Prometheus:
    Almacenamiento de métricas recogidas y expuestas por MON en una TSDB (Time Series DataBase).
  - Grafana:

Interfaz de analítica que permite consultar, visualizar y generar alerta de las métricas almacenadas en Prometheus.

- VNF Configuration and Abstraction (VCA):
  Este servicio lo provee Canonical (Juju). Realiza la configuración inicial de VNF utilizando las funcionalidades de Juju. Teniendo en cuenta este propósito, el módulo VCA puede considerarse como un VNFM genérico con un conjunto de características limitado. Esta capa ofrece la gestión de la funcionalidad de VNF y es la responsable de los atributos de configuración de procesamiento.

En su versión 5, tiene las siguientes novedades:

- Network Slicing para 5G.
- Soporte para funciones de red físicas e híbridas.
- Orquestación multidominio
- Mejoras en el monitoreo y las políticas del framework para operaciones de ciclo cerrado.
- Nuevas capacidades para la configuración y despliegue de VNF.
- Mejoras en la experiencia de usuario.

OSM simplifica enormemente el proceso de gestión de VNF, permitiendo que las operaciones puedan ser automatizadas más allá de una simple instanciación.

6.2.3 Vim-emu

Vim-emu [43] es una plataforma de emulación NFV multi-PoP que nace como parte del proyecto SONATA que es a su vez parte del megaproyecto europeo Horizonte 2020. Permite ejecutar funciones de red virtualizadas como contenedores Docker, en topologías de red simuladas vía mininet. Se tiene acceso a APIs similares a las de Openstack, para lograr la comunicación con OSM.
Está basado en la referencia NFV de ETSI, utiliza la herramienta Containernet para extender el framework de emulación mininet y permitir el uso de contenedores Docker como dentro de la red emulada. En el Anexo 2, podemos ver más información de cómo utilizar esta herramienta.

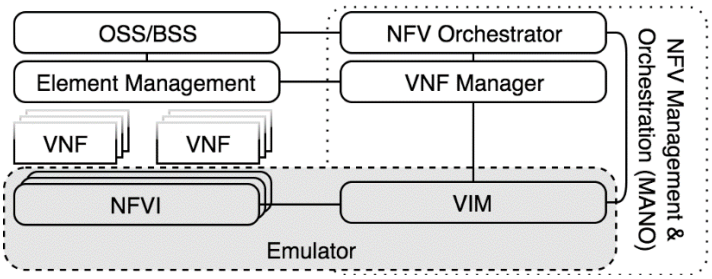


Figura 57. Nivel de emulación de vim-emu [44]

6.3 Diseño del Escenario

En esta sección se describen la topología utilizada, los componentes del diseño y los pasos a seguir para la implementación en las herramientas de prueba.
El escenario consiste en el despliegue una solución FIWARE multiprotocolo como un servicio de red ofertado por una operadora telefónica. Esta solución puede ser desplegada tanto en una arquitectura Edge como desde el Core de la operadora, según las necesidades.
Los dispositivos se comunican con un agente IoT, mediante un enlace radio 5G, LoRaWAN, SigFox, entre otros, según el tipo de dispositivo y el protocolo de comunicación que utilicen, y este a su vez se comunica con Orion, el cual se encarga de transmitir la información a plataformas externas para su monitoreo, análisis o visualización. Adicionalmente se puede añadir al servicio un controlador virtualizado, tanto en la infraestructura del operador, como en la infraestructura propia del cliente.

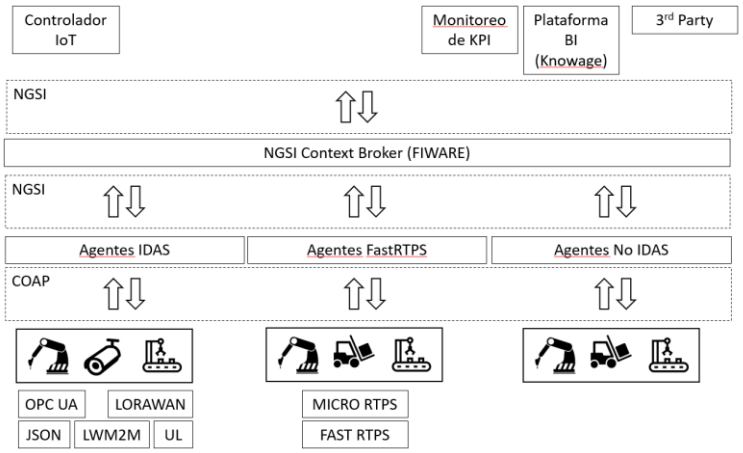


Figura 58. Comunicación entre elementos

Esta arquitectura consiste de 3 elementos básicos, para garantizar la comunicación inicial hacia los dispositivos IoT:

- Orion:
  Recibe las peticiones del Agente IoT mediante la API NGSI y se comunica con la capa de administración
- Agente IoT:
  Se comunica con los dispositivos IoT según el protocolo utilizado y envía las peticiones tanto al Broker como a los dispositivos IoT. Recibe las peticiones Southbound mediante NGSI y las convierte a cualquiera de los protocolos utilizado por los dispositvos para enviar los comandos.
- Base de Datos:
  En este caso una base de datos MongoDB, es utilizada por Orion para guardar información de entidades, suscripciones o registros y también es utilizada por el Agente IoT para guardar direcciones URL o llaves de autenticación.

Para lograr comunicación con plataformas externas, es necesario agregar un cuarto elemento que funciona como proxy.

- Proxy NGSI:
  Recibe peticiones de las capas superiores, cada una con sus propias APIs y formatos propietarios y transmite estas peticiones al Context Broker mediante NGSI.

Por último, dependiendo del tipo de protocolo de comunicación puede ser necesario un quinto elemento, un Broker entre los dispositivos IoT y el Agente IoT:

- Broker (Según Protocolo):
  Este elemento no es necesario en todos los escenarios. Se encarga de hacer publicar los mensajes, si se utiliza un protocolo publicar-suscribir, como MQTT.

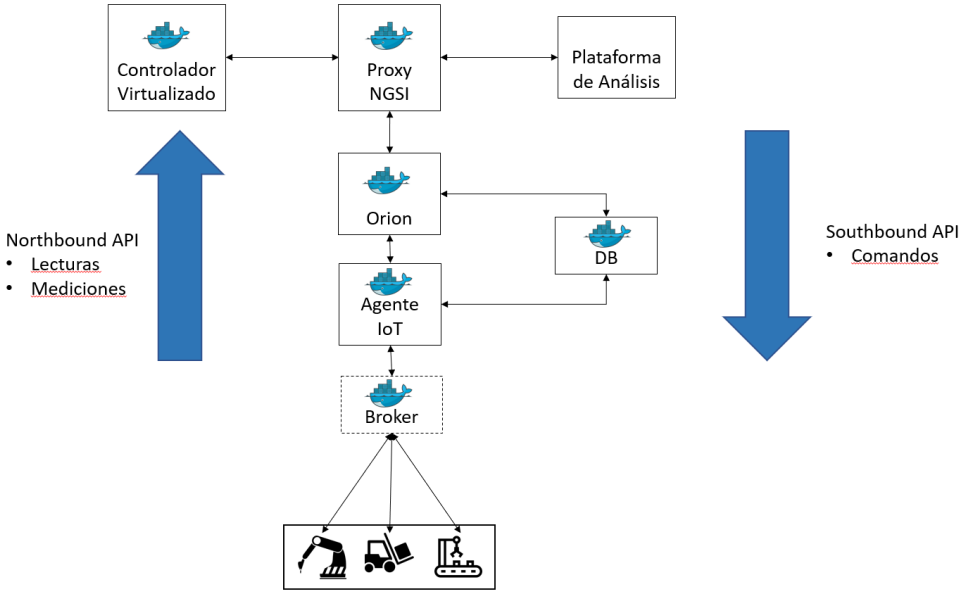


Figura 59. Comunicación entre contenedores

Este escenario se diseña pensando en utilizar como MANO Open Source MANO (OSM) versión 5.0, debido a que en esta versión se incluye la posibilidad de desplegar Network Slices, integrado con un VIM, que en este caso se elige, vim-emu. Inicialmente se intentó conectar OSM con Openstack versión Queens, pero se encontraron inconvenientes de comunicación en el entorno simulado, por lo cual continuamos la implementación con vim-emu que nos resultó favorable debido a la implementación basada en Docker y a la fácil gestión de VNFs como contenedores.

6.4 Prueba de Concepto

Para el despliegue de este servicio se ejecutan cada uno de los elementos de la plataforma FIWARE como un contendor Docker. La plataforma MANO, tiene acceso a estos contenedores para orquestar el servicio de red.

6.4.1 Generación de Imágenes

Se crea un archivo Dockerfile para generar la imagen base con los requerimientos del context broker, el agente IoT y la base de datos, en el cual se detallan la red en la que se comunican los contenedores, los puertos expuestos y los volúmenes de almacenamiento.

A continuación, tenemos un ejemplo de Dockerfile, para generar la nueva imagen del Context Broker en su versión 2.2.0:

```
FROM fiware/orion:2.2.0

RUN -dbhost mongo-db -logLevel DEBUG

ADD start.sh start.sh
RUN chmod +x start.sh
ADD ipconfig.sh ipconfig.sh
RUN chmod +x ipconfig.sh

WORKDIR /


EXPOSE 1026

# normal entry point
ENV FIWARE_CMD ./start.sh
CMD /bin/bash
```

Plantilla Dockerfile Orion 2.2.0

Una vez generadas las imágenes base, en el escenario de pruebas que estamos ejecutando, es necesario lanzar los contenedores en el VIM para que OSM pueda hacer llamadas a los contenedores a la hora de desplegar el servicio de red.

6.4.2 Orquestación en OSM

Una vez generadas las imágenes base, en el escenario de pruebas que estamos ejecutando, es necesario lanzar los contenedores en el VIM para que OSM pueda hacer las llamadas a la hora de desplegar el servicio de red.

Para lograr esta orquestación es necesario generar el catálogo con las VNFs necesarias, para luego poder desplegar el servicio de red bajo demanda.

VNF Descriptor (VNFD):

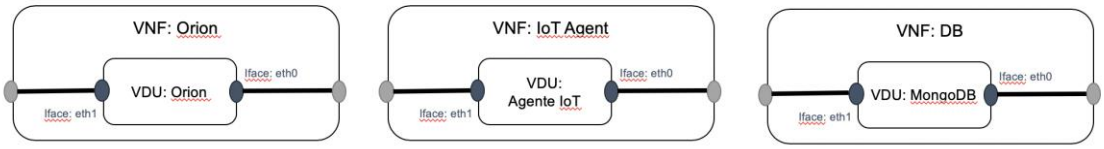


Figura 60. Diagramas VNFD

Se genera un archivo YAML con un puerto de datos y un puerto de administración. Se usa como base de la VNF el contenedor anteriormente añadido al VIM.
Antes de escribir el archivo YAML, para confirmar los detalles, se puede diseñar la VNF de manera visual con un diagrama de referencia, como el que se muestra en la Figura 60, o rellenando una tabla (Tabla 4), como sugiere la comunidad OSM. En este diseño las VNFs de Orion y la base de datos son únicas, pero la VNF del Agente IoT puede variar según los diferentes protocolos disponibles.

Nombre VNF	Nombre VDU	Nombre de la Imagen	Flavor		
			vCPU	vMem	vDisk
orion	orion-vdu	orion-cont	1	512 Mb	5 GB
agent	agent-vdu	agent-cont	1	512 Mb	5 GB
mongo	mongo-vdu	mongo-cont	1	512 Mb	5 GB

Tabla 4. Diseño VNF

A continuación, tenemos plantillas de referencia para el despliegue de cada una de las VNFs, en el caso de Agente IoT la imagen base, agent-contX, varía según el protocolo utilizado por los dispositivos IoT a conectar.

```
vnfd-catalog:
    vnfd:
    -   connection-point:
        -   name: mgmt
            type: VPORT
        -   name: data
            type: VPORT
        description: IoT Agent VNF (Protocolo)
        id: agent-vnf
        mgmt-interface:
            cp: mgmt
        name: agent-vnf
        short-name: agent-vnf
        vdu:
        -   id: agent-vnf
            image: agent-contX #nombre de la imagen según el protocolo IoT
            interface:
            -   external-connection-point-ref: mgmt
                name: agent-vnf-mgmt
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
            -   external-connection-point-ref: data
                name: agent-vnf-data
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
            name: agent-vnf
            vm-flavor:
                memory-mb: 512
                storage-gb: 5
                vcpu-count: 1
        version: 1.0
```

Plantilla VNFD agente-iot

```
vnfd-catalog:
    vnfd:
    -   connection-point:
```

```
        -   name: mgmt
            type: VPORT
        -   name: data
            type: VPORT
        description: mongoDB VNF
        id: mongo-vnf
        mgmt-interface:
            cp: mgmt
        name: mongo-vnf
        short-name: mongo-vnf
        vdu:
        -   id: mongo-vnf
            image: mongo-cont
            interface:
            -   external-connection-point-ref: mgmt
                name: mongo-vnf-mgmt
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
            -   external-connection-point-ref: data
                name: mongo-vnf-data
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
            name: mongo-vnf
            vm-flavor:
                memory-mb: 512
                storage-gb: 5
                vcpu-count: 1
        version: 1.0
```

Plantilla VNFD Base de Datos

```
vnfd-catalog:
    vnfd:
    -   connection-point:
        -   name: mgmt
            type: VPORT
        -   name: data
            type: VPORT
        description: Orion Context Broker VNF
        id: orion-vnf
        mgmt-interface:
            cp: mgmt
        name: orion-vnf
        short-name: orion-vnf
        vdu:
        -   id: orion-vnf
            image: orion-cont
            interface:
            -   external-connection-point-ref: mgmt
                name: orion-vnf-mgmt
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
            -   external-connection-point-ref: data
                name: orion-vnf-data
                type: EXTERNAL
                virtual-interface:
                    type: VIRTIO
```

```
            name: orion-vnf
            vm-flavor:
                memory-mb: 512
                storage-gb: 5
                vcpu-count: 1
        version: 1.0
```

Plantilla VNFD Context Broker

Una vez validado el descriptor en la plataforma, debemos recibir un mensaje de éxito (Figura 61), y podemos tener una vista gráfica de la topología (Figura 62).

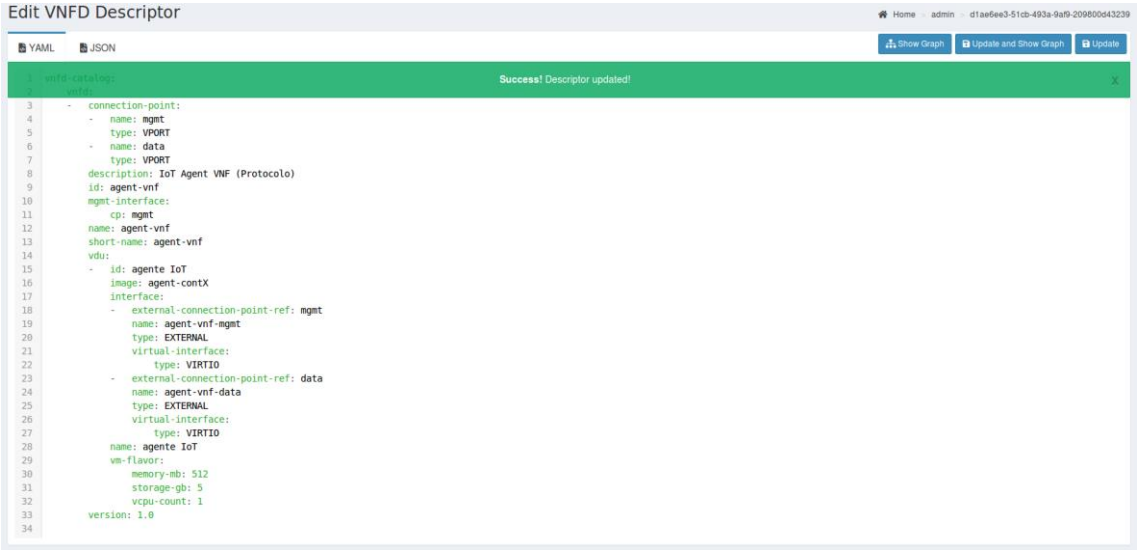


Figura 61. Validación YAML VNFD

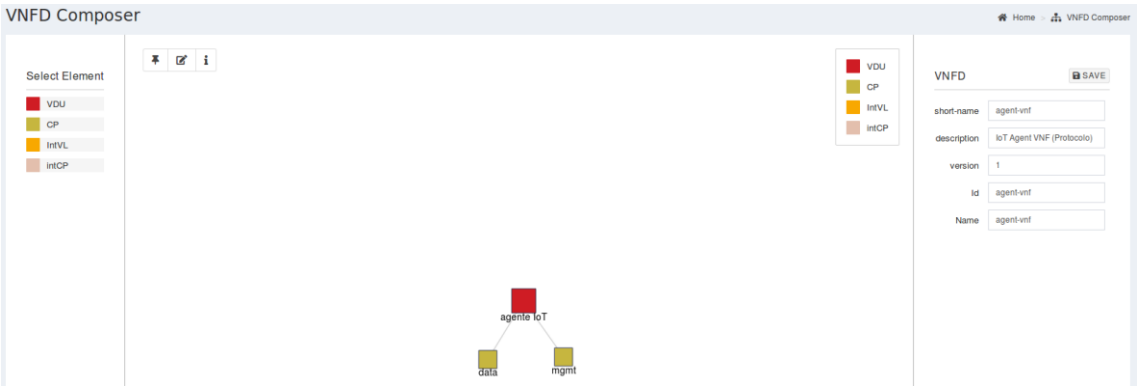


Figura 62. Topología VNF

Network Service Descriptor (NSD):

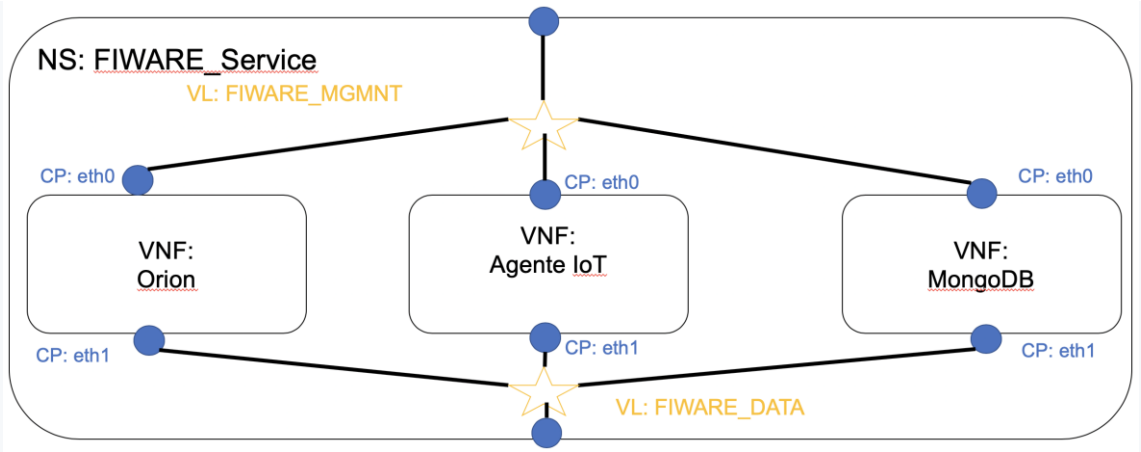


Figura 63. Diagrama NSD

Una vez creado el catálogo de VNFs, podemos agregar el descriptor del servicio de red, que hace referencia a las VNFs agregadas al catálogo.
Para ayudar con el diseño del descriptor, podemos usar el diagrama de referencia mostrado en la Figura 61, en el cual podemos ver que se utilizan 3 VNFs, una por cada uno de los servicios utilizados en la plataforma FIWARE
Éstas VNFs están conectadas a una red de datos para transmitir la información de los sensores y a una red de administración para ejecutar tareas de mantenimiento o modificaciones a los servicios.

A continuación, tenemos el código base que utilizamos para describir el servicio de red:

```
nsd:nsd-catalog:
    nsd:
    -   connection-point:
        -   name: FIWARE_ns_nsd_vld_mgmt
            vld-id-ref: FIWARE_ns_nsd_vld_mgmt
        -   name: FIWARE_ns_nsd_vld_data
            vld-id-ref: FIWARE_ns_nsd_vld_data
        constituent-vnfd:
        -   member-vnf-index: 1
            vnfd-id-ref: agent-vnf
        -   member-vnf-index: 2
            vnfd-id-ref: orion-vnf
        -   member-vnf-index: 3
            vnfd-id-ref: mongo-vnf
        description: FIWARE Network Service
        id: FIWARE_ns_nsd
        name: FIWARE_ns_nsd
        short-name: FIWARE_ns_nsd
        version: '1.0'
        vld:
        -   id: FIWARE_ns_nsd_vld_mgmt
            mgmt-network: 'true'
            name: management
            short-name: management
            type: ELAN
            vnfd-connection-point-ref:
            -   member-vnf-index-ref: 1
                vnfd-connection-point-ref: mgmt
```

```
            vnfd-id-ref: agent-vnf
        -   member-vnf-index-ref: 2
            vnfd-connection-point-ref: mgmt
            vnfd-id-ref: orion-vnf
        -   member-vnf-index-ref: 3
            vnfd-connection-point-ref: mgmt
            vnfd-id-ref: mongo-vnf
    -   id: FIWARE_ns_nsd_vld_data
        name: FIWARE_ns_nsd_vld_data
        short-name: FIWARE_ns_nsd_vld_data
        type: ELAN
        vnfd-connection-point-ref:
        -   member-vnf-index-ref: 1
            vnfd-connection-point-ref: data
            vnfd-id-ref: agent-vnf
        -   member-vnf-index-ref: 2
            vnfd-connection-point-ref: data
            vnfd-id-ref: orion-vnf
        -   member-vnf-index-ref: 3
            vnfd-connection-point-ref: data
               vnfd-id-ref: mongo-vnf
```

Plantilla NSD

Una vez validado el código, podemos observar la topología que tiene nuestro servicio de red. (Figura 64)

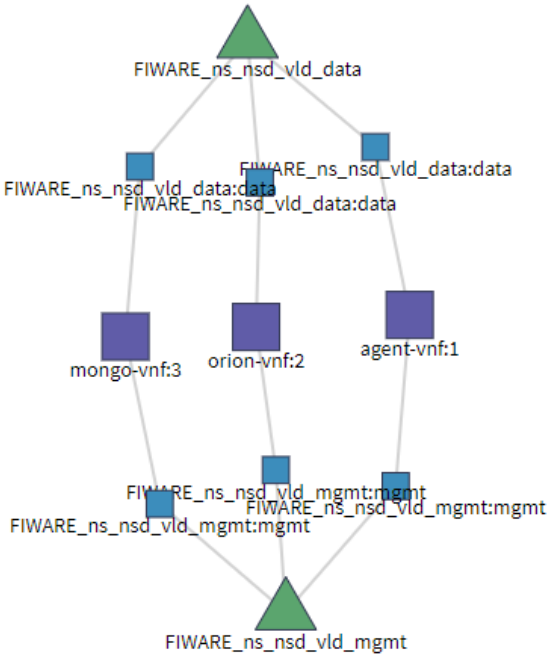


Figura 64. Topologías del Servicio de Red

Luego de esto podemos instanciar el servicio de red (Figura 65):

Name	Identifier	Nsd name	Operational Status	Config Status	Detailed Status	Actions
FIWARE	80308c38-db33-4139-b896-ab3f263f1fc7	FIWARE_ns_nsd	init	init	Creating nsd=FIWARE_ns_nsd at RO	Actions

Figura 65. Orquestación del servicio de red.

Una vez finalizado el proceso tendremos el servicio de red listo para ser utilizado a la hora de instanciar un Network Slice (Figura 66):

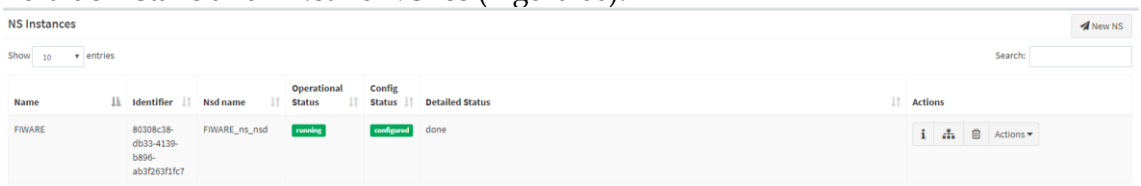


Figura 66. Servicio de red orquestado

Con el servicio de red instanciado, podemos realizar varias pruebas en este escenario de pruebas, por ejemplo, podemos validar tanto la distribución de contenedores en los Datacenters (Figura 67). En este caso se encuentran todos en el Datacenter 1, debido a que OSM removió el soporte para indicar los puntos de presencia en los cuales desplegar los recursos. Esto lo podemos revisar con el siguiente comando:

```
sudo docker exec vim-emu vim-emu datacenter list
```

```
test@ventas:/$ sudo docker exec vim-emu vim-emu datacenter list
+---------+-----------------+----------+----------------+--------------------+
| Label   | Internal Name   | Switch   |   # Containers |   # Metadata Items |
+=========+=================+==========+================+====================+
| dc2     | dc2             | dc2.s1   |              0 |                  0 |
+---------+-----------------+----------+----------------+--------------------+
| dc1     | dc1             | dc1.s1   |              6 |                  0 |
+---------+-----------------+----------+----------------+--------------------+
```

Figura 67. Lista de Datacenters.

También podemos confirmar los contenedores que han sido desplegando revisando los nodos de computación en funcionamiento con el siguiente comando:

```
sudo docker exec vim-emu vim-emu compute list
```

```
test@ventas:/$ sudo docker exec vim-emu vim-emu compute list
+--------------+-------------------------+-------------------------+-----------------------+--------------------------+
| Datacenter   | Container               | Image                   | Interface list        | Datacenter interfaces    |
+==============+=========================+=========================+=======================+==========================+
| dc1          | dc1_FIWARE-1-agent-vnf-1 | ubuntu:trusty           | agent-vnf-0,agent-vnf-1 | dc1.s1-eth2,dc1.s1-eth3  |
+--------------+-------------------------+-------------------------+-----------------------+--------------------------+
| dc1          | dc1_1-2-orion-vnf-1     | fiware/orion:2.2.0      | orion-vnf-2,orion-vnf-3 | dc1.s1-eth10,dc1.s1-eth11 |
+--------------+-------------------------+-------------------------+-----------------------+--------------------------+
| dc1          | dc1_1-1-agent-vnf-1     | fiware/iotagent-ul:1.8.0 | agent-vnf-2,agent-vnf-3 | dc1.s1-eth8,dc1.s1-eth9  |
+--------------+-------------------------+-------------------------+-----------------------+--------------------------+
| dc1          | dc1_FIWARE-2-orion-vnf-1 | ubuntu:trusty           | orion-vnf-0,orion-vnf-1 | dc1.s1-eth4,dc1.s1-eth5  |
+--------------+-------------------------+-------------------------+-----------------------+--------------------------+
| dc1          | dc1_FIWARE-3-mongo-vnf-1 | ubuntu:trusty           | mongo-vnf-0,mongo-vnf-1 | dc1.s1-eth6,dc1.s1-eth7  |
+--------------+-------------------------+-------------------------+-----------------------+--------------------------+
| dc1          | dc1_1-3-mongo-vnf-1     | mongo:latest            | mongo-vnf-2,mongo-vnf-3 | dc1.s1-eth12,dc1.s1-eth13 |
+--------------+-------------------------+-------------------------+-----------------------+--------------------------+
```

Figura 68. Lista de nodos de computación

En este caso vemos 6 contenedores desplegados (Figura 68), debido a que ejecutamos el escenario de pruebas dos veces, una con una imagen base Ubuntu Trusty y otra con las imágenes oficiales de Orion, Agente Ultralight y MongoDB.

Al tener estas 2 instancias del servicio de red podemos comparar el consumo de memoria de un servicio de red con imágenes limpias con el consumo de recursos de las imágenes preparadas para ofrecer los servicios requeridos. Para revisar esto usamos el siguiente comando:

```
docker stats $(docker ps --format '{{.Names}}')
```

```
CONTAINER ID        NAME                                                      CPU %               MEM USAGE / LIMIT     MEM %
aeb6f93fba08        osm_elk_elasticsearch.1.ow33cnfjqctwsgauc0jl3zp46         0.00%               0B / 0B               0.00%
51be4337f06a        mn.dc1_1-3-mongo-vnf-1                                    0.18%               42.02MiB / 23.45GiB   0.17%
269e2ba8b6b2        mn.dc1_1-2-orion-vnf-1                                    0.00%               2.555MiB / 23.45GiB   0.01%
43b1d21407c3        mn.dc1_1-1-agent-vnf-1                                    0.12%               121.9MiB / 23.45GiB   0.51%
c28970f4aa0a        mn.dc1_FIWARE-3-mongo-vnf-1                               0.00%               932KiB / 23.45GiB     0.00%
7f792b716010        mn.dc1_FIWARE-2-orion-vnf-1                               0.00%               928KiB / 23.45GiB     0.00%
b73b2bea8eba        mn.dc1_FIWARE-1-agent-vnf-1                               0.00%               932KiB / 23.45GiB     0.00%
```

Figura 69. Utilización de recursos de contenedores

Aquí podemos ver que el consumo de memoria de las imágenes oficiales es bastante reducido, por lo cual puede desplegarse en equipos con pocos recursos haciendo de ésta solución muy flexible.

Por último, podemos entrar a cada contenedor de manera individual, en caso de ser necesario, para eso solo necesitamos el nombre del contenedor de la VNF a la que queremos acceder y luego entrar a la consola de la siguiente manera:

```
sudo docker exec -it mn.dc1_1-1-agent-vnf-1 /bin/bash
```

```
test@ventas:/$ sudo docker exec -it mn.dc1_1-1-agent-vnf-1 /bin/bash
node@dc1_1-1-agent-vnf-1:/opt/iotaul$ ls
CHANGES NEXT RELEASE    LICENSE    bin              config.js  docs     index.js  mkdocs.yml    package-lock.json  rpm                      template
ContributionPolicy.txt  README.md  client-config.js  docker     ghpages  lib       node_modules  package.json       sonar-project.properties  test
node@dc1_1-1-agent-vnf-1:/opt/iotaul$
```

Figura 70. Línea de comandos de la VNF

Network Slice Template (NST):

Para la instanciación de Network Slice, cargamos el archivo YML con la descripción del servicio indicando los servicios de red a orquestar y el tipo de servicio:

```yml
nst:
-   id: FIWARESlice_nst
    name: FIWARESlice_nst
    SNSSAI-identifier:
        slice-service-type: URLLC
    quality-of-service:
        id: 1

    netslice-subnet:
    -   id: FIWARE_ns_nsd
        is-shared-nss: 'false'
        nsd-ref: FIWARE_ns_nsd
    -   id: FIWARE_ns_nsd
        is-shared-nss: 'false'
        nsd-ref: FIWARE_ns_nsd

    netslice-vld:
    -   id: FIWARE_vld_mgmt
        name: FIWARE_vld_mgmt
        type: ELAN
        mgmt-network: 'false'
        nss-connection-point-ref:
        -   nss-ref: FIWARE_ns_nsd
            nsd-connection-point-ref: FIWARE_ns_nsd_vld_mgmt
        -   nss-ref: FIWARE_ns_nsd
            nsd-connection-point-ref: FIWARE_ns_nsd_vld_mgmt
```

```
-   id: FIWARE_vld_data
    name: FIWARE_vld_data
    type: ELAN
    nss-connection-point-ref:
    -   nss-ref: FIWARE_ns_nsd
        nsd-connection-point-ref: FIWARE_ns_nsd_vld_data
    -   nss-ref: FIWARE_ns_nsd
        nsd-connection-point-ref: FIWARE_ns_nsd_vld_data
```

Plantilla NST

Luego de cargada la plantilla esto podremos instanciar el Network Slice en OSM:

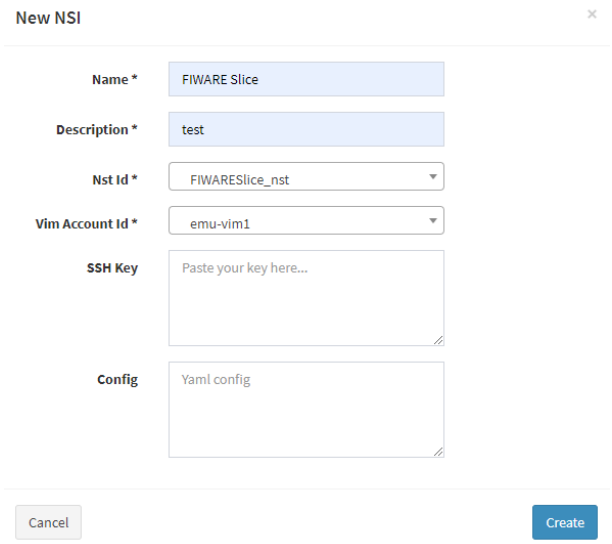


Figura 71. Nueva instancia de Network Slice

Durante la investigación encontramos problemas a la hora de intentar desplegar una network slice en OSM. Después de revisar distintas fuentes y revisar foros y comunidades de soporte de diferenets herramientas, pudimos confirmar que OSM en su integración con vim-emu tiene problemas con la instanciación de Network Slices a la hora de gestionar las redes de comunicación entre los servicios de red, esto es un problema que ya ha sido identificado.

Dentro del grupo de investigación, Andrés Cárdenas contactó con uno de los desarrolladores de vim-emu a través en su canal oficial de soporte, Manuel Peuster, el cual confirmó nuestras sospechas como podemos ver en la Figura 72. Una recomendación para hacer pruebas es desactivar la red de administración descrita en la plantilla.

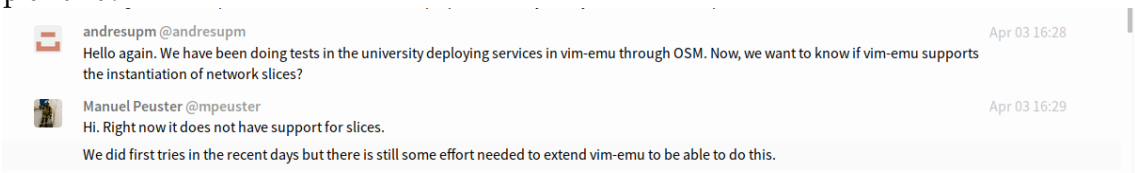


Figura 72. Respuesta sobre incidencias con vim-emu

Igualmente, decidimos continuar con el diseño incluyendo la plantilla NST, que puede ser desplegada utilizando otro VIM como OpenStack directamente, en el cual se pueden realizar despliegues de contenedores utilizando el proyecto Zun [45].

7 Conclusiones

Para lograr la implementación del concepto de Industria 4.0 es necesario replantear el concepto de redes industriales, ya que estas redes no fueron pensadas para los requerimientos que tienen las tecnologías involucradas en la digitalización de los procesos de manufactura. Los procesos industriales son procesos en tiempo real y que requieren muy baja latencia. Las redes inalámbricas usadas en la industria actualmente no son capaces de satisfacer ambos requerimientos. Un solo estándar inalámbrico no podrá cumplir con todos los requisitos en la era de la Industria 4.0. Por lo tanto, se requiere de una combinación de varias tecnologías para proporcionar una red eficiente. Por lo que es necesario buscar una manera de unificar distintos estándares dentro de una sola red y es aquí donde aparece 5G, la que es vista por muchos como la red de redes.

Es cierto que el estándar 5G no ha terminado de definirse y cada vez más aparecen nuevos escenarios en los cuales puede implementarse. Es por esto que hay que aprovechar que tanto la Industria 4.0 está en sus inicios, como el hecho de que las redes 5G siguen en proceso de estandarización. Pero aun cuando las redes 5G ofrecen ésta posibilidad debido a su flexibilidad de despliegue y mejoras en los métodos de acceso a la red, la transición de las redes industriales tradicionales a las redes de nueva generación, al menos en el futuro cercano, pasa por una mezcla de soluciones tradicionales, con las primeras implementaciones de los nuevos estándares. En el papel, 5G presenta una gran flexibilidad para despliegue de soluciones innovadoras y versátiles, como las que hemos detectado en el transcurso de esta investigación.

La virtualización de recursos nos provee de una gran flexibilidad para despliegue de servicios y con ello una gran oportunidad para las operadoras de generar nuevos modelos de negocio con la implementación de redes de nueva generación. En la práctica, con las prueba de concepto que hemos realizado hemos visto de primera mano esta flexibilidad que provee esta nueva generación, pero aún queda mucho por investigar y probar. Afortunadamente esta nueva generación de redes móviles se basa en gran parte en soluciones abiertas, por lo cual tenemos muchas herramientas en desarrollo y así mismo mucha información para desarrollar nuevas herramientas y soluciones que hagan uso de las mismas. De igual manera, hay mucha diversidad en los diferentes proyectos e investigaciones, cada uno tratando de solucionar problemas, a veces muy particulares. A pesar de ser una tarea algo complicada, es necesario unificar soluciones y proyectos en pro de una implementación más rápida tanto de la nueva revolución industrial, que afecta no sólo a la industria, sino a nuestras vidas en general, como también de las redes de nueva generación.

7.1 Líneas Futuras

Con el trabajo realizado en esta investigación hemos identificado varias líneas futuras de investigación:

- Orquestación de servicios en línea con los escenarios de la Release 16 de 3GPP.
- Implementación del escenario diseñado en otras plataformas MANO tanto opensource como propietarias (Gohan, SONATA, ONAP)
- Despliegues de servicios de red en diferentes puntos de presencia y medición del desempeño.
- Integración OSM en su versión 5.0 o en su versión 6.0, prevista para Q3 2019, con OpenStack en su versión Stein para desplegar el escenario diseñado y realizar pruebas de Network Slicing y gestión de contenedores para la orquestación de servicios de red.
- Diseño e implementación de un servicio FIWARE multicontenedor utilizando Kubernetes.

Anexo I: Pruebas con OSM 5.0

Para las pruebas realizadas con OSM, utilizamos la versión 5.0 que soporta orquestación de Network Slices.
La instalación es bastantes sencilla, puede realizarse una instalación bare-metal o en una máquina virtual y solo basta con usar alguna distribución de Ubuntu 16.04 y tener instalado Docker, ya que el despliegue de OSM se realiza con contenedores. Luego de esto solo es necesario bajar y ejecutar el script de instalación.

```
wget https://osm-download.etsi.org/ftp/osm-5.0-five/install_osm.sh

chmod +x install_osm.sh

./install_osm.sh --elk_stack --pm_stack --vimemu
```

OSM puede instalarse con los siguientes componentes opcionales mediante el uso de flags:

- Emulador VIM: vim-emu
- Plataforma de Monitorización de Desempeño: Grafana/Prometheus.
- Plataforma de Gestión de Fallas: Stack ELK

En nuestro caso realizamos la instalación de vim-emu para realizar las pruebas de despliegue de manera más ligera y aprovechar la utilización de contenedores en vez de máquinas virtuales.

Podemos validar que todos tanto OSM como vim-emu han sido instalados y se están ejecutando de manera correcta con el siguiente comando:

```
docker ps
```

```
CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS                          PORTS
9ac01e48a381        prom/prometheus:v2.4.3         "/bin/prometheus --c…"   12 seconds ago      Up 1 second                     9090/tcp
ometheus.1.yexpnyn5oq5vphfqfgi2pdt22
d9aeb32fdfc0        vim-emu-img                    "/son-emu/utils/dock…"   About a minute ago  Up 57 seconds                   4000/tcp, 5000-5001/tcp, 6001/tcp, 808
u
5c3c69945993        opensourcemano/mon:latest      "/bin/sh -c '/bin/ba…"   About a minute ago  Up 54 seconds (health: starting)  8000/tcp
n.1.kb0kmh6v27oldqngaipi884ci
c7351c082cff        opensourcemano/lcm:latest      "python3 /usr/lib/py…"   5 minutes ago       Up 5 minutes (healthy)          9999/tcp
m.1.i9x6s87zv7t3t1jm4xvyzmydy
d2692e36d38a        opensourcemano/nbi:latest      "python3 /usr/lib/py…"   19 hours ago        Up 19 hours (healthy)           9999/tcp
i.1.yztxkzed0fkdto42qnl8rayxu
76c160dbf196        opensourcemano/pol:latest      "/bin/sh -c osm-poli…"   19 hours ago        Up 19 hours
l.1.rkzr21bpg4w10908a326g8cx9
4cdc846a0c04        mongo:latest                   "docker-entrypoint.s…"   19 hours ago        Up 19 hours                     27017/tcp
ngo.1.bpsb2fcfp402k6gnudlvqnrox
b432ccde364a        mariadb:10                     "docker-entrypoint.s…"   19 hours ago        Up 19 hours                     3306/tcp
ystone-db.1.sy7tp40ixr2rdio5pmrwzehb5
e6005dbf1da6        mysql:5                        "docker-entrypoint.s…"   19 hours ago        Up 19 hours                     3306/tcp, 33060/tcp
-db.1.5elgy12k3k4f50qros34idyeb
f156d0c6177b        wurstmeister/zookeeper:latest  "/bin/sh -c '/usr/sb…"   19 hours ago        Up 19 hours (healthy)           22/tcp, 2181/tcp, 2888/tcp, 3888/tcp
okeeper.1.zhql8ricy48b60fk8a9rgyim7
7ef7ef0eaa99        wurstmeister/kafka:2.11-1.0.2  "start-kafka.sh"         19 hours ago        Up 19 hours
fka.1.whvpydbh2rxc10h3cmcwmav7t
5e652b896a8a        opensourcemano/keystone:latest "/bin/sh -c ./start.…"    19 hours ago        Up 19 hours                     5000/tcp
ystone.1.hbd5ahtyrw0yb052vwwngkpfn
abe29ed4c95f        opensourcemano/ro:latest       "/bin/sh -c /bin/RO/…"   19 hours ago        Up 19 hours (healthy)           9090/tcp
.1.cs93j6err6o9j91c3pctrkkm9
b2269cc91b34        opensourcemano/light-ui:latest "supervisord -n"         19 hours ago        Up 19 hours                     80/tcp
ght-ui.1.38zucp31u9uuhcsyva1zsgyie
```

Al ejecutar el script se instala el emulador VIM, pero todavía hace falta añadirlo a OSM para poder orquestar contenedores. Para hacer esto necesitamos la IP del contendor en el cual se está ejecutando vim-emu lo cual podemos conseguir con el siguiente comando:

```
export VIMEMU_HOSTNAME=$(sudo docker inspect -f '{{range .NetworkSettings.Net
works}}{{.IPAddress}}{{end}}' vim-emu)
```

Una vez obtenida la ip, podemos agregar vim-emu a OSM con el siguiente comando:

```
osm vim-create --name emu-vim1 --user username --password password --auth_url
Error! Hyperlink reference not valid. --tenant tenantName --account_type open
stack
```

```
openstack@openstack:~$ osm vim-create --name emu-vim1 --user username --pa
74c0784e-dafe-4a45-9b09-09cb3824ff8f
```

Podemos confirmar que se ha agregado vim-emu de manera exitosa de la siguiente manera:

```
osm vim-list
```

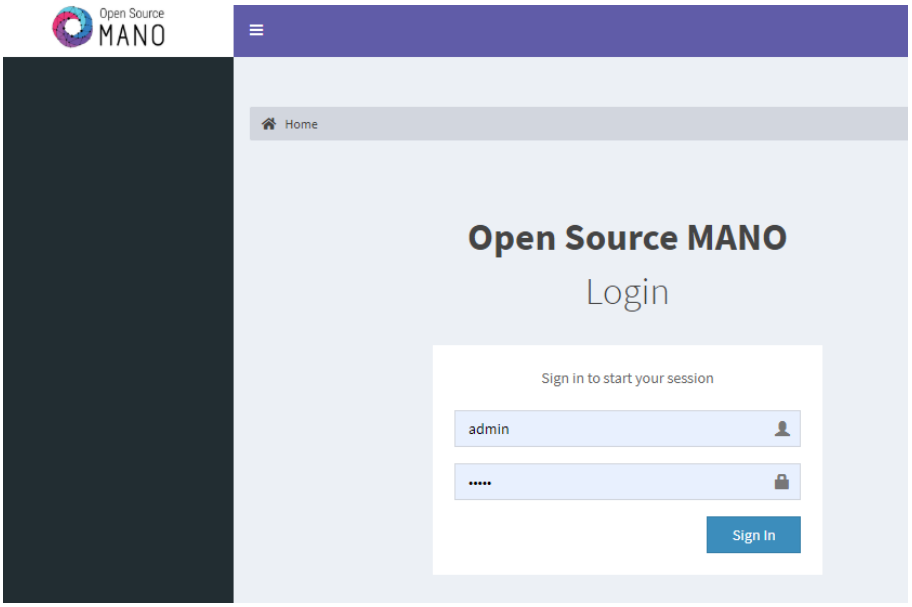
```
openstack@openstack:~$ osm vim-list
+----------+--------------------------------------+
| vim name | uuid                                 |
+----------+--------------------------------------+
| emu-vim1 | 74c0784e-dafe-4a45-9b09-09cb3824ff8f |
+----------+--------------------------------------+
```

Si llegase a haber un problema, podemos eliminar el VIM con el siguiente comando:

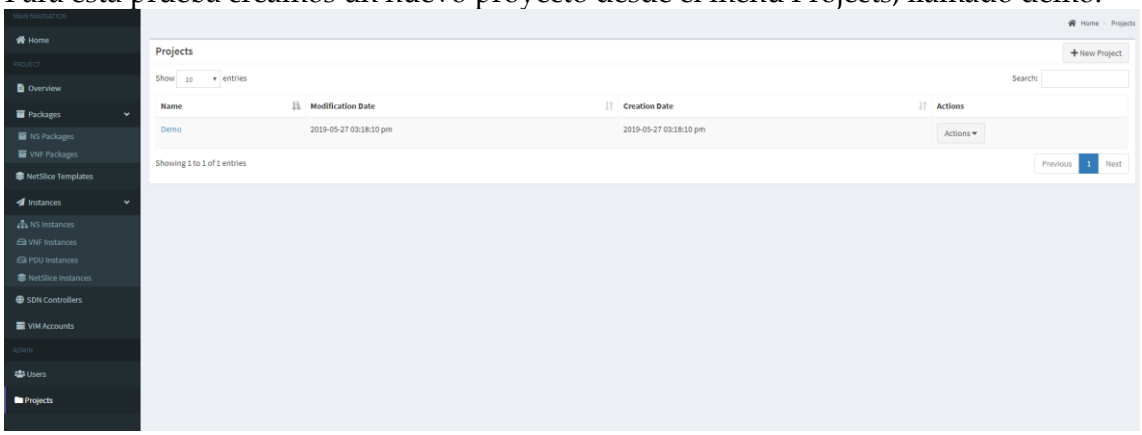
```
osm vim-delete --force emu-vim1
```

```
openstack@openstack:~$ osm vim-delete --force emu-vim1
Deletion in progress
```

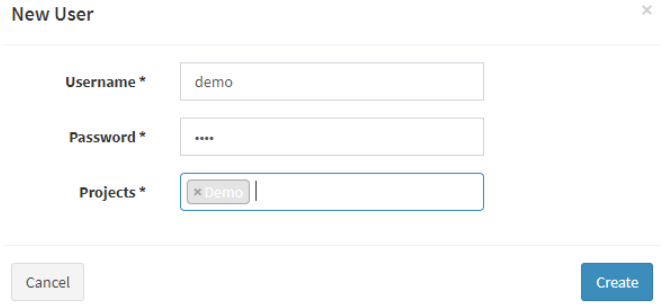
Una vez finalizada la instalación tenemos acceso al dashboard, accediendo a **http://localhost/auth/?next=/home** e introduciendo los credenciales por defecto: admin/admin.



En OSM podemos gestionar diferentes usuarios y proyectos, con sus respectivas medidas de seguridad, para limitar el acceso a los proyectos.
Como usuario podemos crear nuevos usuarios y asignarles acceso a diferentes proyectos.
Para esta prueba creamos un nuevo proyecto desde el menú Projects, llamado demo:



Una vez creado el proyecto, creamos un nuevo usuario con los credenciales demo/demo al que le asignamos este proyecto:



Procedemos a cerrar la sesión de administrador y entrar nuevamente con el nuevo usuario.

Una vez dentro del dashboard podemos observar los paquetes de VNF y NS que han sido validados e igualmente las instancias en ejecución de estos paquetes. Estos paquetes pueden ser agregados mediante la línea de comandos o vía web.

VNF

Antes de agregar una VNF es necesario crear el paquete o validar el código YAML en el portal web. Para ayudar con el diseño de la VNF a la hora de escribir el código,

podemos usar una referencia visual que suministra el proyecto OSM, en el cual vamos definiendo todos los componentes de nuestra VNF:

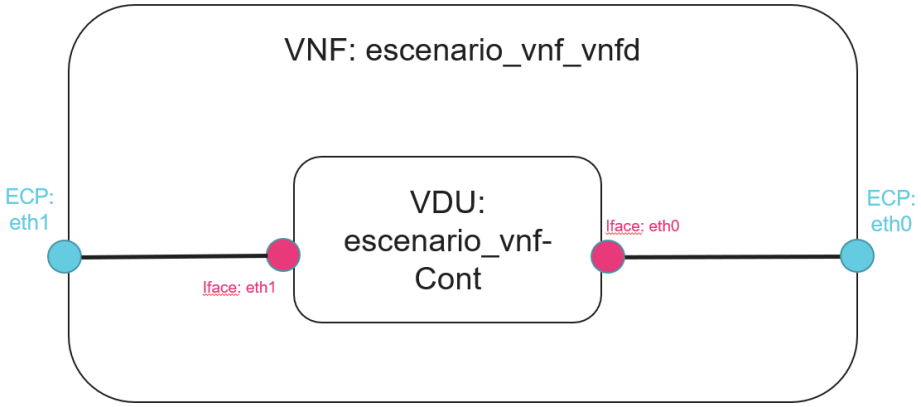
- Virtual Deployment Unit (VDU): Elemento básico de la VNF, son las máquinas virtuales o contenedores.
- External Connection Point (ECP): Interfaces de la VNF
- Interface (Ifce): Interfaces de la VDU
- Virtual Links (VL): Conexión entre las interfaces de la VDU y los ECP

Con estos elementos básicos podemos crear una VNF sencilla, para experimentar con ella y luego desplegar un servicio de red.

Crearemos una VNF con los siguientes parámetros:

Elemento	ID	Cantidad
VNF	escenario_vnf_vnfd	1
VDU	escenario_vnf_vnfd-Cont	1
Iface	eth0, eth1	2
ECP	eth0, eth1	2
VL		2

El cual debe verse de esta manera:

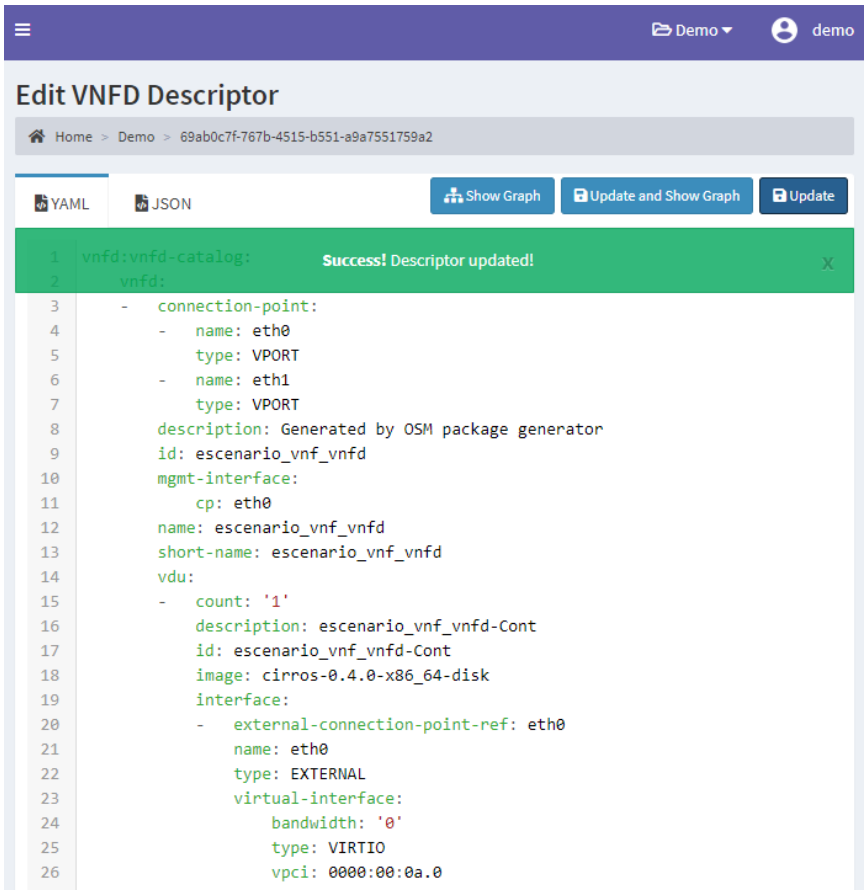


Traspasamos el diseño al lenguaje YANG:

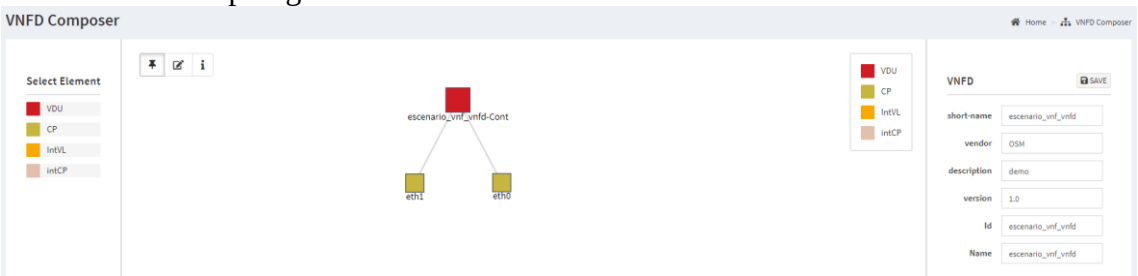
```
vnfd:vnfd-catalog:
    vnfd:
    -   connection-point:
        -   name: eth0
            type: VPORT
        -   name: eth1
            type: VPORT
        description: demo
        id: escenario_vnf_vnfd
        mgmt-interface:
```

```
            cp: eth0
        name: escenario_vnf_vnfd
        short-name: escenario_vnf_vnfd
        vdu:
        -   count: '1'
            description: escenario_vnf_vnfd-Cont
            id: escenario_vnf_vnfd-Cont
            image: cirros-0.4.0-x86_64-disk
            interface:
            -   external-connection-point-ref: eth0
                name: eth0
                type: EXTERNAL
                virtual-interface:
                    bandwidth: '0'
                    type: VIRTIO
                    vpci: 0000:00:0a.0
            -   external-connection-point-ref: eth1
                name: eth1
                type: EXTERNAL
                virtual-interface:
                    bandwidth: '0'
                    type: VIRTIO
                    vpci: 0000:00:0a.0
            name: escenario_vnf_vnfd-VM
            vm-flavor:
                memory-mb: 512
                storage-gb: 5
                vcpu-count: 1
        vendor: OSM
        version: '1.0'
```

Validamos el descriptor en la plataforma web desde el menú VNF Packages:

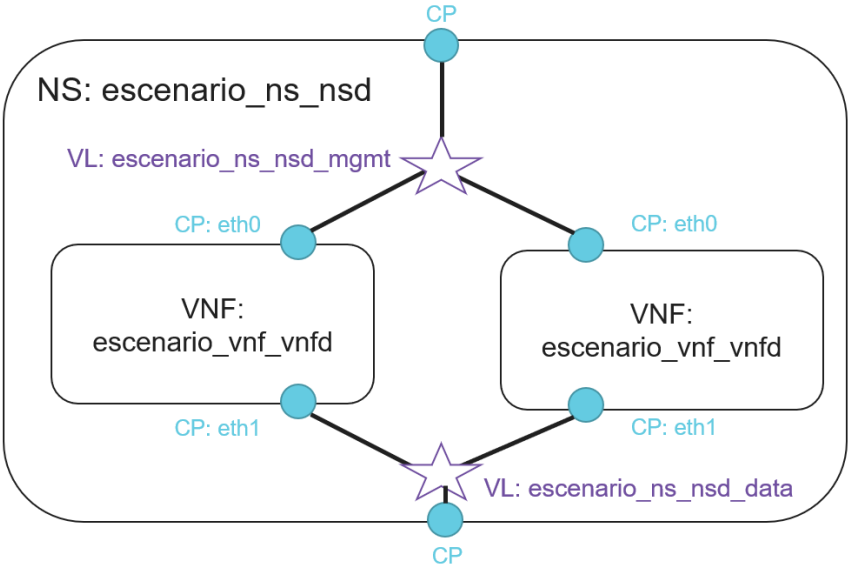


Y validamos la topología:



Network Service

Con esta VNF básica, podemos diseñar un servicio de red como se muestra en la diagrama:

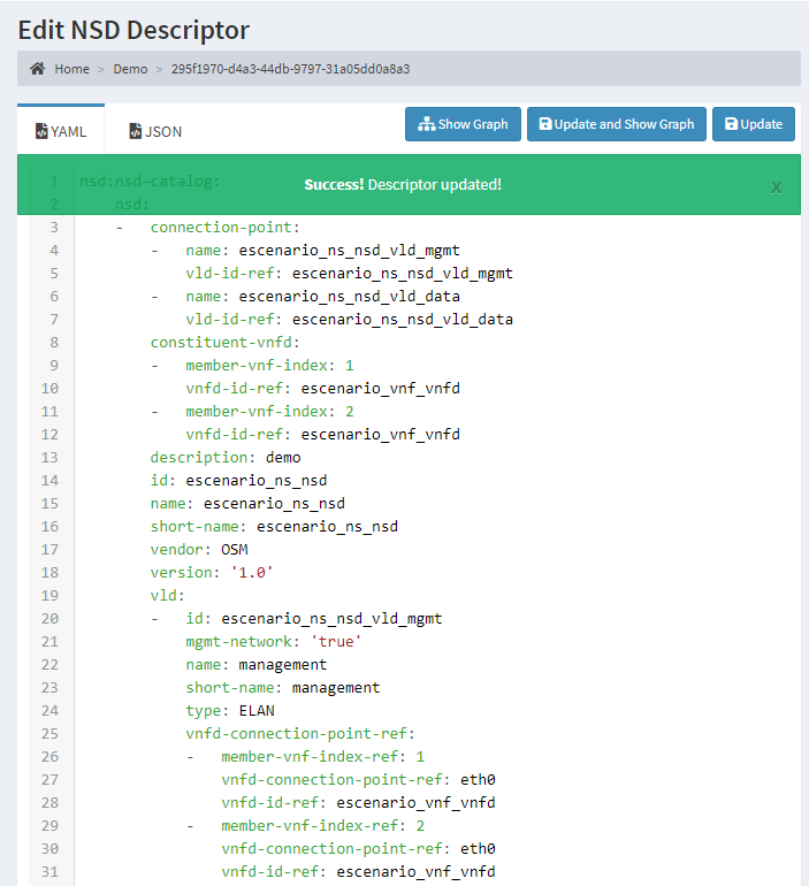


Una vez validado el diagrama, traspasamos el diseño al lenguaje YANG:

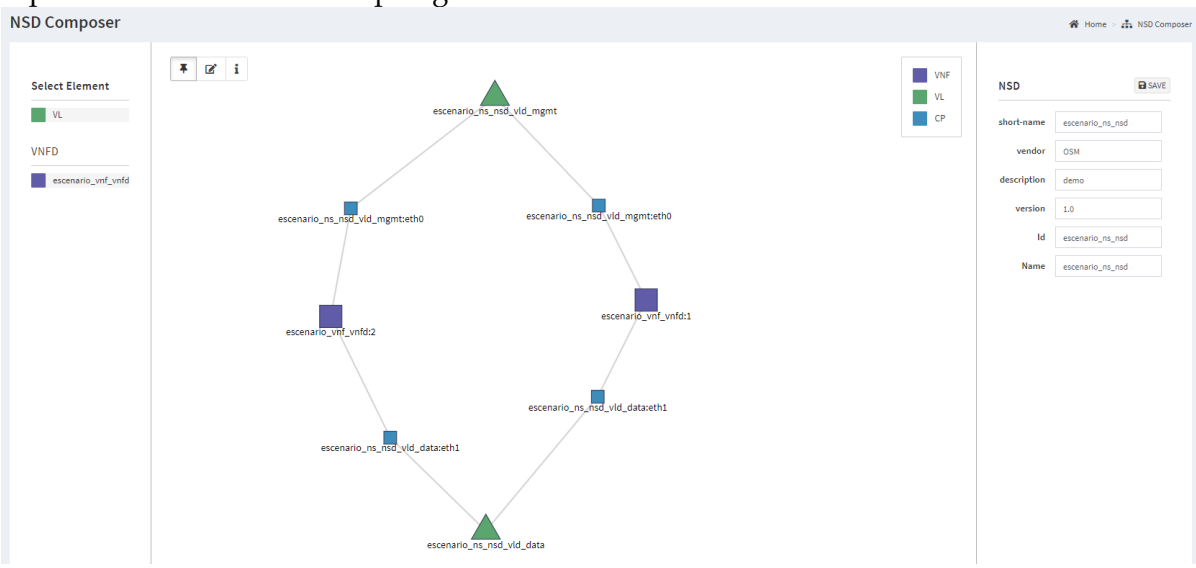
```
nsd:nsd-catalog:
    nsd:
    -   connection-point:
        -   name: escenario_ns_nsd_vld_mgmt
            vld-id-ref: escenario_ns_nsd_vld_mgmt
        -   name: escenario_ns_nsd_vld_data
            vld-id-ref: escenario_ns_nsd_vld_data
        constituent-vnfd:
        -   member-vnf-index: 1
            vnfd-id-ref: escenario_vnf_vnfd
        -   member-vnf-index: 2
            vnfd-id-ref: escenario_vnf_vnfd
        description: demo
        id: escenario_ns_nsd
        name: escenario_ns_nsd
        short-name: escenario_ns_nsd
        vendor: OSM
        version: '1.0'
        vld:
        -   id: escenario_ns_nsd_vld_mgmt
            mgmt-network: 'true'
            name: management
            short-name: management
```

```
            type: ELAN
            vnfd-connection-point-ref:
            -   member-vnf-index-ref: 1
                vnfd-connection-point-ref: eth0
                vnfd-id-ref: escenario_vnf_vnfd
            -   member-vnf-index-ref: 2
                vnfd-connection-point-ref: eth0
                vnfd-id-ref: escenario_vnf_vnfd
        -   id: escenario_ns_nsd_vld_data
            name: escenario_ns_nsd_vld_data
            short-name: escenario_ns_nsd_vld_data
            type: ELAN
            vnfd-connection-point-ref:
            -   member-vnf-index-ref: 1
                vnfd-connection-point-ref: eth1
                vnfd-id-ref: escenario_vnf_vnfd
            -   member-vnf-index-ref: 2
                vnfd-connection-point-ref: eth1
                vnfd-id-ref: escenario_vnf_vnfd
```

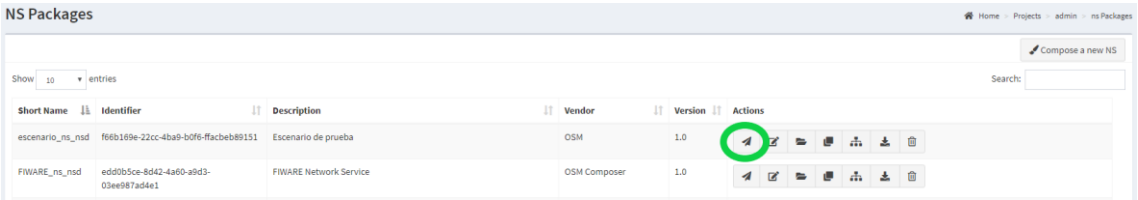
Validamos el descriptor en la plataforma web desde el menú NS Packages:

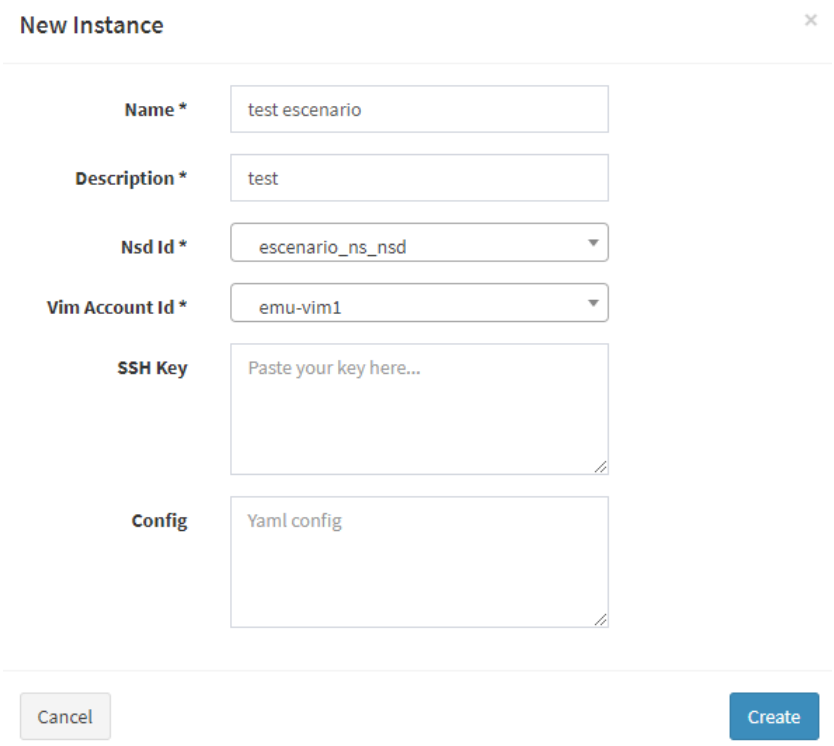


Y por último validamos la topología:



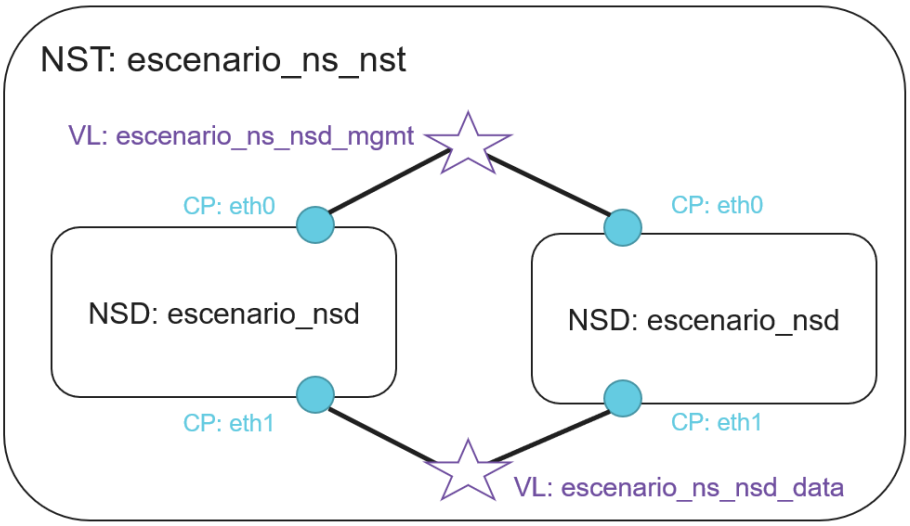
Una vez validado el paquete, podemos instanciar el servicio de red:





Network Slice

Para la instanciación de Network Slice, cargamos el archivo YML con la descripción del servicio indicando los servicios de red a orquestar y el tipo de servicio:




```
nsd:nsd-catalog:
    nsd:
    -   connection-point:
        -   name: escenario_ns_nsd_vld_mgmt
            vld-id-ref: escenario_ns_nsd_vld_mgmt
        -   name: escenario_ns_nsd_vld_data
```

```
            vld-id-ref: escenario_ns_nsd_vld_data
        constituent-vnfd:
        -   member-vnf-index: 1
            vnfd-id-ref: escenario_vnf_vnfd
        -   member-vnf-index: 2
            vnfd-id-ref: escenario_vnf_vnfd
        description: demo
        id: escenario_ns_nsd
        name: escenario_ns_nsd
        short-name: escenario_ns_nsd
        vendor: OSM
        version: '1.0'
        vld:
        -   id: escenario_ns_nsd_vld_mgmt
            mgmt-network: 'true'
            name: management
            short-name: management
            type: ELAN
            vnfd-connection-point-ref:
            -   member-vnf-index-ref: 1
                vnfd-connection-point-ref: eth0
                vnfd-id-ref: escenario_vnf_vnfd
            -   member-vnf-index-ref: 2
                vnfd-connection-point-ref: eth0
                vnfd-id-ref: escenario_vnf_vnfd
        -   id: escenario_ns_nsd_vld_data
            name: escenario_ns_nsd_vld_data
            short-name: escenario_ns_nsd_vld_data
            type: ELAN
            vnfd-connection-point-ref:
            -   member-vnf-index-ref: 1
                vnfd-connection-point-ref: eth1
                vnfd-id-ref: escenario_vnf_vnfd
            -   member-vnf-index-ref: 2
                vnfd-connection-point-ref: eth1
                vnfd-id-ref: escenario_vnf_vnfd
```

Como mencionamos anteriormente OSM en su integración con vim-emu tiene problemas con la instanciación de Network Slices a la hora de gestionar las redes de comunicación entre los servicios de red, siendo esto es un problema que ya identificado. En nuestras pruebas, sólo llegamos a validar el código de la plantilla.

Uso de CLI:

Para poder generar VNFDs y NSDs mediante la CLI, es necesario descargar el repositorio devops de OSM:

```
git clone https://osm.etsi.org/gerrit/osm/devops.git
```

Luego ejecutar el siguiente comando:

```
./devops/descriptor-packages/tools/generate_descriptor_pkg.sh -t vnfd --image
cirros-0.4.0-x86_64-disk -c escenario_vnf
```

Se crea un folder con la estructura <Nombre_VNF>_vnfd y todos los archivos necesarios para el paquete VNF.
Editamos el archivo YAML con el código YANG de nuestra VNF <Nombre_VNF>_vnfd.yaml.
Una vez todos los archivos estén listos se puede generar el paquete tar.gz co el siguiente comando:

```
./devops/descriptor-packages/tools/generate_descriptor_pkg.sh -t vnfd -N <Nom
bre_VNF>_vnfd
```

Luego de haber generado los paquetes necesarios ya podemos agregarlos a OSM con el siguiente comando:

```
osm vnfd-create <Nombre_VNF>.tar.gz
```

Agregar el descriptor del servicio de red mediante el siguiente comando:

```
osm nsd-create <Nombre_NSD>.tar.gz
```

Anexo II: Pruebas con vim-emu

Vim-emu se despliega como un contenedor Docker al realizar la instalación de OSM con el parámetro –vim-emu.
Podemos validar que la instalación fue exitosa y el contenedor está corriendo con el siguiente comando:

```
docker ps | grep vim-emu
```

De no estar en ejecución, podemos iniciarlo con el siguiente comando:

```
sudo docker run --name vim-emu -t -d --rm --privileged --pid='host' -v /var/r
un/docker.sock:/var/run/docker.sock vim-emu-img python examples/osm_default_d
aemon_topology_2_pop.py
```

vim-emu emula topologías de red en mininet que son descritas en un archivo .py. Para las pruebas con OSM usamos la topología por defecto osm_default_daemon_topology_2_pop.py. Ésta puede ser modificada según las necesidades, pero en la versión más reciente de OSM no tiene ningún efecto utilizar otra topología. En las pruebas realizadas con vim-emu y OSM todos los contenedores son desplegados en el Datacenter 1 de la topología, debido a que en la versión 5 de OSM removieron la posibilidad de indicar desde el orquestador, el Datacenter en el cual instanciar los contenedores. Se espera que para la versión 6 este problema esté resuelto.

Para ejecutar comandos en vim-emu debemos acceder a la consola del contendor Docker en el que se está ejecutando y desde ahí podremos utilizar comandos para interactuar con la API emulada de OpenStack.

Podemos ver la lista de los Datacenter emulados y la cantidad de contenedores que se han desplegado en cada Datacenter con el siguiente comando:

```
sudo docker exec vim-emu vim-emu datacenter list
```

```
openstack@openstack:~$ docker exec vim-emu vim-emu datacenter list
+---------+----------------+----------+----------------+--------------------+
| Label   | Internal Name  | Switch   |  # Containers  |  # Metadata Items  |
+=========+================+==========+================+====================+
| dc2     | dc2            | dc2.s1   |              0 |                  0 |
+---------+----------------+----------+----------------+--------------------+
| dc1     | dc1            | dc1.s1   |              0 |                  0 |
+---------+----------------+----------+----------------+--------------------+
```

Podemos ver los nodos de computación desplegados en cada Datacenter con el siguiente comando:

```
sudo docker exec vim-emu vim-emu compute list
```

```
openstack@openstack:~$ docker exec vim-emu vim-emu compute list
+--------------+-------------+---------+-----------------+-------------------------+
| Datacenter   | Container   | Image   | Interface list  | Datacenter interfaces   |
+==============+=============+=========+=================+=========================+
+--------------+-------------+---------+-----------------+-------------------------+
```

En esta ocasión no vemos ningún nodo de computación ya que no se han orquestado servicios de red desde OSM.

Bibliografía

[1] T. Carlsson, «Industrial Ethernet is now bigger than fieldbuses,» HMS Networks, 16 Febrero 2018. [En línea]. Available: https://www.anybus.com/about-us/news/2018/02/16/industrial-ethernet-is-now-bigger-than-fieldbuses. [Último acceso: 2019].

[2] Z. Lin y S. Pearson, «An inside look at industrial Ethernet communication protocols,» Texas Instruments, 2018.

[3] ODVA, «CIP on CAN Technology,» 2016.

[4] PROFINET University, «PROFINET Communication Channels,» [En línea]. Available: https://profinetuniversity.com/profinet-basics/profinet-communication-channels/.

[5] Mettis Aerospace, «THE WIRELESS BROADBAND ALLIANCE AND METTIS AEROSPACE ANNOUNCE WORLD'S FIRST WI-FI 6 INDUSTRIAL TRIAL,» [En línea]. Available: https://www.mettis-aerospace.com/wifi6/.

[6] WiFi Alliance, «Wi-Fi CERTIFIED 6™ coming in 2019,» [En línea]. Available: https://www.wi-fi.org/news-events/newsroom/wi-fi-certified-6-coming-in-2019.

[7] ITU-R, «IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond,» 2015.

[8] B. Bertenyi, «3GPP system standards heading into the 5G era,» 3GPP, [En línea]. Available: https://www.3gpp.org/news-events/1614-sa_5g.

[9] NGMN Alliance, «Description of Network Slicing Concept,» 2016.

[10] «LA TRANSFORMACIÓN DIGITAL DE LA INDUSTRIA ESPAÑOLA - INFORME PRELIMINAR,» Gobierno de España - Ministerio de Industria, Energia y Turismo.

[11] European Comission, «Platforms for Connected Factories of the Future,» de *Workshop on Platforms for Connected Factories of the Future*, Brussels, Belgium, 2015.

[12] Policy Department A: Economic And Scientific Policy, «Industry 4.0 - Study,» 2016.

[13] S. S. Kamble, A. Gunasekaran y S. A. Gawankar, «Sustainable Industry 4.0 framework: A systematic literature review identifying the current trends and future perspectives,» *Process Safety and Environmental Protection,* vol. 117, pp. 408-425, 2018.

[14] KUKA Robotics, [En línea]. Available: https://www.kuka.com/en-de/products/mobility/mobile-robots.

[15] IoT Analytics, «State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating,» [En línea]. Available: https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/.

[16] Statista, «Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions),» [En línea]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

[17] K. Ashton, «That 'Internet of Things' Thing,» [En línea]. Available: https://www.rfidjournal.com/articles/view?4986.

[18] Sam Solutions, «Internet of Things (IoT) Protocols and Connectivity Options: An Overview,» [En línea]. Available: https://www.sam-solutions.com/blog/internet-of-things-iot-protocols-and-connectivity-options-an-overview/.

[19] G. George, E. Osinga, D. Lavie y B. Scott, «Big data and data science methods for management research: From the Editors,» *Academy of Management Journal,* vol. 59, nº 5, 2016.

[20] T. L. Nguyen , «A Framework for Five Big V's of Big Data and Organizational Culture in Firms,» de *2018 IEEE International Conference on Big Data (Big Data)*, 2018.

[21] P. Fraga-Lamas y T. Fernández-Caramés, «A Review on Industrial Augmented Reality Systems for the Industry 4.0 Shipyard,» *IEEE Access,* 2018.

[22] M. Karrenbauer, A. Fellan y H. D. Schotten , «Towards a Flexible Architecture for Industrial Networking».

[23] 3GPP, «Release 16,» [En línea]. Available: https://www.3gpp.org/release-16.

[24] A. Neumann , L. Wisniewski y R. SivaSiva Ganesan, «Towards Integration of Industrial Ethernet with,» de *14th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2018 .

[25] K. V. Reddy, «Time Sensitive Networking for 5G,» de *Building Tomorrow's Networks*, Geneva, 2018.

[26] 5GPPP, «5G and the Factories of the Future,» 2015.

[27] ETSI, «ETSI GS MEC-IEG 004 V1.1.1 Mobile-Edge Computing (MEC): Service Scenarios».

[28] SCOPE AR, [En línea]. Available: https://www.scopear.com/products/remote-ar/.

[29] A. Mueller, «5G in Manufacturing,» Robert Bosch GmbH.

[30] J. Chen, «"Industrial Cellular Radio Access Networking - ICRAN",» US Wireless Lab Futurewei Technologies.

[31] T. Taleb, I. Afolabi y M. Bagaa, «Orchestrating 5G Network Slices to Support Industrial Internet and to Shape Next-Generation Smart Factories,» *IEEE Network,* 2018.

[32] 5G-ACIA, «5G for Connected Industries and Automation,» 2019.

[33] CEF Digital, «CEF Digital Wiki,» [En línea]. Available: https://ec.europa.eu/cefdigital/wiki/display/SDBX/What+is+Context+Broker.

[34] FIWARE, «FIWARE IOT AGENT FRAMEWORK,» [En línea]. Available: https://iotagent-node-lib.readthedocs.io/en/latest/.

[35] FIWARE, «FIWARE Tutorials,» [En línea]. Available: https://github.com/FIWARE/.

[36] FIWARE, «tutorials.IoT-Sensors,» [En línea]. Available: https://github.com/FIWARE/tutorials.IoT-Sensors.

[37] FIWARE, «FIWARE-IOTAGENT-UL,» [En línea]. Available: https://fiware-iotagent-ul.readthedocs.io/en/latest/usermanual/index.html.

[38] FIWARE, «fiware/iotagent-ul,» [En línea]. Available: https://hub.docker.com/r/fiware/iotagent-ul/.

[39] ECLIPSE Foundation, «Eclipse Mosquitto,» [En línea]. Available: https://mosquitto.org/.

[40] FIWARE, «IoT over MQTT,» [En línea]. Available: https://github.com/FIWARE/tutorials.IoT-over-MQTT.

[41] FIWARE, «Fast-RTPS-Micro-RTPS,» [En línea]. Available: https://github.com/FIWARE/tutorials.Fast-RTPS-Micro-RTPS.

[42] N. Nitin, «Docker Container-Based Big Data Processing System in Multiple Clouds for Everyone,» de *2017 IEEE International Systems Engineering Symposium (ISSE)*, Viena, 2017.

[43] M. Peuster, H. Karl y S. v. Rossem, «MeDICINE: Rapid Prototyping of Production-Ready Network Services in Multi-PoP Environments,» de *IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, Palo Alto, CA, USA, 2016.

[44] ETSI, «Vim-emu: A NFV multi-PoP emulation platform,» [En línea]. Available: https://osm.etsi.org/wikipub/images/5/58/Vim-emu-etsi-mapping.png.

[45] OpenStack, «Zun,» [En línea]. Available: https://wiki.openstack.org/wiki/Zun.