Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación

ESTUDIO SOBRE LA APLICACIÓN DE LAS REDES DEFINIDAS POR SOFTWARE A LA INTERNET DE LAS COSAS. DISEÑO DE UN ESCENARIO DE PRUEBAS

TRABAJO FIN DE MÁSTER

Liz Panamá Carrasco Espinosa

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

ESTUDIO SOBRE LA APLICACIÓN DE LAS REDES DEFINIDAS POR SOFTWARE A LA INTERNET DE LAS COSAS. DISEÑO DE UN ESCENARIO DE PRUEBAS

Autor
Liz Panamá Carrasco Espinosa

Director
David Fernández Cambronero

Departamento de Ingeniería de Sistemas Telemáticos

2019

Resumen

La internet de las cosas, o por sus siglas en inglés IoT (Internet of Things), consiste en la interconexión de gran cantidad de dispositivos a la internet. Estos dispositivos están encargados de recopilar, compartir, procesar datos y tomar decisiones inteligentes en base a la información recopilada. Cada año, el número de dispositivos conectados a internet va creciendo exponencialmente, dando como resultado redes heterogéneas en las cuales su gestión, control y monitoreo tiende a ser complejo según la infraestructura que tenemos en redes tradicionales. Por esto, se tiene la necesidad de una nueva arquitectura de red donde se pueda mejorar el funcionamiento de las redes y así permitir que sean más inteligentes, eficientes, seguras y escalables.

Las Redes Definidas por Software, o por sus siglas en inglés SDN (Software Defined Networking) se caracterizan por separar el plano de control del plano de datos. El plano de control posee un controlador de software inteligente, que envía señales a la capa de datos y gestiona todos los dispositivos que estén dentro de la red. La capa de datos es la que contiene los conmutadores programables, los cuales realizan las operaciones mediante un protocolo llamado OpenFlow, el cuál es el medio de comunicación entre el plano de control y datos. Este funcionamiento permite al administrador de red tener una visión global de la red y controlar o gestionar el tráfico de acuerdo a las necesidades de cada momento.

Una de las tecnologías que puede aportar a reducir los diferentes inconvenientes que están ocurriendo en la implementación de la IoT son las SDN que prometen una mejora en la gestión y control de la red por su arquitectura más flexible y escalable.

A la fecha, se tienen diferentes propuestas sobre los impactos que tienen las tecnologías SDN a las diferentes aplicaciones de la IoT. Por ello, este trabajo consiste en un estudio sobre los aportes que traen las SDN a la IoT, haciendo una revisión del estado actual de cada tecnología, identificando los problemas que está atravesando la IoT y de qué manera se pueden solucionar utilizando SDN. Esto se mostrará mediante el diseño e implementación de un caso de estudio en escenarios virtuales, simulando una red residencial. El escenario se realizará mediante la herramienta de creación de escenarios virtuales VNX, donde se simulará una red IoT - SDN con dispositivos con SO TinyCore, Linux y el controlador RYU.

Abstract

The Internet of Things (IoT) describes the interaction of a broad range of electronic devices with the internet. These devices aim at collecting, sharing, processing data and making intelligent decisions that are based on previously collected information. The annually exponentially increasing number of devices connected to the internet gives rise to heterogenous networks which require complex processes of control and monitoring, as they are based on traditional networks. Consequentially, a new network architecture that allows for improving the network performance, making it more intelligent, efficient, secure and scalable has become not only a goal, but a necessity.

Software Defined Networking (SDN) is characterized by the separation of the control plane from the data plane. The control plane has an internal controlling system, which sends signals to the data layer and manages all the devices that are connected to the network. The data layer contains programmable switchers that operate according to a protocol called "OpenFlow", which is the medium of communication between the control plane and the data plane. This setup allows the controller to have a global vision of the network and to manage and/or redirect the traffic in order to meet the necessities at all times.

One of the technologies that can contribute to the reduction of different disadvantages that occur during the IoT's implementation are SDNs, that aim at improving the management and control of the network based on a more flexible and scalable architecture.

Until now there have been different approaches concerning the impact that SDN technologies have on the different applications of the IoT. Therefore, this paper focuses on the contribution of SDN to the IoT. It contains a study that focuses on the current state of the different technologies and the role that the IoT takes when posed with different problems. A special emphasis will be laid on how SDN can be used to solve those problems by integrating it in the architecture of the IoT. Doing so, a case study of virtual scenarios that simulates a residential network will be conducted. The scenario will be carried out using VNX, a tool of creating virtual scenarios. This scenario contains an IoT-SDN network containing devices running the TinyCore and Linux operating system and a RYU controller.

Índice General

Índice de Figuras

Siglas

IoT	Internet of Thinks
SDN	Software Defined Networking
6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
M2M	Machine to Machine
NFV	Network Function Virtualization
SO	Sistema Operativo
RFID	Radio Frequency Identification
VNX	Virtual Networks over linuX
IP	Internet Protocol
OVS	Open vSwitch
RFID	Radio Frequency Identification
HW	Hardware
SW	Software
XML	Extensible Markup Languajes
XMPP	Extensible Messaging Presence Protocol
COAP	Constrained Application Protocol
MQTT	Message Queue Telemetry Transport
QoS	Calidad de Servicio
UDP	User Datagram Protocol
HTTP	Hypertext Transfer Protocol
RPL	Routing Protocol
PHY	Physical Layer
MTU	Maximum Transmission Unit

MAC	Media Access Control
NFV	Virtualización de las Funciones de Red
VANET	Vehicular Ad-Hoc Network
TCP	Protocolo de Control de Transmisión
GW	Gateway
SO	Sistema Operativo
LBR	LoW Border Router
PLC	Programmable Logic Controller

1 Introducción

Estos últimos años se han dado a conocer muchas tecnologías disruptivas, que mediante su desarrollo han tenido un gran efecto sobre la sociedad, esto ha traído un gran crecimiento en las redes de comunicación, lo cual ha llevado a mejorar la arquitectura tradicional por el alto tráfico generado, exigiendo mayores requisitos.

Una de estas tecnologías disruptivas es la IoT, la cual consiste en la interconexión de dispositivos, con cualquier otro que esté a su alrededor al internet, permitiendo que las personas y los objetos interactúen entre sí. Esta tecnología está creciendo a un ritmo acelerado, en busca de que todos los dispositivos se comuniquen entre sí y, por lo tanto, sean más inteligentes e independientes.

En el informe de cisco "Visual Networking Index - Forecast and Trends" publicado en noviembre, 2018 se predicen las diferentes tendencias de las redes de comunicaciones, de acuerdo al consumo de tráfico de 2017 a 2022. Indican que el tráfico IP global se triplicará en los próximos 5 años, se tendrán 28,5 mil millones de dispositivos en red para 2022, frente a los 18 mil millones en 2017 [1].

Según esta predicción, las conexiones M2M representarán el 51 por ciento del total de dispositivos y conexiones. Es decir, que será una de las categorías de más alto crecimiento, lo cual acelera el desarrollo y crecimiento de la IoT, además de que gran parte del tráfico de Internet, se generará a través de redes IPv6, lo cual trae una buena oportunidad a los operadores de redes, proveedores de contenido y usuarios finales, que buscan obtener la escalabilidad y los beneficios de rendimiento de IPv6 para habilitar el Internet de las cosas (IoT) [2]. A pesar que las conexiones aumentan significativamente, el tráfico IP M2M global crece aún más, debido a aplicaciones que requieren mayor ancho de banda y poca latencia como telemedicina, sistemas de navegación inteligente para automóviles y aplicaciones de video.

Un dispositivo IoT (sensor) por lo general requiere de poca capacidad para el procesamiento de la información, permitiendo el ahorro de energía y no genera un gran ancho de banda, pero una red IoT no está formada por un dispositivo, son miles de dispositivos conectados en una sola red. Todos estos dispositivos si generan ingentes datos, además, su direccionamiento, conexiones y seguridad, llevan a que las redes IoT sean complejas. Como mencionó Dave Anderson en [3] “La tecnología que hay detrás de cada dispositivo conectado es extremadamente compleja”. Podemos

decir que es un desafío, ya que se tiene un gran número de usuario y dispositivos conectados al mismo punto.

El surgimiento de las SDN se dio por la necesidad de mejorar la arquitectura de las redes tradicionales, prometiendo una mejora en cuanto a su escalabilidad y flexibilidad en la gestión de la red. Las SDN ya están más definidas en el mercado, mientras tanto las redes IoT a pesar de que se conocen desde hace mucho tiempo y cada año se tiene un aumento considerado en la utilización de las mismas, no tienen una arquitectura en común o un estándar especifico y esto trae diferentes inconvenientes como rendimiento, seguridad, etc.

Según [4] existe una brecha en cuanto a las investigaciones de cómo integrar la IoT a otras soluciones de red, por ello, en este trabajo se intenta investigar si puede haber una fusión entre ambas tecnologías SDN - IoT.

1.1 Objetivos Generales y Específicos

1.1.1 Objetivo General

El objetivo principal de esta memoria, es realizar un estudio del estado del arte de la internet de las cosas y de las redes definidas por software, para así comprender su funcionamiento, y poder identificar los aportes de las SDN en entornos IoT. Para conseguirlo, se diseñará un escenario virtual donde los datos transmitidos por dispositivos IoT sean gestionados por el controlador SDN.

1.1.2 Objetivos Específicos

- Estudio del estado del arte de IoT
- Estudio sobre las características principales de SDN
- Investigación sobre las aplicaciones de SDN a la IoT
- Resumen de las diferentes propuestas repasadas
- Diseño de un entorno virtual para simular la red IoT - SDN
- Selección de las herramientas adecuadas para la creación del entorno de pruebas.

1.2 Metodología

Para llevar a cabo este trabajo, se ha realizado un estudio del estado del arte de cada tecnología, con el fin de conocer a detalle el funcionamiento de las mismas. Luego de esto, se revisaron las diferentes propuestas sobre la integración de las SDN en la IoT.

Cabe destacar que al ser tecnologías que se están implementando, existen pocos escenarios o proyectos basados en este tema. Al analizar estas propuestas, se decidió diseñar un escenario virtual, simulando una red residencial. Este escenario se ha desarrollado en una herramienta llamada VNX, donde se ha emulado una red que contiene un Open Vswitch (OVS), un controlador y que este se conecte a una pasarela de IoT.

1.3 Estructura de la Memoria

A continuación, se detallan los capítulos que se describen en cada sección de la memoria.

El capítulo uno (1), se centra en el contexto de la investigación y el objetivos generales y específicos que se pretenden realizar a lo largo de la investigación.

Así, en el Capítulo dos (2) se introducirá una descripción sobre el estado de la internet de las cosas, detallando sus arquitecturas, protocolos y puntos principales, además de una breve revisión de las redes definidas por software, su arquitectura y complementos, esto para comprender la fusión de las mismas.

En el Capítulo tres (3) se detallarán las propuestas que se han investigado, sobre los aportes que trae la SDN a la IoT.

El capítulo cuatro (4), muestra el escenario virtual que se ha diseñado para una red SDN-IoT, además de algunas propuestas adicionales de diseño.

Al final del documento se formulan las conclusiones y recomendaciones, además se enumeran las referencias utilizadas durante la redacción del informe.

2 Estado del Arte

A lo largo de este capítulo se presenta de manera detallada las principales características de las SDN e IoT para así poder analizarlas e identificar, en que aporta las SDN a entornos de IoT. Podemos decir que la IoT tiene una arquitectura en capas, que utiliza múltiples tecnologías en cada nivel y SDN, tiene que ver con el enrutamiento y mejora en la gestión de la red [5]. Esto indica que IoT puede aprovechar los beneficios de control de SDN para mejorar sus complejidades.

2.1 Internet de las Cosas

El término IoT se dio a conocer en 1999, por Kevin Ashton, mediante el Auto-ID grupo de investigación del Instituto de Tecnología de Massachusetts (MIT) donde se hacían estudios sobre la identificación por radiofrecuencia en red (RFID) y tecnologías de sensores. [6]

Es un término muy global, 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. 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 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 de las cuales en este trabajo utilizaremos de la que se habla en [7].

Por lo general, cuando se habla de IoT, se utilizan una serie de dispositivos para formar la red, estos pueden ser sensores, actuadores, microcontroladores, transceptores etiquetas RFID y dispositivos de comunicación inalámbrica conectados a Internet en un entorno inteligente [8]. 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 límites de 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 [8]. 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, Wi-Fi, 6LoWPAN o redes móviles como se muestra en la Figura 1.

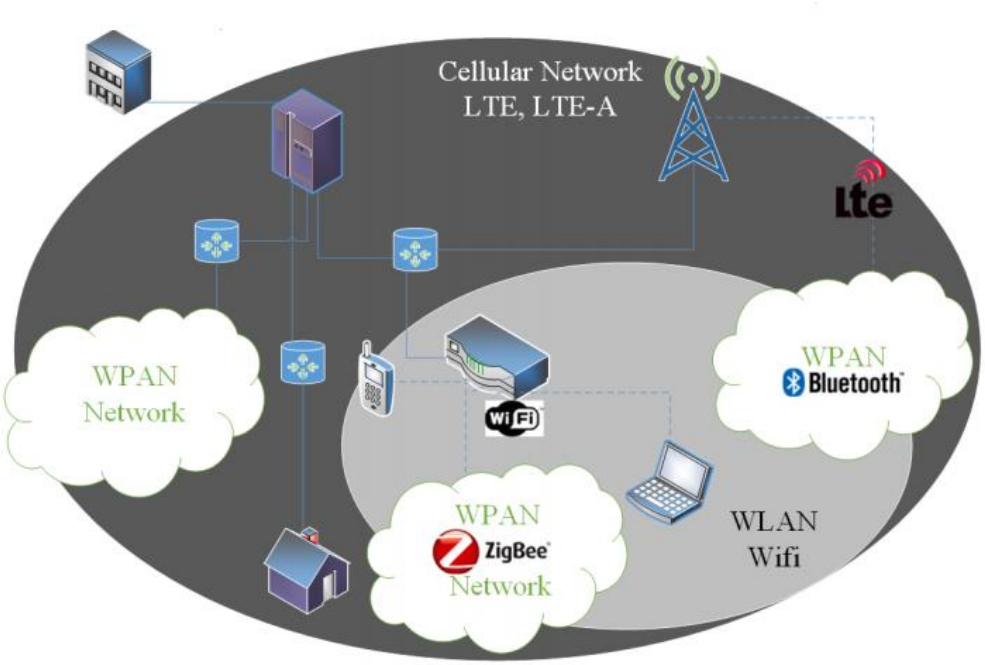

Figura 1. Tecnologías IoT [9]

2.1.1 Arquitectura

En la actualidad no se tiene una arquitectura especifica aplicable a la IoT y se considera que la estandarización de una arquitectura, para que el IoT pueda desarrollarse, puede darse al pasar varios años [4]. De las distintas propuestas revisadas, como he mencionado anteriormente, tomaremos la que se presenta en [7] como una arquitectura de tres capas.

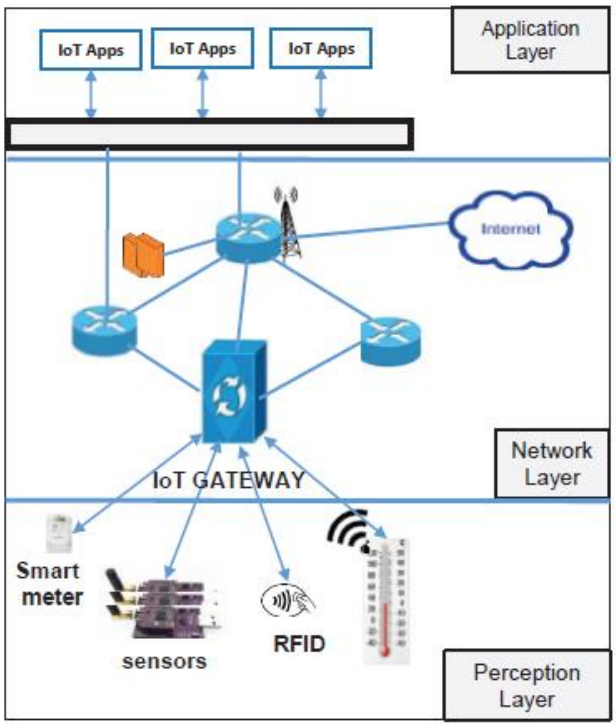

Figura 2. Arquitectura de IoT [10]

Capa de percepción: En esta capa se ubican los dispositivos físicos como sensores, actuadores, etc. Estos recopilan los datos del entorno, además de identificar objetos, utilizando RFID y redes de sensores.

Capa de red: Se encarga de transmitir los datos de los dispositivos, a la puerta de enlace, para su procesamiento de la información recopilada. Para llevar a cabo la transferencia de datos, la capa de red utiliza distintos tipos de tecnologías y protocolo como ZigBee, bluetooth, Wi-Fi, 6LowPAN. Esta capa tiene comunicación y transferencia de datos con la capa de percepción y la capa de aplicación.

Capa de aplicación: Se ocupa de las aplicaciones y servicios que demanda el usuario.

Capa de middleware: Como son dispositivos heterogéneos, esta capa permite, que los dispositivos se puedan comunicar entre sí, traduciendo el mensaje sin ver el hardware. En [7] comentan que esta capa está asociada con la gestión del servicio, el direccionamiento y la denominación del servicio solicitado. Aunque esta capa no está dentro de la arquitectura mencionada la hemos incluido por considerarla importante.

2.1.2 Principales Protocolos

IoT lleva consigo muchos protocolos de red, que son útiles para su funcionamiento, en la Figura. 3 se muestran los mismos clasificados según las capas TCP/IP. Algunos de estos protocolos son ya conocidos y otros se han creado específicamente para IoT. A continuación, se va a dar una breve definición de los más utilizados clasificados por

capa y en la Tabla 1. se muestra una comparación de los protocolos inalámbricos para dispositivos IoT.

Capa de aplicación (comunica el HW con el SW)

- XMPP (Extensible Messaging and Presence Protocol - Protocolo extensible de mensajería y comunicación de presencia)

  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.

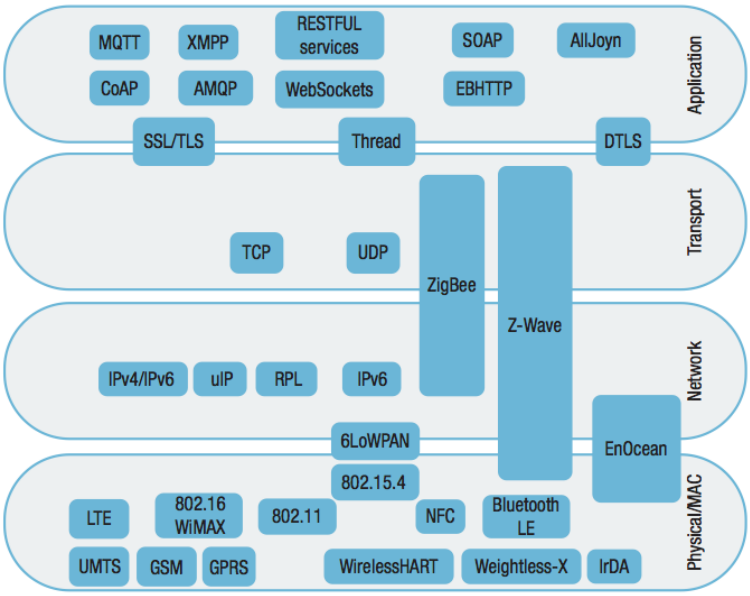

Figura 3. Protocolos de IoT [11]

- CoAP (Constrained Application Protocol - Protocolo restringido de aplicación)

  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. Es de tipo UDP

- MQTT (Message Queuing Telemetry Transport)

  Es un protocolo de la capa de aplicación de tipo publicación/suscripción, diseñado para dispositivos de baja energía, tiene tres modos de

funcionamiento mediante QoS, como menos fiable, más fiable, al menos una vez. Para IoT los subscriptores son las aplicaciones interesadas en datos específicos y los publicadores son sensores que envían la información o datos.

Capa de red

- RPL

  Es un protocolo que surgió, para darle solución a los problemas de enrutamiento, que se tienen en 6LoWPAN. Desde su aparición ha sido el protocolo líder en enrutamiento IPv6, para redes de baja potencia. Es capaz de crear rápidamente rutas de red, distribuir el conocimiento de enrutamiento entre nodos y adaptar la topología de una manera muy eficiente.

Capa física

- Wifi

  Protocolo de la capa de enlace, el mismo está disponible en la mayoría de los edificios comerciales y hogares. Esto es una gran ventaja para los productos IoT dirigidos a esos mercados. No necesitan redes de malla no confiables para extender el rango. Tienen acceso instantáneo a la nube. Tiene un consumo de energía muy alto. Está basado en el estándar 802.11

- Bluetooth Low Energy (Bluetooth de Baja Energía)

  Protocolo de comunicación de la capa de enlace, de corto alcance, que tienen altas velocidades en datos y bajo consumo de energía. Ya viene integrado en los teléfonos inteligentes. Tiene arquitectura maestro/esclavo. Se pueden conectar pocos dispositivos. Con la incorporación de Internet Protocol Support Profile (IPSP), permite conectarse directamente a internet mediante IPv6/6LoWPAN.

Capa red/transporte/física

- Zigbee Smart Energy (Zigbee Energía Inteligente)

  Protocolo de la capa de enlace, diseñado para la mayoría de las implementaciones de IoT, como aplicaciones domóticas e industriales. Solo puede ser administrado por un controlador y admite varias gamas de topologías. Se basa en el estándar IEEE 802.15.4, operando a las 2.4GHz

- Z-Wave

  Protocolo de la capa de enlace de baja potencia, creado para domótica y se ha implementado en IoT, especialmente para hogares inteligentes y pequeños dominios comerciales. Es totalmente compatible con redes de topología de malla, no necesita un nodo coordinador y es muy escalable. Utiliza CSMA / CA para detectar colisiones y mensajes ACK, para una transmisión confiable. Estándar: Z-Wave Alliance ZAD12837 / ITU-T G.9959

Protocolos inalámbricos para IoT	Frecuencia	Velocidad de datos	Topología	¿Propio o abierto?	Gestionado	Seguridad	Uso previsto
ZigBee	2,4 GHz, 915 MHz (EE. UU.), 868 MHz (UE)	250 kbps (2.4) 40 kbps (915) 20 kbps (868)	Malla	Abierto	ZigBee Alliance (Comcast, Kroger, Samsung, TI)	Encriptado	Productos de bajo costo dirigidos en el hogar
Z-Wave	915 MHz (EE. UU.) 868 MHz (UE)	40 kbps (915) 20 kbps (868)	Malla	Propiedad	Alianza Z-Wave	Encriptado	Productos para el hogar
Bluetooth de baja energía (BLE)	2.4 GHz	10kB / s	PAN	Abierto	Grupo de interés especial de Bluetooth (3k miembros)	Encriptado	Productos portátiles
Wifi	2.4GHZ / 5GHz	7Gbps	Estrella	Abierto	IEEE	Opcional	Productos independientes dirigidos a la casa o el negocio
6LoWPAN	2.4 GHz	250 kbps	Malla	Abierto	IEEE	Opcional	Productos dirigidos a la casa o edificios comerciales que necesitan comunicarse con otros productos o sistemas.

Tabla 1. Comparación de protocolos inalámbricos para dispositivos IoT [12]

2.1.3 6LoWPAN

Por la importancia que tiene el protocolo 6LoWPAN con respecto a las redes IoT vamos a ver su arquitectura y funcionamiento con más detalles. Es un protocolo de capa de red o adaptación que tiene objetivo que los paquetes de protocolo de internet se puedan llevar de manera eficiente dentro de pequeñas tramas de capa de enlace de datos como los que están definido por IEEE 802.15.4.

Entendiendo que todo estará conectado, cada dispositivo debe tener una dirección IP que lo identifique y de esta manera conectarse a internet de diferentes maneras. En este caso, para asignar las direcciones IP a cada dispositivo, se utiliza IPv6, pero el mismo fue diseñado para redes de escenarios abundantes (por ejemplo, Ethernet), mientras que en IoT en general, los entornos de red son significativamente más limitados, por esto se creó el protocolo de capa de adaptación 6LoWPAN.

Es un protocolo de encapsulamiento de la capa de red, el cual permite que los nodos o dispositivos de una red inalámbrica, puedan comunicarse directamente con otros dispositivos IP.

6LoWPAN permite que los dispositivos más pequeños, con capacidad de procesamiento limitada y de bajo consumo, puedan transmitir datos de forma inalámbrica, utilizando un protocolo de internet. Se identifica por admitir un ancho de banda bajo, distintos tipos de topología, en general malla.

Se caracteriza por el envío eficiente de paquetes IPv6, dentro de tramas pequeñas de la capa de enlace de datos, como las definidas en el estándar IEEE 802.15.4.

IETF (Internet Engineering Task Force), es una organización que se encarga de brindar aportes a la ingeniería de internet en cuanto a encaminamiento y transporte. Desarrolló el estándar IEEE 802.15.4 para encapsular o fragmentar de manera eficiente los datagramas IPv6, de tal manera que no excedan de los 128 bytes.

La capa inferior de la tecnología 6LoWPAN, adopta los estándares de capa física PHY y la capa de acceso al medio MAC de IEEE 802.15.4. estas brindan los servicios de transmisión de datos. Como se muestra en la Figura

Para utilizar IoT se requiere IPv6, las direcciones IPv6 son de 128 bits en lugar de 32 bits como se da en IPv4. Con esta cantidad de direcciones se puede tener un mayor número de equipos conectados. Pero estas al ser muy extensas no caben en las tramas de datos de IoT, que son mucho más pequeñas. Por este motivo surge el protocolo de la capa de red 6LoWPAN.

IETF logró fragmentar los paquetes, introduciendo una capa de adaptación entre las capas de red y física de TCP/IP. Figura 4. Adicional a esto, se restructuró el formato de los paquetes IPv6.

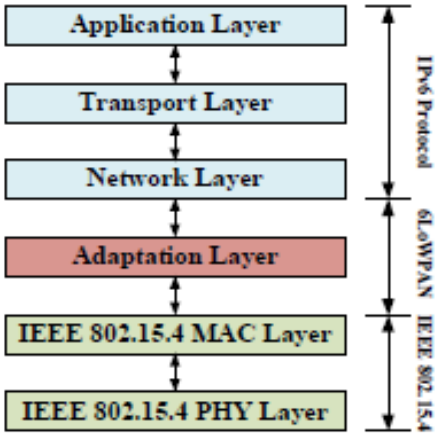

Figura 4. Pila de protocolo 6LoWPAN [13]

A pesar que 6LoWPAN fue creado para redes inalámbricas de baja potencia IEEE 802.15.4 en la banda de 2,4 GHz, actualmente se están adaptando y utilizando una variedad de otros medios de redes incluyendo RF de baja potencia, Bluetooth, Inteligente, PLC y Wi-Fi de baja potencia.

Arquitectura

Un sistema 6LoWPAN se compone normalmente, de muchos nodos de baja potencia en forma de malla. Los nodos pueden ser intercambiados libremente, mediante el uso de routers frontera, que actúan como puerta de enlace hacia el resto del internet.

Las aplicaciones que se ejecutan en los nodos, pueden pasar los paquetes IP a un servidor en internet, sin pensar en el router frontera, el cual retransmite los paquetes de la red 6LoWPAN a la internet.

Las LoWPAN no tienen conocimientos de otras redes, no transportan tráfico de otras y se comunican mediante routers frontera, como se mencionó. A este tipo de redes se les llama stub, ya que solo pueden enviar o recibir paquetes IP. No transportan tráfico de otras redes externas través de ellas.

Están conformadas por dos tipos de dispositivos, nodos de rutado, los cuales son los que permiten, la conexión de paquetes a los demás dispositivos de la red LoWPAN y al router frontera 6LoWPaN y los nodos terminales solo pueden recibir datos, cada cierto tiempo le preguntan al dispositivo al que está conectado, si tiene algo pendiente, al no estar siempre pendiente, permite ahorro de consumo energético.

El router frontera, es quien se encarga del intercambio de datos entre los dispositivos 6LoWPAN y las redes IPv6, como también entre estos dispositivos e Internet directamente. Estos routers, son los encargados de direccionar los datos a nivel de capa de red. Los routers frontera también tienen que convertir los paquetes IPv4 en IPv6, ya que la implementación de IPv6 sobre IEEE 802.15.4, no pueden realizar una comunicación directa con redes IPv4. Por lo tanto, los routers frontera, deben ser capaces de convertir los paquetes IPv4, en paquetes IPv6 y viceversa, para ello se usa NAT64 [14].

Las redes 6LoWPAN permiten tres tipos de configuraciones según [15] y se muestra en la Figura 6.

Simple LoWPAN, red con un único router. Se conecta mediante el router frontera a otra red IP.

Extended LoWPAN, varios routers frontera, compartiendo un enlace troncal (Ethernet) que los interconecta. Además, tienen un servidor que los gestiona. Se utilizan para redes de gran tamaño y así separarlas en redes independientes.

Ad-hoc LoWPAN, no hay routers, forma parte de una red local exclusiva. No está conectada a internet, opera sin una infraestructura.

En general 6LoWPAN, permite la conectividad ubicua e interoperabilidad entre dispositivos heterogéneos y se utiliza en IoT, para redes de área personal inalámbrica de baja potencia.

Su estándar se basa en el documento RFC6282 y su frecuencia es adaptable a múltiples capas físicas, como Bluetooth Smart (2.4GHz), ZigBee o comunicación RF de bajo consumo (sub-1GHz)

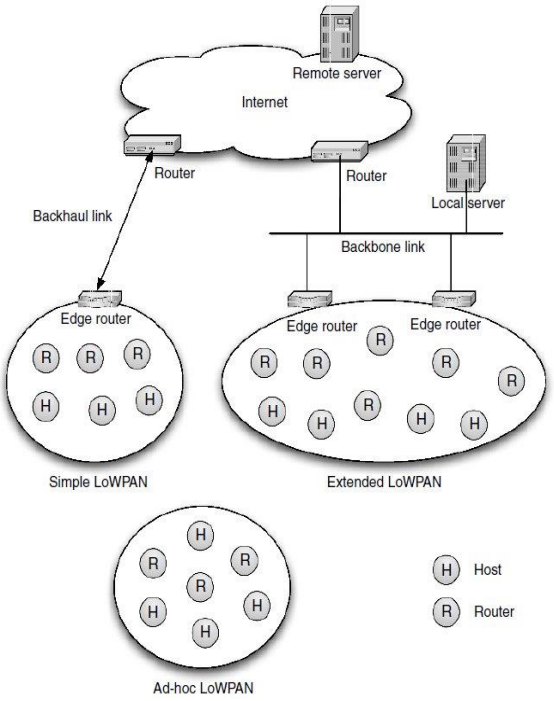

Figura 5. Arquitectura de una red 6LoWPAN

Características

- Fragmentación de los paquetes:

El mismo se debe evitar, ya que generan un mayor consumo de energía, en este caso, el receptor debe estar activo, para recolectar el paquete original y en cuanto a almacenamiento, porque el dispositivo tiene que reservar un espacio de memoria, para el tamaño del paquete final, los cuales son limitados para dispositivos 802.15.

Los paquetes IPv6 tienen 40bytes de cabecera, el objetivo de IETF es comprimir esos 40 bytes para así aprovechar el uso que los dispositivos hacen en la red. Para esto se

decidió crear una cabecera llamada IPHC, con el fin de unificar en el menor tamaño posible, toda la capacidad de direccionamiento que hay en Ipv6.

El estándar IEEE 802.15.4 dispone de un MTU de 127 bytes, mientras que el de IPv6 es de 1280 bytes. En el caso de 6LoWPAN, se decidió que el MTU mínimo fuera de 1280 bytes.

Todas las redes que tengan un MTU inferior, al tamaño de paquete que se quiere enviar, necesitan fragmentación. La misma consiste en dividir el paquete en varios segmentos más chicos e introducir la información adicional, en el header de cada segmento, para la reconstrucción exacta del paquete original, cuando llegue al destino. Cuando todos los segmentos hayan llegado al destino, se procede con la unión del paquete, quitando información adicional recuperando el formato que tenía.

En 6LoWPAN la fragmentación es de tipo mesh-routing (malla de enrutamiento). No está permitida la fragmentación en un nodo intermedio (solo el nodo fuente y el nodo destino). El nodo que envía el paquete de primero, es quien decide si el paquete se fragmenta o no, pero debe saber cuál es el MTU del camino que seguirá el paquete. Para conocer el MTU se utiliza PMTUD (Path MTU Detection) como en IPv6 el cual permite al nodo conocer el mínimo MTU, de todos los nodos identificando así, cual es el MTU del camino. Con esto se aprovecha para elegir de todos los caminos, el MTU con mayor capacidad y así evitar la fragmentación del paquete.

Cada fragmento debe estar formado por el contenido del paquete y un encabezamiento con la información necesaria, para la reconstrucción.

- Capa de adaptación:

Como el marco IEEE 802.15.4 no puede empaquetar todo el paquete de datos de IPv6, es necesario introducir la capa de adaptación, para lograr las funciones de fragmentación y reensamblaje por esto, 6LoWPAN además de las capas del modelo OSI, añade una capa más en medio de la capa de enlace y la capa de red. Fig.2.

El objetivo de esta capa de adaptación, es darle solución a los inconvenientes que estaba teniendo el estándar. La capa de adaptación es quien decide, cual es el nodo a nivel de capa de conexión, que se le dirigirán los datos hasta el destino final.

- Generación automática de direcciones IP:

Stateless auto- configuration, la configuración automática en IPv6, permite un dispositivo, para generar automáticamente su dirección IP de cada miembro de la red 6LoWPAN, sin ninguna interacción externa con un servidor DHCP o de realizar la configuración manualmente.

Esta es una las características que se distinguen en 6LoWPAN, su capacidad de la asignación dinámica de direcciones cortas de 16 bits. Al utilizar esta dirección corta, se puede emplear el enrutamiento jerárquico.

- Comprensión de las cabeceras:

Como tenemos un sistema con muchos dispositivos, es necesario reducir el uso que cada dispositivo hace de la red, por esta razón se ha llegado a generar un nuevo tipo de cabecera estandarizado por el IEFT, que permite comprimir de los 42 bytes de tamaño de cabecera de una ipv6 normal, a sólo 6 bytes en su mínima implementación. Para esto se creó la cabecera IPCH, con la cual se reduce la capacidad de direccionamiento de IPv6, la misma cuenta con un campo Dispatch, que cuenta con 24 bits con estos, el dispositivo identifica que tipo de paquete va a leer en los próximos bits.

- Autoconfiguración de la red usando Neighbor Discovery

Un nodo que se acaba de conectar a la red, descubre la presencia de otros nodos en el mismo enlace, además de ver sus direcciones IP. ND ofrece detección de prefijo de subred, resolución de direcciones y detección de direcciones duplicadas, fue diseñado para escenarios de red transitivos, de alta energía y de ancho de banda, donde la red de nodos y las interfaces están siempre activadas, como también el uso agresivo de multidifusión de ND, es una buena opción. A pesar de que ND es útil para reducir las operaciones que hace el usuario y que es una buena opción para entorno IoT, no se recomienda para redes de nodos restringidos, que a menudo están en ciclo de trabajo, ya que ofrecen enlaces no transitivos, y asignación por multidifusión IPv6 a la difusión de toda la red. El ND optimizado para 6LoWPAN (6LoWPAN ND) es compatible con hosts en reposo (dormidos) al proporcionar interacción iniciada por el host con enrutadores y minimiza el uso de multicast. Además, 6LoWPAN ND proporciona difusión de contexto opcional para 6LoWPAN HC.

- Soporta unicast, multicast y broadcast
- Soporte para IP routing (IETF RPL)
- Soporte para el uso de link-layer mesh (malla de capa de enlace)
- Problema de seguridad

Como el uso de mecanismo de seguridad requiere alto procesamiento y ancho de banda, no es adecuado para estos equipos de baja potencia. Los mecanismos de seguridad que ofrece IEEE 802.15.4 en la capa de enlace son un poco flexibles, por lo que se comenta que la seguridad se debe mejorar en el protocolo 6LowPAN.

2.1.4 IEEE 802.15.4

IEEE 802.15.4 se creó para sistemas con baja transmisión de datos, procesamiento baja, larga duración en la alimentación y bajo costo. Es el estándar IoT más utilizado para MAC. Define los encabezados que incluyen direcciones de origen y destino, y cómo los nodos se pueden comunicar entre sí. [16] Cumple con los requisitos de comunicación de IoT como alta fiabilidad, bajo costo, sincronización de tiempo, formación de red.

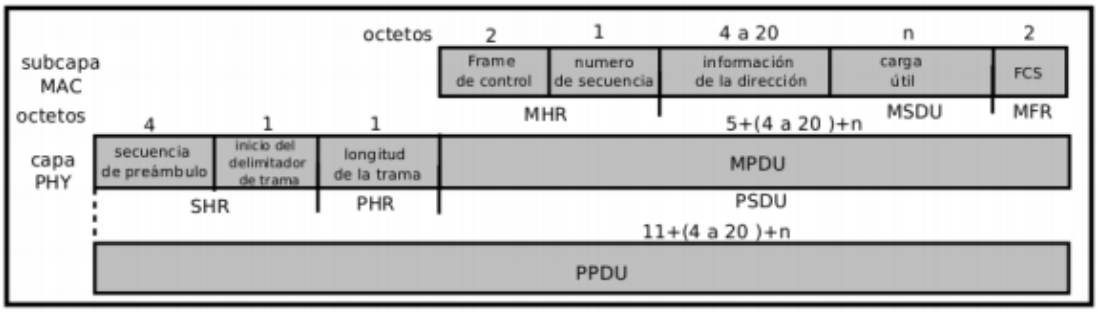

Figura 6. Cabecera 802.15.4

Como mencionamos anteriormente 6LoWPAN utiliza para el encabezado lo establecido en IEEE 802.15.4 capa física (PHY) y capa MAC

PHY es la primera cabecera que lee el dispositivo cuando le llega un paquete, la misma posee los bits de control para la lectura del resto de la cabecera. Luego de leer el Frame Lenght, leerá el resto de los paquetes con los bytes anteriores. El MPDU de un paquete en 6loWPAN debe ser menor que el MTU, es decir, el máximo número de bytes que se debe transmitir en un paquete en una red WPAN debe ser 127, esto se detalla más adelante.

Posee algunas características como:

Estructura del Slotframe: La estructura de trama IEEE 802.15.4 está diseñada indicarle a cada nodo qué hacer y que los mismos sean programados. Un nodo puede estar en reposo, enviar o recibir información. Cuando el nodo está en reposo, se apaga para estar en modo ahorro, en ese caso almacena todos los mensajes que requiera enviar cuando tenga otra transmisión. Al transmitir, envía sus datos y espera una respuesta de recibido. Al recibir, el nodo se enciende antes de la hora de recepción programada, recibe los datos, envía un mensaje de recibido, entrega los datos a las capas superiores y vuelve a estar en reposo.

Sincronización: Los nodos con sus vecinos y las puertas de enlace requieren tener conectividad. Se tiene dos tipos de conectividad, una con mensaje de recibido que consiste en que los nodos ya están operativos y en comunicación y envían el mensaje

de recibido y la otra en cuadros y el modo en cuadros, los nodos no están en comunicación.

Salto de canales: Esto es con el fin de acceder por intervalos de tiempo al medio inalámbrico. Se necesita cambiar el canal de la frecuencia de manera aleatoria, permitiendo así que no existan interferencias y que se tenga diversidad de frecuencias.

Este estándar determina las comunicaciones en la capa física y de control de acceso al medio. Actualmente es utilizado por otro protocolo además de 6LoWPAN llamado Zigbee. Este estándar define la capa física, la frecuencia, potencia, modulación y condiciones del enlace inalámbrico. [16]

2.1.5 Sistemas Operativos

Como se menciona en [17] la IoT tiene dispositivos de alta gama como Rasberry Pi que pueden utilizar sistemas operativos como Linux y dispositivos de gama baja con menos recursos y se requiere otro tipo de sistema operativo. La mayoría de los dispositivos son de gama baja.

Vamos a mencionar los sistemas operativos de IoT más comunes en el mercado y que son de código abierto.

Contiki

Está diseñado para sistemas con escasa memoria, conecta dispositivos pequeños de bajo costo y de bajo consumo de energía, se utiliza en sistemas inalámbricos. Se ejecuta en microcontroladores de 8 y 16 bits y dispositivos con ARM de 32 bits.

Es compatible con IPv6 e IPv4 y también con protocolos de la IETF, recientemente estandarizados como 6lowpan, RPL, CoAP. Contiene un simulador llamado Cooja, que te permite crear entornos de simulación. [18]

RIOT

Se puede ejecutar en microcontroladores de 32, 16 y 8 bits. Se puede utilizar en la mayoría de los dispositivos de bajo consumo de IoT. Compatible con 6lowpan, IPv6, RPL, CoAP.

Se conoce como "*el Linux de la Internet de las Cosas*". Se basa en una arquitectura microkernel y tiene la ventaja que permite la programación de aplicaciones y ofrece capacidades de multihilo y en tiempo real. [19]

TinyOS

Fue diseñado para redes de sensores inalámbricas, soporta plataforma HW de 8 y 16 bits. Está escrito en un lenguaje de progrmación de C llamado nesC. Implementa la pila 6LoWPAN y provee un ambiente de simulación llamado TOSSIM

Tinycore

No es considerado como sistema operativo de IoT. Es un Sistema Operativo que se basa en el kernel de Linux 2.6 y por ser tan pequeño (10MB), se puede instalar en un CD o USB, funciona para simular un dispositivo de IoT en un entorno virtual.

Operating System	Action	Programming Language	RAM required (Kb)	Kernel Implementation	Service management	Kernel management	Model
Contiki	Event based	C	2	Preemptive multithreading	Dynamic	Hybrid	-
RIOT OS	Task based	C/C++	1.5	Multithreading	Dynamic	Static	-
TinyOS	Event	NesC	1	Partial	Static	Dynamic	Concurrency model

Tabla 2. Comparativa de Sistemas Operativos de IoT [4]

2.2 Redes Definidas por Software

Las redes SDN han traído un cambio de paradigma en las redes de comunicación. Tienen como objetivo facilitar la implementación e implantación de servicios de red de una forma dinámica y escalable y evitar que el administrador de red, tenga que gestionar estos servicios [7]. La infraestructura de red tradicional se caracteriza, por ser bastante rígida y SDN se caracteriza por su facilidad de programación en la infraestructura de red.

2.2.1 Arquitectura

En una red tradicional el plano de control y el plano de datos están en un solo dispositivo y se gestiona de manera individual en cada enrutador o conmutador.

El plano de control se encarga de tener las políticas de reenvío de datos, como tablas de enrutamiento, tablas de topología, etc., es el que decide que acción tomar, al recibir un paquete con cierto flujo de tráfico y el plano de datos, se encarga de realizar el procedimiento de reenvío.

SDN propone la separación del plano de control y el plano de datos, como se muestra en la figura. Ahora se tiene un controlador SDN, dispositivo central de la red, que se encarga de gestionar todos los dispositivos, a través de la centralización del plano de control y el plano de datos, gestiona la transferencia de paquetes, desde equipos de red, tales como conmutadores y enrutadores, a través de tablas de enrutamiento, con las decisiones que recibe desde el plan de control.

La arquitectura SDN, como mostramos en la Figura 7, se divide en tres capas:

Capa de aplicación: Es un conjunto de aplicaciones software, que indican sus requisitos de red a la capa de control de SDN, mediante una interfaz y este lo envían los dispositivos de red.

Capa de control: Aquí se encuentran los controladores SDN, quienes tienen la administración de la red, también se le dicen que son el cerebro de la red. El controlador se comunica con las otras capas, mediante interfaces y protocolos específicos.

Capa de infraestructura: En esta capa se encuentran los diferentes nodos de red y se toman algunas decisiones de transferencia de datos, para reenviar el tráfico o procesarlo.

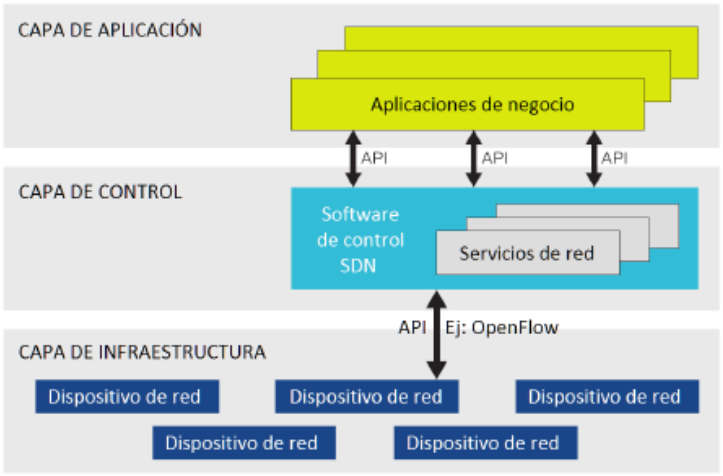

Figura 7. Arquitectura de SDN

Desde el plano de control se diferencian dos interfaces de programación, como se muestra en la Figura 8, estas permiten la comunicación entre la capa de control y las diferentes capas.

Southbound APIs son todos los mensajes que van desde el controlador hasta los dispositivos de red y viceversa. Los conmutadores le preguntan al controlador que hacer con los paquetes, mediante el protocolo OpenFlow.

Northbound APIs son las aplicaciones que se conectan directamente, con el controlador y este se comunica con los dispositivos.

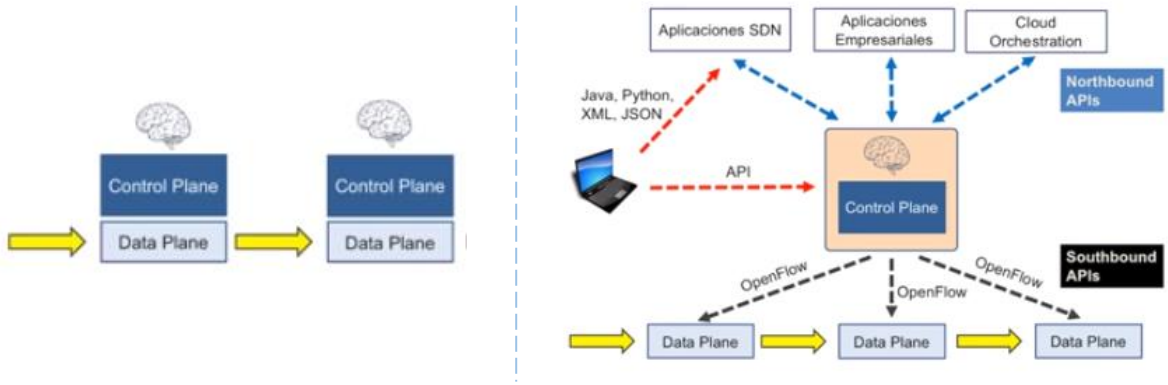

Figura 8. Comparativa de red tradicional y SDN [20]

OpenFlow es un protocolo que permite la comunicación entre el plano de control y el plano de datos mediante Southboud API. Este se encarga de enviar los paquetes por la interfaz adecuada a los conmutadores o enrutadores. La mayoría de los sistemas SDN utilizan OpenFlow.

Como hemos visto, las SDN prometen ser una buena solución para los distintos inconvenientes que están teniendo la internet de las cosas, por esto en el capítulo siguiente podremos ver como se fusionan están tecnologías y qué aportes nos traen.

3 Propuestas de Aplicación de SDN a IoT

Ahora que tenemos claros los conceptos más relevantes de la IoT y de dar un repaso sobre las SDN, vamos a describir las diferentes propuestas que hemos revisado de investigaciones sobre la fusión de estas tecnologías, aportes, arquitecturas y aplicaciones principales.

3.1.1 Aportes de SDN a la IoT

En esta sección vamos a mencionar los principales aportes de SDN para las tecnologías de IoT. [7]

- Visibilidad de la red: La administración de la red es simplificada para el usuario, dispositivos y aplicaciones.

- Gestión de la Red: En la IoT se tiene una gran cantidad de equipos conectados, lo cual lleva consigo, una generación de trafico enorme, estos datos requieren ser procesados a tiempo y de manera eficiente. Por este motivo es de suma importancia, la gestión de la red (gestionar la gran cantidad de dispositivos y la información que es generada por ellos). Se requiere el uso de tecnologías adecuadas, para distribuir y controlar el flujo de tráfico en las redes, su balance de carga y minimizar el tiempo de respuesta de la red. Aquí entra SDN, ya que nos permite tener la visión global de la red, de una manera centralizada, ofrece mayores posibilidades para encaminar de forma inteligente el tráfico. Resumiendo, las tecnologías basadas en SDN, se pueden aplicar para la gestión de la red IoT en los siguientes aspectos: balance de carga, mejor utilización del ancho de banda, enrutamiento inteligente.

- Virtualización de la red: NFV, es una tecnología que permite que los dispositivos o equipos físicos actuales, utilizados en la red (router, firewalls, balanceadores de carga, etc.) sean virtualizados mediante el uso de software y técnicas basadas en la nube estando ahora en máquinas virtuales. Los enfoques basados en SDN, juegan un papel importante, para la realización del concepto de NFV en una red de la IOT a gran escala. Por ejemplo, una ciudad inteligente requiere de varios centros de datos, oficina central, etc. y esto lleva a tener costes de hardware, interfaz de gestión y personal especializado, para mantenimiento del hardware y si se necesita aumentar la red, estos costos también van aumentando. Con NFV estas funciones pueden estar en una sola plataforma de Hardware de computación, como aplicaciones de SW y esto nos permite ahorro en costo, espacio, consumo de energía, despliegue rápido de las funciones, y mejor administración remota.

- Acceso a la información desde cualquier lugar: Para la IoT es una necesidad, que los usuarios puedan acceder a los distintos dispositivos que tengan conectados a la red, desde el lugar donde se encuentren, teniendo las facilidades de controlar y modificar las funciones o configuración de sus dispositivos, dependiendo de sus necesidades de una manera transparente. Esto se puede controlar, con la ayuda de las tecnologías basadas en SDN, con la centralización de la red.

- Flexibilidad: Como se van a manejar tantos datos, se requiere un mapeo eficiente de las solicitudes de los usuarios, para una mejor utilización de los recursos, a fin de maximizar la utilidad de la red y evitar que la red no sea eficiente en cuanto a su rendimiento. En SDN, el reenvío de tráfico basado en reglas de flujo, ayuda a mejorar la utilización de los recursos de la red. La solicitud de múltiples usuarios puede reenviarse a través de la ruta deseada, de acuerdo con las reglas de flujo decididas por el controlador SDN. SDN brinda flexibilidad en la configuración permite cambiar dinámicamente el comportamiento de la red, en función de los patrones de tráfico, los incidentes detectados y los cambios de políticas de seguridad, sobre todos los dispositivos conectados en un entorno de IoT.

- Gestión de la energía: Para el procesamiento de la gran cantidad de datos que genera la IoT, se requiere una gran cantidad de centros de datos, que recopilen la información generada por miles de dispositivos. Para alimentar estos centros de datos, se consumirá una gran cantidad de energía. Los sistemas de administración de energía inteligente, también deben garantizarse para una red de centros de datos eficiente energéticamente. En redes de centros de datos basadas en SDN, el tráfico se puede asignar a los servidores adecuados de manera eficiente o bien utilizar NFV.

- Seguridad y privacidad: Para la IoT la seguridad es de suma importancia y en la actualidad es más bien un problema, ya que deben proteger los datos, para que miles de dispositivos, proveedores y usuarios participen en una sola plataforma y que los usuarios tengan la seguridad de la privacidad de sus datos [8]. Como gran parte de los dispositivos utilizan redes wifi tienen más vulnerabilidad. El control detallado de los flujos que utilizan SDN mejora la seguridad y la privacidad del tráfico de la red.

- Escalabilidad: Como una red IoT tiene un gran número de dispositivos conectados y se espera que crezca exponencialmente, se requiere una infraestructura de redes escalable. SDN, permite la automatización de sistemas con protocolos que permitan la gestión de flujos de datos.

- Programabilidad: Debido a la visión global de la red SDN, ofrece la oportunidad de configurar nuevas funcionalidades y aplicaciones de red sin necesidad de configurarlo en cada dispositivo.

- Interoperabilidad: Como hemos mencionado anteriormente en la IoT se tienen el problema de heterogeneidad por las diversas tecnologías, formatos de datos y protocolos que utilizan los dispositivos, este desajuste puede afectar el rendimiento de la red. Según [10], la flexibilidad que brinda SDN permite que varios objetos conectados a redes heterogenias se comuniquen entre sí y asé se podrá tener conexiones simultáneas de varias tecnologías de comunicación.

3.1.2 Propuestas de Aplicación de SDN

En el estudio realizado hemos encontrado distintas propuestas sobre las aplicaciones de SDN a IoT. Hemos seleccionado algunas de ellas para presentarlas en esta memoria, cabe destacar que la seguridad es un campo muy interesante para la IoT que se puede mejorar con SDN, pero en este trabajo no entraremos en detalle sobe esta aplicación.

Propuesta 1

En [10] presentan una arquitectura de SDN – IoT como la que se muestra en la Figura [9] donde el administrador puede gestionar, todos los servicios de la red, por medio un controlador SDN. Ahora la capa de red tendrá diferentes redes de acceso, que proporcionan conectividad a dispositivos, por medio de redes heterogéneas. En la Figura [2] vemos una arquitectura típica de IoT, ahora esa capa de red se clasifica en la capa de datos y capa de control.

En la capa de datos "*Data Plane*", están ubicados los conmutadores y enrutadores SDN, estos reciben y transmiten información de la capa de percepción, pero no toman ninguna decisión. La decisión la tiene la capa de control, la cual se comunica con los conmutadores y enrutadores programables, por medio de *Southbound APIs* con OpenFlow u otros protocolos similares. La capa de control, es la capa de mayor importancia en esta arquitectura, ya que, por tener la visión global de la red, trae soluciones en términos de desarrollo e implementación de sistemas. Tiene muchas funcionalidades como, calcular las reglas para el reenvío de paquetes, esquemas de planificación y enrutamiento, servicios de red de importancia como QoS, movilidad, equilibrio de carga, los cuales pasan a la capa de aplicación por medio de *Northbound APIs.*

Los autores comentan que el objetivo principal de la arquitectura descrita anteriormente, es sustituir las pasarelas tradicionales de IoT, por una pasarela SDN.

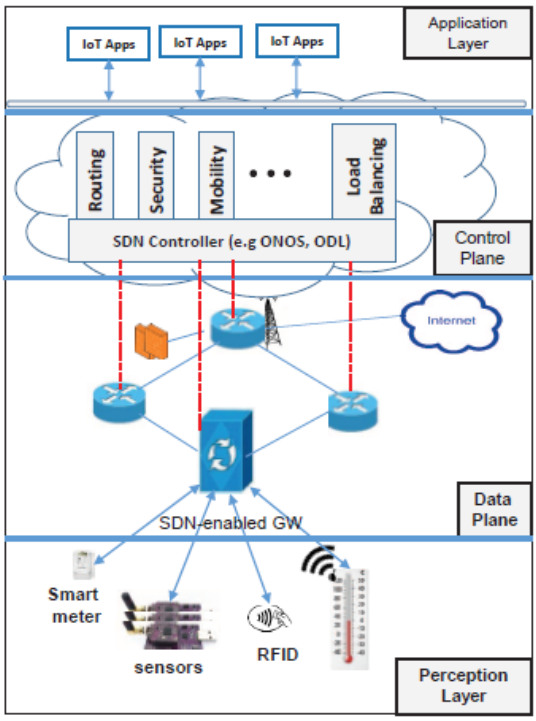

Figura 9. Arquitectura SDN - IoT [10]

Introducir una *SDN-enabled GW* como se muestra en la imagen [9] ofrece muchas ventajas. Los sensores/actuadores se conectan a la red, mediante pasarelas con tecnologías como 6LowPAN, Bluetooth, ZigBee e incluso cableado y esto lleva a la puerta de enlace de IoT (*IoT Gateway)* a tener flexibilidad, para gestionar todos los datos, mantener la QoS, seguridad, entre otras cosas. Y como no es posible que pueda abarcar la heterogeneidad de la red, se sustituye por una puerta de enlace con SDN, la cual tiene la facilidad de manejar grandes cantidades de tráfico, provenientes de diferentes objetos y poseen técnicas/protocolos de enrutamiento inteligentes, para llevar el tráfico a través de las rutas menos congestionadas. También comentan que, si a esta arquitectura se le añade NFV, se puede solucionar el problema de escalabilidad de las redes IoT.

En [6] sostienen, si se le da la responsabilidad de tomar decisiones de enrutamiento al controlador, quien tiene la gestión global de la red, se reduce la sobrecarga y esto podría reducir el costo, ya que no se tendría que tener tantos dispositivos de enrutamiento en la red.

Propuesta 2

En [7] proponen una arquitectura SDN-IoT similar que [21] pero indican que las arquitecturas con un solo controlador son poco prácticas, que se debería utilizar un esquema distribuido con múltiples controladores. También argumentan que se debe

considerar la movilidad, ya que los dispositivos IoT se mueven con frecuencia, desde un punto de acceso a otro.

Ponen como ejemplo una red en diferentes áreas geográficas (particiones de red) y que cada una tenga un controlador local, que solo tiene la visión de la red en la que se encuentra. Según ellos, los problemas de escalabilidad se pueden resolver, asignando cada dispositivo de IoT, a un controlador basado en un "algoritmo de hashing distribuido", y para manejar los traspasos de dispositivos, se utiliza un protocolo de coordinación especial entre los controladores separados. Luego que se han asignado los dispositivos a las diferentes particiones de red, una unidad especial, que tiene en cuenta los requisitos de servicio de cada flujo y la condición de cada partición de red en términos de carga, reasigna los dispositivos para acceder a otros puntos. Un ejemplo real para esta propuesta es una Red Ad Hoc Vehicular (VANET). En la cual los vehículos pueden comunicarse entre sí. Propone utilizar SDN en una red VANET, donde los conmutadores SDN son los vehículos y dispositivos en la carretera, reciben los mensajes de control de la unidad central para realizar las acciones de enrutamiento.

Propuesta 3

Según [22] la integración de SDN en IoT, resuelve la rigidez de la arquitectura de red tradicional (heterogeneidad de la red), pero no se enfoca en la heterogeneidad de los dispositivos. Por esto proponen una solución, que se basa en la utilización de Docker implementados en dispositivos. Docker permite crear contenedores con las aplicaciones que sean necesarios y utilizarlos en cualquier máquina que tenga docker instalado, sin necesidad de descargas o actualizaciones, es un enfoque ligero, que brinda portabilidad a los desarrolladores, para crear e implementar aplicaciones sin complejidad.

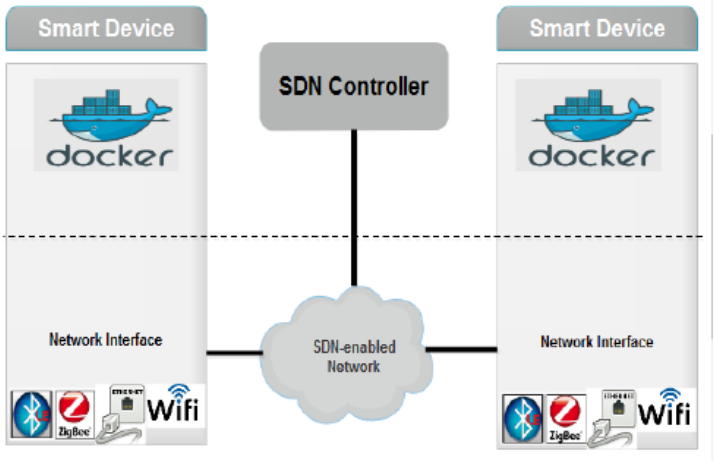

Figura 9. SDN- Docker -IoT

Ellos validaron sus propuestas, mediante una arquitectura, que establecía comunicación entre dispositivos diferentes, a través de una red basada en SDN. El escenario fue emulado en Mininet y el uso de un controlador de SDN centralizado, observando la conectividad entre dispositivos heterogéneos, vinculados a redes heterogéneas, a través de diferentes flujos de tráfico, comprobando la estabilidad en diferentes escenarios con host diferentes. El escenario que proponen costa de servidor host, cliente host, docker actuando como host, un conmutador virtual y un controlador. El objetivo de su escenario, figura 10 es comprobar la facilidad de añadir, cambiar o eliminar host, conmutadores y docker de la red, al hacer esos cambios verificaban la inactividad UDP, tráfico TCP y retraso entre dispositivos. Según los resultados la conectividad entre todos los hosts se mantiene gracias al contenedor docker implementado en dispositivos inteligentes.

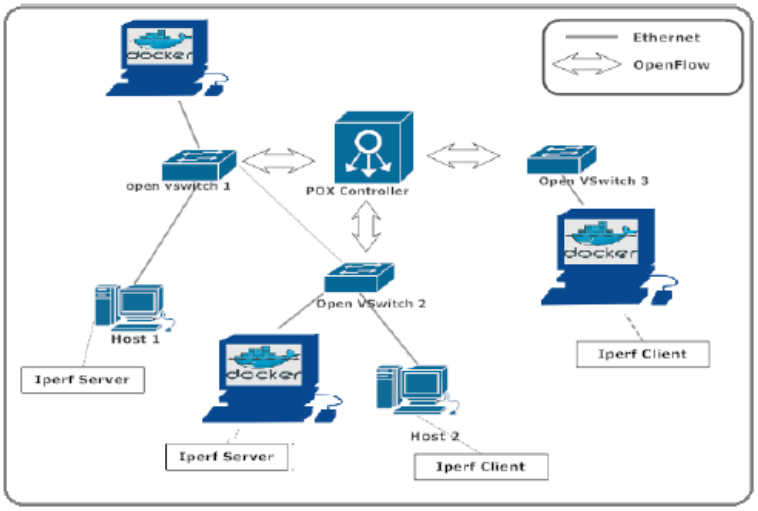

Figura 10. Arquitectura – Pruebas

Propuesta 4

En [4] presentan el marco UbiFlow el cual tiene arquitectura integrando la SDN y la IoT. El enfoque es similar al mencionado en [24] con respecto la división de la red IoT y la movilidad. Proponen un "control de flujo eficiente y la gestión de la movilidad en múltiples redes con controladores distribuidos de SDN"

La red se divide en varios segmentos y cada segmento tiene su controlador SDN físicamente distribuido. Los dispositivos IoT pueden estar conectados a diferentes puntos de acceso y los controladores distribuidos se comunican para la administración de la movilidad y para seleccionar a que puntos de acceso, se debe conectar el dispositivo, de manera escalable y confiable. Brindan balanceo de carga para la red IoT.

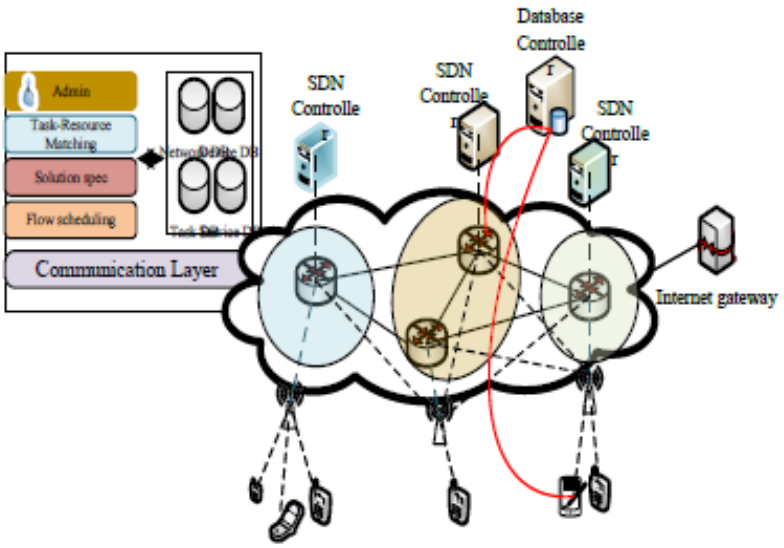

Figura 11. Arquitectura Ubiflow

Propuesta 5

En [23] proponen una arquitectura SDN – IoT para solucionar el problema de la gran cantidad de datos (big data). Su arquitectura consta de cuatro capas como se muestra en la imagen.

Percepción, puerta de enlace, red (donde se encuentra el controlador) y aplicación. Figura [11].

Capa de percepción: sensores y actuadores, los cuales evalúan el entorno y envían datos a las capas superiores. La big data ocurre en esta capa y las tecnologías típicas de IoT como Bluetooh o Zigbee, requieren de la puerta de enlace para enviar los paquetes a internet. Estos dispositivos envían datos que incluso no son útiles y de igual manera se envían a las capas superiores.

Capa de puerta de enlace: Reenvía los paquetes generados por la capa de percepción a internet. En esta capa la puerta de enlace está basada en SDN y le llaman OF- GW (Gateway – OpenFlow) y se encargan de analizar la utilidad de cada paquete recibido y si debe ser descartado o enviado a internet. Esto lo hacen mediante las reglas enviadas por el controlador.

Las reglas creadas por el controlador pueden cambiar dinámicamente. Mencionan que hay una diferencia entre OF-SW (Switch – OpenFlow) típico de una red SDN a un OF-GW. Los OF-SW deciden como reenviar el paquete de acuerdo a las reglas y los OF-GW no toman decisiones sobre como reenviar, solo deciden si el paquete es útil o no.

Capa de red: Reciben los paquetes que se decidieron enviar en la capa de puerta de enlace y se enrutan. Los OF-GW se comunican con el controlador, el cual determina las reglas según las necesidades de la capa de aplicación y las envía a OF-GW.

Según esta propuesta, se evita la congestión de la red con la capa de puerta de enlace, al evitar que se envíen datos innecesarios a internet.

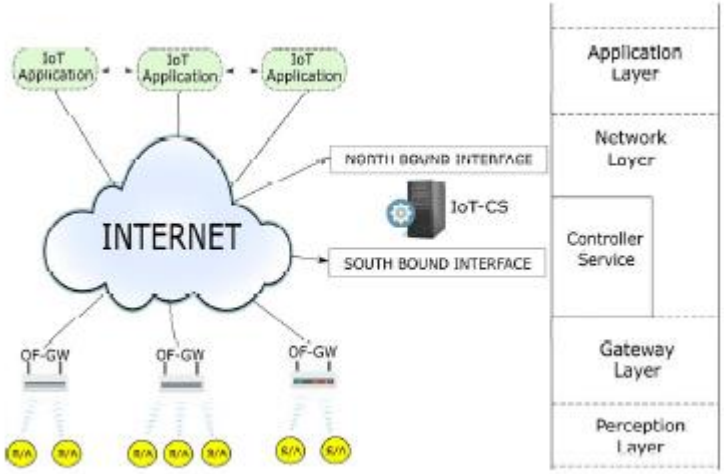

Figura 12. Arquitectura de 4 capas

4 Diseño de Arquitectura SDN – IoT

En base a lo mencionado en el capítulo 3 sobre los aportes y propuestas de aplicación que trae la SDN a la IoT, se ha planteado el diseño de un escenario virtual sencillo el cual se basará en una red residencial. Se ha tomado como referencia una red residencial ya que el campo de aplicación más típico de la IoT es una casa inteligente "Smart Home".

La arquitectura clásica de una red residencial consiste en la conexión por cable de los equipos (ordenadores) y un *router* que se comunica mediante una antena a los dispositivos inalámbricos. A este escenario le aplicaremos los conceptos de SDN – IoT sustituyendo el *router* por un conmutador SDN que tenga conexión a un controlador y también incluiremos dispositivos con sistemas operativos de IoT.

Hemos mencionado en reiteradas ocasiones que una de las grandes ventajas es la "Gestión de la Red", se le da tanta importancia por el hecho de la programabilidad que brinda SDN. Los proveedores de servicios e incluso el usuario evitará tener contacto con los equipos de red y esta gestión pasará a ser remota permitiendo lo siguiente: identificación de fallos, configuración de los dispositivos, activación remota de los dispositivos, flexibilidad, interoperabilidad de diferentes tipos de redes, simplificar la arquitectura de la red, priorizar tráfico de IoT, entre otras. Con este escenario se pretende demostrar que al reemplazar la pasarela GW- IoT por un *switch*-SDN y permitiendo que el controlador sea el cerebro de la red se tendrá una mejor gestión de las redes. También se propone aplicarle a controlador la función de *router.*

4.1 Herramientas Utilizadas

4.1.1 VirtualBox

Para facilitar el manejo del escenario, el mismo se realizará en un software de virtualización que nos permita arrancar una máquina virtual en un sistema operativo, es este caso Ubuntu. El Software de virtualización que utilizaremos será VirtualBox, el cual se puede instalar en los sistemas operativos más comunes Windows, Linux, Max OS X, etc.

Otra razón por la cual desarrollamos en entorno en VirtualBox es por la capacidad de importar y exportar máquinas virtuales en formato OVA o OVF, de esta manera podremos tener el escenario empaquetado sin necesidad de realizar nuevamente cualquier tipo de instalación. Además, VirtualBox da la facilidad de guardar el estado de la máquina virtual permitiendo que al volver al escenario esté en el mismo estado en que se dejó.

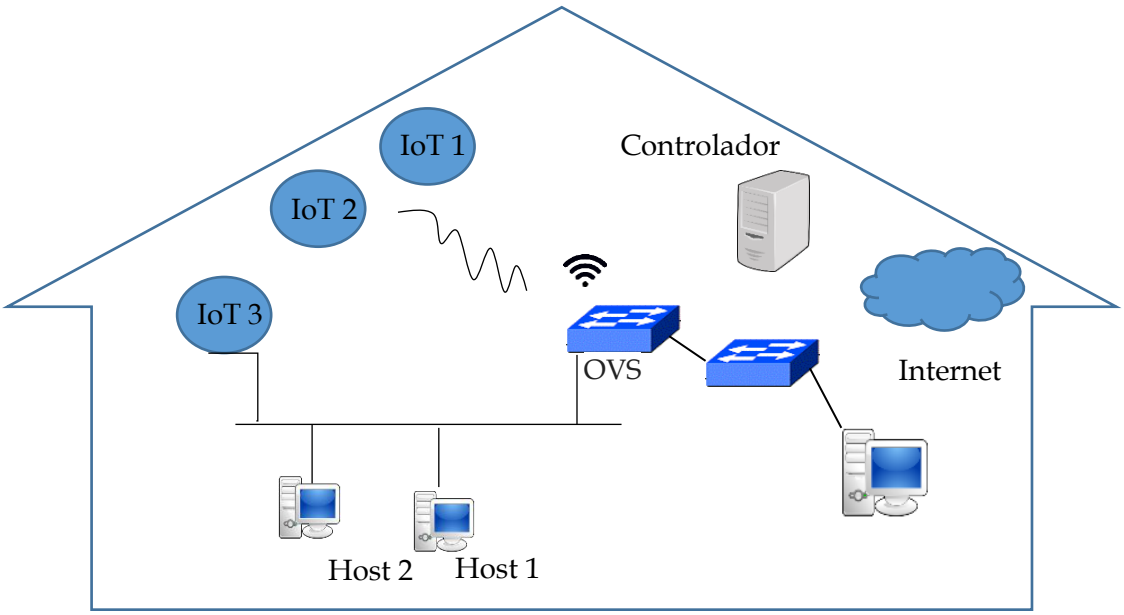

Figura 13. Diseño de Red Residencial IoT - SDN

4.1.2 Virtual Networks over Linux

Para el desarrollo del escenario se ha utilizado una herramienta de código abierto llamada Virtual Networks over Linux, la cual permite la creación de escenarios virtuales.

VNX ha sido desarrollada por el Departamento de Ingeniería Telemática de la Universidad Politécnica de Madrid [24].

Nos da la facilidad de desplegar automáticamente escenarios con máquinas virtuales de sistemas operativos diferentes. VNX se ejecuta sobre una maquina Linux VNX y para la creación de los escenarios virtuales se utiliza el lenguaje XML donde se detalla todo el entorno que se quiere desarrollar.

4.2 Desarrollo

En esta sección mencionaremos todo lo que necesitamos para cumplir con el diseño de la topología de red propuesta.

4.2.1 Componentes

OpenVSwitch

Se ha diseñado para permitir la creación de un *switch* virtual, multicapa y de código abierto, admite distintos protocolos e interfaces de administración estándar como ejemplo OpenFlow.

Controlador

Como mencionamos en el capítulo 2, el controlador SDN, es quien tiene la inteligencia de la red. En el mercado existen diferentes controladores de código abierto con excelentes funcionalidades para el desarrollo de proyectos con SDN, ejemplo de estos: OpenDaylight, NOX, POX, ONOS, RYU.

En nuestro caso vamos a utilizar el controlador RYU, es de código abierto, brinda componentes de software con una API que permite crear nuevas aplicaciones de control y gestión de red de manera sencilla para el desarrollador, además de incluir algunas aplicaciones básicas. Tiene compatibilidad con el protocolo OpenFlow hasta la versión 1.5. Su código fuente se encuentra en GitHub, escrito en Python bajo la licencia Apache 2.0 [25].

4.2.2 Entorno Virtual

Ahora que hemos detallado cada uno de los componentes que utilizaremos en nuestro escenario, vamos a describir su desarrollo.

En la página web de VNX está disponible el enlace para la instalación y también una imagen OVA para importar la máquina virtual. En nuestro caso utilizamos el OVA con VNX instalado, para utilizarlo tenemos que tener VirtualBox instalado e importar la máquina. Como utilizamos una máquina virtual de VNX tenemos que tener en cuenta que el usuario es "root" y la contraseña "xxxx".

Con la maquina importando y funcionando debemos actualizar VNX para evitar inconvenientes al correr el escenario mediante el siguiente comando:

```
vnx_update
```

Para la puesta en marcha del escenario, es necesario utilizar un archivo XML donde se describe la descripción del escenario según especificaciones que tiene VNX y las cuales se detallan en su página web.

La red residencial se creará basada en un OpenvSiwtch mediante las líneas siguientes en el XML de VNX:

```
<net name="Net0" mode="openvswitch" hwaddr="00:00:00:00:00:01"
controller='tcp:127.0.0.1:6633'
        of_version="OpenFlow13" fail_mode='secure'/>
```

Le detallaré lo que indica cada línea a continuación;

hwaddr: MAC asociada al switch que se utilizará para construir el identificador del switch (dpid).

controller: protocolo, dirección IP y puerto para conectarse al controlador. Esto le indica que cuando arrancamos el controlador RYU desde el host y arranquemos el escenario, el OVS se conectará al controlador a través de la interfaz loopback (127.0.0.1).

of_version: versión de Openflow que utilizará el switch para la comunicación con el controlador. Es importante que la versión del controlador que se coloque aquí, sea la misma con la cual se realicen la instalación de aplicaciones en RYU.

fail_mode: modo de funcionamiento del conmutador en caso de no estar conectado al controlador, se conectará o funcionará como un conmutador Ethernet. Un OVS por defecto arranca como un conmutador Ethernet "fail_mode=standalone "si no está conectado al controlador. Para que funcione siempre como conmutador Openflow, le colocamos en fail_mode=secure.

Los dispositivos con capacidad IoT los vamos a simular con el SO tinycore, estos trabajan sobre KVM y para correr eficientemente tienen que tener las extensiones de virtualización las cuales VirtualBox no ofrece. Por esto se arranca con qemu, de esta manera se puede ejecutar de una mejor manera dentro de una máquina virtual de VirtualBox.

Luego de tener en nuestro archivo de XML, toda la configuración del escenario, lo arrancamos con el siguiente comando.

```
sudo vnx -f VNX_file.xml -v -t
```

El escenario arrancará las máquinas virtuales: con SO tinycore y con linux con el servidor de dhcp activado, además de los *swtich*.

La instalación del controlador RYU la podemos encontrar en [25], se puede realizar de dos maneras. Desde el repositorio de GitHub o la instalación directa usando pip.

```
pip install ryu
```

Para comprobar que se ha instalado bien utilizamos:

```
ryu-manager
```

Luego de tener instalado el controlador procederemos a verificar que aplicación tiene listas para instalar con el siguiente comando:

```
% ls ryu/app/
```

Nos muestra las distintas aplicaciones que ofrece RYU

```
bmpstation.py        rest_router.py          simple_switch_lacp_13.py
cbench.py            rest_router.pyc         simple_switch_lacp.py
conf_switch_key.py   rest_topology.py        simple_switch.py
example_switch_13.py rest_vtep.py            simple_switch.pyc
gui_topology         simple_monitor_13.py    simple_switch_rest_13.py
__init__.py          simple_switch_12.py     simple_switch_snort.py
__init__.pyc         simple_switch_13.py     simple_switch_stp_13.py
ofctl                simple_switch_13.pyc    simple_switch_stp.py
ofctl_rest.py        simple_switch_14.py     simple_switch_websocket_13.py
rest_conf_switch.py  simple_switch_15.py     wsgi.py
rest_firewall.py     simple_switch_igmp_13.py wsgi.pyc
rest_qos.py          simple_switch_igmp.py   ws_topology.py
```

De estas utilizaremos: simple_switch_13.py - rest_router.py

La primera aplicación que utilizaremos es para verificar que el *switch* esta funcionando como capa L2 mediante pruebas de ping y también comprobamos que el *swtich* se esté conectando con el controlador. El número 13 del comando indica la versión de Openflow.

La segunda aplicación que utilizaremos es para que el *swtich* funcione como router, esto se realiza mediante una API REST que se debe configurar. Se debe indicar en el archivo XML las rutas del router.

Antes de empezar a utilizar la API REST debemos tener instalado. cURL herramienta que permite hacer peticiones HTTP. Se instala mediante lo siguiente:

```
sudo apt-get install curl
```

La funcionalidad de routing se implementa para que el router residencial utilice un OVS que esté controlado de RYU.

Al comprobar todas las conexiones y que el controlador esté funcionando como router, proponemos utilizar *Wireshark* para realizar pruebas de tráfico que viene de dispositivos IoT.

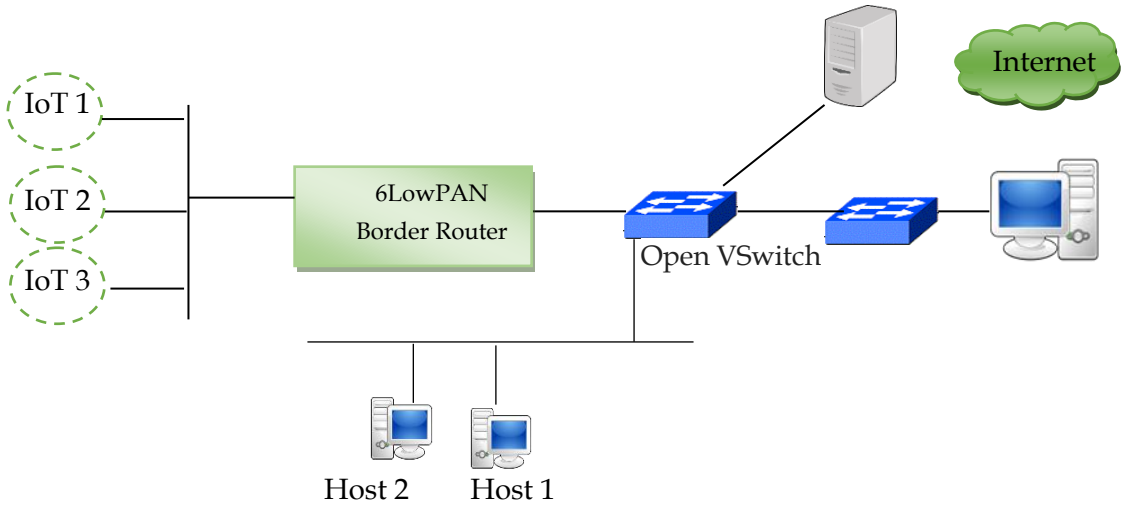

Figura 14. Diseño de red IoT- SDN – 6LowPAN

Otra propuesta de escenario que se propone es agregarle a la topología un router border 6LoWPAN (LBR) el cual actúa como GW y tendrá la función de conectar los dispositivos IoT basados en. IEEE 802.15.4 con la red ethernet. Para simular este entorno tenemos que tener una máquina virtual con que tenga instalado 6LoWPAN.

Tendríamos una nueva red que se conecte a la pasarela 6LoWPAN mediante una red inalámbrica, pero se puede simular como otro segmento de red a donde se conectaran los sensores

5 Conclusiones

Durante la realización de esta memoria, se ha logrado tener más claro el concepto de internet de las cosas, que no simplemente son objetos conectados, si no que hay un gran mundo detrás de esta tecnología. A pesar que se escucha de ella desde hace mucho tiempo, no tiene una arquitectura estándar, se tienen muchos protocolos que por su variedad se forma la heterogeneidad de los dispositivos.

Los usuarios que tiene IoT actualmente dicen que no están satisfechos, con el rendimiento de los dispositivos y tienen temor de que algunos que son para casos importantes, fallen en cierto momento. La IoT está teniendo problemas como seguridad, heterogeneidad, manejo de los grandes datos, gestionar sus redes, escalabilidad, entre otras.

También hemos notado, la importancia que tienen las redes definidas por software, en diferentes aspectos y que su principal aporte para la IoT, es gestionar la red, permitiendo reducir costes, por la disminución de equipos como enrutadores. Debido a que el controlador tiene el cerebro de la red y es totalmente programable, se permite tener una mejor administración de la red y esto es una ventaja para las empresas y usuarios. Otra ventaja que se ha visto, es la disminución del ancho de banda, ya que con lo que se menciona en la propuesta 5, que al adicionar un Gateway – OpenFlow este puede ir descartando paquetes que no sean necesarios.

SDN brinda flexibilidad en la configuración, permite cambiar dinámicamente el comportamiento de la red, en función de los patrones de tráfico, los incidentes detectados y, sobre todos los dispositivos conectados en un entorno de IoT.

El objetivo principal era identificar las diferentes aplicaciones que tenía la SDN en la IoT y a pesar de que su recopilación fue ardua, se ha logrado encontrar diferentes beneficios de la fusión de esta tecnología.

Las diferentes propuestas que se vieron durante la investigación y desarrollo de la memoria, no muestran un entorno virtual para simulación claro, por eso se tomó la decisión de diseñar un escenario de pruebas.

En resumen, la mejora de los problemas que está teniendo la IoT, se pueden mejorar, si se enfocan en buscar tecnologías adicionales que optimicen su rendimiento, en este caso SDN o NFV.

Líneas futuras

Durante el desarrollo del trabajo se ha querido abarcar en otros aspectos importantes para la IoT, como es el caso de la seguridad y privacidad, existen muchas investigaciones que mencionan los beneficios que traen las SDN en este aspecto.

Con respecto al escenario de red se propone realizar pruebas de escalabilidad, simulando varios dispositivos de IoT, que por temas de capacidad del PC no se pudieron realizar, además de pruebas de priorización de tráfico, calidad de servicios, rendimiento, detección de fallos, probar con dispositivos de SO de IoT distintos para verificar la heterogeneidad de la red. Adicional, crear una aplicación en RYU, para que el controlador funcione como DHCP.

Para realizar pruebas sobre el segundo escenario propuesto, se recomienda utilizar un equipo físico para la simulación de IoT, ya que la pasarela 6LBR solo funciona con sensores externos.

Bibliografía

[1] Conferencia Cisco Internet Business Solutions Group (IBSG) 2013

[2] Cisco Visual Networking Index: Forecast and Trends, 2017–2022.

[3] Estudio realizado por Dynatrace Dave Anderson, Disponible en: https://www.dynatrace.com/news/press-release/el-52-de-los-consumidores-usa-de manera-habitual-dispositivos-con-tecnologia-de-internet-de-las-cosas-en-todo-el mundo-y-un-64-declara-fallos-continuos/.

[4] Sahrish Khan Tayyaba, Munam Ali Shah, Naila Sher Afzal Khan, Yousra Asim, Wajeeha Naeem and Muhammad Kamran, "Software-Defined Networks (SDNs) and Internet of Things (IoTs): A Qualitative Prediction for 2020" International Journal of Advanced Computer Science and Applications(IJACSA), 7(11), 2016.

[5] Khan, Sahrish & Shah, Munam & Khan, Omair & Wahab Ahmed, Abdul. (2017). Software Defined Network (SDN) Based Internet of Things (IoT): A Road Ahead. 1-8. 10.1145/3102304.3102319.

[6] El internet de las cosas Disponible en: https://es.wikipedia.org/wiki/Internet_de_las_cosas.

[7] N. Bizanis and F. A. Kuipers, "SDN and Virtualization Solutions for the Internet of Things: A Survey," in IEEE Access, vol. 4, pp. 5591-5606, 2016.

[8] J. Fernández, "Integración de Plataformas de Computación en la Nube dedicadas al Internet de las Cosas con soporte de Redes 5G," 2018 Universidad Politécnica de Madrid.

[9] Parmar, Jekishan & Desai, Ankit. (2016). IoT: Networking Technologies and Research Challenges. International Journal of Computer Applications. 154. 1-6. 10.5120/ijca2016912181.

[10] M. Ojo, D. Adami and S. Giordano, "A SDN-IoT Architecture with NFV Implementation," 2016 IEEE Globecom Workshops (GC Wkshps), Washington, DC, 2016, pp. 1-6. doi: 10.1109/GLOCOMW.2016.7848825

[11] Network Connected Devices (Internet of Things) Disponible en: https://blogs.deusto.es/master-informatica/network-connected-devices-internet-of-things-riesgos-parte-2-final/

[12] Table Comparing Wireless Protocols for IoT Devices. Disponible en: http://glowlabs.co/wireless-protocols/

[13] X. Ma and W. Luo, "The Analysis of 6LowPAN Technology," 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, Wuhan, 2008, pp. 963-966. doi: 10.1109/PACIIA.2008.72

[14] M. Bangulo, P. Matthews. (2011) “Sateful NAT64” Disponible en: https://tools.ietf.org/html/rfc6146

[15] Luis Jiménez Ruiz (2016) “Diseño e Implementación de Etapa de Comunicación Basada en 6lowpan para Plataforma Modular de Redes de Sensores Inalámbricas” pp.164

[16] Tara Salman, “Networking Protocols and Standards for Internet of Things” Universidad de Washington en St. Louis, pp. 28. Disponible en: https://www.cse.wustl.edu/~jain/cse570-15/ftp/iot_prot.pdf

[17] C. Gonzalez, O.Flauzac, F. Nolot, “Evolution and Contribution for the Internet of Things by the Emerging Software-defined networking”

[18] Contiki Disponible en: http://www.contiki-os.org/

[19] RIOT Disponible en: https://www.riot-os.org/

[20] CCNA R&S v3.0 - Conceptos básicos de SDN (Software Defined Networks)

[21] S. Bera, S. Misra and A. V. Vasilakos, "Software-Defined Networking for Internet of Things: A Survey," in IEEE Internet of Things Journal, vol. 4, no. 6, pp. 1994-2008, Dec. 2017.

[22] I. Bedhief, M. Kassar and T. Aguili, "SDN-based architecture challenging the IoT heterogeneity," 2016 3rd Smart Cloud Networks & Systems (SCNS), Dubai, 2016, pp.

[23] M. T. Kakiz, E. Öztürk and T. Çavdar, "A novel SDN-based IoT architecture for big data," 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, 2017, pp. 1-5.

[24] Virtual Networks over linuX (VNX), http://vnx.dit.upm.es/.

[25] RYU Controller https://osrg.github.io/ryu/

[26] J. Suárez, Interconexión entre plataformas IoT mediante el uso de SDN. Universidad Politecnica de Valencia.