UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

MAESTRÍA EN TELECOMUNICACIONES

TEMA: ALARMAS COMUNITARIAS BASADAS EN ARQUITECTURAS SDN E IOT

Trabajo de Titulación, previo a la obtención del Grado Académico de Magíster en Telecomunicaciones

Autor: Ingeniero Franklin German Placencia Camacho

Director: Ingeniero Víctor Santiago Manzano Villafuerte, Magíster.

Ambato - Ecuador

2021

APROBACIÓN DEL TRABAJO DE TITULACIÓN

A la Unidad Académica de Titulación de la Facultad de Ingeniería en Sistemas Electrónica e Industrial. El Tribunal receptor de la Defensa del Trabajo de Titulación, presidido por la Ingeniera Elsa Pilar Urrutia Urrutia Magíster, e integrado por los señores Ingeniero Juan Pablo Pallo Noroña Magíster e Ingeniero Marco Antonio Jurado Lozada Magíster designados por la Unidad Académica de Titulación de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial de la Universidad Técnica de Ambato, para receptar el Trabajo de Titulación con el tema: "ALARMAS COMUNITARIAS BASADAS EN ARQUITECTURAS SDN E IOT", elaborado y presentado por el señor Ingeniero Franklin German Placencia Camacho, para optar por el Grado Académico de Magíster en Telecomunicaciones; una vez escuchada la defensa oral del Trabajo de Titulación el Tribunal aprueba y remite el trabajo para uso y custodia en las bibliotecas de la Universidad Técnica de Ambato.

Ing. Elsa Pilar Urrutia Urrutia Mg.

Presidente y Miembro del Tribunal de Defensa

Ing. Juan Pablo Pallo Noroña, Mg.

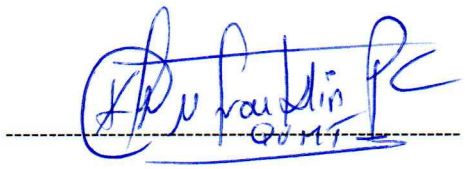
Miembro del Tribunal de Defensa

Ing. Marco Antonio Jurado Lozada, Mg.

Miembro del Tribunal de Defensa

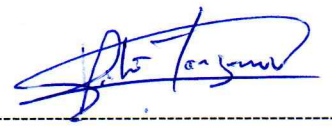
AUTORÍA DEL TRABAJO DE TITULACIÓN

La responsabilidad de las opiniones, comentarios y críticas emitidas en el Trabajo de Titulación presentado con el tema: "ALARMAS COMUNITARIAS BASADAS EN ARQUITECTURAS SDN E IOT", le corresponde exclusivamente a: Ingeniero Franklin German Placencia Camacho, Autor bajo la Dirección del Ingeniero Víctor Santiago Manzano Villafuerte, Magister, Director del Trabajo de Titulación; y el patrimonio intelectual a la Universidad Técnica de Ambato.



Ingeniero Franklin German Placencia Camacho

AUTOR



Ingeniero Víctor Santiago Manzano Villafuerte, Magister.

DIRECTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que el Trabajo de Titulación, sirva como un documento disponible para su lectura, consulta y procesos de investigación, según las normas de la Institución.

Cedo los Derechos de mi Trabajo de Titulación, con fines de difusión pública, además apruebo la reproducción de este, dentro de las regulaciones de la Universidad Técnica de Ambato.

Ingeniero Franklin German Placencia Camacho

C.C. 1804612362

ÍNDICE GENERAL

CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

AGRADECIMIENTO

Al Creador por darme la mejor familia, salud y vida para surcar sus terrenos.

Al Ingeniero Santiago Manzano; más que tutor, un buen amigo y excelente ser humano. Gracias por tus palabras de motivación y superación. «Tu pupilo no se da por vencido!»

A Karen Orellana, gracias por tu tiempo, esfuerzo, paciencia y regaños. Gracias por estar presente durante mis triunfos y derrotas. Sin tí, esta etapa de mi vida no sería posible.

A Cristina y Gonzalo, gracias por acompañarme en esta nueva etapa educativa y su amistad sincera e incondicional.

A mis amigos del mejor equipo del mundo «LDI», gracias por estar siempre presentes y aunque las condiciones hayan cambiado continuen reuniendose para hacer lo que nos gusta.

A mis amigos de ruta, Javier V. y nuevos integrantes del grupo. Por secundarme en los viajes y vivencias sobre dos ruedas.

A Juan C. J., Mauro C., Diego B., David C., gracias todo su apoyo moral.

A Kathe, Lau y Francisco, aunque la pandemia nos haya distanciado. Siempre los tengo presentes, gracias.

Al grupo de Asistentes, Analistas y Docentes Investigadores de la UTA - FISEI 2016 - 2018.

A todos mis amigos y clientes, gracias por su confianza. Este logro es por y para ustedes.

Franklin Placencia

DEDICATORIA

A mi Madre, por su amor, apoyo y palabras de ánimo en cada etapa de vida en las que he incursionado y ser el motor que me impulsa a alcanzar mis metas.

A mis hermanas Mary, Marcia y Sara y mi hermano Pablo por formar los pilares fundamentales de mi gran familia y acompañarme en este camino llamado vida.

A mi cuñado y hermano Víctor Hugo por su confianza, apoyo y palabras de motivación.

A mis sobrinos Esteban, Dana y Matías por alegrar mis días con sus alocadas ocurrencias.

Franklin Placencia

UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

MAESTRÍA EN TELECOMUNICACIONES

TEMA:

ALARMAS COMUNITARIAS BASADAS EN ARQUITECTURAS SDN E IOT

AUTOR: Ingeniero Franklin German Placencia Camacho

DIRECTOR: Ingeniero Víctor Santiago Manzano Villafuerte, Magíster

LÍNEA DE INVESTIGACIÓN:

- TECNOLOGÍAS, SEGURIDAD Y GESTIÓN DE REDES DE COMUNICACIONES

FECHA: 17 de marzo de 2021

RESUMEN EJECUTIVO

Los sistemas de seguridad ciudadana hoy más que nunca representan una herramienta de alto impacto y de vital importancia para la sociedad del siglo XXI. Su madurez tecnológica le ha permitido ser incluida en varias zonas geográficas como parte de las políticas gubernamentales e inclusive ha logrado ser catalogada constitucionalmente como un derecho de todos los ciudadanos.

Con el crecimiento de la demanda de sistemas de seguridad ciudadana y los continuos avances tecnológicos ligados a ello. Los datos que transitan sobre las redes computacionales que soportan dichos sistemas, sufren constantemente de problemas de saturación y cuellos de botella causados por la falta de políticas de gestión de tráfico y calidad de servicio.

En las topologías de red tradicional, los datos son considerados por igual sin distinguir protocolo o prioridad; debido a que los enrutadores centrales contienen políticas precargadas y preconfiguradas por los fabricantes del hardware. Sin embargo, en los segmentos de red donde se requiere optimizar la gestión del tráfico; es necesario crear políticas personalizadas por cada uno de los dispositivos que conformen dicha red.

En términos de control de red, esto representa otro gran obstáculo durante la expansión, migración o actualización de redes de gran tamaño como las destinadas al monitoreo de alarmas comunitarias que incluyen tráfico crítico de video, audio y datos.

La investigación propone una reestructuración de las redes de datos tradicionales, utilizando dos tecnologías emergentes diseñadas para la descongestión. En primera instancia se utiliza una arquitectura de red definida por software (SDN por sus siglas en inglés) para separar el plano de control del plano de datos en los conmutadores de red y con ello solventar los problemas de expansión, migración o

actualización de redes de gran tamaño. En segundo lugar, se reemplaza la infraestructura de las alarmas comunitarias activadas por dispositivos de radiofrecuencia por una arquitectura de red IoT con protocolo de comunicaciones MQTT.

Para finalizar, se presentan los resultados de un análisis de ancho de banda de una red tradicional sin políticas de control sobre las que transitan servicios de audio digital (VoIP), videovigilancia (CCTV), datos y paquetes de control IoT; demostrando que los servicios de tráfico crítico tienden a acaparar el 80 % del ancho de banda total de la red.

Y en adición, se evalúan las ventajas y resultados de una topología de red definida por software diseñada para trabajar de forma sinérgica con protocolos IoT. Además de proporcionar sencillez y simplicidad durante la implementación de políticas de tráfico y calidad de servicio configuradas a través de un entorno web que puede ser implementado y replicado en entornos de red de alta demanda.

Descriptores: Calidad de Servicio, Ingeniería de tráfico, Internet de las cosas, OpenWRT, Open-vSwitch, Redes definidas por software, Redes tradicionales, Seguridad ciudadana, Servicios de red virtualizados, Servicios Web de Amazon, .

UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

MAESTRÍA EN TELECOMUNICACIONES

THEME:

COMMUNITY ALARMS BASED ON SDN AND IOT ARCHITECTURES

AUTHOR: Ingeniero Franklin German Placencia Camacho

DIRECTED BY: Ingeniero Víctor Santiago Manzano Villafuerte, Magíster

LINE OF RESEARCH:

- TECHNOLOGIES, SECURITY AND MANAGEMENT OF COMMUNICATIONS NETWORKS

DATE: March 17, 2021

EXECUTIVE SUMMARY

Citizen security systems of today more than ever represent a tool of high impact and vital importance for the Society of the 21th Century. Its technological maturity has allowed it to be included in several geographical areas as part of government policies and has even managed to be constitutionally catalogued as a right of all citizens.

With the growth of the demand for citizen security systems and the continuous technological advances linked to it. The data that transits over the computational networks that support such systems, constantly suffer from saturation problems and bottlenecks caused by the lack of traffic management policies and quality of service.

In traditional network topologies, data is considered equally without distinguishing protocol or priority; because central routers contain policies preloaded and preconfigured by hardware manufacturers. However, in network segments where traffic management is required to be optimized; it is necessary to create custom policies for each of the devices that make up that network.

In terms of network control, this represents another major obstacle during the expansion, migration or upgrade of large networks such as those intended for monitoring community alarms that include critical video, audio and data traffic.

The research proposes a restructuring of traditional data networks, using two emerging technologies designed for decongestion. In the first instance, a software-defined network architecture (SDN) is used to separate the control plane from the data plane on the network switches and thereby solve the problems of expansion, migration or upgrade of large networks. Second, the infrastructure of community alarms activated by radio frequency devices is replaced by an IoT network architecture with MQTT communications protocol.

Finally, we present the results of a bandwidth analysis of a traditional network without control policies over which digital audio services (VoIP), video surveillance (CCTV), data and IoT control packets pass; demonstrating that critical traffic services tend to capture 80 % of the total bandwidth of the network. And in addition, the advantages and results of a software-defined network topology designed to work synergistically with IoT protocols are evaluated. In addition to providing simplicity and simplicity during the implementation of traffic and quality of service policies configured through a web environment that can be implemented and replicated in high-demand network environments.

Descriptors: Amazon Web Services, Citizen security, Internet of things, OpenWRT, Open-vSwitch, quality of Service, Software-defined networks, traffic engineering, traditional networks, virtualized network services, .

CAPÍTULO I

EL PROBLEMA DE INVESTIGACIÓN

1.1. Introducción

En la actualidad, la seguridad ciudadana es un factor de gran importancia para toda sociedad desarrollada o en desarrollo. Debido a los altos indices delictivos que van ascendiendo día con día, los ciudadanos han convertido al acceso a sistemas de monitoreo, alarmas y videovigilancia en una necesidad latente y en crecimiento a nivel global, regional y nacional, lo que ha obligado a buscar soluciones tecnológicas más efectivas.

En Latinoamérica, la seguridad ciudadana ha logrado ser incluida como parte de la política gubernamental de los estados. Sin embargo, debido al gran impacto social y económico que ello representa; alcanzar la madurez en este campo requiere de atención prioritaria a largo plazo basándose en los resultados generados por la implementación de acciones preventivas y correctivas cuyo pilar es el monitoreo continuo [1].

Con el afán empoderar a la ciudadanía en torno a sistemas de seguridad ciudadana, en Ecuador, se han instalado sistemas de video vigilancia de alta definición, vinculados a la red del Servicio Integrado de Seguridad (ECU911). Dichas soluciones tecnológicas están compuestas por un aproximado de 4500 cámaras localizadas en puntos estratégicos y de alto tráfico ciudadano; además de 70000 kits de alarma móvil sobre vehículos de transporte público, enlazados a 16 centrales de monitoreo y operaciones a lo largo del territorio nacional trabajando durante 24 horas, los 365 del año [2, 3].

Adicionalmente, el Ministerio del Interior ecuatoriano a través del cuerpo de policía nacional y con el aporte de la ciudadanía, ha implementado soluciones comunitarias para cubrir la brecha tecnológica de barrios y parroquias rurales, a través de alarmas activadas mediante nodos de radiofrecuencia (RF) que son socorridos por unidades policiales locales o cercanas.

No obstante, pese a contar con gran cantidad de soluciones de diferentes tecnologías aún existen falencias relacionadas con el mecanismo de activación de alarmas, tiempo de respuesta a llamados de emergencia, mecanismos de geolocalización de la zona en riesgo y sobre todo la gestión de redes de datos destinadas a obtener, procesar y transmitir la data de alarmas y video en tiempo real [4].

En consecuencia y gracias a la tecnología emergente, se ha desarrollado gran cantidad de utilitarios para resolver la problemática a través de la gran red llamada internet. Gracias a la aceleración de las unidades de procesamiento y la infraestructura de red, los usuarios pueden ver, manipular y descargar contenidos de alta calidad con poco retardo. Inclusive se ha permitido a los usuarios tomar el control completo de estaciones de trabajo y manipular el flujo los datos para reducir aún más los retardos de red.

Parte de estos avances se deben al trabajo de desarrolladores de red, quienes han programado y rediseñado los protocolos de red preexistentes así como migrado topologías y arquitecturas. Entre las nuevas tecnologías se destacan las Redes Definidas por Software (SDN, por sus siglas en inglés) destinado a desaturar y descentralizar redes tradicionales y protocolos ligeros para el Internet de las Cosas (IoT, por sus siglas en inglés) para reducir la carga sobre la red [5].

Por lo tanto, se ha considerado necesario el análisis de una nueva topología de red para alarmas comunitarias con un núcleo de red que vaya más allá del sistema tradicional. Y en adición, proporcionar soluciones que permitan conocer el lugar y momento de oportuno se requiera de ayuda inmediata de los efectivos policiales; a través del uso de las nuevas tecnologías para gestión y desaturación de tráfico de datos trabajando de forma complementaria dentro de los sistemas de comunicaciones.

1.2. Justificación

El gobierno ecuatoriano, dentro de las políticas de seguridad ciudadana, ha invertido gran cantidad de recursos económicos y humanos para prevenir y combatir el delito organizado. En paralelo con el ECU 911, se han implementado proyectos tecnológicos para zonas descentralizadas conocidos como Sistemas de Alarmas Comunitarias. Normalmente, estos sistemas cuentan con cámaras de monitoreo, altoparlantes y sirenas accionadas por botones de pánico que son ubicados en urbanizaciones, casas, oficinas, hoteles, locales comerciales, empresas y entidades financieras con el afán de generar una señal electrónica que permita a la policía nacional socorrer a su llamado.

Sin embargo, la solución descentralizada que se ha adoptado para abarcar áreas extensas suele generar retrasos por términos de logística. La principal causante de ellos está directamente relacionada con la plataforma tecnológica dispersa en distintos puntos, ocasionando que en algunos sectores sea imposible determinar el punto exacto o localización del nodo que accionó la alarma.

Pese a que el despliegue de unidades policíacas y de emergencia es inmediata, a menudo su rapidez es opacada por el tiempo que conlleva hallar el sitio donde la ayuda es requerida. Como ejemplo referencial se puede mencionar que resultaría imposible determinar con exactitud el punto de activación de una alarma de un nodo no georreferenciado, dentro del grupo de 32.000 botones de pánico que han sido instalados en la ciudad de Guayaquil - Ecuador [6].

El trabajo de investigación propone un nuevo mecanismo de gestión del tráfico de la información para sistemas de seguridad comunitaria centralizada. Basados en una arquitectura SDN para optimizar recursos tecnológicos, reordenar tráfico de entrada y salida y aprovechar el ancho de banda de las redes de datos durante la visualización de sistemas de videovigilancia y en adición proporcionar alertas visuales para alarmas, mediante la reestructuración de botones de pánico georeferenciados que utilizan protocolos y arquitectura IoT.

1.3. Objetivos

1.3.1. Objetivo General

Diseño de alarmas comunitarias basadas en arquitecturas SDN e IoT

1.3.2. Objetivos Específicos

- Evaluar la calidad de servicio de una arquitectura de red IoT sin control de flujo de datos con transmisión paralela de audio y video.
- Implementar un entorno de ensayos de una arquitectura de red IoT sobre una red SDN para alarmas comunitarias con transmisión paralela de audio y video.
- Implementar un sistema de alarmas comunitarias basada en arquitectura SDN e IoT.
- Evaluar la calidad de servicio de una arquitectura de red IoT con control de flujo de datos sobre una Red Definida por Software destinada a la seguridad comunitaria.

CAPÍTULO II

ANTECEDENTES INVESTIGATIVOS

2.1. Estado del Arte

Durante los últimos 10 años, los avances tecnológicos han transformado la manera de comunicar al ser humano a casi absolutamente todo lo que lo rodea y con el surgimiento del IoT, la cantidad de objetos que son conectados a la red global crece a una velocidad nunca antes vista, en el apartado en curso se analiza el estado del arte sobre investigaciones que buscan la convergencia entre las SDN e IoT con el afán de mitigar la problemática que supone el crecimiento desmesurado de las redes, así como sus aplicaciones en el ámbito educativo, empresarial y/o comunitario.

En la publicación de Vijay Varadharajan [7] se menciona que los dispositivos superarían los 30 mil millones a inicios de la década en curso. Este crecimiento desmesurado del IoT, plantea nuevos desafíos para las redes de Internet tradicional en aspectos como: tecnologías, protocolos de comunicación, requisitos de calidad de servicio específico, afluencia masiva de datos y condiciones de red impredecibles.

Según el Departamento de Ciencias e Ingeniería del Instituto Hindú de Tecnología Kharagpur, las SDN son un enfoque prometedor para controlar la red de manera unificada mediante la administración basada en reglas. Las abstracciones proporcionadas por SDN permiten el control descentralizado de la red mediante políticas de alto nivel, sin preocuparse por problemas de configuración de bajo nivel; por lo tanto, es ventajoso implementar SDNs para sobrellevar la heterogeneidad y los requisitos específicos del IoT [8].

Existen varias razones por lo que en segmentos privados y públicos de las redes existentes aún no se haya migrado a SDN. En el estudio de Nicolae Paladi y Christian Gehrmann se puntualiza que: gran cantidad de desarrolladores de SDN se han enfocado en el mejoramiento de la abstracción de los planos de reenvío y control, es decir se ha alcanzado la madurez de los algoritmos de control (interfaz SUR: control de la red) pero se han descuidado a los entornos de nivel de aplicación (interfaz NORTE: entorno gráfico para comandar la interfaz SUR). Ante tal circunstancia, su investigación se centra en la mejoramiento de la interfaz sur para mejorar los mecanismos rudimentarios y proporcionar herramientas de control suficientes a proveedores de red, operadores y desarrolladores de aplicaciones, permitiéndolos desplegar múltiples aplicaciones y funcionalidades en una infraestructura SDN, sin embargo es de responsabilidad el administrador de la red implementar sus propias interfaces de control[9].

En la investigación de Mamdouh Alenezi y Khaled Almustafa se evalúa al uso de las arquitecturas complementarias SDN y NVF (virtualización de funciones de red) como solución efectiva a la sobrecarga de red que supone el crecimiento de los dispositivos IoT que se conectan a internet. Se

destaca también, que con dicho crecimiento los costos de expansión y administración de las redes serían directamente proporcionales en términos de hardware y energía. Por lo tanto, se establece que la mejor solución para la migración o implementación de redes emergentes es utilizar arquitecturas SDN sobre NVF; aún más, cuando estas redes son destinadas a soportar gran cantidad de tráfico de diferentes dispositivos. Los experimentos de laboratorio pese a tener buenos resultados requieren ser llevados a entornos reales para verificar su validez [10].

A su vez, los resultados publicados por Michael Baddeley demuestran que la implementación de redes flexibles y convergentes basándose en arquitecturas SDN y protocolos de comunicación para dispositivos IoT; son capaces de reducir significativamente la latencia, consumo energético y entrega de paquetes sin sobrecargar la red, inclusive se ha demostrado que la fusión de estas dos tecnologías de comunicación puede llegar a igualar o superar el rendimiento y escalabilidad de redes de bajo consumo basados en IEEE 802.15.4-2012. En adición, ésta investigación ha demostrado que la migración o la búsqueda de unificación de redes de nodos inalámbricos IEEE 802.15.4 con redes SDN puede suponer una labor compleja, debido a la falta de control sobre los mecanismos de comunicación que inundan la red o a su vez, dependen de modificaciones tanto para los protocolos control SDN así como para los nodos de red[11].

Las SDN no únicamente han surgido para facilitar la tarea de descongestionar redes o reducir el tiempo en la implementación de políticas de bajo nivel, como menciona Chaitanya Aggarwal en su investigación, absolutamente todo tráfico que transita sobre una red es vulnerable a la piratería y monitoreo ilegal e incrementa aún más cuando existe crecimiento descontrolado y carencia de políticas de estandarización. El autor utiliza el paradigma de redes inteligentes SDN como gestor de tareas de configuración rápida y automatizada para dispositivos IoT a través de la redirección, control y acceso al tráfico; lo cual afecta positivamente sobre la seguridad y mecanismos de control de acceso a la información que se comparte sobre la red. Estos resultados, al igual que en anteriores investigaciones pese a tener excelentes respuestas en redes de laboratorio, carecen de información proveniente de redes reales por lo que se proponen trabajos futuros para comprobar su validez [12].

Entre los trabajos más actuales desarrollados en torno a la temática de SDN e IoT, Sergio Baza Alonso utilizó una infraestructura de red virtualizada para integrar las SDN híbridas compuestas por alámbricas e inalámbricas para analizar y gestionar el tráfico inálambrico en un un entorno SDN no tan explotado como son las comunicaciones inalámbricas y controlarlas mediante NVF para reducir los costes de equipos de red físicos. Los resultados que ha obtenido en las pruebas de laboratorio, demostraron lo ventajoso que resulta el uso de las SDN para la gestión del tráfico generado por dispositivos IoT al asociarlos a enrutadores SDN inalámbricos y comandados por un controlador remoto. Sin embargo, la mayoría de las soluciones replicables planteadas, utilizan hardware propietario y NVF locales sobre VirtualBox; lo que limitó su investigación a redes privadas locales controladas [13].

Por lo tanto, de las investigaciones analizadas previamente y con el afán de generar una investigación

sinérgica en la que se posibilite la obtención de resultados provenientes de entornos reales. Para el proyecto investigativo se propuso evaluar los beneficios de vincular las tecnologías SDN e IoT comandados remotamente a través de un entorno NVF para obtener resultados ponderables, recolectados de una red que supondría una tarea caótica al momento de gestionar la gran cantidad de información en tránsito.

2.2. Marco Teórico

Las redes de computadoras actuales en conjunto con su capacidad de conectar casi cualquier objeto a la nube y permitir el acceso a servicios tecnológicos desde cualquier lugar; han forzado a la sociedad digital a transformar las redes IP tradicionales debido a la complejidad y dificultad de administración; es decir, en una red IP no resulta tan simple reestructurar, reconfigurar o modificar las políticas predefinidas por el fabricante de los dispositivos ante posibles fallos, sobrecargas de tráfico o cambios para mejorar la calidad de servicio.

Adicional a ello y para dificultar aún más la posibilidad de mejorar la experiencia del usuario en una red, tanto el plano de control y el plano de los datos se encuentran fusionados y codependientes en todos los dispositivos de red antiguos, impidiendo implementar políticas que únicamente afecten solo a uno de los dos planos mencionados [14].

En relación a esta temática y para afrontar la problemática de control de red, han surgido nuevos mecanismos de control y gestión que se describen a continuación y a lo largo de este apartado.

2.2.1. Redes de computadoras tradicionales

Desde la aparición de las primeras redes de computadoras, el surgimiento y mejora de protocolos, dispositivos, enlaces y tecnologías ha ido en aumento, en la actualidad es casi imposible observar un ambiente que no cuente con al menos un tipo de tecnología de acceso a internet; es decir en cada hogar, negocio o lugar de esparcimiento se cuenta con una pequeña red gestionada a través de un controlador basado en políticas embebidas en hardware propietario de propósito general.

Los enrutadores de las redes tradicionales que persisten en actualidad, no tomaron en cuenta la gestión a nivel de software que los dispositivos requerían, sino más bien, fueron desarrollados tomando como base al hardware como herramienta principal y dejando al software en segunda posición; estrategia que en la actualidad ya no tiene buenos resultados.

A partir de los años 2000, el software de red empezó y continúa experimentando cambios en busca de los niveles más altos de calidad, orientado a mejorar la experiencia del usuario y reducir la complejidad del diseño de red organizada en capas; este concepto de red permite al usuario final acceder a servicios de la forma más rápida posible, sin que éste perciba los detalles del estado de la información, el tipo de enlace, la topología de red o los algoritmos de encapsulación de los datos [15].

El funcionamiento básico de este tipo de red es que cada capa N de un ordenador se comunique con su similar de otro ordenador para mantener un canal de transmisión de datos, limitándose por reglas muy bien establecidas denominadas protocolos de capa, como se visualiza en la figura 2.1; los dispositivos de red llegan a un acuerdo de intercambio de información que es encapsulada en cada paquete de datos provenientes de una capa hacia su inmediata inferior hasta completar un mensaje que pueda ser transmitido por un medio físico y luego descifrado por cada capa homónima del dispositivo receptor.

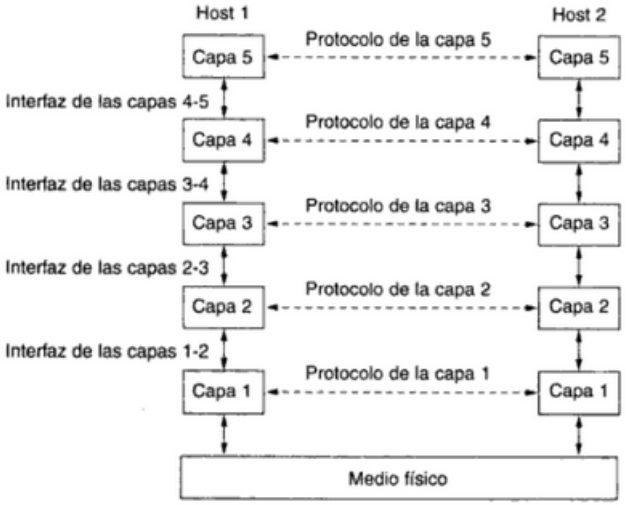


Figura 2.1: Capas, protocolos e interfaces de una red tradicional
Fuente: [15]

Los protocolos de red de transporte y control que se ejecutan dentro de los enrutadores y conmutadores sobre una red como la de la figura 2.2 son las tecnologías clave que permiten que la información, en forma de paquetes digitales, viaje alrededor del mundo. Para expresar las políticas de red de alto nivel deseadas, los operadores de red deben configurar cada dispositivo de forma individual, utilizando comandos de bajo nivel y, a menudo, específicos del proveedor. Además de la complejidad de la configuración, los entornos de red también deben soportar la dinámica de las fallas y adaptarse a los cambios de carga [15].

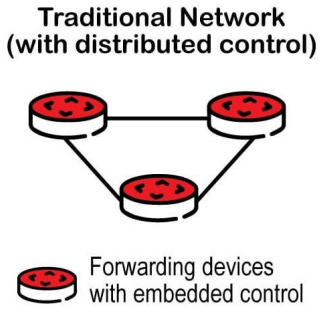


Figura 2.2: Arquitectura de dispositivos de red tradicionales
Fuente: [16]

Plano de control y Plano de datos

En las redes tradicionales, el plano de control determina como manejar el tráfico de red y el plano de datos define como se reenvía el tráfico de acuerdo con las decisiones tomadas por el plano de control, ambos normalmente se agrupan y son codependientes dentro de los dispositivos de red, lo que reduce la flexibilidad, dificulta la innovación y limita la evolución de la red.

La transición de IPv4 a IPv6, que comenzó hace más de una década y aún está en gran medida incompleta, ha sido testigo del desafío de la limitación que representa la agrupación de ambos planos. Basándose en aquella premisa, debido a la inercia de las redes IP actuales, un nuevo protocolo de enrutamiento puede tomar hasta diez años el ser completamente diseñado, evaluado e implementado; de igual forma, en el caso de cambiar la arquitectura de Internet, al requerir demasiado tiempo para despliegue, sería una tarea desalentadora tanto logística como en términos de capital y gastos operativos [14, 15].

2.2.2. Redes de datos para Monitoreo de Seguridad Comunitaria

Debido a que la seguridad es la base para la supervivencia y el desarrollo de una sociedad, hoy en día también es importante fortalecer la tecnología de seguridad y comunicaciones de forma que sea posible proporcionar seguridad a través de herramientas controladas y soportadas remotamente desde uno o varios puntos de monitoreo.

Una red de datos para monitoreo de seguridad comunitaria como la presentada en la figura 2.3, es una combinación de tecnología electrónica, tecnología de sensores, tecnología informática y tecnología de comunicación moderna; la fusión de todas ellas, desempeña un papel difícil o imposible de controlar completamente en la prevención y la lucha contra la delincuencia, el mantenimiento del orden social, la prevención de accidentes de desastres y la reducción de la propiedad estatal, colectiva y la vida de las personas [17].

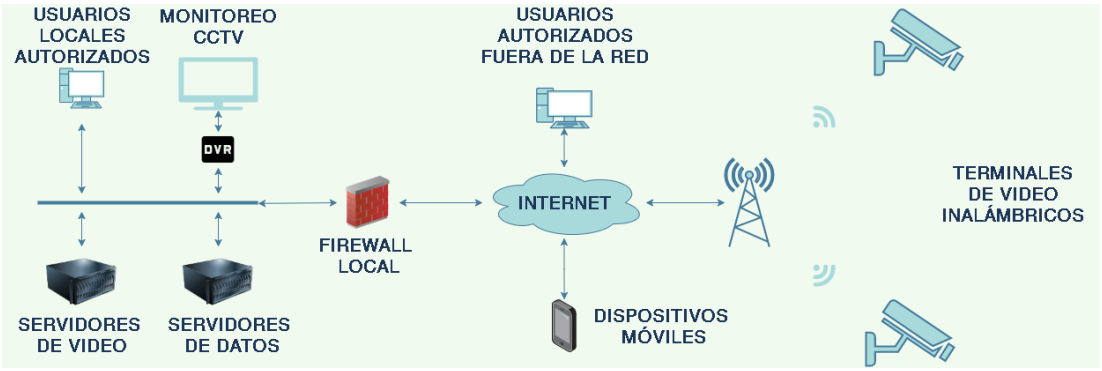


Figura 2.3: Topología de red para seguridad comunitaria
Fuente: [17]

En términos de seguridad ciudadana, el entorno de implementación varía enormemente en dependencia de la zona a monitorear y aún más cuando deben trabajar de forma continua, evitando fallas en comunicaciones y energía. Por lo tanto, la red debe proporcionar: fiabilidad a nodos de video y disponibilidad para módulos inalámbricos especialmente en lugares donde la cobertura de red se torna débil y en consecuencia, una red de datos destinada a soportar un sistema de seguridad ciudadana requiere de una gran cantidad de diseño de red, radiofrecuencia y optimización de los algoritmos de protocolos de software para garantizar la estabilidad del diseño y el entorno de red [17].

2.2.3. Redes Definidas por Software

La red definida por software (SDN por sus siglas en inglés) es un paradigma de red emergente que da la esperanza de cambiar las limitaciones de las infraestructuras de redes actuales. En primera instancia, rompe la integración vertical al separar la lógica de control de la red o plano de control de los enrutadores y conmutadores subyacentes que envían el tráfico o plano de datos; en segunda, con la separación de los planos de control y datos, los conmutadores de red se convierten en simples dispositivos de reenvío y la lógica de control se implementa en un controlador centralizado o sistema operativo de red, lo que simplifica el cumplimiento de la política y la re-configuración y evolución de la red; en la figura 2.4 se muestra una vista simplificada de esta arquitectura.

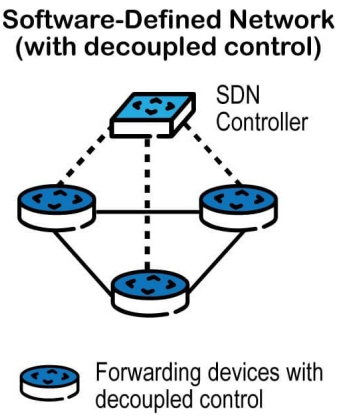


Figura 2.4: Arquitectura de red SDN
Fuente: [16]

Es importante enfatizar que un modelo centralizado lógicamente no postula un sistema físicamente centralizado, de hecho, la necesidad de garantizar niveles adecuados de rendimiento, escalabilidad y confiabilidad impediría tal solución.

Por lo tanto, los diseños de red SDN a nivel de producción recurren a planos de control distribuidos físicamente y la separación del plano de control y el plano de datos se realiza a través de una interfaz de programación bien definida entre los conmutadores y el controlador SDN; de esta forma, el controlador

ejerce control directo sobre el estado en los elementos del plano de datos a través de esta interfaz de programación de aplicaciones (API) bien definida, como ejemplo más notable de tal API se tiene a OpenFlow [14, 15].

2.2.4. OpenFlow

Un conmutador OpenFlow tiene una o más tablas de reglas de manejo de paquetes o tablas de flujo, cada regla coincide con un subconjunto del tráfico y realiza ciertas acciones ante caídas, reenvío, modificación, etc. en el tráfico de red; dependiendo de las reglas instaladas por una aplicación de control el conmutador OpenFlow puede ser instruido a través de una conexión local o remota como se visualiza en la figura 2.5 y adicionalmente puede comportarse como un enrutador, conmutador, firewall o realizar funciones de equilibrador de carga, modelador de tráfico, etc.

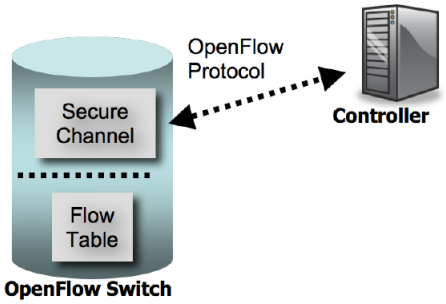


Figura 2.5: Arquitectura de un switch openflow
Fuente: [18]

Una consecuencia importante de los principios SDN es la separación de las tareas introducidas entre la definición de políticas de red, su implementación a nivel de hardware en conmutadores y el reenvío de tráfico; esta separación es clave para la flexibilidad deseada, dividiendo el problema de control de la red en segmentos manejables que facilita la creación e introducción de nuevas abstracciones en las redes, simplificando la administración de la red y posibilita la evolución e innovación de la red [18].

Aunque SDN y OpenFlow comenzaron como experimentos académicos, han ganado gran influencia en la industria de las redes de los últimos años, a través de proveedores de conmutadores comerciales que incluyen soporte de OpenFlow API en sus equipos. El impulso de SDN ha sido lo suficientemente fuerte como para hacer que Google, Facebook, Yahoo, Microsoft, Verizon y *Deutsche Telekom financien Open Networking Foundation* (ONF) opten por incluir funcionalidades en sus redes con el objetivo de promocionarlas; a través del desarrollo de estándares abiertos y a medida que se abordan las preocupaciones de red tradicional mediante soluciones SDN, los mitos preexistentes se desvanecen, un claro ejemplo de ello es la centralización lógica en reemplazo de un controlador centralizado físicamente [19].

2.2.5. Open vSwitch

Open vSwitch (OvS) es una herramienta de software diseñado para operar como un conmutador de red de código abierto, la funcionalidad primaria de OVS es permitir la implementación de conmutadores de calidad comercial que permitan la integración de interfaces de red de propósito general y habilitar funciones de control y envío de tráfico automático.

A inicios, OvS fue diseñado con el afán de proporcionar soluciones de red a gran cantidad de servidores reales o virtualizados basados en Linux repotenciado por los beneficios de la cultura de fuente abierta que permite el acoplamiento y adaptación de su código basado en lenguaje C a casi cualquier sistema operativo [20]; actualmente OvS ha sido explotado para implementar conmutadores en plataformas que proveen servicios de red virtualizados debido a la abstracción de hardware, la estructura base de un dispositivo OvS puede observarse en la figura 2.6.

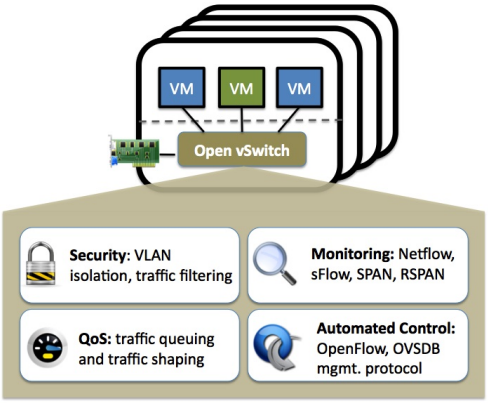


Figura 2.6: Estructura de Open vSwitch
Fuente: [20]

Como se muestra en la figura 2.6, los hipervisores de servidores en la nube, necesitan la capacidad de puentear el tráfico entre máquinas virtuales (VM por sus siglas en inglés) y el mundo exterior; para esta función, anteriormente en los sistemas basados en Linux solía utilizarse conmutadores L2 debido a su rapidez y confiabilidad y es por ello que actualmente Open vSwitch representa una solución similar pero con mayor efectividad y motivo por el que recomienda utilizarlo en entornos de múltiples servidores/servicios donde la arquitectura de red tradicional no es óptima [21].

Las ventajas de OvS permiten satisfacer condiciones como:

- Movilidad de estado: permite migrar configuraciones y estados de red incluyendo reglas SPN y QoS sin necesidad de apagar el entorno.
- Respuesta dinámica: proporciona características para control de red adaptativa dependiendo del tipo de entorno.

- Etiquetas lógicas: incluye mecanismos de marcación y etiquetado de paquetes en transito para su gestión remota.
- Integración de hardware: diseñado para modificar el procesamiento de los paquetes de datos en uno o varios conjuntos de conmutadores reales o virtuales. Permitiendo la implementación de dispositivos de control de red basados en hardware o software puro.

2.2.6. Protocolo abierto de administración de bases de datos vSwitch (OvSDB)

El proyecto Open vSwitch incluye implementaciones de cliente y servidor bajo el protocolo de código abierto para administración de bases de datos (OvSDB por sus siglas en inglés), este protocolo de administración OvSDB que emplea JSON-RPC versión 1.0 está destinado a permitir el acceso programático a la base de datos Open vSwitch.

La base de datos es el asistente lógico que contiene la configuración para un servicio o demonio (daemon en linux) Open vSwitch, aunque se debe tener en cuenta que esta información describe por completo el comportamiento de un conmutador virtual más no el comportamiento o la configuración de un sistema de enrutamiento [22].

En la figura 2.7 se ilustran los componentes principales de Open vSwitch y las interfaces para un clúster de control y administración de red, una instancia de OVS común se compone de: un servidor de base de datos «servidor OvSDB», un demonio vswitch «ovs-vswitchd» y opcionalmente un módulo que realiza el reenvío de ruta rápida.

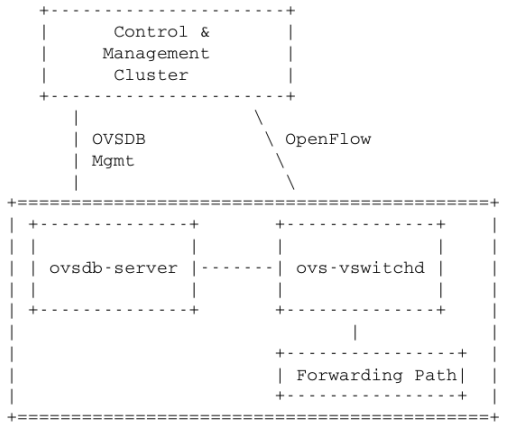


Figura 2.7: Infraestructura de una instancia OvS
Fuente: [22]

El "clúster de gestión y control" descrito en la figura 2.7 puede estar compuesto por uno o varios controladores SDN. El o los controladores utilizan el protocolo de administración OvSDB para administrar instancias OVS y es necesario mencionar que toda instancia de OVS es administrada por al menos un administrador que utiliza OpenFlow para instalar el estado de flujo en los conmutadores.

Por un lado, la interfaz de administración de OvSDB es utilizada para realizar operaciones de administración y configuración en la instancia de OVS, mientras que OpenFlow se encarga de las operaciones de administración de OvSDB que se producen en una escala de tiempo relativamente larga [22].

Las operaciones de red que son compatibles con OvSDB incluyen:

- Creación, modificación y eliminación de rutas de datos OpenFlow, de las cuales puede haber muchas en una sola instancia de OVS.
- Configuración de controladores a los que debe conectarse una ruta de datos OpenFlow.
- Configuración del conjunto de administradores a los que debe conectarse el servidor OvSDB.
- Creación, modificación y eliminación de puertos en rutas de datos OpenFlow.
- Creación, modificación y eliminación de interfaces de túnel en rutas de datos OpenFlow.
- Creación, modificación y eliminación de colas también conocidas como *queues* en inglés.
- Configuración de políticas de calidad de servicio y conexión de esas políticas a las colas.
- Colección de estadísticas.

2.2.7. Red Virtual Abierta (OVN)

Una Red Virtual Abierta (OVN por sus siglas en inglés) es una solución SDN basada en Open vSwitch para suministrar servicios de red de uso común en los servicios virtualizados de RedHat, la topología de red referencial se puede observar en la figura 2.8; en síntesis un OVN permite conectar mediante software a grupos de instancias utilizando herramientas de propósito general con la capacidad replicarse y crecer como red.

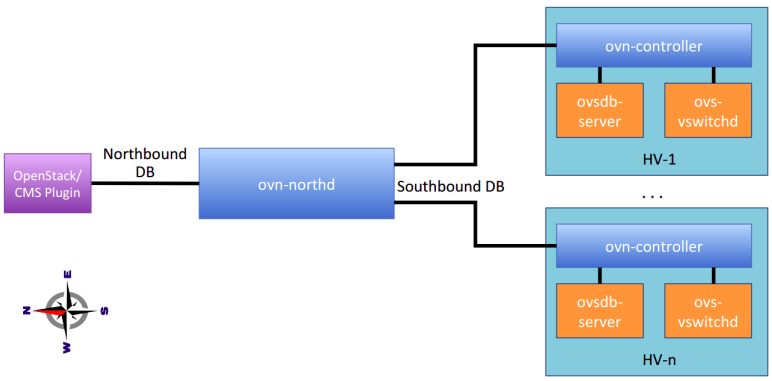


Figura 2.8: Mecanismo de Control OVN
Fuente: [23]

La arquitectura de una OVN reemplaza el segmento de capa 2 con el complemento modular OVN de capa 2 (ML2 por sus siglas en inglés) para admitir cambios generados a través de la API de red como se muestra en la figura 2.9 [23], los componentes de la arquitectura son:

- Complemento OVN ML2: convierte las configuraciones de red específica en configuraciones de red lógica sin importar la plataforma.
- Base de datos de norte OVN-NB: almacena la configuración de la red lógica del complemento OVN ML2.
- Servicios OVN de norte ONV-NORTBOUND: convierte la configuración de la red almacenada en OVN-NB y las complementa con las de OVN.SB.
- Base de datos de sur OVN-SB: almacena la ruta de los flujos de datos lógicos.
- Controlador OVN: Se conecta a OVN-SB y se encarga de monitorear y controlar el tráfico en la red.

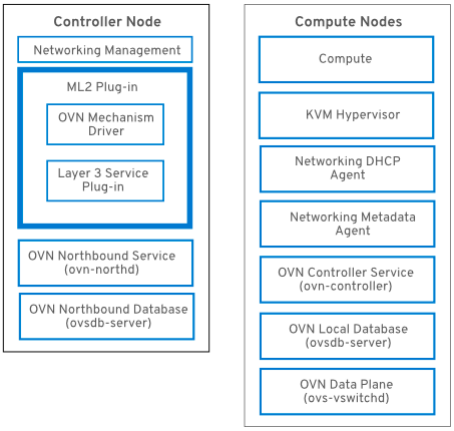


Figura 2.9: Arquitectura OVN
Fuente: [23]

2.2.8. Controladores SDN

Existen varias soluciones a nivel de software para la implementación de controladores SDN, en la tabla 2.1 se presenta el análisis comparativo de los principales controladores y de alta relevancia para la investigación desarrollada; basándose en las características evaluadas, los cuatro aplicativos hasta la actualidad han alcanzado la madurez necesaria para posibilitar la implementación de soluciones de alto nivel para el mejoramiento de redes emergentes.

Sin embargo, tanto Floodlight como ONOS, pese a que soportan OpenFlow (OF), no cuentan con soporte para dispositivos con estructura OVSDB; motivo por el cual en la investigación cursada no fueron seleccionadas como herramientas para el desarrollo. En el caso de OpenDayLight, ha permitido

ha desarrolladores implementar redes SDN para datacenters y capa de transporte en redes WAN SDN de alto rendimiento; no obstante, no ha sido optimizada para la implementación en campo a diferencia de Ryu y Floodlight.

Finalmente, el controlador RYU además de ser compatible con OF versiones 1.0 a 1.5 y OVSDB ha demostrado estar en la capacidad de soportar soluciones para redes de campo; aunque no es un controlador multiplataforma y carece de un entorno de configuración, el lenguaje de programación Python con el que ha sido implementado, otorga a los desarrolladores la capacidad de generar un sin número de soluciones ligeras para gestión, monitoreo y control multitarea y multiplataforma acorde a las necesidades del usuario final [13].

Tabla 2.1: Controladores para redes definidas por software

	RYU	**Floodlight**	**OpenDayLight**	**ONOS**
Interfaz SUR	OpenFlow 1.0 - 1.5, Netconf, Ofconfig, OVSDB	OpenFlow 1.0 - 1.5	OpenFlow 1.0 - 1.5, Netconf/Yang, OVSDB, PCEP, BGP, LISP, SMTP, Ofconfig	OpenFlow 1.0 - 1.5, Netconf
Interfaz de Programación	Si	Si	Si	Si
Interfaz Gráfica	Fase de desarrollo	WEB/basado en java	WEB	WEB
Modularidad	Mediana	Mediana	Alta	Alta
Soporte de orquestación	Si	Si	Si	No
Sistema Operativo	Linux	Linux, Windows, MacOS	Linux, Windows, MacOS	Linux, Windows, MacOS
Documentación	Muy buena	Buena	Muy buena	Buena
Lenguaje de programación	Python	Java	Java	Java
Soporte mulihilera	Si	Si	Si	Si
Soporte TLS	Si	Si	Si	Si
Entornos de virtualización	OVS y Mininet	OVS y Mininet	OVS y Mininet	OVS y Mininet
Dominios de aplicación	redes en campo	redes en campo	Datacenter y capa de transporte en redes WAN SDN	Datacenter y capa de transporte en redes WAN SDN
Desentralización	No	No	Si	Si

Fuente: Elaborado por el investigador basado en [13][24]

2.2.9. Funciones de Red Virtualizadas (NVF)

La computación en la nube y virtualización son tecnologías directamente relacionadas que han surgido para aligerar los paradigmas de redes de computadoras preexistentes, las funciones de red virtualizadas

(NVF por sus siglas en inglés) proporcionan una solución efectiva en base a hardware computacional configurable alojado en la nube del proveedor del servicio; solución que otorga al usuario final el acceso servicios de red como servidores, servicios específicos, infraestructura de red y gestores y bases datos, sin requerir de los altos costos de inversión en hardware ni las limitantes relacionadas con elasticidad, flexibilidad o disponibilidad.

El principio de la teoría de las NVF es la separación de los servicios de red del hardware sobre el que funciona normalmente, en la figura 2.10 se muestra como el hardware necesario es abstraído virtualmente en la nube del proveedor para proporcionar al usuario final el servicio de forma transparente; es decir, no se percibe si se utiliza un servicio basado en hardware o software, sin embargo, esta abstracción de recursos y consumo bajo demanda de los servicios virtualizados, genera un impacto positivo sobre términos energéticos, económicos y ambientales [25].

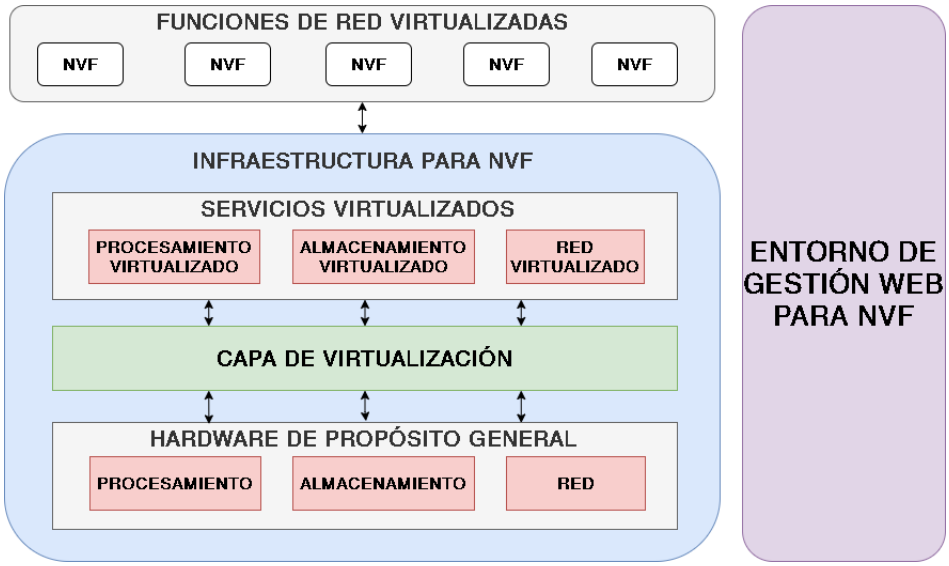


Figura 2.10: Arquitectura de sistemas para Funciones de Red Virtualizadas
Fuente: Desarrollado por el investigador basado en [25] y [26]

2.2.10. Internet de las Cosas (IoT)

El alcance de internet ha abarcado a objetos y dispositivos cotidianos que hasta la actualidad pueden incluir algún tipo de inteligencia, dichos dispositivos están en la capacidad de compartir información con el mundo que los rodea, de allí donde toma el nombre conocido como Internet de las Cosas (IoT por sus siglas en inglés). Para el año en curso se ha pronosticado que se duplicaría la cantidad de dispositivos conectados que para el año 2010 alcanzaba una cifra cercana a los 25 mil millones con al menos un dispositivo inteligente por persona en el mundo.

El IoT ha habilitado funciones de acceso a datos y servicios complementarios para áreas cotidianas como estudio, salud, seguridad, transporte, etc. simplificando las tareas de la humanidad moderna y se espera que sea la herramienta ideal para dotar al ser humando de redes inteligentes y servicios personalizados en

base a necesidades específicas; lo que a su vez conlleva a un incremento significativo en recursos y datos transmitidos, debido a que los dispositivos requieren adquirir y proporcionar información digitalizada a redes bidireccionales para su consumo a gran escala [27]. En la figura 2.11 se pueden observar los tres pilares en los que IoT genera convergencia, dotando a objetos e individuos capacidades que antes carecían.

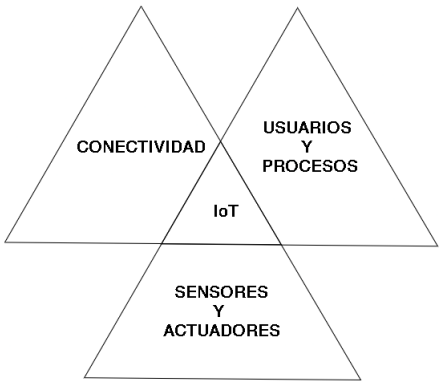


Figura 2.11: Concepto de IoT
Fuente: [27]

Nuevas investigaciones a nivel mundial, han logrado reducir la sobrecarga de información transmitida, modificando la estructura de los datos; la infraestructura de IoT puede dividirse en 4 capas de información:

- Capa de detección de objetos: Sensores, los objetos físicos y la obtención de datos.
- Capa de intercambio de datos: Transmisión transparente de datos a través de redes de comunicación
- Capa de integración de la información: El procesamiento de la información incierta adquirida de las redes, filtrado de datos no deseados e integración de información principal en conocimiento útil para los servicios y los usuarios finales.
- Capa de servicios y aplicaciones: Da servicios de contenido a los usuarios.

Para alcanzar el nivel de respuesta que se espera, la tendencia de la red obliga a incrementar velocidades de transferencia y reducir la latencia en la conectividad, obligando a la red global a duplique su tamaño cada 5.3 años; por lo tanto, no es de extrañarse que una de las herramientas tecnológicas importantes sean los servicios de computación en la nube con el motivo de permitir a los usuarios acceder a tecnologías maduras existentes sin profundizar en conocimientos o experiencia [27].

Protocolo de Mensajería Ligero MQTT

En una red congestionada por docenas de dispositivos transmitiendo datos de forma recurrente, los protocolos HTTP preexistentes dejaron de representar una solución efectiva debido a los retardos o cuellos de botella que se generan durante el intercambio de información; en consecuencia, ha surgido el protocolo para Transporte de Telemetría MQ (MQTT por sus siglas en inglés), un protocolo ligero diseñado para comunicaciones Máquina-Máquina (M2M) o dispositivos IoT, en la figura 2.12 se puede observar la arquitectura de red que utiliza MQTT [28].

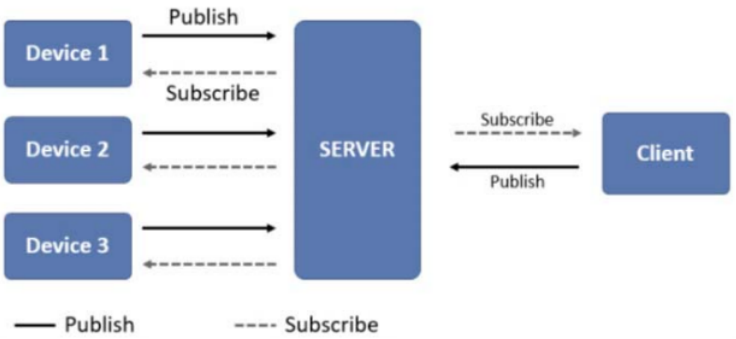


Figura 2.12: Arquitectura de una red de dispositivos IoT
Fuente: [29]

Generalmente MQTT trabaja en la cima de la pila de TCP/IP con un mecanismo de intercambio de información publicar/suscribir coordinado por un gestor *o BROKER*, el funcionamiento estándar de un sistema de comunicaciones basado en MQTT consiste en que los clientes de publicación envían sus datos a colas de mensajería transitoria llamados *TOPICS*; todos los mensajes dentro de un *topic* son enrutados por el *broker* para ser leídos por clientes suscritos y pueden ser organizados jerárquicamente a través de separadores "/" similares a directorios de un sitio web denotando su nivel [30].

Los componentes lógicos primarios de una arquitectura IoT MQTT son:

- Identificador de paquete (*Packet Id*): aplica para mensajes con QoS diferentes a cero. Sirve para identificar el paquete y posibles respuestas relacionadas al paquete.
- Bandera (*dup*): puede tomar el valor de uno o cero. Sirve para generar redundancia en un mensaje que ha sido publicado.
- Calidad de Servicio (*QoS*): Establece un nivel de QoS para un mensaje. Sirve para otorgar importancia al mensaje publicado en un tema específico y puede tomar valores de 0 a 2 donde cero es el valor de mayor relevancia.
- Conservar (*Retain*): cumple funciones de bandera uno o cero y habilita la función de almacenamiento del mensaje en el servidor MQTT en el caso de que sea necesario.

- Tema (*topic*): es una cadena de texto que establece el nombre del *topic* al que los suscriptores envían los mensajes con determinado valor de QoS.

Mosquitto MQTT

Tomando como punto de referencia la premisa del protocolo MQTT, el proyecto de fuente abierta nombrado «*Mosquitto*» fue diseñado por Eclipse para cumplir con funciones de gestor central de una red IoT con la capacidad de soportar las versiones 3.1, 3.1.1 y 5 del protocolo MQTT. La ventaja predominante de Mosquitto es que no requiere de hardware de altas prestaciones para funcionar, permitiendo a los desarrolladores integrarlo tanto en dispositivos microcontrolados de baja potencia como en servidores de alto rendimiento [31].

Actualmente, Mosquitto ofrece soluciones de software integrables en dispositivos IoT relativamente minúsculos gracias al avance de lenguajes de programación basados en C; ante tal circunstancia, Mosquito fue seleccionado como software primario para gestionar la arquitectura de IoT compuesta por teléfonos, computadoras embebidas y microcontroladores que el proyecto de investigación requería.

2.2.11. Calidad de Servicio (QoS)

La Calidad de Servicio (QoS por sus siglas en inglés) hace referencia a evaluar la experiencia que tiene un usuario final de red al interactuar con servicios de internet, estos parámetros permiten a los clientes elegir que proveedor de servicios de internet consumir y a los proveedores conocer las preferencias y necesidades de sus clientes; en base a ello, es posible diseñar nuevos productos o mejorar los ofertados donde los principales puntos a considerar en términos de QoS son:

- ancho de banda
- retardos entre la transmisión/recepción de paquetes, también llamado *jitter*
- y retardo o pérdida de paquetes

Al mantener como objetivo primario estos parámetros al otorgar un servicio, los proveedores y fabricantes de dispositivos pueden prevenir inconformidades, a través de la modificación, optimización o expansión de infraestructura de red y otorgar nuevas funcionalidades al equipamiento existente [32].

CAPÍTULO III

METODOLOGÍA

3.1. Ubicación

El trabajo de investigación fue desarrollado en la ciudad de Ambato, dentro del casco urbano de la misma. Para efectos de investigación, se seleccionaron dos puntos de vigilancia en las parroquias La Matriz y Celiano Monge respectivamente, debido a la facilidad de acceso a redes y posibilidad de manipulación directa de sus enrutadores primarios.

3.2. Equipos y materiales

Los equipos y materiales utilizados para el desarrollo de la investigación se describen en la tabla 3.2.

Tabla 3.2: Equipos y materiales

Equipos, servicios o materiales	**Costo (USD)**
Equipos	690
Materiales	100
Transporte	100
Gastos varios	200
Total	1090

Fuente: Elaborado por el Investigador

3.3. Tipo de investigación

3.3.1. Investigación Bibliográfica

Se ha seleccionado la metodología de investigación bibliográfica debido a que gran cantidad de información requerida para la implementación del estudio, pudo hallarse de forma dispersa en artículos científicos, publicaciones de estudios de maestría o doctorado y libros o manuales provistos por desarrolladores y fabricantes.

3.3.2. Investigación Exploratoria

La investigación también se definió como de tipo exploratoria debido a que se inicia desde una visión generalizada del tema de vinculación de dos arquitecturas y se la lleva a una realidad determinada. Además de que existen pocos estudios previos del tema, no existían hasta la fecha de presentación del trabajo una publicación sobre la arquitectura propuesta fuera de entornos de laboratorio.

3.4. Prueba de hipótesis

El presente trabajo de investigación, por su naturaleza «Alarmas Comunitarias basadas en Arquitecturas SDN e IoT», corresponde a una modalidad de desarrollo de tipo exploratorio. Es muy conocido que

los tipos de tecnología que la investigación aborda «SDN e IoT» son tecnologías relativamente nuevas, desarrolladas para aligerar la carga sobre las redes de datos tradicionales y han alcanzado una madurez estable y continúan en fase de desarrollo en ámbito de gestión centralizada y seguridades; sin embargo la implementación y prueba de redes que correlacionen ambas tecnologías de forma interoperable es relativamente nueva y los estudios en desarrollo se encuentra en etapas de prueba y evaluación.

La investigación pretende vincular la madurez de dos tecnologías y evaluar los beneficios que aportaría cada una de ellas; a través de un análisis experimental en un ambiente de red controlado que proporcione información relevante sobre la interacción, control y gestión unificada y los beneficios o debilidades que supone la correlación de ambas tecnologías en una misma red que requiere respuestas en tiempo real.

Hipotesis

Según (Sampieri H 1997 Pág. 13) los estudios exploratorios se efectúan, normalmente cuando el objetivo es examinar un tema o problema de investigación poco estudiado o no abordado antes y por lo tanto carecen de hipótesis**.**

Variables

Al no contar con una hipótesis no es posible establecer variables de operacionalización.

3.5. Población y muestra

Debido al tipo de investigación desarrollado, no fue necesario definir población y muestra. Todos los resultados obtenidos en la etapa de pruebas han sido considerados para el estudio.

CAPÍTULO IV

ANÁLISIS E INTERPRETACIÓN DE RESULTADOS

4.1. Análisis de tráfico en red convergente tradicional

Debido a la naturaleza exploratoria de la investigación, se propuso inicialmente el análisis de tráfico de una topología de red tradicional previo a la implementación de políticas de control de calidad de servicio sobre una SDN, de tal manera que permita evidenciar las desventajas de utilizar una red con planos de control y datos agrupados y sin políticas de QoS dedicada a:

- Monitoreo de video en tiempo real,
- transmisión y recepción de audio en intranet (VoIP),
- control de dispositivos a través de protocolos IoT
- y navegación web y/o generación de tráfico de datos.

Es necesario puntualizar que todos los dispositivos que conforman la topología descrita en la figura 4.13 forman parte de una red local privada que cuenta con un ancho de banda contratado a su proveedor de servicios (ISP) de 20Mbps a través de un hilo de fibra de compartición 2:1.

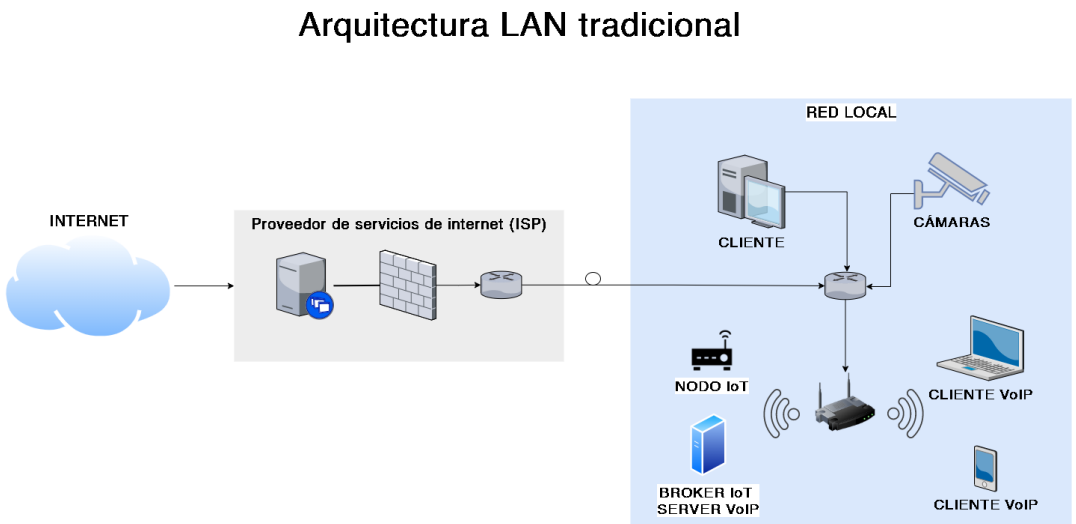


Figura 4.13: Arquitectura de red LAN tradicional
Fuente: Desarrollado por el Investigador

Para la etapa de videovigilancia, la arquitectura de estudio contó con una grabadora de video digital (DVR) de 4 canales, 2 de ellos dedicados netamente al monitoreo continuo de puntos estratégicos de alto tráfico vehicular y peatonal; a su vez, con el afán obtener tráfico de voz digitalizada sobre la red de

estudio, se ha utilizado un servidor para VoIP basado en CentOS y Asterisk con al menos dos clientes internos, quienes además tienen la capacidad de manipular y monitorear la data proveniente de nodos IoT.

Y en adición, se contó con al menos un cliente fijo con la capacidad de navegar sobre la red y acceder a servicios web sin ninguna política de restricción implementada salvo las provistas por las reglas predefinidas en el enrutador primario.

Un primer análisis del ancho de banda provisto por el ISP se muestra en la figura 4.14 donde es posible evidenciar la capacidad del canal contratado, unicamente con dispositivos de red de uso común en una red local tradicional. Es necesario tomar en cuenta que para evaluar la capacidad del canal y obtener el valor exacto de ancho de banda contratado, fue en horas en las que los dispositivos no saturen la red debido al consumo de servicios.

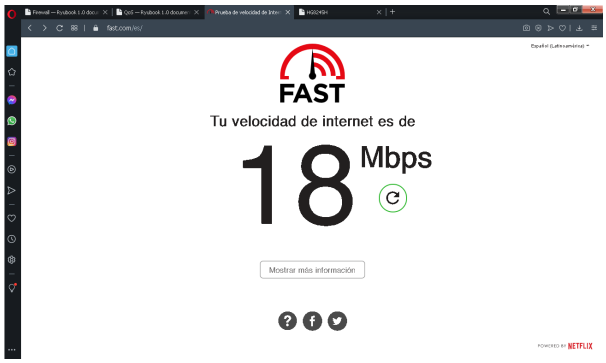


Figura 4.14: Ancho de banda de salida contratado con compartición 2:1 sin saturación
Fuente: Desarrollado por el investigador

En el caso de desarrollar el análisis del ancho de banda en horas laborables, inclusive sin utilizar la red para servicios adicionales como monitoreo de redes de Circuito Cerrado de Televisión (CCTV), dispositivos IoT, servicios de audio por VoIP, etc. es posible evidenciar en la figura 4.15 la saturación en el ancho de banda de red al punto de afectar el correcto funcionamiento de uno o varios servicios.

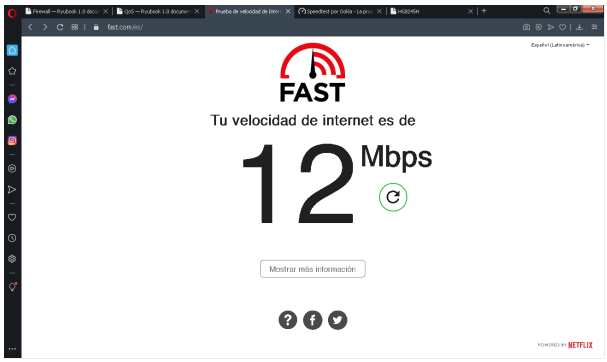


Figura 4.15: Ancho de banda de salida limitada por sobrecarga generada por dispositivos comunes
Fuente: Desarrollado por el investigador

En la figura 4.16 se muestra el listado completo de los dispositivos conectados al enrutador primario y que la gran cantidad de dispositivos que acceden a la red son el motivo por el cual las redes tienden a saturarse debido a la falta de políticas de control, por lo tanto el alto consumo de ancho de banda de determinados servicios son los causantes de cuellos de botella en redes tradicionales.

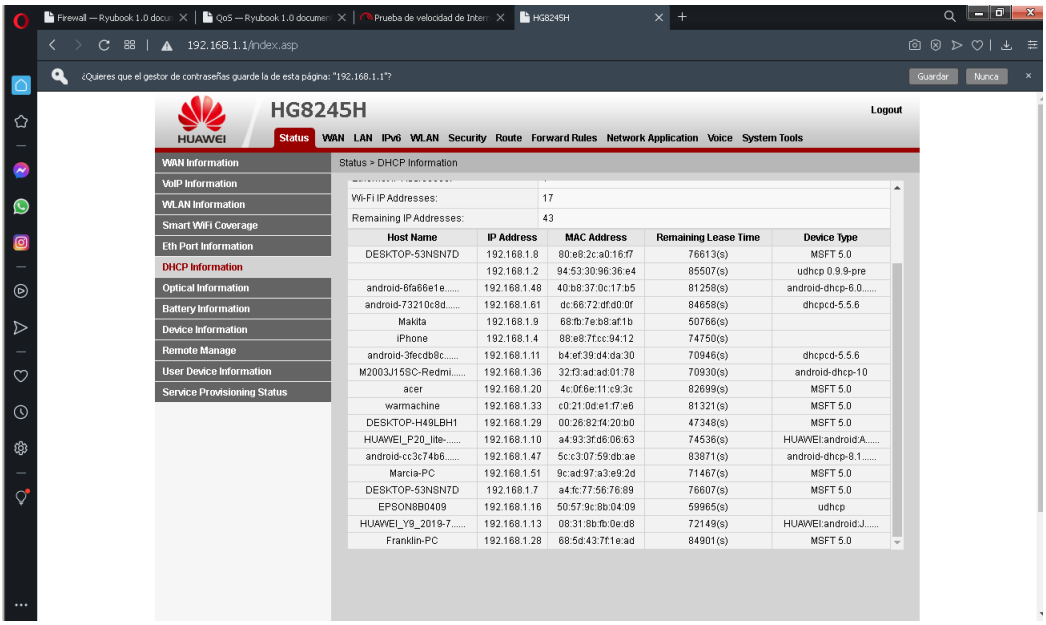


Figura 4.16: Listado de dispositivos conectados al enrutador primario
Fuente: Desarrollado por el investigador

4.1.1. Pruebas de ancho de banda en red tradicional sin servicios IoT, VoIP o CCTV

Para evaluar las limitaciones de ancho de banda en un segmento de red, se diseñó un entorno de prueba simple pero efectivo basado en el proyecto de software libre *JPERF* [33] creado para pruebas de rendimiento en redes, la topología de análisis propuesta se puede visualizar en la figura 4.17 en donde el principio de análisis consistió en generar ráfagas de sobrecarga sobre el protocolo UDP entre dos dispositivos servidor - cliente conectados a través de un enrutador primario como punto de fallo.

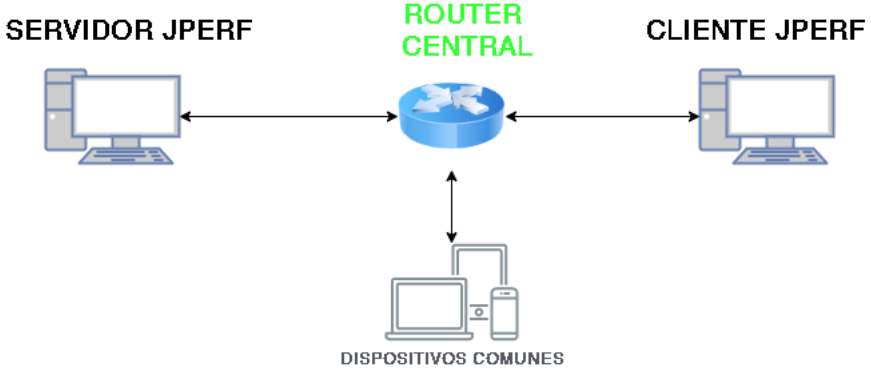
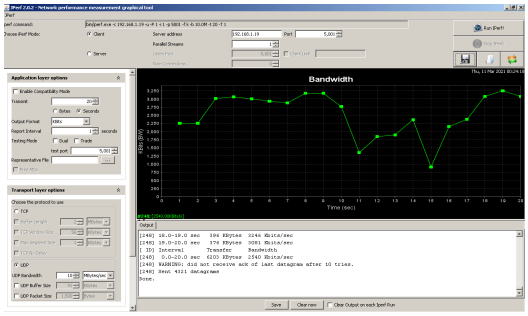


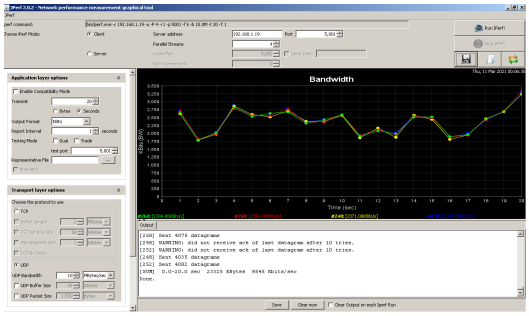
Figura 4.17: Topología de evaluación de red
Fuente: Desarrollado por el investigador

Los resultados presentados en la figura 4.18, obtenidos de las pruebas de rendimiento de una topología como la de la figura 4.17 demostraron que inclusive en una red sin tráfico crítico tiende a saturarse debido a la falta de políticas de gerencia.

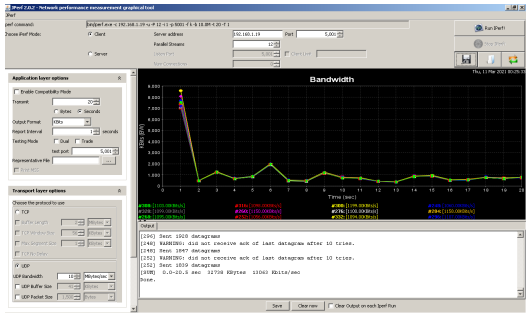
De la figuras 4.18, es posible evidenciar que durante una prueba de rendimiento con duración de 20 segundos entre un servidor JPERF con IP 192.168.1.19 y el cliente JPERF con IP 192.168.1.28; pese a que se estableció un umbral de ancho de banda de 10 Megabits por segundo (Mbps), únicamente se alcanzaron anchos de banda de 3.5Mbps de 1 a 4 hilos paralelos como se puede observar en las figuras 4.18.a y 4.18.b y en conformidad con el crecimiento de dispositivos o hilos que saturen la red con paquetes de datos, estos umbrales tienden a ser mucho más bajos de hasta máximo 2Mbps como se muestra en la figura 4.18.c.



a) Análisis de ancho de banda entre cliente - servidor con 1 flujo de datos



b) Análisis de ancho de banda entre cliente - servidor con 4 flujos de datos paralelos



c) Análisis de ancho de banda entre cliente - servidor con 12 flujo de datos paralelos

Figura 4.18: Resultados de análisis de ancho de banda en red con dispositivos tradicionales
Fuente: Desarrollado por el investigador

4.1.2. Pruebas de ancho de banda en red tradicional con servicios IoT, VoIP o CCTV

Como se evidenció en el apartado anterior, con el crecimiento de la red los problemas de saturación tienden a crecer de forma proporcional y aún más al tratarse de la incorporación de servicios de tráfico crítico de audio y video. En la figura 4.19 se muestra la arquitectura de red tradicional anteriormente presentada en la figura 4.17, complementada con servicios de CCTV, VoIP que comúnmente son los causantes de la saturación de las redes convencionales y dispositivos IoT locales.

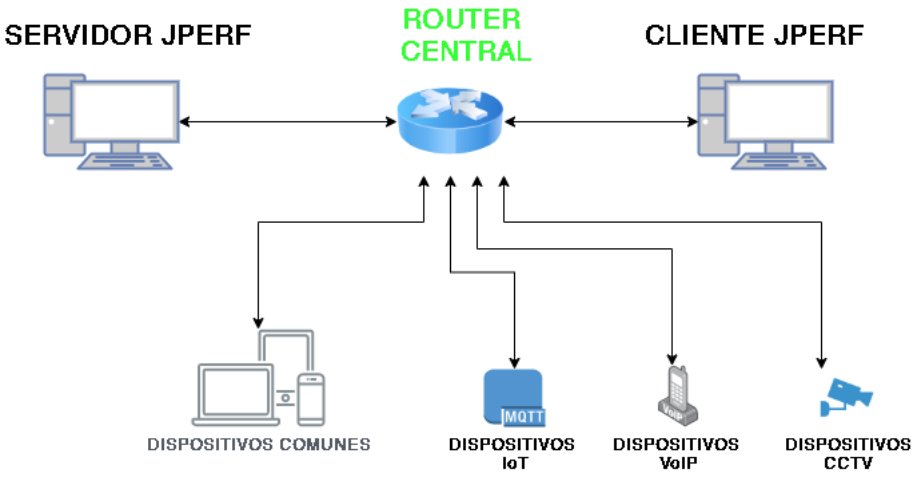
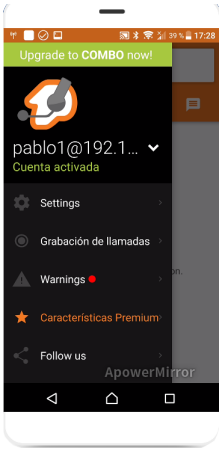


Figura 4.19: Topología de evaluación de red
Fuente: Desarrollado por el investigador

En primera instancia, como se muestra en la figura 4.20 se habilitaron dos usuarios de servicios de VoIP-SIP con funcionalidades para recibir llamadas entre sí; permitiendo saturar la red con servicios de audio digitalizado debido que los codecs de compresión de audio consumen ancho de banda de red para garantizar un servicio de calidad.



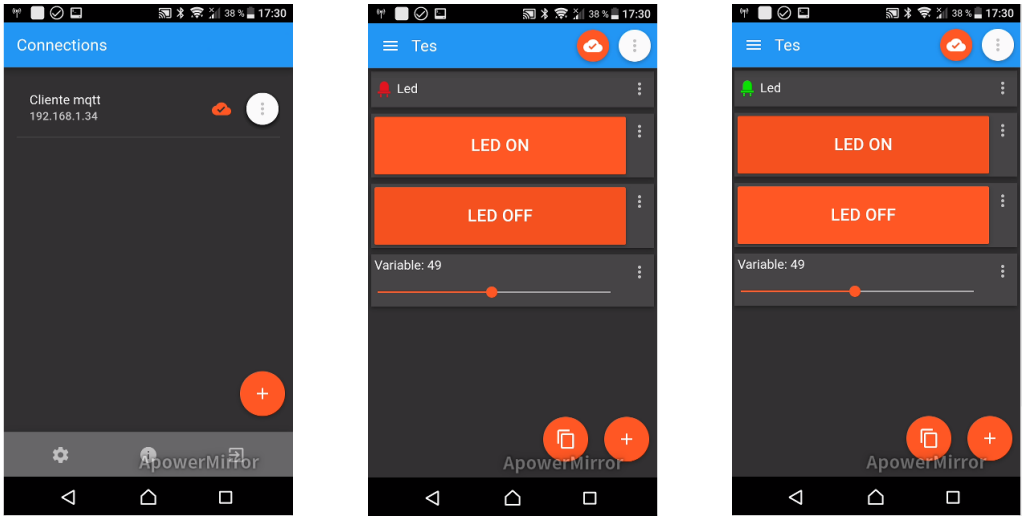
a)Terminal VoIP 1 en Zoiper

b)Terminal VoIP 2 en 3CX

Figura 4.20: Activación de cuentas de VoIP sobre red local
Fuente: Desarrollado por el investigador

En adición, se habilitaron dos dispositivos locales que trabajan con protocolos IoT-MQTT, ambos con la capacidad de trabajar como publicadores y suscriptores para la activación de actuadores simples.

Como se muestra en la figura 4.21.a el dispositivo se conectó a un broker local y en las figuras 4.21.b y 4.21.c se comprobó su óptimo funcionamiento para transmitir señales tanto discretas provenientes de la activación de dispositivos como señales continuas generadas por variante aleatorias.



a) Registro de suscriptor b) Prueba de conexión c) Prueba de activación

Figura 4.21: Activación de dispositivos IoT sobre red local
Fuente: Desarrollado por el investigador

La evaluación del ancho de banda mostrada en la figura 4.21 demostró que los servicios preexistentes continúan saturando la red; sin embargo, los servicios de VoIP pueden llegar a limitar el tráfico de salida inclusive si únicamente son servicios locales y en adición se comprobó que la variación en el valor de ancho de banda presentado en la figura 4.21 no varía durante la modificación de variables en los dispositivos de IoT.

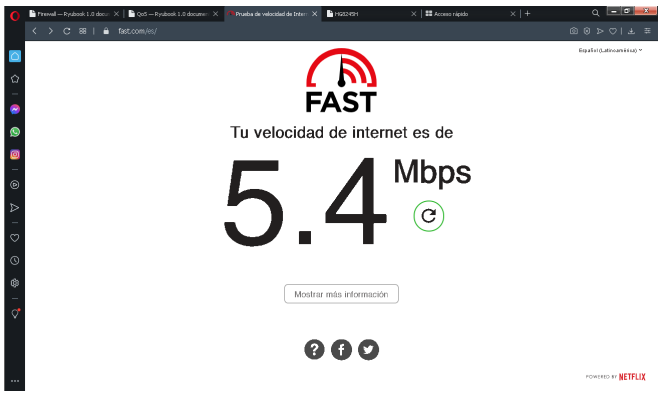
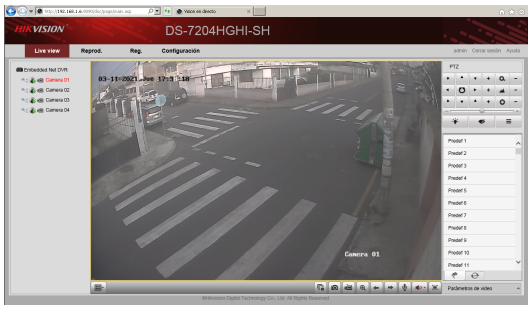
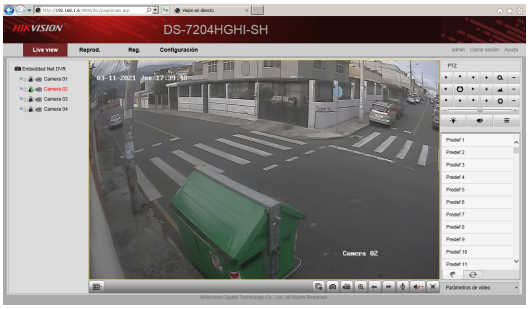


Figura 4.22: Evaluación de tráfico de salida con servicios de VoIP e IoT
Fuente: Desarrollado por el investigador

Para culminar con las pruebas de red tradicional, se desarrolló un análisis de ancho de banda después de habilitar el sistema de CCTV para generar carga de tráfico de video de forma local como se muestra en la 4.23.a y 4.23.b y se otorgaron funcionalidades de acceso al servicio de monitoreo de forma remota a través de un monitor remoto como se observa en las figuras 4.23.c y 4.23.d.



a) Acceso local a Cámara 01



b)Acceso local a Cámara 2

c) Acceso remoto a Cámara 01

d)Acceso remoto a Cámara 2

Figura 4.23: Habilitación de servicios de videovigilancia, local y remota CCTV
Fuente: Desarrollado por el investigador

El análisis de ancho de banda del tráfico de salida presentado en la figura 4.23 evidenció que en efecto, la ejecución de los servicios de video, audio y datos sobre una red tradicional sin políticas de calidad de servicio tienden a saturar la red de forma descontrolada e inclusive pueden llegar a afectar la funcionalidad de servicios de red como navegación web y video streaming por citar un par de ellos.

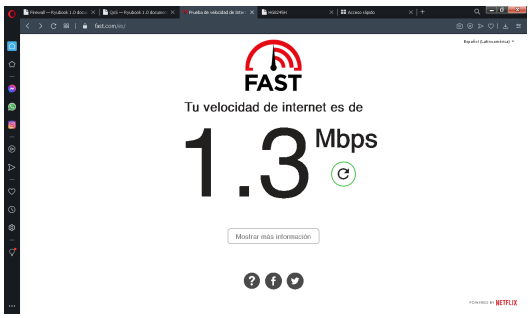
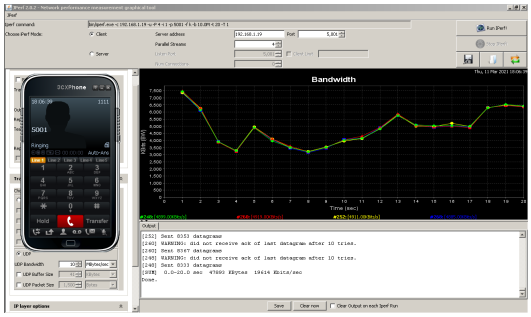
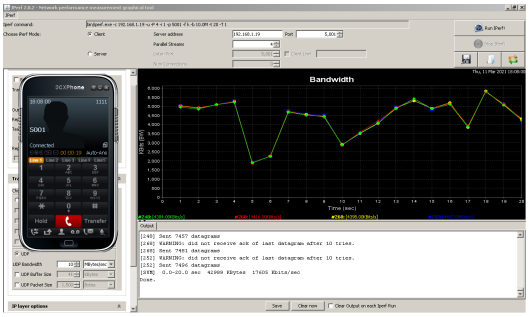


Figura 4.24: Evaluación de tráfico de salida con servicios de Video, VoIP, datos y tráfico sobre protocolo IoT
Fuente: Desarrollado por el investigador

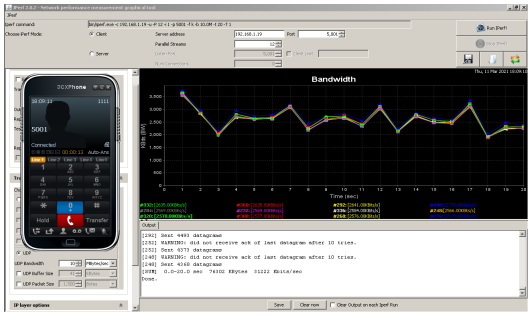
Finalmente, las pruebas obtenidas de la ejecución de todos los servicios mencionados con anterioridad dentro de una LAN, han demostrado que no unicamente se ve afectada la estabilidad de la red en cuanto a servicios provenientes de internet, sino que también afectan la estabilidad de servicios locales como se muestra en la figura 4.25.



a) Análisis de tráfico entre dos puntos de red LAN durante llamada entrante VoIP



b)Análisis de tráfico entre dos puntos de red LAN durante asignación de canal VoIP



c) Análisis de tráfico entre dos puntos de red LAN durante canal VoIP ocupado

d)Análisis de tráfico entre dos puntos de red LAN una vez liberado el canal de comunicación VoIP

Figura 4.25: Evaluación de tráfico en red LAN con servicios de VoIP e IoT y CCTV
Fuente: Desarrollado por el investigador

El análisis de las imágenes del comportamiento de la red LAN se presenta en la tabla 4.3, previamente es necesario recalcar que durante todas las pruebas, cuyos resultados son presentados en las figuras 4.25, también se encontraban en ejecución los servicios de navegación y acceso a servicios web, CCTV local y remoto y monitoreo de señales provenientes de dispositivos IoT.

Tabla 4.3: Análisis del ancho de banda obtenido dentro de una LAN con servicios de VoIP e IoT y CCTV.

Referencia	**Análisis de resultados**
Figura 4.25.a	Durante el inicia llamada VoIP entre dos clientes, el ancho de banda de red presenta una reducción en sus valores máximos que pasan de 7.5Mbps a 3.5Mbps para estabilizarse entre 5 y 7 Mbps.
Figura 4.25.b	Los valores de ancho de banda se reducen considerablemente cuando se establece el canal de VoIP. Alcanzando un valor crítico de 1.8Mpbs aproximadamente para de forma intermitente estabilizarse entre 4 y 6 Mbps.
Figura 4.25.c	Durante la duración de la llamada VoIP se pudo observar que la red presenta una reducción en sus valores de ancho de banda. Limitando sus niveles entre 1.8 y 3.8 Mbps.
Figura 4.25.d	Finalmente, cuando el canal VoIP es liberado. Las políticas predefinidas en el hardware del enrutador primario tardan en restablecer la red a sus valores normales. Y se presenta una limitación en el ancho de banda con valores que fluctúan entre 1.75 y 3Mbps

Fuente: Desarrollado por el investigador

Una vez culminadas las pruebas del comportamiento de una red LAN tradicional con servicios de video, audio, datos y tráfico de paquetes IoT-MQTT; se procedió a la implementación de un entorno con arquitectura de red IoT sobre una SDN destinada al tráfico de datos provenientes de sistemas de alarmas comunitarias como entorno de pruebas real.

Tanto el diseño, implementación y resultados obtenidos se presentan en los siguientes apartados.

4.2. Diseño de arquitectura red SDN con IoT

Una vez obtenidos los resultados de una arquitectura tradicional sin políticas de control de tráfico, se planteó la implementación de una red convergente de las mismas características y dedicada netamente a la seguridad ciudadana repotenciada por tecnologías de red emergentes.

En la figura 4.26 se observa la infraestructura de red de destinada al monitoreo de Alarmas comunitarias basadas en arquitecturas SDN e IoT, en base al marco teórico estudiado con anterioridad y con el afán de obtener resultados a los paradigmas generados en el estado del arte analizado, la arquitectura propuesta se compone de varios subsistemas.

Arquitectura de red para Alarmas comunitarias basadas en arquitecturas SDN e IoT

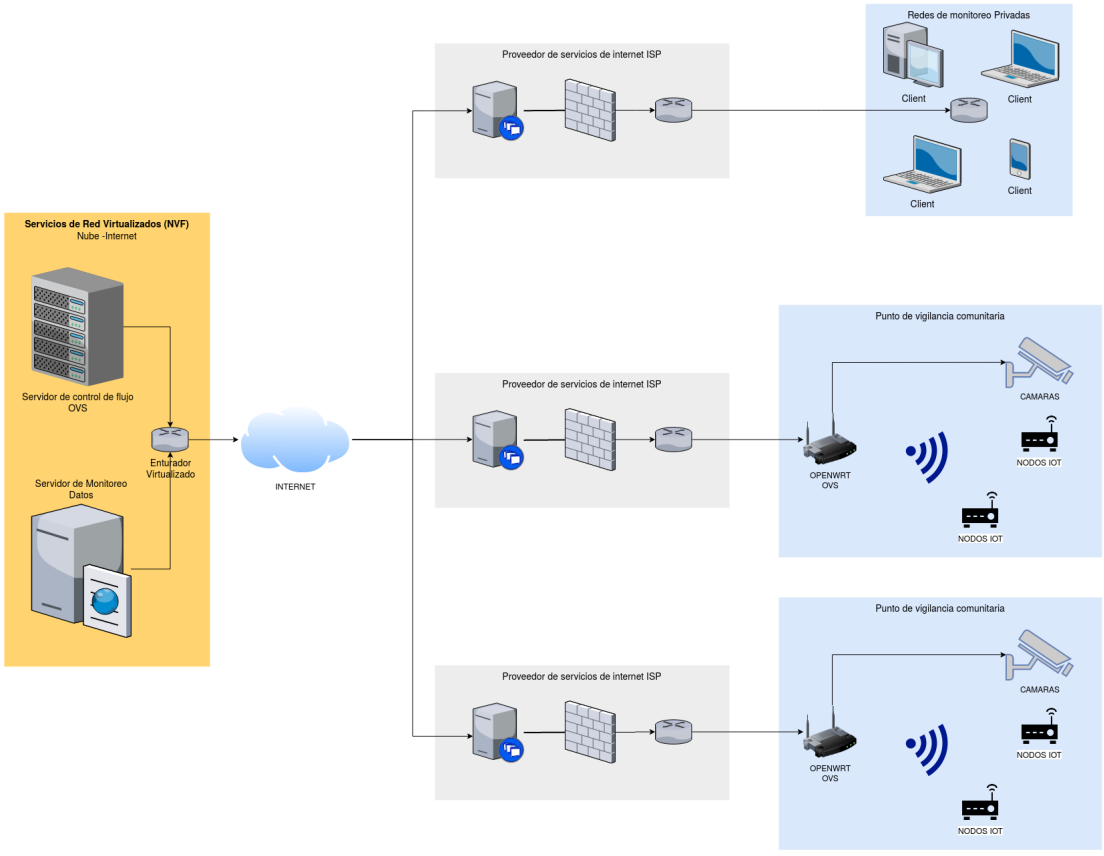


Figura 4.26: Diseño de arquitectura para Alarmas comunitarias basadas en SDN e IoT
Fuente: Desarrollado por el Investigador

El primero de ellos, es un segmento de red pública o privada destinado al monitoreo de los nodos IoT y cámaras de videovigilancia; segmento que a su vez, también puede contener nodos IoT o cámaras. El segundo subsistema es un segmento compuesto por cámaras de videovigilancia comunitaria y dispositivos RF para activación de alarmas de forma alámbrica o in alámbrica; para ambos casos, se planteo la sustitución de los dispositivos de red tradicional por dispositivos SDN implementados y configurados completamente por el investigador.

Es necesario preestablecer para fines investigativos, que ambos segmentos pueden encontrarse dentro de una misma red o distribuida a lo largo de una región pero para que sea posible la comunicación entre ambos, cada segmento de monitoreo o vigilancia requiere de acceso de a internet sin importar proveedor o tecnología.

Para la etapa de control de red y una vez lograda la separación del plano de control y el plano de datos a través de dispositivos SDN, la gestión centralizada de la red se logra a través de un controlador implementado utilizando NVF; es decir, cada dispositivo SDN que se vincula a la red de videovigilancia, requiere de políticas creadas en un controlador RYU implementado en un servidor virtualizado con acceso mundial.

En adición, se ha implementado un entorno completamente visual tanto para la gestión del controlador SDN, así como para el monitoreo de los dispositivos de accionamiento de alarmas RF reemplazados por nodos IoT debidamente georreferenciados en un mapa mundial; facilitando la gestión de tráfico de redes de gran magnitud, así como la generación de alertas visuales remotas, auditivas locales con el menor retardo posible y de tal manera que permita el rápido accionar de cuerpos del orden.

En los siguientes apartados se describe el proceso de implementación de cada uno de los componentes descritos en la figura 4.26.

4.2.1. Implementación de servidor primario en entorno de NVF

Como se estableció previamente, las funciones NVF representan la solución más óptima para el desarrollo y despliegue de redes de alto tráfico y crecimiento, permitiendo implementar servicios de red sin requerir un datacenter ni cualquier tipo de hardware para alojamiento, además de que los gastos de funcionamiento y mantenimiento son relativamente bajos; en consecuencia, las NVF han logrado garantizar un nivel de disponibilidad superior a los que se pueden alcanzar en datacenter físicos y en crecimiento que el proyecto puede requerir.

Para la investigación se utilizaron los servicios web de Amazon (AWS), debido a que proporcionan un stack completo de soluciones NVF para un sin número de soluciones de propósito general, AWS cuenta con soluciones para infraestructura, almacenamiento, bases de datos, inteligencia artificial e inclusive su propia plataforma de internet de las cosas; y su gestión se la desarrolla a través de una plataforma web de acceso mundial [34].

El proceso de creación de un servidor o instancia EC2, además de ser gratuita durante el primer año es relativamente sencillo. Basta con seguir los siguientes pasos:

1. Crear una cuenta de AWS a través del link https://portal.aws.amazon.com/billing/signup#/start, para lo cual se requiere un correo de uso personal y una tarjeta de crédito/débito internacional solo para validación.

2. Dirigirse la consola de administración de AWS y seleccionar el servicio EC2 deseado como en la figura 4.27

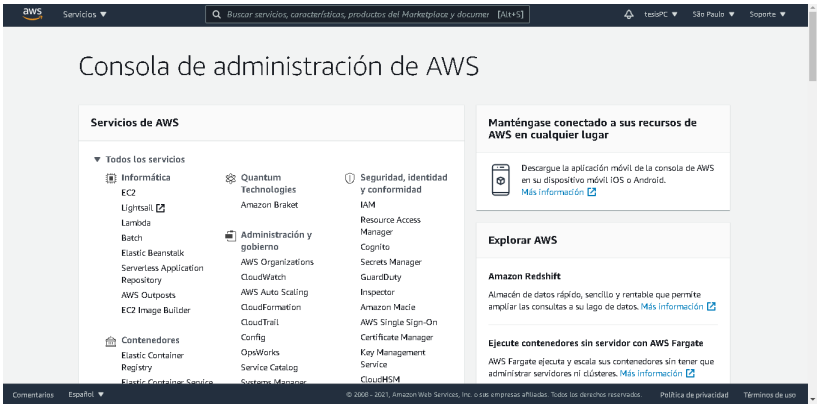


Figura 4.27: Consola de administración de AWS
Fuente: Desarrollado por el investigador

3. Seleccionar el tipo de servidor a implementar de la lista de posibles opciones listadas como en la figura 4.28

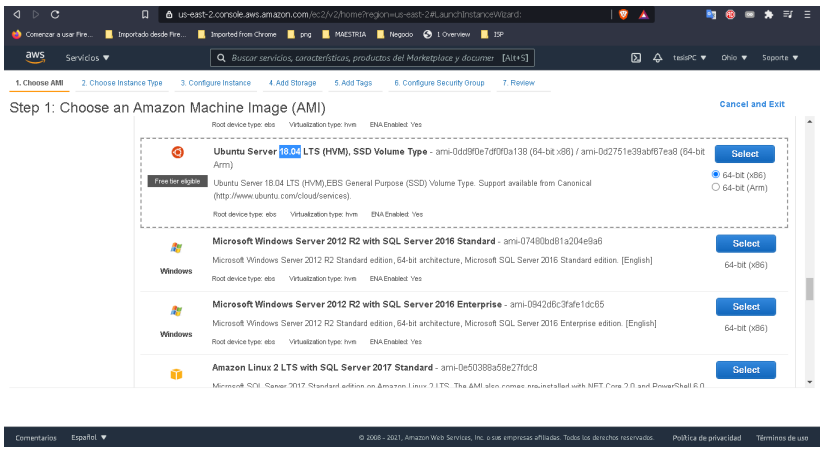


Figura 4.28: Listado de servidores virtualizados disponibles en AWS
Fuente: Desarrollado por el investigador

4. Seleccionar el tipo de instancia para el servidor seleccionado como se muestra en la figura 4.29, la instancia hace referencia al número de nucleos de la CPU, memoria RAM, capacidad de almacenamiento y capacidades de red que tendrá el servidor.

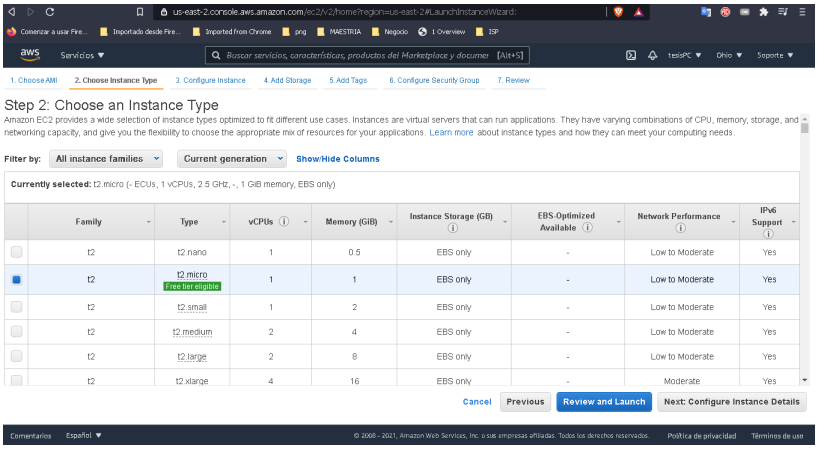


Figura 4.29: Características de servicios virtualizados disponibles en AWS
Fuente: Desarrollado por el investigador

5. Configurar el grupo de seguridad del servidor con el que se va a trabajar como en la figura 4.30. Esto controlará o permitirá a posterior generar políticas del firewall virtual que AWS otorga en conjunto con el servidor.

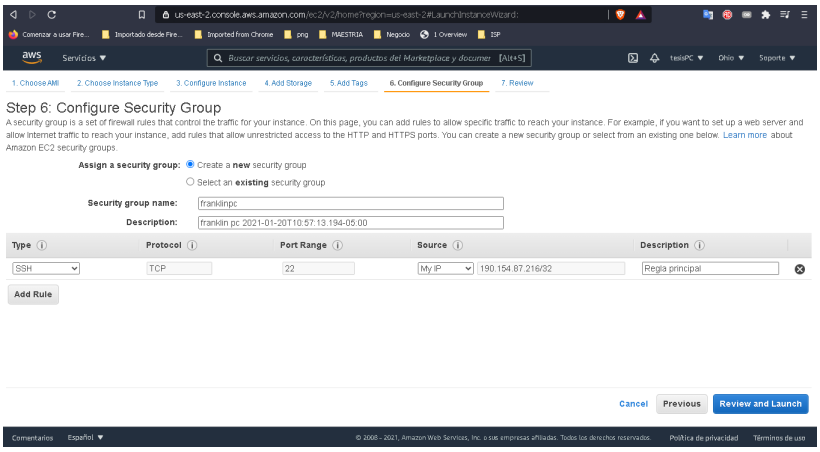


Figura 4.30: Configuración de firewall de servidor virtualizado en AWS
Fuente: Desarrollado por el investigador

6. Finalmente, en la pestaña de *REVIEW* se presiona el botón de *LAUNCH* para iniciar el nuevo servidor como el presentado en la figura 4.31.

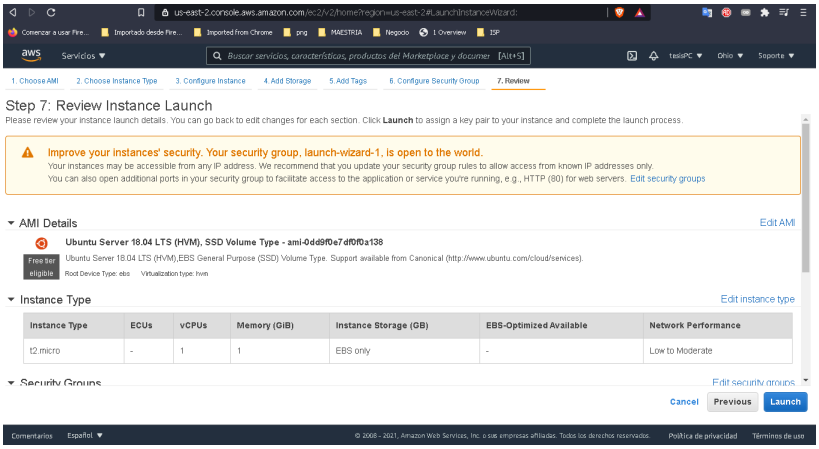


Figura 4.31: Listado de características seleccionadas para servidor virtualizado en AWS
Fuente: Desarrollado por el investigador

Una vez creado e iniciado el servidor de trabajo, es necesario que el usuario genere un par de llaves pública y privada con las cuales el usuario se autentica y accede de forma remota al nuevo servidor; para lo cual, una vez que la máquina haya sido creada con éxito, AWS despliega una ventana emergente para la creación de las llaves como en la figura 4.32.

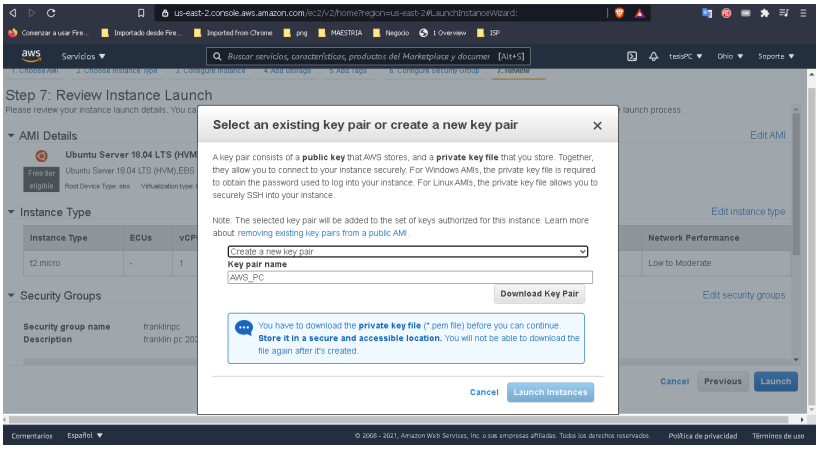


Figura 4.32: Generación de llaves SSH para acceso remoto a AWS
Fuente: Desarrollado por el investigador

Si todos los pasos fueron exitosos, en la sección de instancias de la plataforma de AWS aparecerá el servidor creado similar a los resultados mostrados en la figura 4.33.

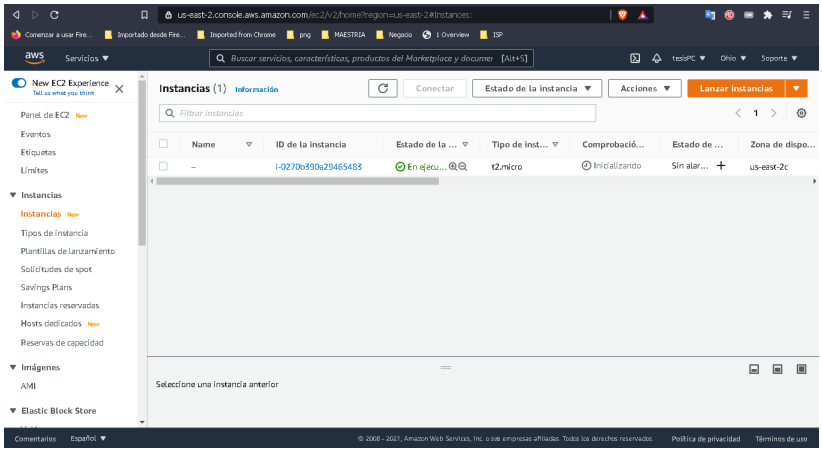


Figura 4.33: Listado de instancias creadas en AWS
Fuente: Desarrollado por el investigador

Previo a la culminación de este apartado, es necesario importar las llaves proporcionadas por AWS con extensión PEM a un gestor de conexiónes remotas como Putty, sin embargo, este último no soporta llaves de extensión PEM; para solucionar dicho conflicto, la herramienta de libre acceso puttygen permite convertir una llave PEM en un PPK que si es aceptada por Putty.

Finalmente, teniendo el servidor iniciado en un entorno NVF es necesario acceder al mismo mediante el gestor de comunicaciones remotas y actualizar el sistema operativo seleccionado y sus dependencias a las mas actuales con el comando mostrado en la figura 4.34.

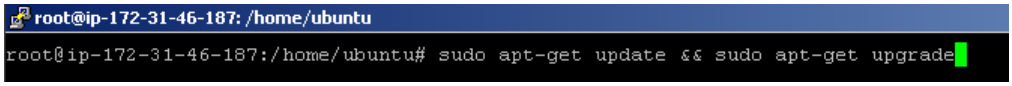


Figura 4.34: Primera actualización de repositorios en instancia virtualizada.
Fuente: Desarrollado por el investigador

Configuración de firewall virtual de AWS

Al ser una instancia completamente nueva y como en todos los entornos de red sean fisicos o virtuales, los servidores y redes NVF otorgados por AWS, requieren de configuración manual del firewall interno para permitir las comunicaciones de entrada y salida. Para lo cual, es necesario dirigirse al apartado de Red y Seguridad del entorno de AWS y seleccionar *security groups*, en el apartado como el de la figura 4.35, se debe seleccionar el grupo de seguridad creado al generar la instancia y modificar las reglas de entrada y salida del firewall de acuerdo a las necesidades de la red.

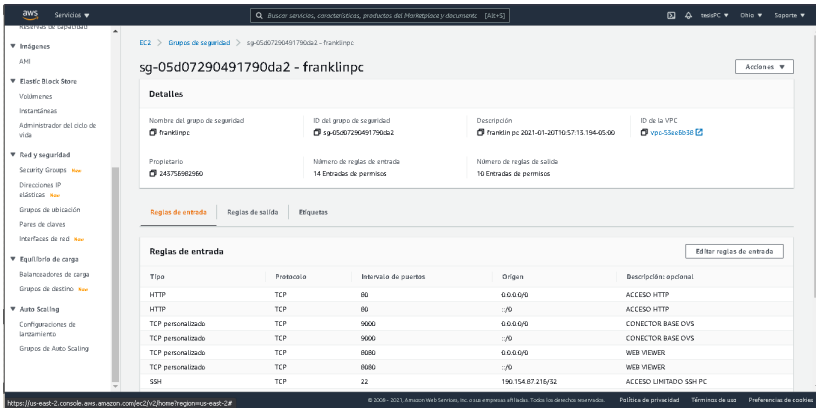


Figura 4.35: Configuración de red y seguridad de la instancia en AWS
Fuente: Desarrollado por el investigador

Para el presente estudio fue necesario habilitar las reglas de entrada descritas en la tabla 4.4 y las reglas de salida de la tabla 4.5.

Tabla 4.4: Reglas de entrada para la solución propuesta

Tipo	Protocolo	Puerto	Origen	Descripción
HTTP	TCP	80	Anywhere	Acceso Http
TCP personalizado	TCP	9000	Anywhere	Acceso a servicio OVS-DB para control SDN
TCP personalizado	TCP	8080	Anywhere	Acceso a servicio WEBVIEWER de controlador SDN
SSH	TCP	22	IP pública específica	Acceso al servidor de forma segura desde la IP pública del admisitrador de red
TCP personalizado	TCP	1884	Anywhere	Acceso a servicio MQTT-IoT por websockets
TCP personalizado	TCP	1883	Anywhere	Acceso a servicio MQTT-IoT por nodos IoT

Fuente: Desarrollado por el investigador

Tabla 4.5: Reglas de salida para la solución propuesta

Tipo	Protocolo	Puerto	Origen	Descripción
HTTP	TCP	80	Anywhere	Acceso Http
TCP personalizado	TCP	9000	Anywhere	Acceso a servicio OVS-DB para control SDN
TCP personalizado	TCP	6633	Anywhere	Acceso a servicios del controlador SDN RYU
HTTPS	TCP	443	Anywhere	Acceso Https
Todos los puertos TCP (opcional)	TCP	0-65535	Anywhere	Acceso desde el servidor hacia la red (reemplaza todas las políticas anteriores)

Fuente: Desarrollado por el investigador

Las políticas del firewall de entrada y salida para el servidor NVF solucionarían en su totalidad cualquier

tipo de fallos en las que el servidor no permita la instalación de paquetes, actualización de los mismos o comunicación entre dispositivos SDN e IoT.

4.2.2. Instalación y configuración de controlador RYU para Firewall y QoS

El marco de control SDN seleccionado para la investigación fue RYU en base a las características evaluadas en el capitulo II del presente. Es recomendable que para obtener mayor control y acceso a todas las funcionalidades que otorga este marco, se instale RYU sobre el servidor a partir del código fuente del mismo, tal y como se describe en la página oficial de los desarrolladores [35]; si el proceso de instalación ha sido desarrollado con éxito, bastaría con ejecutar el comando ryu-manager en una consola del servidor principal para comprobar su funcionamiento y obtener el resultado igual o similar al de la figura 4.36.

Figura 4.36: Comprobación del funcionamiento del marco de control RYU
Fuente: Desarrollado por el investigador

Dentro de las dependencias del marco de control de RYU, se encuentran contenidos varios aplicativos de nivel de red que permiten controlar y gestionar el tráfico de red de manera directa e inclusive contienen el código fuente para que de ser necesario se puedan modificar los aplicativos con el afán de crear servicios personalizados.

En la investigación desarrollada se han utilizado tres marcos primarios proporcionados por los desarrolladores del proyecto y que son compatibles con el protocolo OpenFlow desde las versiones 1.0 a 1.5. Los mismos que se describen a continuación:

- ofctl_rest: aplicativo diseñado para obtener y actualizar las estadísticas de los dispositivos de la red SDN[36], en la investigación se utilizó este aplicativo con el objetivo de consultarle al controlador el listado completo de los dispositivos vinculados a la red.

  Ejemplo de uso: `ryu-manager ryu.app.ofctl_rest aplicativo_de_red_adicional`

- ofctl_firewall: aplicativo diseñado para habilitar y controlar las políticas de firewall de los dispositivos de la red SDN [37], en la investigación presentada, este aplicativo permitió crear políticas de firewall para controlar el tráfico sobre los dispositivos en los segmentos de red de monitoreo y videovigilancia.

  Ejemplo de uso: `ryu-manager ryu.app.rest_firewall`

- rest_qos: aplicativo diseñado para implementar funciones básica o complejas de QoS en los dispositivos de la red SDN [38], este aplicativo posibilitó la habilitación de políticas de transferencia de datos y la priorización del tráfico dependiendo del tipo de datos video, voz, datos o IoT; inclusive está en la capacidad de reservar ancho de banda constante para uno o varios protocolos de comunicación si así lo requieren los administradores de red.

    - Previo a la utilización de este aplicativo, es necesario modificar el marco de control de la arquitectura SDN con el que se ejecuta un switch bajo el protocolo openflow 1.3 de tal manera que se permita registrar las entradas en las tablas de flujo del controlador SDN, procedimiento que se puede lograr con el siguiente comando:

      ```
      sed '/OFPFlowMod(/,/)/s/)/, table_id=1)/' ryu/ryu/app/simple_switch_13.py >
                          ryu/ryu/app/qos_simple_switch_13.py
      ```

    - Una vez realizado el cambio sobre el aplicativo de dispositivo lógico, es necesario reconfigurar y reinstalar el marco de control RYU para registrar internamente los cambios desarrollados con el siguiente comando:

      ```
      cd ryu/; python ./setup.py install
      ```

    - Si los cambios fueron realizados con éxito, se habrá habilitado un nuevo marco de control de QoS para dispositivos lógicos de capa 2 sobre RYU.

      Ejemplo de uso: `ryu-manager ryu.app.rest_qos ryu.app.qos_simple_switch_13 ryu.app.rest_conf_switch`

Una vez comprobada la funcionalidad de cada uno de los marcos y que el controlador haya sido instalado desde el código fuente, el aplicativo ryu-manager está en la capacidad de ejecutar varios marcos a la vez sobre un único controlador SDN para otorgar el control total de la red al administrador. El comando de ejecución para RYU compuesto de los aplicativos anteriormente descritos es:

```
ryu-manager ryu.app.ofctl_rest ryu.app.rest_qos ryu.app.qos_simple_switch_13
                ryu.app.rest_conf_switch ryu.app.rest_firewall
```

Con la finalidad de automatizar la ejecución del controlador SDN, es recomendable crear un archivo de ejecución que contenga el comando completo para evitar posibles fallos de sintaxis, en la figura 4.37 es posible observar la respuesta del servidor una vez que RYU ejecuta el controlador con varios aplicativos de red unificados.

Figura 4.37: Prueba de funcionamiento de controlador
Fuente: Desarrollado por el investigador

4.2.3. Implementación de dispositivos SDN OVS RPI

Los dispositivos controlados SDN son la pieza fundamental para la optimización de la red propuesta y debido a que los datos que transitan sobre la red provienen de dispositivos alámbricos e inalámbricos, se propuso un dispositivo SDN con una arquitectura de dichas características.

A nivel de sofware se seleccionó al proyecto OpenWRT debido a que hasta la actualidad no tiene oponente en cuanto a: estabilidad trabajando como punto de acceso inalámbrico, simplicidad de manejo, compatibilidad con gran cantidad de dispositivos de red comerciales, soporta a Open vSwitch para SDN y sobre todo de fuente abierta.

En cuanto a hardware, la mayoría de dispositivos que soportan OpenWRT son dispositivos comerciales de alto costo, procesamiento reducido y de difícil acceso en determinadas partes del mundo; razón por la cual y debido a que el proyecto Raspberry a inundado prácticamente a todo el globo con sus dispositivos de bajo costo, alta capacidad de procesamiento, distribuido bajo licencias de libre modificación y con amplio soporte técnico brindado por un colectivo de desarrolladores a nivel mundial.

El concordancia con lo propuesto previamente, los dispositivos seleccionados como de red fueron placas Raspberry pi 3 y pi 4 modelo B debido a que las funcionalidades y características de los miniordenadores aportarían al dispositivo velocidad de procesamiento de 1.2 a 1.5 Ghz, 1Gb a 4Gb de memoria RAM y hasta 4 puertos USB que permitieron convertirlos en interfaces ethernet o wifi de hasta 1Gbps, en figura 4.38 se describe gráficamente la arquitectura propuesta para ambos modelos, obteniendo los mismos resultados.

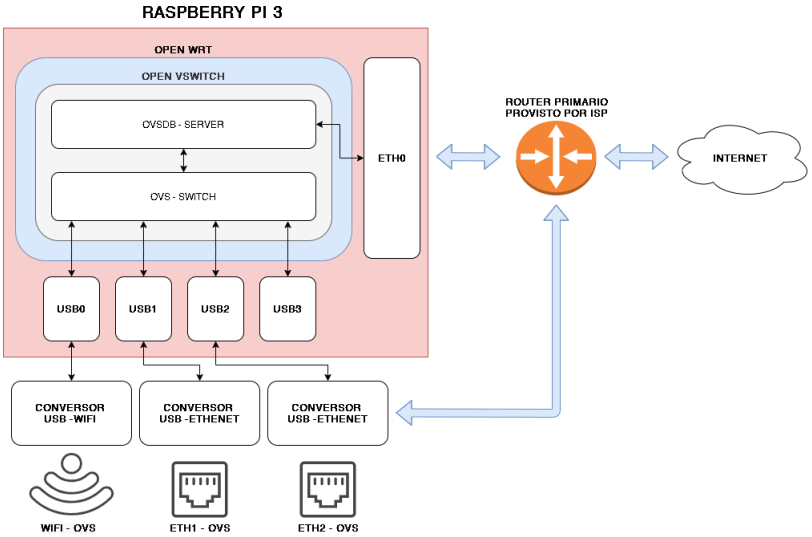


Figura 4.38: Arquitectura de dispositivo SDN OVS RPI
Fuente: Desarrollado por el investigador

El software utilizado para proporcionar las funcionalidades requeridas al hardware descrito fue: OpenWRT para proporcionar al dispositivo funcionalidades de comunicación inalámbrica, además de un entorno gráfico amigable con el usuario y Open-Vswitch para habilitar la creación de un conmutador virtual dentro del software OpenWRT, a razón de que este último permite enlazar el dispositivo de red al controlador SDN NVF y con ello el control del flujo de datos de los segmentos alámbricos e inalámbricos.

Para la instalación de OpenWRT en el hardware, unicamente se requiere descargar la versión más actual del software de la página oficial [39] y cargarlo sobre la tarjeta SD del dispositivo, posteriormente y para habilitar las funcionalidades de OVS es necesario seguir los siguientes pasos:

- Actualizar los repositorios de OpenWRT utilizando el comando:

```
opkg update
```

- Instalar el paquete de OVS utilizando el comando:

```
opkg install openvswitch
```

- Instalar los controladores de los dispositivos USB adicionales USB-ETHERNET y USB-WIFI dependiendo del tipo de hardware utilizado y los drivers listados en [40]:

```
opkg install controladorUSB
```

- Configurar el dispositivo de USB-WIFI con el comando:

```
wifi config
```

- Modificar el archivo de configuraciones que se genera en el paso anterior y que vincule el dispositivo USB-WIFI con una red wireless presentado en el Anexo 1.

```
nano /etc/config/network
```

- Configurar el firewall para habilitar el tráfico de la nueva red wireless de acuerdo al Anexo 2.

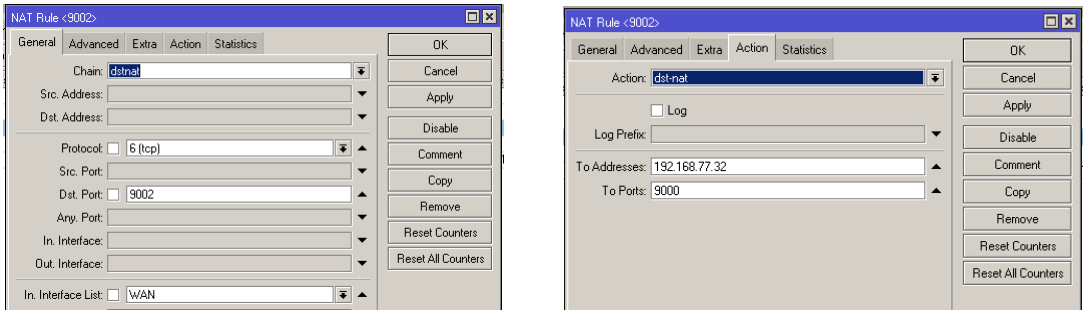
```
nano /etc/config/firewall
```

- Habilitar la nueva interfaz inalámbrica con el comando

```
wifi up
```

- Crear un archivo de configuración para la creación automática de OvSDB-Server y OvS-Switch y su comunicación directa con el controlador SDN en NVF como se puede verificar en el Anexo 3.

Con el archivo de configuración creado para comunicarse al controlador, previo a la prueba de comunicaciones; es necesario habilitar y redireccionar el puerto asignado a OvSDB-Server para permitir el intercambio de información desde el exterior de la red como se presenta en las figuras 4.39 a y b.



a) Creación de regla de redireccionamiento origen desde puerto exterior perteneciente a IP pública

b) Creación de regla de redireccionamiento destino hacia puerto de IP privada de OVS

Figura 4.39: Redireccionamiento de puerto para intercambio de información entre controlador y OVS
Fuente: Desarrollado por el investigador

Una vez configurado el acceso hacia el puerto del OvDSB-Server de la red interna desde la red externa, la prueba de comunicaciones debe permitir el intercambio de información entre el controlador y el conmutador como se muestra en la figura 4.40.

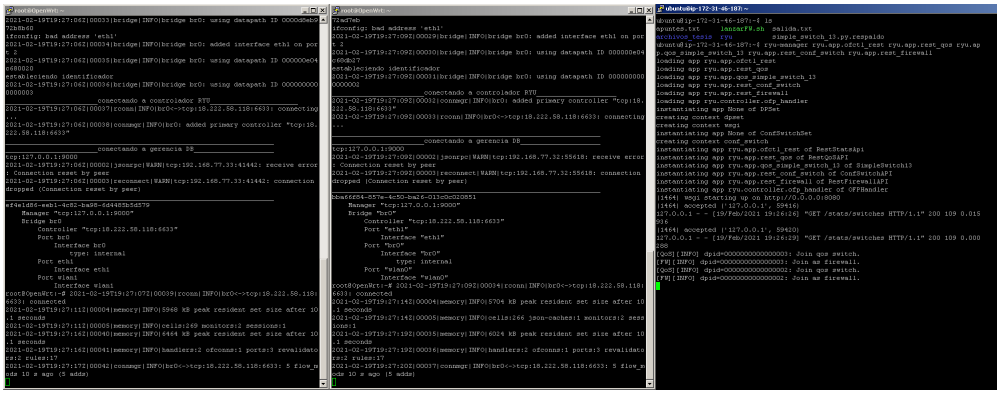


Figura 4.40: Comprobación de comunicaciones bidireccional entre OVS y controlador SDN
Fuente: Desarrollado por el investigador

4.2.4. Implementación de entorno de control web para SDN

Como se describió en el apartado del capítulo II, una de las ventajas principales de RYU es la política open-source bajo el que se ha edificado el proyecto, además del amplio soporte y documentación sobre el marco de control para SDN. En síntesis, RYU es completamente modificable y adaptable de acuerdo a las necesidades del usuario y se encuentra escrito en lenguaje de alto nivel.

Parte del proyecto investigativo y con el afán de proporcionar un entorno amigable a administradores de red y con ello reducir el tiempo destinado a la creación de políticas de control para el marco SDN, se implementó un entorno web presentado en la figura 4.41 basado en los aplicativos proporcionados por la comunidad de desarrollo del proyecto RYU utilizando la información publicada en el libro *RYU SDN Framework* [37].

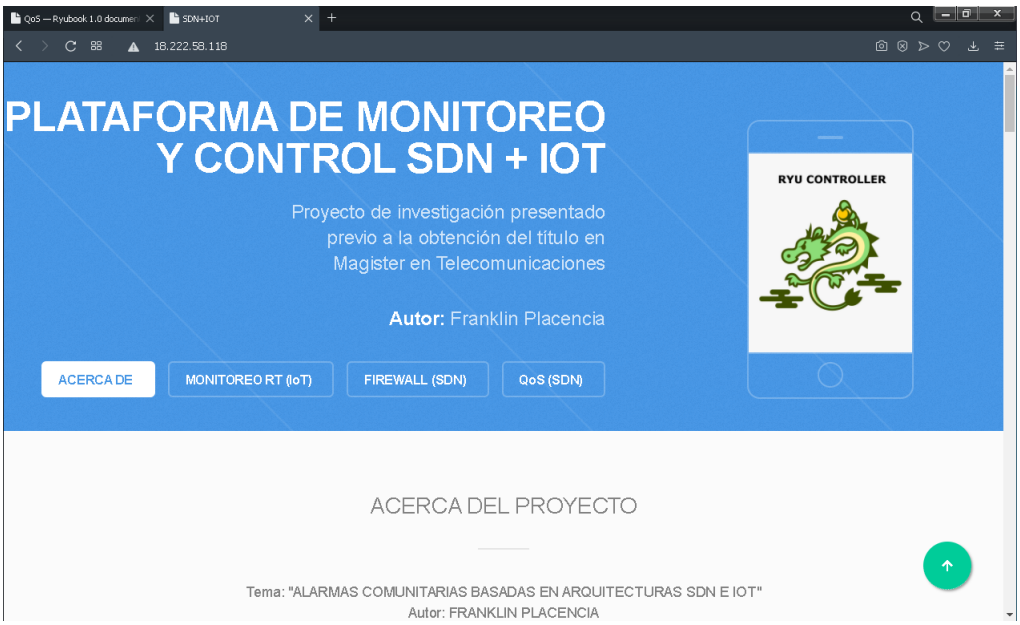


Figura 4.41: Entorno de control WEB para controlador SDN e IoT
Fuente: Desarrollado por el investigador

Se requirieron al menos dos páginas web dedicadas netamente a controlar por separado el aplicativo de firewall y el de QoS de RYU, los entornos implementados y basados en lenguaje PHP proporcionan al administrador de red el envío de comandos directamente al controlador SDN sin la necesidad de escribir o estructurar código de consola; para lo cual, dentro de los requerimientos para el óptimo funcionamiento del entorno fueron necesarios las siguientes dependencias:

- Servidor LAMP como se describe en [41] para proporcionar el servicio de alojamiento web así como las funcionalidades de acceso a base de datos y soporte de lenguaje PHP.
- Paquete de software cURL [42] junto a sus dependencias y librerías adicionales para ejecutar comandos de transmisión de datos a través de línea de comandos y PHP.
- Adicionalmente pero no obligatorio: Software editor de texto y software de transferencia de archivos de preferencia del programador.

Aplicativo web para Firewall SDN

Para el entorno de control del firewall se adaptaron los comandos proporcionados en el REST API List de RYU [37] divididos en tres subsecciones como se muestra en la figura 4.42.

Figura 4.42: Entorno web de FIREWALL SDN
Fuente: Desarrollado por el investigador

La pestaña de «Dispositivos» presenta visualmente los dispositivos conectados al controlador y el estado del firewall interno que se habilita una vez el dispositivo SDN se vincula al controlador, la estructura de uso del mismo descrita en *Acquiring Enable/Disable State of All Switches* se logra a través de una solicitud HTTP GET hacia la URL del servidor que contiene el controlador principal como en el siguiente ejemplo:

```
http:ip_servidor:puerto/firewall/module/status
```

Su respuesta HTTP genera una variable de texto con estructura de datos de tipo JSON, que posteriormente se ha des-concatenado para presentar en una tabla los valores separados por identificador de conmutador, el identificador como el presentado en la figura 4.42 es una representación de tipo cadena de texto con estructura única e irrepetible a través del cual controlador SDN reconoce.

La pestaña de «Configurar» está estructurada para enviar políticas a uno o todos los conmutadores que se crean dentro de los dispositivos SDN al conectarse al controlador, en la figura 4.43 es posible observar los componentes que requiere la petición HTTP POST tomada de *Adding Rules* para ser enviada en forma de cadena JSON a la URL del controlador SDN.

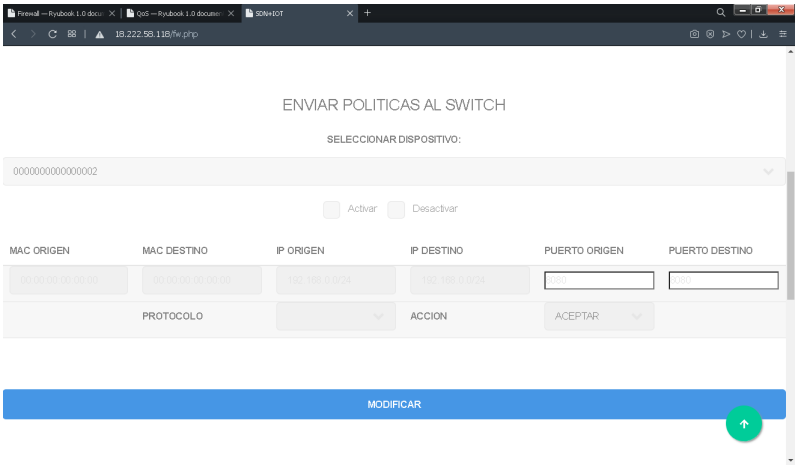


Figura 4.43: Entorno web para agregar reglas al firewall SDN
Fuente: Desarrollado por el investigador

La estructura de la URL para adicionar políticas al conmutador utilizada fue:

```
http:ip_servidor:puerto/firewall/rules/{switch_id,all}/{vlan,all}
```

Adicionalmente, el entorno de firewall previo a la carga del entorno visual, realiza peticiones al controlador para obtener el listado de los dispositivos disponibles a configurar y habilita las funciones de activación del firewall tomado de *Changing Enable/Disable State of All Switches* los cuales requieren peticiones HTTP PUT con la estructura de URL:

```
http:ip_servidor:puerto/firewall/{--opcion=enable,disable}/{switch_id,all}
```

La pestaña de «Verificar» como la presentada en la figura 4.44 contiene el código que le permite al administrador de red listar las políticas que contienen los conmutadores, comando tomado de *Acquiring All Rules* y cuya solicitud que se desarrolla utilizando una petición HTTP GET hacia la URL:

```
http:ip_servidor:puerto/firewall/rules/{switch_id,all}/{vlan,all}
```

y finalmente, una vez que el administrador conozca la tabla de reglas de firewall habilitadas; se posibilita la eliminación de políticas no deseadas utilizando los comandos de *Deleting Rules* a través de una petición HTTP DELETE a la misma URL.

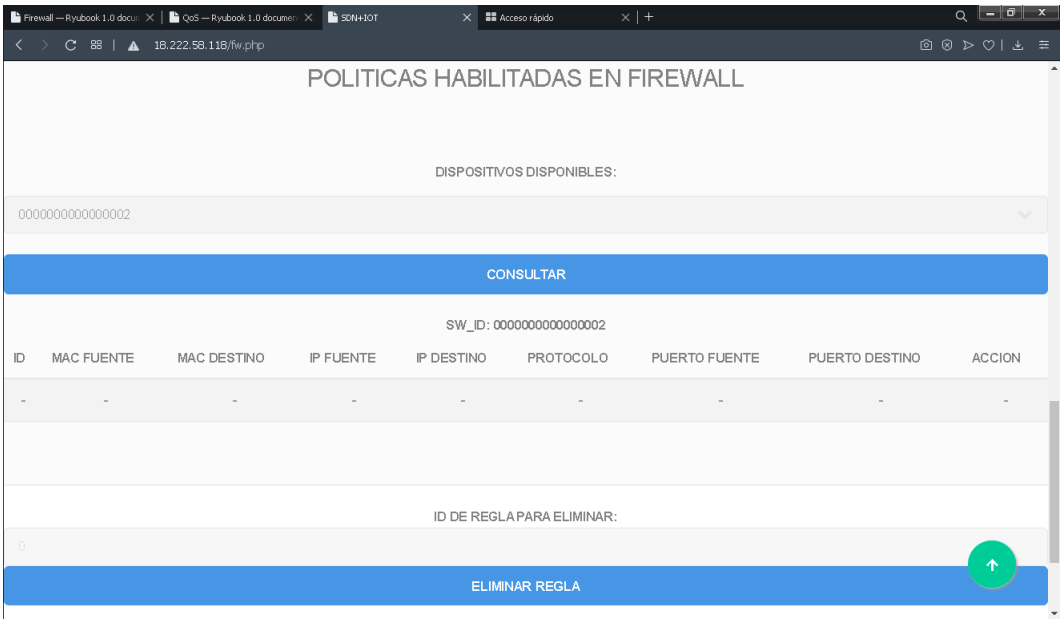


Figura 4.44: Entorno web para verificar o eliminar reglas del firewall SDN
Fuente: Desarrollado por el investigador

Internamente, la información que el administrador modifique en el entorno gráfico tanto para la habilitación como para el envío de políticas hacia los conmutadores fue capturada por la misma página PHP, concatenada en un array de texto, transformarla en una sola cadena de texto con estructura JSON y enviadas al marco provisto por aplicativo de gestión del controlador.

Algoritmo 1 Conversión de texto a cadena JSON para configurar Firewall SDN

```
$camposFw = array();
$camposFw["opcion_fw_data"] = strval($valor_a_enviar);
json_encode = $camposFw;
```

El contenido completo del código html y php descrito para el control del Firewall se encuentra en el Anexo 4.

Aplicativo web para QoS SDN

Para la web del aplicativo de control de QoS se diseñó un entorno compuesto por seis subapartados con el afán de otorgar todas las funcionalidades del controlador SDN provistos a través de un único entorno visual e intuitivo como se muestra en la figura 4.45, los comandos fueron adaptados basándose en la pila provista por el marco REST API List de RYU [38].

Figura 4.45: Entorno web para control de políticas de QoS SDN
Fuente: Desarrollado por el investigador

El apartado de «Dispositivos» que aparece en la figura 4.45 se proporciona la lista de dispositivos conectados al controlador SDN divididos por posición de conmutador en la tabla del controlador e identificador único. En el apartado de «Configurar» presentado en la figura 4.46 se diseñó un mecanismo para habilitar la conexión bidireccional entre el controlador y el dispositivo SDN; es decir, el controlador SDN necesita conocer la dirección IP y el puerto del OvSDB-Server al que se enviarán las políticas de QoS que comanda al comutador SDN.

La estructura de la petición HTTP PUT que requiere ser enviada a la URL:

```
http://ip_servidor:puerto/v1.0/conf/switches/{switch_id}/ovsdb_addr
```

fue una cadena de texto tipo JSON que contenía la siguiente información:

```
"tcp:ip_del_switch:puerto_OVSDB_manager"
```

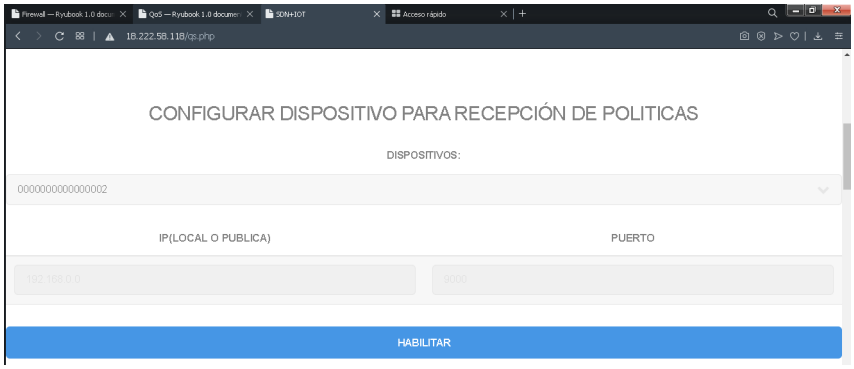


Figura 4.46: Entorno web para configuración de comunicaciones bidireccionales para QoS
Fuente: Desarrollado por el investigador

Una vez que el controlador SDN conoce la IP y Puerto del conmutador al que debe programar, los apartados «Queue Rules» y «QoS Rules» presentados en las figuras 4.47 y 4.48 fueron diseñados para crear políticas de Calidad de servicio que afecten al conmutador que el administrador de red desee y se mantengan de forma persistente dentro del OvSDB-Server del dispositivo SDN.

Figura 4.47: Entorno WEB para configurar colas de QoS
Fuente: Desarrollado por el investigador

Para el apartado de colas o *queues* en inglés fue necesario adaptar peticiones de *Set queue* mediante HTTP POST hacia la URL:

```
http://ip_servidor:puerto/qos/queue/{switch_id,all}
```

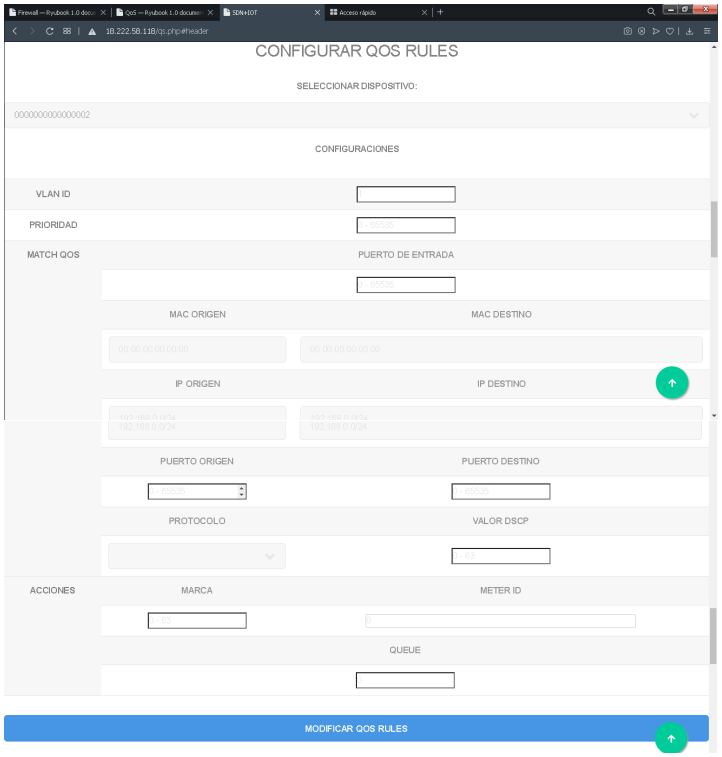


Figura 4.48: Entorno WEB para configurar reglas de QoS
Fuente: Desarrollado por el investigador

Y para el apartado de reglas descrito en *Set a QoS rule* fue necesario adaptar el mismo tipo de peticiones para la URL:

```
http://ip_servidor:puerto/qos/rules/{switch_id,all}/{vlan,all}
```

En ambos casos, la información que el administrador modificó fueron capturadas por el mismo entorno web HTTP GET y codificadas a texto con estructuras JSON para ser enviadas a las URLs. El código completo del presente apartado se lo encuentra en el Anexo 5.

4.2.5. Implementación de broker MQTT

Los nodos RF reemplazados por dispositivos IoT para la activación de alarmas visuales y auditivas, requieren de un broker MQTT encargado de gestionar los mensajes que se publican por clientes IoT suscritos a un hilo determinado y difundirlos a sus suscriptores.

El broker utilizado para la investigación fue Mosquitto por las características descritas en el capítulo II, y su proceso de implementación sobre un servidor Ubuntu actualizado se resume a la ejecución de los comandos:

```
sudo apt-get install mosquitto
```

```
sudo apt-get install mosquitto-clients
```

El primero de los comandos, instala sobre el servidor las dependencias necesarias para la ejecución del broker Mosquitto. El segundo, instala las dependencias para que clientes IoT obtengan acceso por consola o a través de cualquier gestor que soporte el protocolo MQTT, una vez instalados ambos paquetes de software para los dispositivos IoT, es necesario tener en cuenta que los dispositivos requieren de acceso al mensajes de entrada y salida que difunde el broker.

Debido a que el servidor que contiene el broker se encuentra alojado en los servicios NVF de AWS, se requiere la habilitación de los puertos del firewall descritos en el apartado de configuración de AWS y adicionalmente la creación de un archivo de configuración para habilitar la comunicación a través de dispositivos IoT y el servicio de websockets que permite el envío y recepción de mensajes a través de servicios web; para la creación de este último se ejecuta las siguientes líneas de código:

```
sudo nano /etc/mosquitto/conf.d/websockets.conf
```

Agregar las siguientes líneas de código con el afán de asignar puertos específicos para cada servicio:

Algoritmo 2 Contenido del archivo de configuración para habilitar puertos

```
listener 1883
listener 1884
protocol websockets
```

Una vez culminada la edición, se guardan los cambios sobre el archivo y se reinicia el servicio de mosquitto; este último paso mantiene habilitada la comunicación inclusive si se reiniciase por completo el servidor y permite la creación de hilos de publicadores o suscriptores.

4.2.6. Diseño de nodos de alarma IoT

Con el objetivo de reemplazar los nodos convencionales de RF que comúnmente se utilizan en las alarmas comunitarias, se diseñaron nuevos nodos respetando la topología de red pero reemplazando la unidad de procesamiento central por un dispositivo microcontrolado como el representado en la figura 4.49.

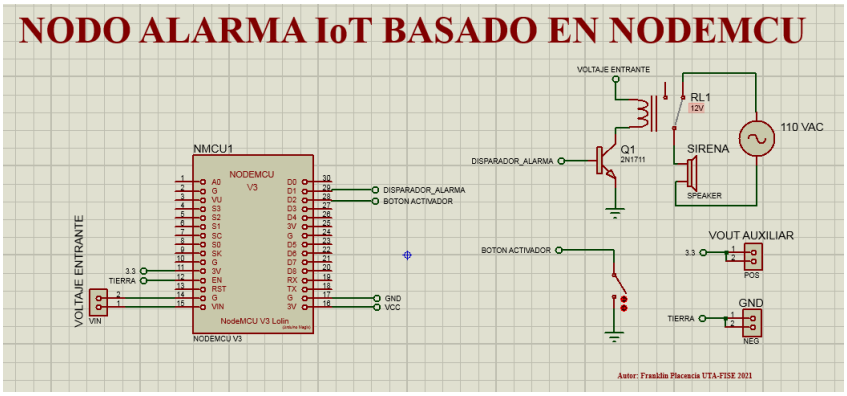


Figura 4.49: Diagrama esquemático de nodo alarma IoT
Fuente: Desarrollado por el investigador

El dispositivo IoT propuesto en la figura 4.49 esta compuesto por:

- Una unidad de microprocesamiento (MCU por sus siglas en inglés), alimentado por una fuente regulada de entre 5 y 12 Vdc.
- Un módulo de relé previamente optoisolado que separa una etapa de potencia de hasta 120 Vac de la etapa controlada por la MCU con señales de entre 3 y 5 Vdc.
- Una sirena de alarma, activada por corriente alterna conectada directamente al contacto normalmente abierto del módulo de relé.
- Adicional se ha provisto de pines de conexión para una entrada de activación física de tipo interruptor.

- Y finalmente, se han incluido pines de conexión para puntos de entrada o salida lógica para circunstancias en las que se requiera expandir las funcionalidades del dispositivo.

En cuanto a los algoritmos de control de dispositivo microcontrolado NMCU1 de la figura 4.49 se implementó un código simple para reducir al mínimo el uso del procesador y configurarlo como suscriptor del hilo MQTT de «pánico», en el algoritmo 3 se presentan los componentes más importantes del mismo y el código completo de programación del dispositivo se encuentra en el anexo 6.

Algoritmo 3 Código de programación de dispositivo microcontrolado como Nodo Alarma IoT

```
// Variables de conexión
#define WLAN_SSID "SSD_ID_RED_INALAMBRICA_SDN"
#define WLAN_PASS "CONTRASEÑA"
#define AIO_SERVER "IP_PUBLICA_DEL_BROKER_IoT"
#define AIO_SERVERPORT 1883
#define AIO_USERNAME "Usuario_de_acceso_a_broker"
#define AIO_KEY "Contraseña_de_acceso_a_broker"

//creación de identificador único
uint16_t identificador = 2;

//Objeto MQTT para conexión
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME,
AIO_KEY);

//Objeto de conexión como publicitante sobre hilo MQTT
Adafruit_MQTT_Publish pánico = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME
"/pánico");

/*******************bucle de monitoreo y activación****************/
Adafruit_MQTT_Subscribe *subscription;
while ((subscription = mqtt.readSubscription(1000)))
{
  if (subscription == &onoffbutton)
  {
    cadena = (char *)onoffbutton.lastread;
    if(cadena== "on")
    {
      digitalWrite(SIRENA, LOW);
    }
    else if(cadena == "off") //logica inversa Activacion en Cero lógico
    {
      digitalWrite(SIRENA, HIGH);
    }
  }
}
if (!pánico.publish(x))
{
  Serial.println(F("Failed"));
}
else
{
  Serial.println(F("OK!"));
}
```

4.2.7. Interfaz de monitoreo web para nodos IoT

En esta sección se describen los puntos más importantes para la creación del entorno gráfico de monitoreo de los nodos IoT presentado en la figura 4.50, encargados de activar alarmas tanto auditivas como visuales, para lo cual se diseño un entorno que permita al usuario monitorear en tiempo real todos los dispositivos conectados al broker y las alteraciones que generan los mismo sobre un mapa mundial.

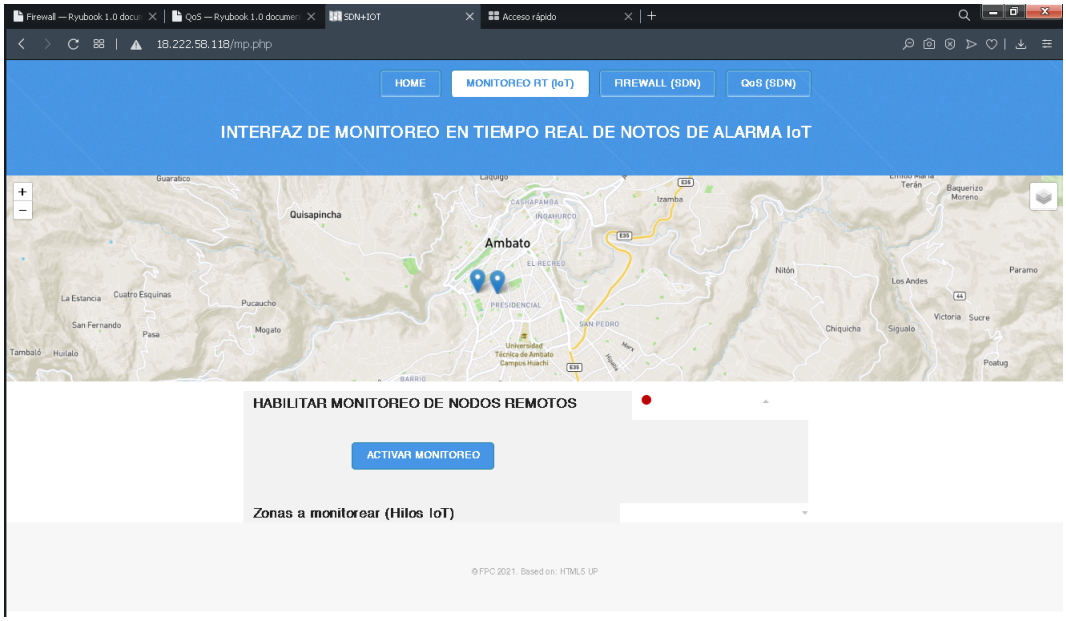


Figura 4.50: Entorno web para monitoreo en tiempo real de nodos de alarma comunitaria IoT
Fuente: Desarrollado por el investigador

Para el monitoreo de la variables o hilos que los suscriptores modifican dependiendo de los estados de los disparadores de las alarmas se tomaron como herramienta base de software a la solución web MQTT de fuente abierta que ofrece HiveMQ [43] a través de websockets y para el entorno del mapa mundial se adaptó la herramienta de acceso libre Leaflet [44]; ambas soluciones de software fueron reprogramadas y adaptadas para generar sinergia y posibilitar el despliegue de alertas visuales sobre el mapa cuando exista una variación sobre el hilo de «pánico» del broker MQTT como se puede observar en la 4.51.

Figura 4.51: Sección del entorno web para conexión al broker MQTT
Fuente: Desarrollado por el investigador

El diagrama de flujo del funcionamiento de entorno de monitoreo se describe en la figura 4.52 y el código completo del entorno visual se puede visualizar en el Anexo 7.

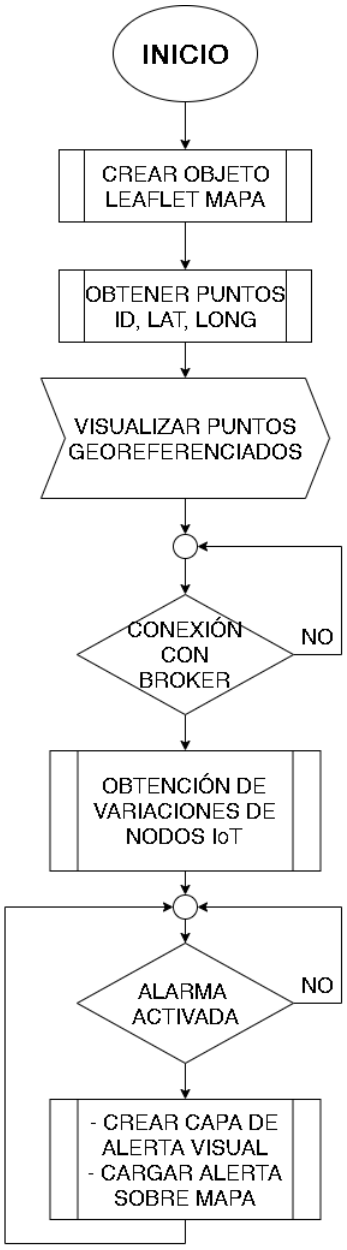


Figura 4.52: Diagrama de flujo del algoritmo de generación de alertas de nodos IoT
Fuente: Desarrollado por el investigador

Leaflet requiere crear una capa con marcadores específicos de latitud y longitud específica por lo que con la ayuda de PHP se creó un algoritmo para la generación N objetos a través de un ciclo while destinado a poblar el mapa mundial con todos los nodos IoT existentes. La sección de código para la creación de N nodos de alarma IoT se describe en el algoritmo 4.

Algoritmo 4 Creación de N nodos IoT sobre mapa mundial

```
//Consulta sobre la base de datos los nodos IoT habilitados
<?php $j=0; while($row = mysqli_fetch_array($result)) {?>

//Verifica los registros existentes previo a la creación de marcadores
<?php if($j+1 != $registros) {?>

  //Crea una instancia en el mapa por cada nodo georeferenciado
  L.marker([<?php echo $row["latitud"]; ?>, <?php echo $row["longitud"];
?>]).bindPopup('<?php echo $row["referencia"]; ?>').addTo(nodos),

<?php }else{?>

  //Crea una instancia en el mapa por cada nodo georeferenciado
  L.marker([<?php echo $row["latitud"]; ?>, <?php echo $row["longitud"];
?>]).bindPopup('<?php echo $row["referencia"]; ?>').addTo(nodos);

<?php } $j = $j+1; }?>
```

El resultado del algoritmo 4 genera la capa con puntos numerados mostrados en la figura 4.53 y georreferenciados sobre el que el aplicativo de MQTT realiza los cambios; razón por la que Hive-MQ al estar la escrita en javascript se requirió adaptar la interfaz gráfica del mapa también desarrollada en el mismo lenguaje de programación para que converjan con la del broker IoT.

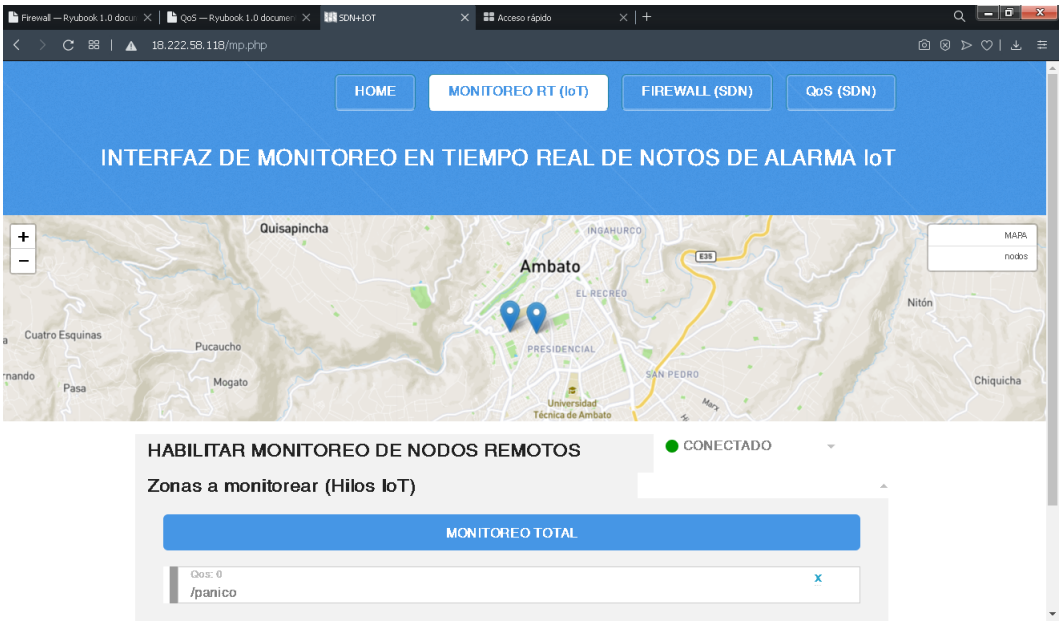


Figura 4.53: Entorno gráfico con 2 nodos de alarma IoT habilitados
Fuente: Desarrollado por el investigador

Ante tal circunstancia, se implementó una función completamente en javascript que fue incluida en conjunto con el código PHP de la pagina web para la creación de la alerta visual. En la sección de código posterior, se muestra el fragmento necesario para que los cambios generados por los websockets del broker IoT realicen acciones sobre el mapa implementado con anterioridad.

Algoritmo 5 Función para creación de capa de alarmas IoT sobre mapa

```
//Definición de variables
var auxC = false;
var auxM = false;

//CREACIÓN DE FUNCIÓN PARA LA CREACIÓN DE ALERTAS GEOREFERENCIADAS
function alertaMapa(lt, lg, estado) {
  var unique_id_C = "c"+lt.toString(); //creacion de identificadores
  var unique_id_M = "m"+lt.toString(); //creacion de identificadores
  var circle = new L.circle([lt, lg], { color: 'red', fillColor: '#f03',
fillOpacity: 0.5, radius: 400 });
  circle.layerID = unique_id_C;
  var icono = new LeafIcon({iconUrl: '../leaflet/sirena.gif'});
  var nmarker = new L.marker([lt, lg], {icon: icono, title: "ALARMA ACTIVADA",
zIndexOffset:-1000}).bindPopup("ALERTA");
  nmarker.layerID = unique_id_M;

  if(estado) {
console.log("GRAFICAR EN:"+lt.toString()+" "+lg.toString());
  //BUSCAR MARCADORES EN EL MAPA
  window.map.eachLayer(function(layer) {
  console.log("identidicador "+layer.layerID);

  //VERIFICAR SI EXISTEN VARIABLES DE ALARMA CREADAS CON ANTERIORIDAD
  if (layer.layerID === unique_id_C) { auxC = true; }
  if (layer.layerID === unique_id_M) { auxM = true; } });

  //VERIFICA SI EXISTEN ALERTAS EN EL PUNTO ESPECIFICO
  if(auxC === false) { window.map.addLayer(circle); }
  if(auxM === false) { window.map.addLayer(nmarker); } }

  else {

  //BUSCAR EL MARCADOR PARA BORRARLO CUANDO LA ALARMA SE DESACTIVE
  window.map.eachLayer(function(layer) {
  if (layer.layerID === unique_id_C) {
    map.removeLayer(layer); auxC = false;
  }
  if (layer.layerID === unique_id_M) {
    map.removeLayer(layer); auxM = false;
  }
  }); }
}
```

En la figura 4.54 se muestran las alertas que se despliegan sobre el mapa en base a los cambios del nodo de alarma IoT que se genera al invocar la función de creación de alarmas descrito.

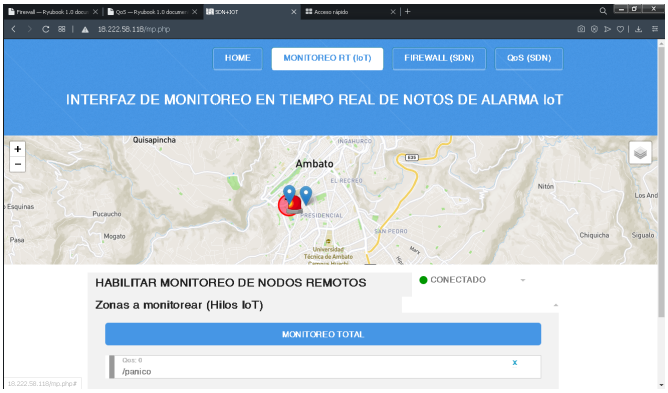


Figura 4.54: Interfaz de monitoreo WEB con alteraciones activadas por nodos IoT
Fuente: Desarrollado por el investigador

El conjunto de las herramientas trabajando en su totalidad posibilitó realización las pruebas de los sistemas de Alarma comunitaria basado en la arquitectura SDN como gestor de red e IoT para evitar la saturación de mensajes que son emitidos en tiempo real por los nodos de alarma.

En los siguientes apartados se analizan los resultados obtenidos al someter a la red a pruebas de tráfico en tiempo real con datos, video, audio y paquetes de IoT-MQTT para el control de alarmas y despliegue visual de alertas visuales.

4.3. Análisis de tráfico en red convergente basado en arquitecturas SDN e IoT

La topología de red diseñada para el monitoreo de alarmas comunitarias basadas en arquitecturas SDN e IoT se presenta en la figura 4.55, a diferencia de la red tradicional estudiada en el apartado inicial del presente capítulo, dentro de la nueva red de monitoreo se ha agregado un dispositivo SDN intermediario entre los dispositivos proporcionados por el ISP y los dispositivos finales alámbricos e inalámbricos.

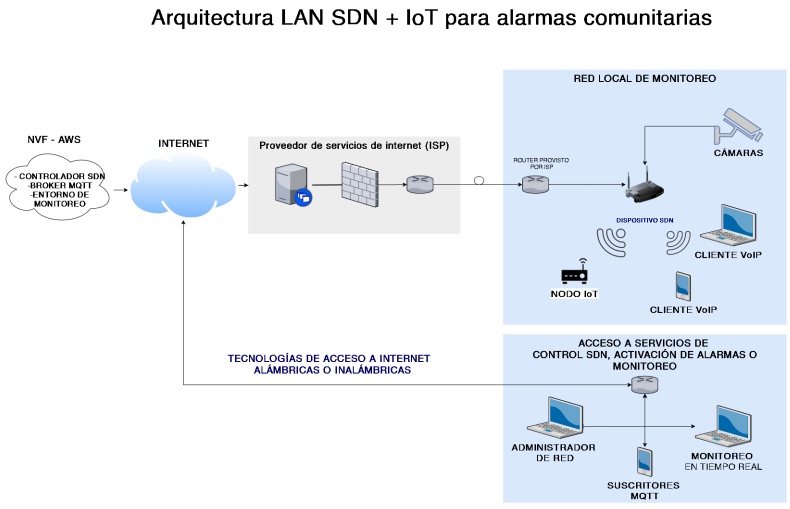
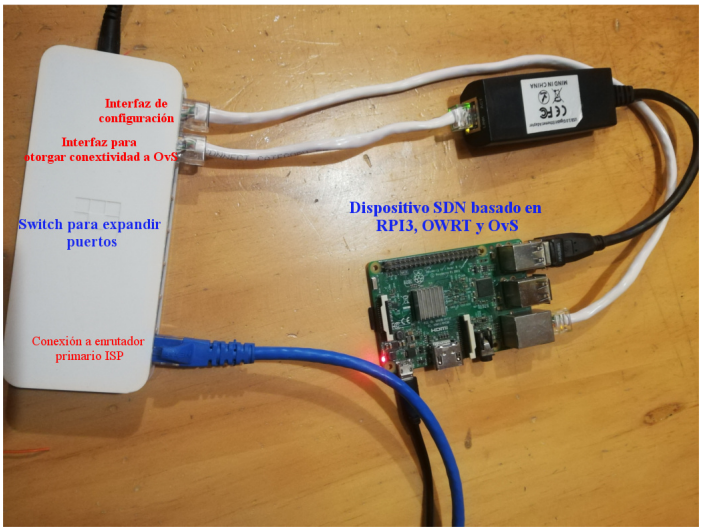


Figura 4.55: Arquitectura de red SDN implementada para monitoreo de alarmas comunitarias
Fuente: Desarrollado por el investigador

El conmutador SDN que se crea internamente en el dispositivo implementado y presentado en la figura 4.56, se conecta directamente hacia el controlador alojado en los servicios NVF de AWS y replica las políticas que el administrador configura para el punto de monitoreo específico o para toda la red. En adición, gracias a que el controlador se implementó en la nube con la garantías de seguridad y disponibilidad que otorga AWS; gran cantidad de dispositivos tanto SDN como IoT a nivel mundial tienen la capacidad de acceder a todos los servicios de control proporcionados por la arquitectura de red SDN, así como a los servicios de monitoreo y control de los nodos de alarma IoT.

Una acotación adicional a tomarse en cuenta es la necesidad de al menos dos puertos ethernet en el dispositivo SDN, haciendo referencia a lo que se describió en la arquitectura de implementación del dispositivo en el apartado de diseño del mismo y como nuevamente se presenta en la figura 4.56.a y 4.56.b. la primera de las interfaces ethernet se destina netamente a la configuración del dispositivo por el controlador SDN y a través del cual se comanda al OvS interno basado en software y la segunda interfaz ethernet es destinada a otorgar acceso a internet a todos los dispositivos alámbricos e inalámbricos conectados al OvS.

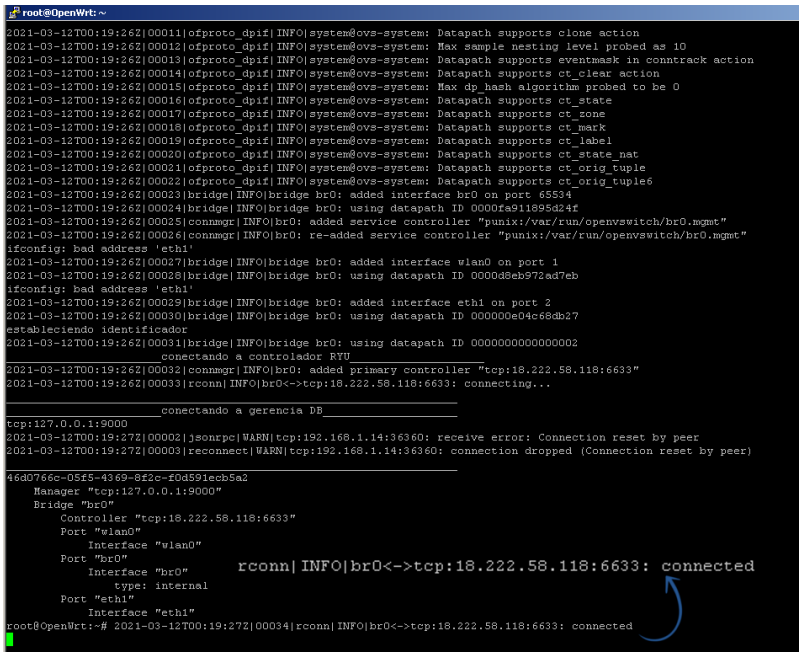


a) vista superior de dispositivo SDN implementado

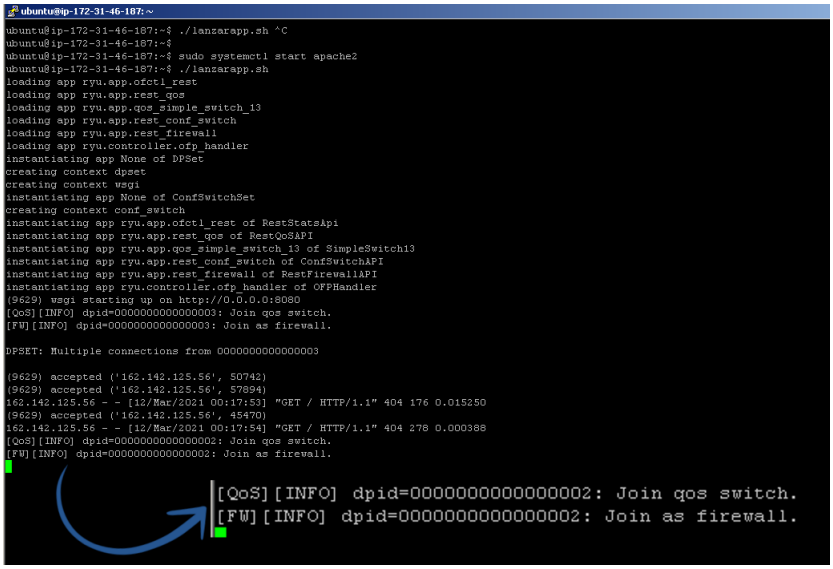
b) Verificación del funcionamiento de los enlaces

Figura 4.56: Dispositivo SDN físico basado en Raspberry Pi 3, OpenWRT y OpenVswitch
Fuente: Desarrollado por el investigador

Inicialmente, fue necesario habilitar el dispositivo SDN descrito en el apartado Implementación de Dispositivos SDN OVS RPI y comprobar que exista comunicación bidireccional, en la figura 4.57.a se puede observar el arranque en etapa temprana del vínculo que se genera entre el dispositivo SDN y el controlador de la figura 4.57.b, donde se denota la conexión establecida y el tipo el comportamiento que el controlador le ha asignado al dispositivo así como el comportamiento de cada uno de los periféricos de red que este contiene.



a) Activación de servicio interno para la creación de dispositivo SDN con la configuración el controlador otorgue.



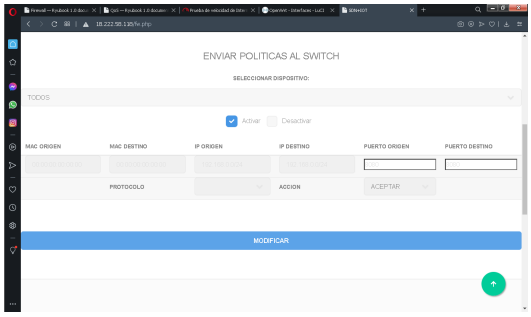
c) comprobación de conexión con controlador RYU

Figura 4.57: Habilitación de controlador SDN basado en RYU
Fuente: Desarrollado por el investigador

Fue posible comprobar visualmente la habilitación del dispositivo SDN, a través de la plataforma que se desarrolló con el afán de evitar el acceso mediante consola, de está forma se verificó que el o los administradores tengan un entorno de control de red de norte SDN de forma transparente.

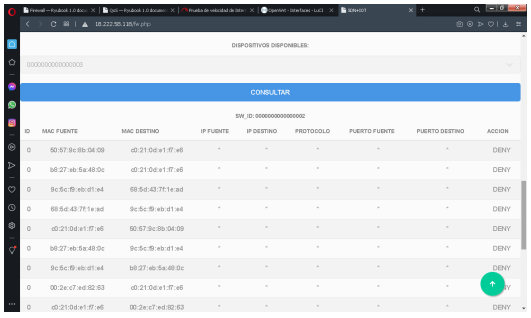
En la figura 4.58.a se puede observar visualmente la creación de un nuevo dispositivo SDN que inicia en un estado deshabilitado y sin políticas de control, para posteriormente en las figuras 4.58.b, 4.58.c y 4.58.d ser habilitado con políticas por defecto.

a) Verificación de dispositivo SDN conectados



b) Activación de dispositivo SDN mediante entorno gráfico

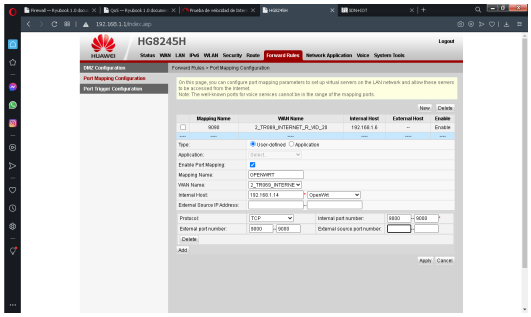
c) Verificación de estado activo de dispositivo SDN



d) Listado de políticas creadas por defecto al iniciar dispositivo

Figura 4.58: Habilitación de dispositivo SDN controlado por RYU instalado en NVF
Fuente: Desarrollado por el investigador

En segunda instancia, fue necesario garantizar la comunicación CONTROLADOR - DISPOSITIVO SDN para configuración de políticas de calidad de servicio, motivo por el cual y como se muestra en la figura 4.59.a fue necesario modificar los puertos del enrutador que el ISP provee al usuario con el afán de instaurar una política de firewall que permita redireccionar las comunicaciones que se envían a la IP pública del cliente hacia el puerto de configuración del OVSDB server del dispositivo SDN; posteriormente, el administrador de red está en la capacidad de configurar el controlador para la transacción de políticas de QoS como se observa en la figura 4.59.b.



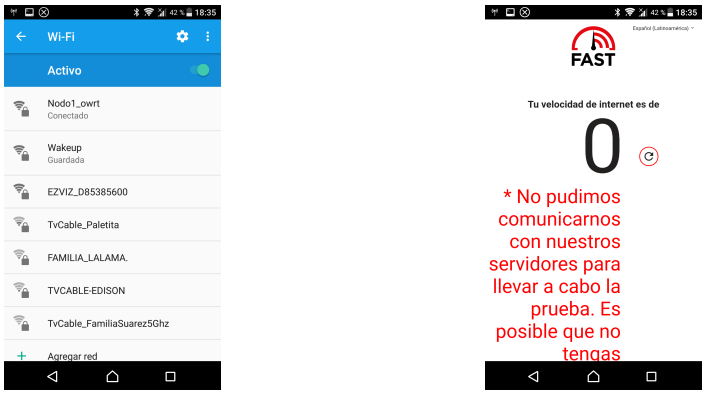
a) Redirección de tráfico puerto público a privado en enrutador primario

b) Habilitación de dispositivo SDN para creación de políticas de QoS

Figura 4.59: Modificación de políticas de redireccionamiento de puertos en enrutador provisto por ISP para intercambio de políticas de QoS
Fuente: Desarrollado por el investigador

4.3.1. Pruebas de ancho de banda sobre red SDN inicial

Un análisis inicial sobre el ancho de banda de la red alámbrica e inalámbrica provistas por el dispositivo SDN, ha demostrado la efectividad del uso de la arquitectura de SDN y se evidenció que pese a que un dispositivo ha sido habilitado como conmutador carece de políticas de control de tráfico, así como de calidad de servicio. En la figura 4.60 se muestran dispositivos correctamente vinculados a la red SDN pero con tráfico nulo.



a) Verificación de conexión a red inalámbrica SDN

b) Prueba de velocidad de navegación limitada por carencia de políticas.

Figura 4.60: Análisis de tráfico de salida limitado por políticas iniciales instanciadas en dispositivo SDN
Fuente: Desarrollado por el investigador

Adicionalmente, el análisis de ancho de banda presentado en la figura 4.61 entre dos dispositivos que forman parte de la red LAN provista por el dispositivo SDN; fue evidencia irrefutable de que el dispositivo es completamente configurable y que carece de políticas que proveen los fabricantes de hardware en dispositivos comerciales, en este caso el tráfico permitido es prácticamente nulo entre 0 y 12kbps; además, se pudo aseverar que el dispositivo depende únicamente de las políticas que el administrador de red implemente para el segmento deseado.

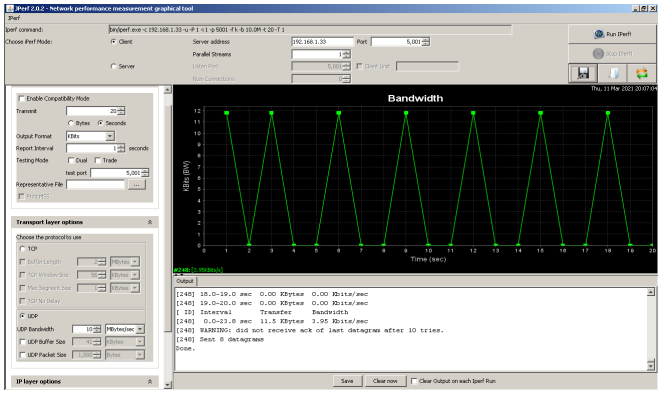
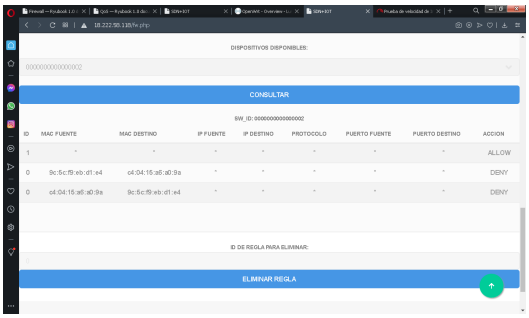


Figura 4.61: Análisis de tráfico entre dos puntos pertenecientes a red LAN SDN sin configuraciones iniciales

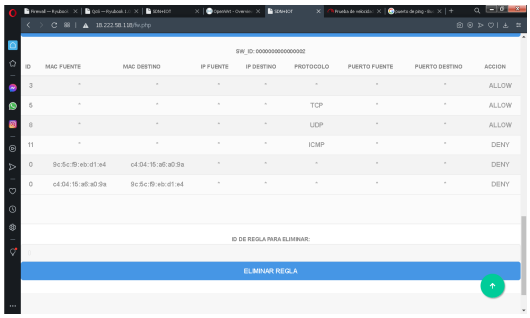
Fuente: Desarrollado por el investigador

4.3.2. Pruebas de tráfico en red de alarmas comunitarias basadas en arquitecturas SDN e IoT

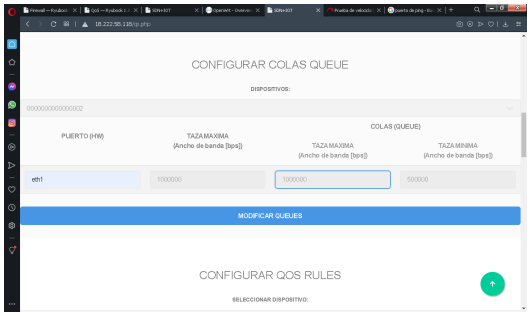
En base a los resultados obtenidos en el apartado anterior, debido al tráfico limitado entre dispositivos de red fue necesario habilitar nuevas reglas para el tráfico de la red a través del entorno gráfico; en la figura 4.62 se presenta una serie de políticas de prueba para verificar el funcionamiento y los efectos positivos que desencadenan sobre la red.



a) Eliminación de políticas de restricción de tráfico y habilitación de todo tipo de tráfico sobre la red.



b) Creación de políticas para habilitación de tráfico bajo protocolos TCP, UDP y denegación de ICMP

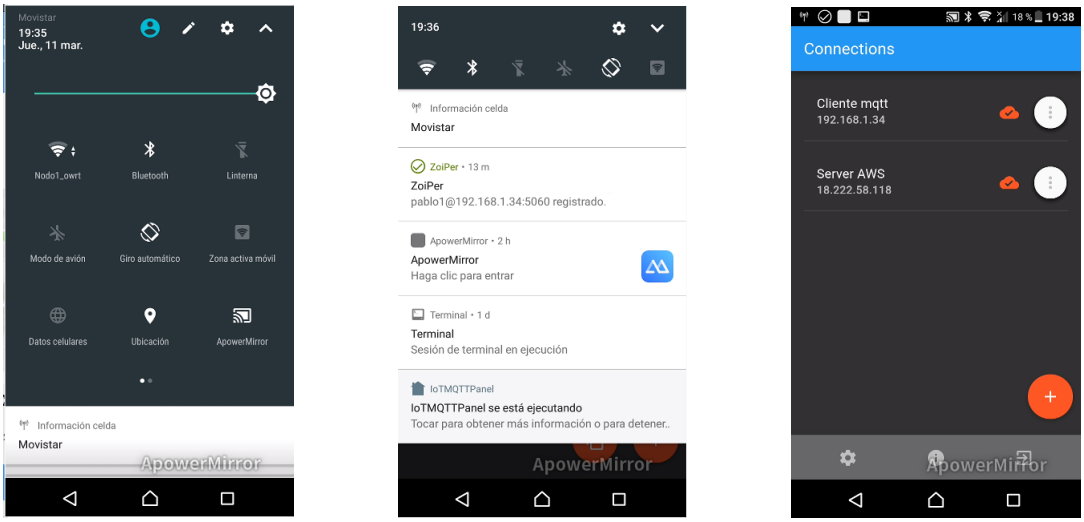


c) Habilitación de política de calidad de servicio por hardware ETH1 con límite de 1Mbps

Figura 4.62: Habilitación de políticas de control de tráfico y QoS
Fuente: Desarrollado por el investigador

Una vez efectuados los cambios, se iniciaron nuevamente los sistemas que generaban sobrecarga sobre la red tradicional para evaluar la calidad de servicio sobre una red gestionada de forma remota con la capa de datos y control por separado.

En la figura 4.59.a es posible observar la conexión al punto inalámbrico del dispositivo SDN para posteriormente en la figura 4.59.b habilitar nuevamente los servicios de VoIP y como último paso crear una conexión suscriptor/publicador para la activación de las alarmas IoT gestionadas por el broker fuera de red como se demuestra en la figura 4.59.c.



a) vinculación de dispositivos terminales a red inalámbrica SDN.

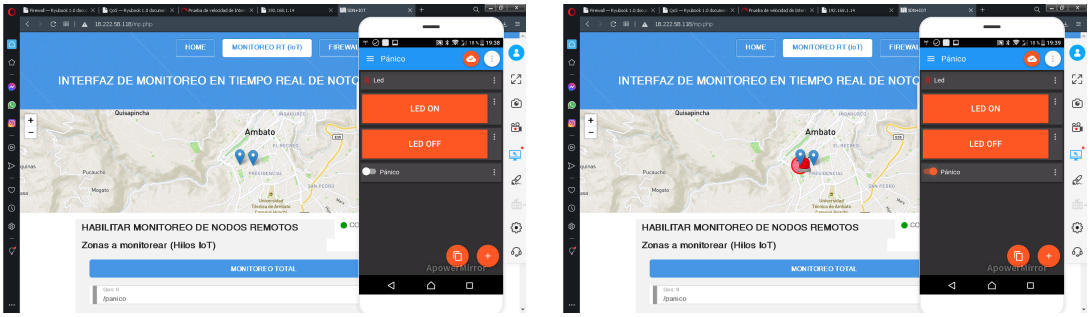
b) Comprobación de servicios VoIP en funcionamiento.

c) habilitación de conexión entre dispositivos IoT con broker NVF.

Figura 4.63: Habilitación de dispositivos para VoIP y botones de pánico IoT sobre red SDN
Fuente: Desarrollado por el investigador

La habilitación del tráfico por completo propuesto, permitió a todos los dispositivos sin excepción conectados a internet a través del dispositivo SDN controlado por RYU, acceder por completo al tráfico de red con la única condición de no superar el 1Mbps en la interfaz eth1 utilizada para proporcionar comunicaciones al conmutador OvS.

En la figura 4.64, se demuestra el correcto funcionamiento de los disparadores de alarmas para los nodos IoT, estos últimos presentaron retardos insignificantes e imperceptibles menores a 1 segundo tanto en el despliegue de alarmas visuales como en alarmas físicas.

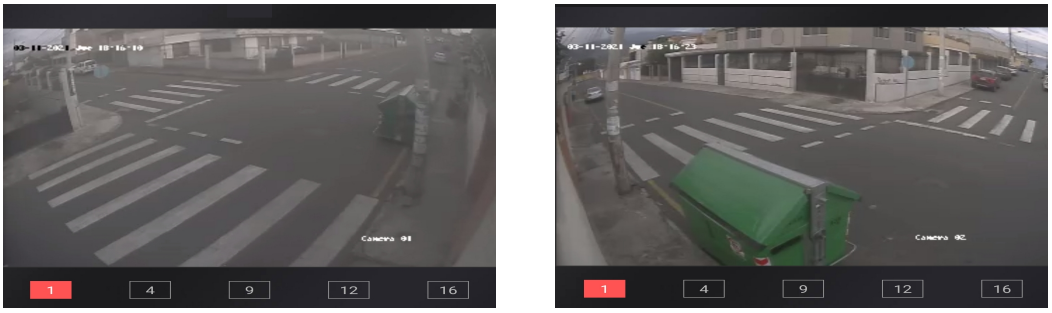


a) Alarma comunitaria georeferenciada desactivada

b) Alarma comunitaria georeferenciada activada

Figura 4.64: Activación de botones de pánico con retardo < 1 seg
Fuente: Desarrollado por el investigador

En concordancia con el análisis de las redes tradicionales, también para la topología de red basadas en arquitecturas SDN e IoT se habilitaron los servicios de videovigilancia remota presentado en la figura 4.64; servicio que generó la mayor sobrecarga en el ancho de banda provisto por el ISP en la red de estudio.



a) Verificación de acceso remoto a Cámara 01

b) Verificación de acceso remoto a Cámara 2

Figura 4.65: Habilitación de servicios de videovigilancia remota CCTV sobre SDN
Fuente: Desarrollado por el investigador

Los resultados de la limitación generada por las políticas implementadas en el controlador SDN y replicado en el conmutador definido por software demostraron la efectividad en términos de control de tráfico que anteriormente saturaban al enrutador primario. En la figura 4.66 se presenta el resultado de la prueba de velocidad de salida que otorga el ISP al usuario.

De los 20 Mbps inicialmente descritos y que posteriormente descendía a 1.3Mbps cuando los servicios de video y audio se habilitaban; con el dispositivo SDN intermediario como ente regidor del segmento de red de audio, video y trafico IoT-MQTT, la velocidad alcanzó los 14.6Mbps; un cambio drásticamente significativo, tomando en cuenta que el restante de dispositivos se conectaba directamente al enrutador sin intermediarios.

Figura 4.66: Prueba de velocidad posterior a la creación de políticas de control de tráfico en switch SDN

Fuente: Desarrollado por el investigador

En términos de ancho de banda interno, la evaluación del tráfico entre dos dispositivos de red pertenecientes a la LAN SDN también presentaron mejoras significativas inclusive cuando se mantenían llamadas de VoIP o visualización local y remota de videovigilancia. En la figura 4.67 se observa que los umbrales del ancho de banda alcanza valores de entre 7.2 y 10Mbps, inclusive generando tráfico paralelo de 12 hilos entre un cliente - servidor PERF; a diferencia del estudio inicial donde se alcanzaban como máximo 4Mbps.

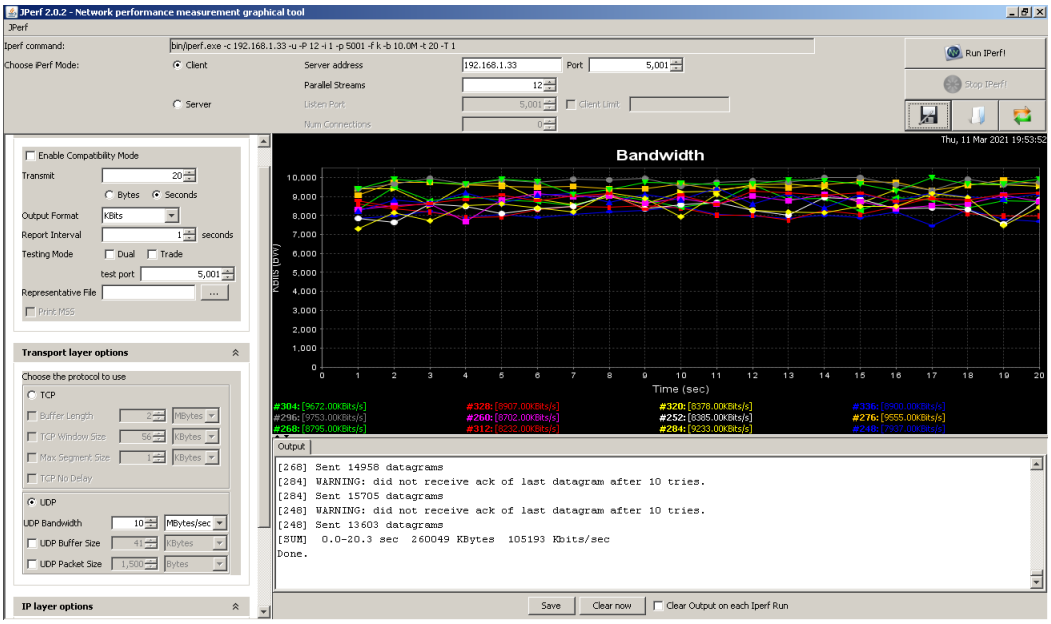


Figura 4.67: Pruebas de tráfico entre dos dispositivos dentro de red controlada SDN con servicios de VoIP, Video, Datos y botones de pánico IoT

Fuente: Desarrollado por el investigador

4.3.3. Pruebas de funcionamiento del controlador SDN diferentes redes destinadas al monitoreo de alarmas comunitarias

Finalmente, para comprobar el funcionamiento de la diversificación de mensajes de control que el el marco RYU implementado en un entorno NVF provee y la capacidad de activación de alarmas desde diferentes localidades con acceso a internet, se implementó al menos un punto de prueba con las mismas características y condiciones de red con un ancho de banda contratado de 10Mbps.

El procedimiento que se llevó a cabo fue el mismo empleado durante la implementación de la red y se describe de forma simplificada a continuación:

1. Comprobación del ancho de banda con servicios de video vigilancia local y remota habilitado, monitoreo de dispositivos IoT y acceso a servicios web, en el resultado presentado en la figura 4.68 se demuestra un retardo significativo en donde el ancho de banda no supera el 1.1Mbps.

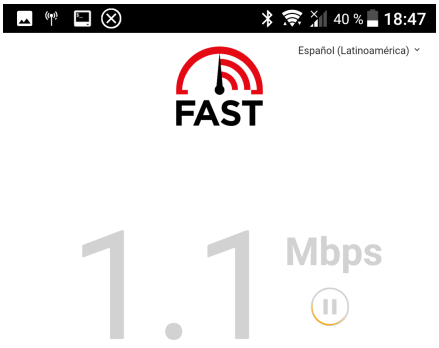


Figura 4.68: Evaluación de tráfico de salida con servicios de CCTV, IoT y datos
Fuente: Desarrollado por el investigador

2. Instalación de dispositivo SDN intermediario y modificación de las configuraciones de red del dispositivo SDN para adaptarlo a la subred que provee el ISP como en la figura 4.69.

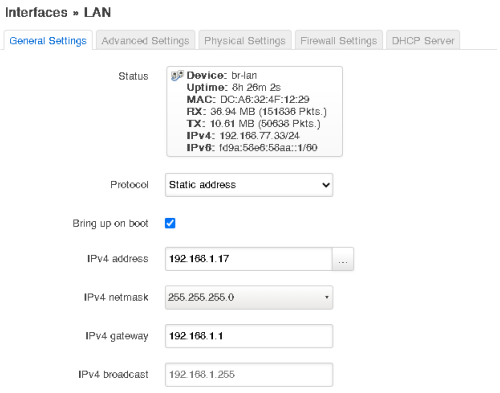


Figura 4.69: Modificación de IP estática asignada a la interfaz de control SDN
Fuente: Desarrollado por el investigador

3. Modificación de políticas en enrutador primario para habilitar comunicación bidireccional entre controlador SDN y conmutador definido por software como se muestra en la figura 4.69.

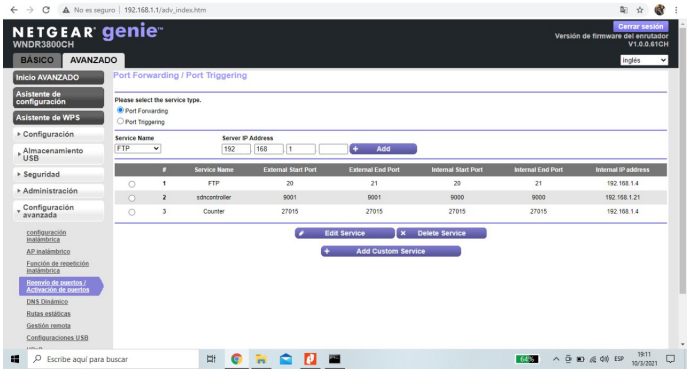


Figura 4.70: Implementación de política de redireccionamiento para QoS
Fuente: Desarrollado por el investigador

4. Verificación del funcionamiento de red inalámbrica controlada por OvS interno como se puede observar en la figura 4.69.

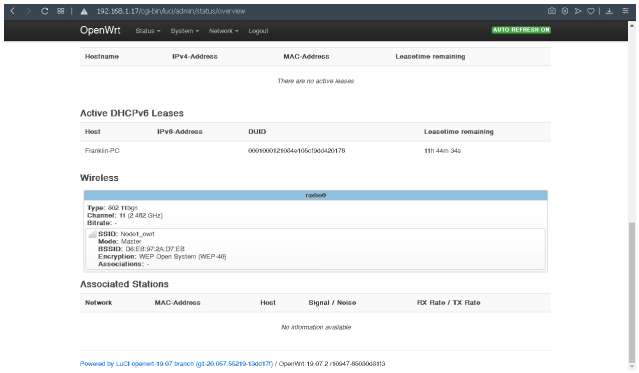


Figura 4.71: Verificación de red local inalámbrica controlada por marco de control SDN RYU
Fuente: Desarrollado por el investigador

5. Habilitación de servicios de video vigilancia remota comprobada en las figuras 4.69, servicio que mayor ancho de banda consume en una red tradicional.

a) Verificación de acceso remoto a sistema CCTV diurno

b) Verificación de acceso remoto a sistema CCTV nocturno

Figura 4.72: Habilitación de dispositivo de video vigilancia CCTV sobre red SDN
Fuente: Desarrollado por el investigador

6. En la figura 4.69 se muestra el resultado de la prueba de velocidad de internet con servicios CCTV, IoT y datos sobre segmento de red SDN controlado por marco RYU.

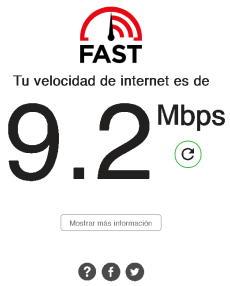
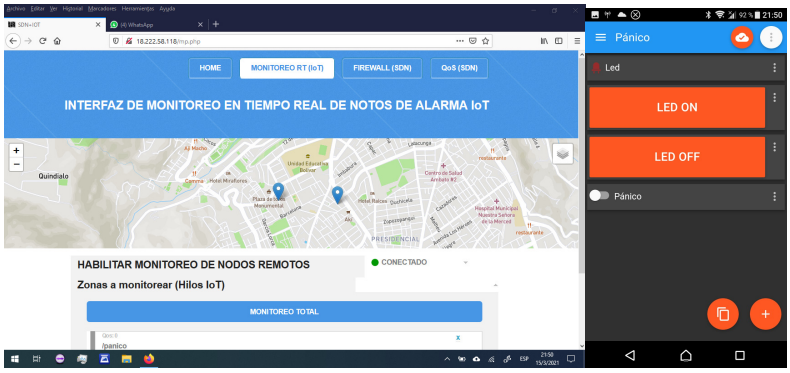
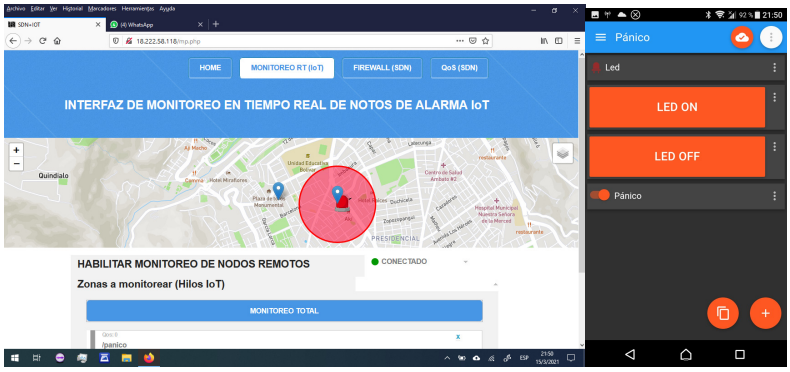


Figura 4.73: Verificación de red local inalámbrica controlada por marco de control SDN RYU
Fuente: Desarrollado por el investigador

7. Adicionalmente, se verificó la respuesta de activación de nodos de alarma visual IoT de forma remota a través de los disparadores habilitados.



a) Interfaz de monitoreo remoto de nodos de alarma comunitaria basado en arquitectura IoT.



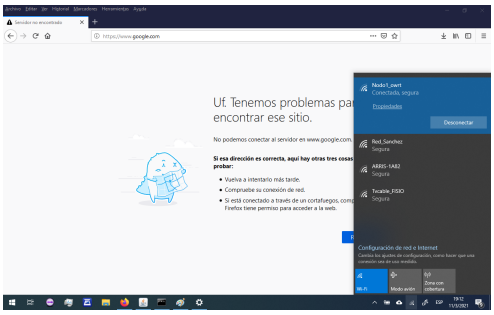
b) Interfaz de monitoreo remoto con sistema de alarma activado de forma inalámbrica a través de red SDN.

Figura 4.74: Verificación del funcionamiento del sistema de monitoreo de alarmas comunitarias basado en arquitectura IoT y control de tráfico basado en el marco SDN RYU desde dos puntos separados geográficamente.
Fuente: Desarrollado por el investigador

8. Finalmente, se desarrollaron pruebas de control de tráfico y habilitación de políticas de QoS

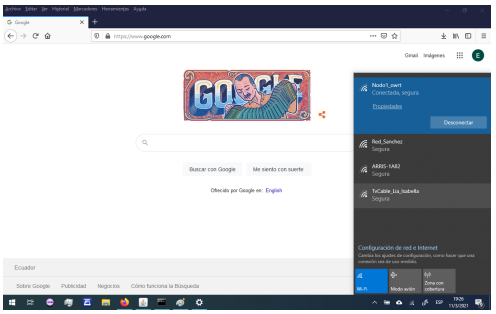
creadas en un segmento de red remoto y diferente a la red local controlada como se muestra en las figuras 4.75.

a) Des-habilitación de política de control de tráfico SDN desde entorno remoto A.



b) Pruebas de control de tráfico limitado a través de dispositivo SDN en entorno remoto B.

c) Habilitación de política de control de tráfico SDN desde entorno remoto A.



d) Pruebas de control de tráfico limitado a través de dispositivo SDN en entorno remoto B.

Figura 4.75: Verificación de políticas de tráfico implementadas en dispositivo SDN a través de marco de control RYU en entorno NVF en dos puntos separados geográficamente.
Fuente: Desarrollado por el investigador

Los resultados obtenidos durante las pruebas de control de tráfico en diferentes segmentos de red, separados geográficamente y basados en arquitecturas SDN e IoT, han demostrado que un marco de control basado en software simplifica las funcionalidades de control para administradores de red y posibilita el despliegue de redes interoperables de gran cobertura de forma sencilla sin importar tecnología de acceso a internet, proveedor de servicios de red e inclusive ancho de banda contratado.

4.4. Análisis de resultados de red tradicional y red SDN e IoT convergentes

A través de la tabla 4.6, finalmente se presenta una síntesis de los resultados obtenidos y descritos anteriormente en la cual se engloban las configuraciones primarias que se han implementado sobre el controlador SDN primario y replicado sobre los conmutadores definidos por software para optimizar el ancho de banda que una red tradicional dispone y evaluar los beneficios que supone la migración hacia una red basada por software.

Tabla 4.6: Resultados globales obtenidos

Arquitectura	Servicios en demanda	Política	Retardo en IoT	AB Intranet	AB Global
Tradicional	Navegación WEB (normal)	Por defecto	-	< 4Mbps	18 Mbps / 20 Mbps
Tradicional	Navegación WEB (pico)	Por defecto	-	< 2Mbps	12 Mbps / 20 Mbps
Tradicional	Navegación WEB (pico), VoIP local, tráfico de dispositivos IoT locales	Por defecto	< 2 seg	< 2Mbps	5.4 Mbps / 20 Mbps
Tradicional	Navegación WEB (pico), VoIP local, tráfico de dispositivos IoT locales, CCTV local y remoto	Por defecto	< 2 seg	< 2Mbps	1.3 Mbps / 20 Mbps
SDN + IoT	Navegación WEB (pico), VoIP local, tráfico de dispositivos IoT locales y remotos	- FW inhabilitado - sin políticas QoS	-	< 20Kbps	0 Mbps / 20 Mbps
SDN + IoT	Navegación WEB (pico), VoIP local, tráfico de dispositivos IoT locales, CCTV local y remoto	- FW habilitado - todo tráfico a través de FW habilitado - sin políticas QoS	< 1 seg	< 4Mbps	6 Mbps / 20 Mbps
SDN + IoT	Navegación WEB (pico), VoIP local, tráfico de dispositivos IoT locales, CCTV local y remoto	- FW Habilitado - habilitación de tráfico TCP Y UDP - bloqueo de paquetes ICMP interno - Limitación de ancho de banda a través de política de QoS	< 1 seg	7Mbps < AB < 10 Mbps	14.6 Mbps / 20 Mbps
Tradicional (Segmento de prueba remoto)	Navegación WEB (pico), VoIP local, tráfico de dispositivos IoT locales, CCTV local y remoto	Por defecto	< 2 seg	< 4 Mbps	1.1 Mbps / 10 Mbps
SDN + IoT (Segmento de prueba remoto)	Navegación WEB (pico), VoIP local, tráfico de dispositivos IoT locales, CCTV local y remoto	- FW Habilitado - habilitación de tráfico TCP Y UDP - bloqueo de paquetes ICMP interno - Limitación de ancho de banda a través de política de QoS	< 1 seg	< 7 Mbps	9.2 Mbps / 10 Mbps

Nota: *AB = Ancho de banda, normal = tráfico en horario diurno, pico = tráfico en horario nocturno, FW = firewall*

Fuente: Desarrollado por el investigador

Es evidente la mejora que se genera en términos de congestión y carga de paquetes de datos al agregar un conmutador SDN entre el enrutador primario y los dispositivos de red que anteriormente solían conectarse de forma directa sin ningún tipo de política de control.

Tomando como punto referencial la información de la tabla 4.6 y sintetizando el análisis de resultados previamente expuesto durante el desarrollo de este capítulo, las políticas por defecto de un enrutador propietario no priorizan el tráfico de información de ningún tipo es decir: se tratan a toda la información por igual y genera cuellos de botella y problemas significativos en el ancho de banda para transmisión y recepción de datos en una red; sin embargo, únicamente con agregar un dispositivo intermediario completamente programable, se habilita una extensa gama de funcionalidades técnicas que permiten clasificar, limitar e inclusive obstruir directamente el tráfico de red local y saliente.

Con base en dicho análisis, se ha demostrado que varios segmentos de red SDN controlados a través de un nodo centralizado, contribuye en gran medida a la replicación de políticas de bajo nivel que anteriormente requerían ser implementadas de forma manual por parte de los administradores de red, esto ha permitido reducir el tiempo de implementación, despliegue de red y actualización de políticas de control sobre redes de gran magnitud; y adicionalmente, se ha proporcionado un marco de control completamente adaptable y configurable con el afán de descongestionar segmentos saturados que dependen de la cantidad de políticas de control e ingeniería de tráfico que un único administrador desarrolle.

4.5. Análisis presupuestario

En concordancia con el análisis técnico expuesto en los anteriores apartados, en esta sección se presenta un análisis económico con el afán de otorgar una perspectiva realista a tomar en cuenta al momento de migrar sistemas completos con arquitectura de red tradicional por sistemas basados en arquitecturas SDN complementadas con IoT sobre entornos NVF. En la tabla 4.8 se presenta el costo referencial de cada nodo que componga la red SDN con N nodos separados geográficamente.

Tabla 4.8: Tabla de presupuesto económico requerido para la implementación de un nodo de alarma comunitaria SDN + IoT

Materiales	**Cantidad**	**Costo unitario**	**Costo total**
Raspberry pi 3 modelo B, RAM = 1Gb	1	70,00	70,00
Adaptadores USB - ETHERNET	1	20,00	20,00
Adaptador USB - WIFI	1	25,00	25,00
Tarjeta SD 32Gb clase 10	1	15,00	25,00
Fuentes reguladas 5Vdc - 2.5A	1	10,00	10,00
Dispositivo microcontrolado NodeMCU	1	8,00	8,00
Módulo relé doble optoisolado	1	4,00	4,00
Interruptor tipo hongo	1	4,00	4,00
Sirena 120Vac	1	45,00	45,00
Materiales y cableado	1	30,00	30,00
		TOTAL	241,00

Fuente: Desarrollado por el investigador

El costo total de dos cientos cuarenta y un dólares americanos no representa un costo excesivo si se considera que se ha invertido por única vez en la sustitución de conmutador de red de hardware propietario por un nodo SDN completamente gestionado desde un punto centralizado y en adición

se sustituye la placa primaria del sistema de alarma por la unidad de microcontrol, adicionalmente y con el despliegue del controlador SDN en un entorno NVF se generaron servicios recurrentes que los encargados de cada zona a monitorear deberían cubrir mensual o anualmente para el óptimo funcionamiento del sistema. En la tabla 4.9 se presentan los costos mencionados.

Tabla 4.9: Tabla de presupuesto económico que los usuarios asumirían anualmente

Servicio	**Costo mensual**	**Costo Anual**
Instancia virtualizada bajo demanda EC2 de AWS	18,40	220,80
Software y licencias adicionales	0,00	0,00
Servicio de acceso a internet dedicado (opcional)	25,00	300,00
Mantenimiento del sistema	25,00	300,00
	TOTAL	820,80

Fuente: Desarrollado por el investigador

El costo total de ocho cientos veinte dólares con ochenta centavos americanos de inversión anual puede representar un costo excesivo a simple vista. Sin embargo, es necesario tomar en cuenta que el costo de la instancia virtualizada sería un rubro que se debería cubrir entre todos los habitantes de los todos sectores beneficiados en los que el controlador SDN actúa y únicamente los costos de servicio de internet dedicado y mantenimiento del sistema serían los valores que se deberían cubrir de forma independiente por cada uno de los sectores que contengan un dispositivo SDN controlado.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- En redes de datos de pequeña y mediana magnitud, las políticas de control de flujo y de calidad de servicio han sido poco exploradas y explotadas; la razón de ello, es debido a las limitaciones proporcionadas por el hardware utilizado como gestor de red. Estos dispositivos contienen los planos de control y datos integrados en un único nodo de conexión gerenciado por políticas preestablecidas por el fabricante que en conformidad con el crecimiento de la red, tienden a saturar por completo la salida y el intercambio de paquete dentro y fuera de la red local.

- De las soluciones a nivel de software creadas para satisfacer la necesidad de control sobre redes en crecimiento. Los marcos de control SDN que hasta la actualidad se han implementado, otorgan un stack completo de gestión para administradores de red tanto en entornos de Norte y Sur; sin embargo, depende de la funcionalidad, características y naturaleza de la red el seleccionar una herramienta que permita explotar en su totalidad las capacidades del hardware destinado a la gerencia del tráfico.

- Durante la implementación de redes de gran magnitud basadas en arquitecturas SDN e IoT de forma complementaria, se concluyó que el dispositivo más importante para garantizar el servicio ininterrumpido no es el conmutador SDN sino el servidor primario que contiene el servicio virtualizado para controlar la red. Pese a que la capacidad de almacenamiento, procesamiento, memoria de intercambio e interfaces de entrada y salida son relevantes, a implementarse como un dispositivo de red que no cuenta con un plano de control incrustado; no está en la capacidad de transaccionar con paquetes de datos sin un controlador central que lo comande.

- Los servicios de tráfico critico como el audio y el video en tiempo real son los principales responsables de saturar cualquier tipo de red sin importar si son basadas en hardware o software. Por lo que se puede concluir que, para otorgar un servicio de calidad a los usuarios de la red, es necesario generar políticas de tráfico bien definidas limitados por umbrales de ancho de banda y organizar los paquetes de red dependiendo la prioridad y funcionalidad que se requiera.

5.2. Recomendaciones

- Para entornos en los que se requiera implementar políticas para explotar la totalidad del ancho de banda contratado, es recomendable reemplazar el enrutador primario por uno con funcionalidades de control de tráfico y de ser posible que soporte políticas de calidad de servicio. Este nuevo dispositivo, no necesariamente debe contar con funcionalidades SDN, aunque simplificaría la tarea de replicar las políticas en segmentos de red idénticos pertenecientes a una misma organización.

- Es necesario puntualizar y recomendar a los administradores y desarrolladores de redes que para el despliegue de un SDN con las características propuestas en el estudio. Es necesario contar con dispositivo enrutador y un conmutador por separado; debido a que en el primero se llevan a cabo las tareas de asignación de asignación de IP, enrutamiento y enmascaramiento del tráfico de entrada y salida, dejando al conmutador basado en software con planos de control y datos separados la tarea de replicación de políticas del comportamiento del tráfico generadas en el marco de control SDN local o remoto.

- Para la implementación de redes de acceso mundial en los que se requiere alterar en tiempo real las políticas de gestión de tráfico, se recomienda utilizar planes de acceso a internet corporativos o domésticos en los que se garantice una IP pública fija; así como el control directo de las políticas de firewall de entrada y salida. Esto suprimirá los puntos de fallo y facilitará la tarea de gerenciar y mantener constantes las reglas de control de tráfico de una o n redes de datos.

- Es recomendable que mientras más extensa sea una red de datos, menos se utilicen los servicios de asignación de IP de forma dinámica y se documente una por una las direcciones asignadas a cada dispositivo que conforme la red. De esta forma se garantizará que el administrador de red mantenga el control completo de la red, posibilitará la creación de políticas personalizada acorde a los requerimientos de uno o varios segmentos y otorgará una capa extra de seguridad a la red.

Referencias

[1] Francisca Gonzales Rodriguez and Hipólito Percy Barbarán Mozo. La seguridad ciudadana como política gubernamental en américa latina en el último quinquenio. *Ciencia Latina Revista Científica Multidisciplinar*, 5(1):422–435, 2021.

[2] Sistema Integrado de Seguridad ECU911. 4500 camaras de videovigilancia servicio - integrado de seguridad ecu 911.

[3] Sistema Integrado de Seguridad ECU911. Cobertura - servicio integrado de seguridad ecu 911.

[4] Daniel Castro Aniyar, Juan Carlos Jácome, and Jorge Mancero. Seguridad ciudadana en ecuador: Política ministerial y evaluación de impacto, años 2010-2014. *Nova criminis: visiones criminológicas de la justicia penal*, (9):111–148, 2015.

[5] Longbin Chen, Meikang Qiu, Wenyun Dai, and Ning Jiang. Supporting high-quality video streaming with sdn-based cdns. *The Journal of Supercomputing*, 73(8):3547–3561, 2017.

[6] Ministerio de Gobierno Ecuador. Ministerio del interior implementa sistema integral de alarmas comunitarias, January 2021.

[7] Vijay Varadharajan. Using sdn to secure iot infrastructures. October 2019.

[8] Wiem Bekri, Rihab Jmal, and Lamia Chaari Fourati. Softwarized internet of things network monitoring. *IEEE Systems Journal*, 2020.

[9] Nicolae Paladi and Christian Gehrmann. Sdn access control for the masses. *Computers & Security*, 80:155–172, 2019.

[10] M Alenezi, K Almustafa, and KA Meerja. Cloud based sdn and nfv architectures for iot infrastructure. egypt. inform. j.(2018), 2018.

[11] Michael Baddeley, Reza Nejabati, George Oikonomou, Mahesh Sooriyabandara, and Dimitra Simeonidou. Evolving sdn for low-power iot networks. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 71–79. IEEE, 2018.

[12] Chaitanya Aggarwal and Kingshuk Srivastava. Securing iot devices using sdn and edge computing. In *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, pages 877–882. IEEE, 2016.

[13] Sergio Baza Alonso. Gestión de dispositivos iot con acceso a la red mediante 802.11 ax y sdn en el edge para orquestar los flujos de datos.

[14] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2014.

[15] Andrew S Tanenbaum. *Redes de computadoras*. Pearson educación, 2015.

[16] LEI Technology. How sdn is fixing existing network bottlenecks with more hardware at the edge, 2019.

[17] YUGE Technology. Yuge technology designed for security applications. online, 2019.

[18] Rebeca Sarai. Software defined network and the openflow protocol live. WebSite, 2019.

[19] Hyojoon Kim and Nick Feamster. Improving network management with software defined networking. *IEEE Communications Magazine*, 51(2):114–119, 2013.

[20] Stephen Finucane. What is open vswitch? online, 2017.

[21] Stephen Finucane. Why open vswitch? online, 2017.

[22] Ben Pfaff and Bruce Davie. The open vswitch database management protocol. *IETF RFC 7047*, 2013.

[23] Corporacion Red Hat. Open virtual network (ovn). online, 2020.

[24] Lusani Mamushiane, Albert Lysko, and Sabelo Dlamini. A comparative evaluation of the performance of popular sdn controllers. In *2018 Wireless Days (WD)*, pages 54–59. IEEE, 2018.

[25] Manuel Eduardo Peña Pernia. *Modelo para virtualización de funciones de redes en aquitecturas interoperables de M-Gobierno en Venezuela*. PhD thesis, UNIVERSIDAD CATÓLICA ANDRÉS BELLO, 2019.

[26] Carlos Peliza, Fernando Dufour, Ariel Serra, Gustavo Micieli, and Facundo Guerrero. Virtualización de funciones de red. In *XXI Workshop de Investigadores en Ciencias de la Computación (WICC 2019, Universidad Nacional de San Juan).*, 2019.

[27] Jordi Salazar and Santiago Silvestre. Internet de las cosas. *Techpedia. České vysoké učení technické v Praze Fakulta elektrotechnická*, 2016.

[28] Andrea Palmieri, Paolo Prem, Silvio Ranise, Umberto Morelli, and Tahir Ahmad. Mqttsa: A tool for automatically assisting the secure deployments of mqtt brokers. In *2019 IEEE World Congress on Services (SERVICES)*, volume 2642, pages 47–53. IEEE, 2019.

[29] Bharati Wukkadada, Kirti Wankhede, Ramith Nambiar, and Amala Nair. Comparison with http and mqtt in internet of things (iot). In *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 249–253. IEEE, 2018.

[30] Gaston C Hillar. *MQTT Essentials-A lightweight IoT protocol*. Packt Publishing Ltd, 2017.

[31] Eclipse Mosquitto. An open source mqtt broker. *Eclipse Mosquitto [cit. 2018-04-23]. Dostupne z: Mosquitto. org*, 2018.

[32] César Alonso Irizar, Jesús Moya Neyra, and Caridad Anías Calderón. Evaluación de qoe en servicios ip basada en parámetros de qos. *Revista Ingeniería Electrónica, Automática y Comunicaciones ISSN: 1815-5928*, 38(3):36–46, 2019.

[33] Andy Grove. Java performancescalability testing framework, 2015. (undefined 12/3/2021 0:6).

[34] Amazon Inc. What is aws, 2011.

[35] Ryu SDN Framework Community. Ryu sdn framework, 2020.

[36] Nippon TT Corporation. Ryu app ofctl rest, 2016.

[37] Ryu Project Team. Firewall ryubook 10 documentation, 2014.

[38] Ryu Project Team. Qos ryubook 10 documentation, 2015.

[39] OpenWrt Project Team. Openwrt wiki raspberry pi, 2019.

[40] OpenWrt Project Team. Openwrt wiki packages, 2018.

[41] Mark Drake. How to install linux apache mysql php lamp stack on ubuntu 1804, 2018.

[42] Daniel Stenberg. Command line tool and library.

[43] HiveMQ GmbH. Hivemq a websockets based mqtt client for your browser, 2015.

[44] Vladimir Agafonkin. Documentation leaflet a javascript library for interactive maps, 2020.

ANEXOS

Anexo 1. Archivo de configuración para red inalámbrica en dispositivo OpenWRT - Open-vSwitch

```
config wifi-device 'radio0'
  option type 'mac80211'
  option channel '36'
  option hwmode '11a'
  option path 'platform/soc/fe300000.mmcnr/mmc_host/mmc1/mmc1:0001/mmc1:0001:1'
  option htmode 'VHT80' option disabled '1'
config wifi-iface 'default_radio0'
  option device 'radio0'
  option network 'lan'
  option mode 'ap'
  option ssid 'OpenWrt'
  option encryption 'none'
config wifi-device 'radio1'
  option type 'mac80211'
  option hwmode '11g'
  option path 'scb/fd500000.pcie/pci0000:00/0000:00:00.0/0000:01:00.0/usb1/1-1/1-1.3/1-1.3:1.0'
  option disabled '0'
  option cell_density '0'
  option channel '8'
config wifi-iface 'default_radio1'
  option device 'radio1'
  option network 'wifi'
  option mode 'ap'
  option ssid 'Nodo2_owrt'
  option encryption 'none'
```

Anexo 2. Archivo de configuración para firewall interno de dispositivo OpenWRT - Open-vSwitch

```
config defaults
  option input 'ACCEPT'
  option output 'ACCEPT'
  option forward 'REJECT'
  option synflood_protect '1'
config zone
  list network 'lan'
  option input 'ACCEPT'
  option output 'ACCEPT'
  option forward 'ACCEPT'
  option name 'lan_wifi'
config zone
  option output 'ACCEPT'
  option forward 'REJECT'
  option masq '1'
  option mtu_fix '1'
  option input 'ACCEPT'
  option name 'wan'
  option network 'lan'
config zone
  list network 'wifi'
  option input 'ACCEPT'
  option output 'ACCEPT'
  option forward 'REJECT'
  option name 'wifi_lan'
config forwarding
  option src 'lan_wifi' option dest 'wifi_lan'
config forwarding
  option dest 'lan_wifi' option src 'wifi_lan'
```

Anexo 3. Archivo de configuración para conexión entre dispositivo OpenWRT - Open-vSwitch y controlador SDN remoto basado RYU

```
#Declaracion de variables echo "declarando variables"
#IP del dispositivo
IP_SW=192.168.77.32
#IP del controlador OpenFlow
IP_CTL=IP_PROPORCIONADA_POR_AWS
#Identificador del DATAPATH
DP_IP=0000000000000002
#ID interfaz OVS
SW=br0
#Interfaces a activar como puertos OpenFlow
PUERTOS="wlan0 eth1"
#Variable para utilizar el comando ovs-vsctl epecificando la IP del switch y el puerto de escucha
VSCTL="ovs-vsctl --db=tcp:$IP_SW:9000"
#PATH para configuracion de la base OVS
OVSDB=/tmp/ovs-vswitchd.conf.db
#Rutina para aguardar por puertos
#sleep 5
espera_puertos(){
  puerto=$1
  while !'netstat -na | grep $puerto';do
    echo -n
    sleep 1
  done
}
#creacion de directorios faltantes mkdir /var/run/openvswitch/
echo "____________________REINICIANDO PROCESOS_____________________"
#Detener proceso del servidores de OVSDB
/usr/bin/killall ovsdb-server
#Detener y elimitar archivos y configuraciones previas de la OVSDB
/usr/bin/killall ovs-vswitchd rm /tmp/.ovs-vswitchd.conf.db.~lock~
#Elminar base OVSDB preexistente y volver a cargarla
rm -f $OVSDB
echo "creando base de datos OVSDB"
ovsdb-tool create $OVSDB /usr/share/openvswitch/vswitch.ovsschema
echo "iniciando servidor OVSDB"
#iniciar el servidor OVSDB
ovsdb-server $OVSDB --remote=ptcp:9000:$IP_SW &
#esperar 5 segundos a la escucha del puerto 9000
sleep 5
```

```
echo "iniciando switch"
#iniciar el switch ovs.vswitchd ovs-vswitchd
tcp:$IP_SW:9000 --pidfile=ovs-vswitchd.pid --overwrite-pidfile -- &
#VSCTL="ovs-vsctl"
echo "agregando puertos"
#agregar puertos
$VSCTL add-br $SW $VSCTL set bridge $SW
#agregar puertos
for i in $PUERTOS; do
  PUERTO=$i
    ifconfig $PUERTOS up
  $VSCTL add-port $SW $PUERTO done
echo "estableciendo identificador"
#establecer IDENTIFICADOR para el datapath del switch
$VSCTL set bridge $SW other-config:datapath-id=$DP_IP
echo "conectando a open floww" #Conectar al controlador OPENFLOW
$VSCTL set-controller $SW tcp:$IP_CTL:6633
$VSCTL show
```

Anexo 4. Código basado en PHP y Javascript para control del marco de Firewall SDN - RYU

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="/estilo.css" media="screen" />
</head>
<?php
include 'funciones.php';
$output = get_query("http://localhost:8080/stats/switches");
//$output = get_query("http://localhost:8080/firewall/module/status");
$arr = json_decode($output, true); //decodificar la respuesta JSON de la pagina
$dev_exist = sizeof($arr); //VERIFICA SI EXISTEN DISPOSITIVOS CONECTADOS AL CONTROLADOR
?>
<body>
<center>
<h1>APLICACION PARA CONTROL DE FIREWAL RYU</h1>
<h2>DISPOSITIVOS CONECTADOS AL CONTROLADOR</h2>
</center>
<?php
if(!$dev_exist)
echo "<center><h3 style='color:red'> NO EXISTEN DISPOSITIVOS CONECTADOS AL
CONTROLADOR</h3></center>";
?>
<div>
<table width="100%">
<tr>
<th>Switch</th>
<th>ID</th>
<th>STATUS</th>
</tr>
<?php
$sw_stat = get_query("http://localhost:8080/firewall/module/status");
$vector = json_decode($sw_stat, true);
$vector_eva = printValues($vector);
$n_dispositivos = repeticiones("switch_id",$vector_eva["keys"]);
if(!$dev_exist)
{
echo "<tr>
<td style='text-align:center'>*</td>
<td style='text-align:center'>*</td>
<td style='text-align:center'>*</td>
</tr>";
}
else
{
```

```
for($i = 0; $i < sizeof($n_dispositivos); $i++)
{
$devices = fw_rules($i, $n_dispositivos, $vector_eva["keys"], $vector_eva["values"]); //es metodo divide los datos
echo "<tr>
<td style='text-align:center'>",$i,"</td>
<td style='text-align:center'>".strtoupper ( $devices["switch_id"])."</td>
<td style='text-align:center'>".strtoupper ( $devices["status"])."</td>
</tr>";
}
}
?>
</table>
</div>
<hr>
<center>
<h2>ENVIAR POLITICAS AL SWITCH</h2>
</center>
<div>
<form action="" method="GET">
<!--- CHECKBOX PARA HABILITACION --->
<label for="dispositivo">DISPOSITIVOS:</label>
<select name="dispositivo" id="dispositivo">
<?php
foreach($arr as $key=>$value)
{
while(strlen($value)<16)
$value = '0'.$value;
echo "<option value='".$value."'>".$value."</option>";
}
echo "<option value='all'>TODOS</option>";
?>
</select>
<!--- CHECKBOX PARA HABILITACION FW EN SWITCH--->
<label for="habilitar">
<input type="checkbox" name="habilitar" id="habilitar" value="1">
Activar
</label>
<label for="deshabilitar">
<input type="checkbox" name="deshabilitar" id="deshabilitar" value="1">
Desactivar
</label>
<br>
<center>
```

```
<!----------FORMULARIO REGLAS---------->
<table>
<tr>
<th>MAC ORIGEN</th>
<th>MAC DESTINO</th>
<th>IP ORIGEN</th>
<th>IP DESTINO</th>
<th>PUERTO ORIGEN</th>
<th>PUERTO DESTINO</th>
<tr>
<tr>
<th><input type="text" id="mac_origen" name="mac_origen" placeholder="00:00:00:00:00:00"></th>
<th><input type="text" id="mac_destino" name="mac_destino" placeholder="00:00:00:00:00:00"></th>
<th><input type="text" id="ip_origen" name="ip_origen" placeholder="192.168.0.0/24"></th>
<th><input type="text" id="ip_destino" name="ip_destino" placeholder="192.168.0.0/24"></th>
<th><input type="number" id="puerto_origen" name="puerto_origen" placeholder="8080"></th>
<th><input type="number" id="puerto_destino" name="puerto_destino" placeholder="8080"></th>
<tr>
<br>
<tr>
<th></th>
<th>PROTOCOLO</th>
<th>
<select id="protocolo" name="protocolo">
<option value=""></option>
<option value="TCP">TCP</option>
<option value="UDP">UDP</option>
<option value="ICMP">ICMP</option>
<option value="ICMPv6">ICMPv6</option>
</select>
</th>
<th>ACCION</th>
<th>
<select id="accion" name="accion">
<option value="ALLOW">ACEPTAR</option>
<option value="DENY">RECHAZAR</option>
</select>
</th>
<th></th>
<tr>
</table>
<br><br>
```

```
<!--- BOTON ENVIAR--->
<input type="submit" value="MODIFICAR">
</form>
</center>
</div>
<?php
$dispositivo = $_GET["dispositivo"];
$habilitar = $_GET["habilitar"];
$deshabilitar = $_GET["deshabilitar"];
$mac_origen= htmlspecialchars($_GET["mac_origen"]);
$mac_destino=htmlspecialchars($_GET["mac_destino"]);
$ip_origen=htmlspecialchars($_GET["ip_origen"]);
$ip_destino=htmlspecialchars($_GET["ip_destino"]);
$puerto_origen=$_GET["puerto_origen"];
$puerto_destino=$_GET["puerto_destino"];
$protocolo=$_GET["protocolo"];
$accion = $_GET["accion"];
//habilitacion
if(isset($habilitar))
{
$url = "http://localhost:8080/firewall/module/enable/".$dispositivo;
$output = put_query($url);
$arr1 = json_decode($output, true); //decodificar la respuesta JSON de la pagina
//var_dump($output);
}
if(isset($deshabilitar))
{
$url = "http://localhost:8080/firewall/module/disable/".$dispositivo;
$output = put_query($url);
$arr1 = json_decode($output, true); //decodificar la respuesta JSON de la pagina
//var_dump($output);
}
/*
<td>".llenar($rules["dl_src"])."</td>
<td>".llenar($rules["dl_dst"])."</td>
<td>".llenar($rules["nw_src"])."</td>
<td>".llenar($rules["nw_dst"])."</td>
<td>".llenar($rules["nw_proto"])."</td>
<td>".llenar($rules["tp_src"])."</td>
<td>".llenar($rules["tp_dst"])."</td>
<td>".llenar($rules["actions"])."</td>
*/
```

```
$camposFw = array();
if(isset($mac_origen))
{
if(strlen($mac_origen)>15)
$camposFw["dl_src"] = strval($mac_origen);
//else
// echo "LA MAC DE ORIGEN: $mac_origen NO ES CORRECTA, POR FAVOR CORRIJALA E INTENTE NUEVAMENTE";
}
if(isset($mac_destino))
{
if(strlen($mac_destino)>15)
$camposFw["dl_dst"] = strval($mac_destino);
//else
// echo "LA MAC DE ORIGEN: $mac_origen NO ES CORRECTA, POR FAVOR CORRIJALA E INTENTE NUEVAMENTE";
}
if(isset($ip_origen))
{
if(strlen($ip_origen)>7)
$camposFw["nw_src"] = strval($ip_origen);
//else
// echo "LA MAC DE ORIGEN: $mac_origen NO ES CORRECTA, POR FAVOR CORRIJALA E INTENTE NUEVAMENTE";
}
if(isset($ip_destino))
{
if(strlen($ip_destino)>7)
$camposFw["nw_dst"] = strval($ip_destino);
}
if(!empty($puerto_origen))
{
$camposFw["tp_src"] = $puerto_origen;
}
if(!empty($puerto_destino))
{
$camposFw["tp_dst"] = $puerto_destino;
}
if(!empty($protocolo))
{
$camposFw["nw_proto"] = strval($protocolo); //agregar el tipo de protocolo
}
/*
```

```
if(!empty($accion))
{
$camposFw["actions"] = $accion; //agregar accion
}
*/
if(!empty($camposFw)) //si hay reglas agregadas al vector de reglas
{
//agregar accion en base a https://osrg.github.io/ryu-book/en/html/rest_firewall.html adding rules
$camposFw["actions"] = $accion;
//echo "PARA FIREWAL: ".print_r(json_encode($camposFw));
$url = "http://localhost:8080/firewall/rules/".$dispositivo;
$output = post_query($url,$camposFw);
$arr2 = json_decode($output, true); //decodificar la respuesta JSON de la pagina
//var_dump($output);
}
?>
<hr>
<!--- LISTAR POLITICAS DEL FW-->
<div>
<center>
<h2>POLITICAS HABILITADAS EN FIREWALL</h2>
</center>
<br>
<BR>
<form method="GET">
<label for="fws">DISPOSITIVOS:</label>
<select name="fws" id="fws">
<?php
foreach($arr as $key=>$value)
{
while(strlen($value)<16)
$value = '0'.$value;
echo "<option value='".$value."'>".$value."</option>";
}
//echo "<option value='all'>TODOS</option>";
?>
</select>
<input type="submit" value="CONSULTAR">
<br>
<?php
$fws = $_GET["fws"];
```

```
if($fws)
{
$url = "http://localhost:8080/firewall/rules/".$fws;
echo $url."<br>";
$output = get_query($url); //solicitar al servidor
echo $output;
$arr = json_decode($output, true); //decodificar la respuesta JSON de la pagina
$result = printValues($arr);
$n_reglas = repeticiones("rule_id",$result["keys"]);//obtener cuantas reglas existen en la cadena de retorno
/*
echo implode(",", $result["keys"]);
echo "<br>";
echo implode(",", $result["values"]);
echo "<br>";
*/
/*
$index = 0;
$n_reglas;
foreach($result["keys"] as $dato)
{
if($dato == "rule_id")
$n_reglas[] = $index;
$index++;
}
*/
}
?>
<!--- tabla de datos del firewall -->
<br>
<center>
<label for="reglasfw"><?php echo "SW_ID: ".$fws;?></label>
</center>
<table width="100%" id="reglasfw">
<tr>
<th>ID</th>
<th>MAC FUENTE</th>
<th>MAC DESTINO</th>
<th>IP FUENTE</th>
<th>IP DESTINO</th>
<th>PROTOCOLO</th>
<th>PUERTO FUENTE</th>
<th>PUERTO DESTINO</th>
<th>ACCION</th>
</tr>
```

```
<?php
if(empty($n_reglas))
{
for($i = 0; $i < 1; $i++)
{
echo "<tr> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td><td>-</td>
<td>-</td> <td>-</td> </tr>";
}
}
else
{
for($i = 0; $i < sizeof($n_reglas); $i++)
{
$rules = fw_rules($i, $n_reglas, $result["keys"], $result["values"]);
echo "<tr>
<td>".llenar($rules["rule_id"])."</td>
<td>".llenar($rules["dl_src"])."</td>
<td>".llenar($rules["dl_dst"])."</td>
<td>".llenar($rules["nw_src"])."</td>
<td>".llenar($rules["nw_dst"])."</td>
<td>".llenar($rules["nw_proto"])."</td>
<td>".llenar($rules["tp_src"])."</td>
<td>".llenar($rules["tp_dst"])."</td>
<td>".llenar($rules["actions"])."</td>
</tr>";
}
}
?>
</table>
</form>
</div>
<center>
<form method="GET">
<?php
$regla = $_GET["regla"];
$sw = $_GET["sw"];
$data = array();
if(isset($fws))
{
echo "<input type='number' id='regla' name='regla' placeholder='0'>";
echo "<input type='hidden' id='sw' name='sw' value=$fws>";
echo "<input type='submit' value='ELIMINAR REGLA'>";
}
```

```
if(isset($regla))
{
$data["rule_id"] = $regla;
$url = "http://localhost:8080/firewall/rules/".$sw;
$output = delete_query($url,$data);
$arr3 = json_decode($output, true); //decodificar la respuesta JSON de la pagina
echo "BORRANDO REGLA <b>$regla</b> DEL SWITCH <b>$sw</b>";
}
?>
</form>
</center>
</body>
```

Anexo 5. Código basado en PHP y Javascript para control del marco de QoS SDN - RYU

```
<html>
<head>
<link rel="stylesheet" type="text/css" href="/estilo.css" media="screen" />
</head>
<?php
include 'funciones.php';
$output = get_query("http://localhost:8080/stats/switches");
//$output = get_query("http://localhost:8080/firewall/module/status");
$arr = json_decode($output, true); //decodificar la respuesta JSON de la pagina
$dev_exist = sizeof($arr); //VERIFICA SI EXISTEN DISPOSITIVOS CONECTADOS AL CONTROLADOR
?>
<body>
<center>
<h1>APLICACION PARA CONTROL DE QOS SWITCH RYU</h1>
<h2>DISPOSITIVOS CONECTADOS AL CONTROLADOR</h2>
</center>
<?php
if(!$dev_exist)
echo "<center><h3 style='color:red'> NO EXISTEN DISPOSITIVOS CONECTADOS AL
CONTROLADOR</h3></center>";
?>
<div>
<table width="100%">
<tr>
<th>SWITCH</th>
<th>ID</th>
</tr>
<?php
$sw_stat = get_query("http://localhost:8080/firewall/module/status");
$vector = json_decode($sw_stat, true);
$vector_eva = printValues($vector);
$n_dispositivos = repeticiones("switch_id",$vector_eva["keys"]);
if(!$dev_exist)
{
echo "<tr>
<td style='text-align:center'>*</td>
<td style='text-align:center'>*</td>
<td style='text-align:center'>*</td>
</tr>";
}
else
{
```

```
for($i = 0; $i < sizeof($n_dispositivos); $i++)
{
$devices = fw_rules($i, $n_dispositivos, $vector_eva["keys"], $vector_eva["values"]); //es metodo divide los datos
echo "<tr>
<td style='text-align:center'>",$i,"</td>
<td style='text-align:center'>".strtoupper ( $devices["switch_id"])."</td>
</tr>";
}
}
?>
</table>
</div>
<center>
<h2>ENVIAR DATOS AL SWITCH</h2>
</center>
<div>
<form action="" method="GET">
<!--- CHECKBOX PARA HABILITACION --->
<label for="dispositivo">DISPOSITIVOS:</label>
<select name="dispositivo" id="dispositivo">
<?php
foreach($arr as $key=>$value)
{
while(strlen($value)<16)
$value = '0'.$value;
echo "<option value='".$value."'>".$value."</option>";
}
echo "<option value='all'>TODOS</option>";
?>
</select>
<!--- CHECKBOX PARA HABILITACION FW EN SWITCH--->
<label for="habilitar">
<input type="checkbox" name="habilitar" id="habilitar" value="1">
Activar
</label>
<label for="deshabilitar">
<input type="checkbox" name="deshabilitar" id="deshabilitar" value="1">
Desactivar
</label>
<br>
```

```
<center>
<!----------FORMULARIO REGLAS---------->
<table>
<tr>
<th>MAC ORIGEN</th>
<th>MAC DESTINO</th>
<th>IP ORIGEN</th>
<th>IP DESTINO</th>
<th>PUERTO ORIGEN</th>
<th>PUERTO DESTINO</th>
<tr>
<tr>
<th><input type="text" id="mac_origen" name="mac_origen" placeholder="00:00:00:00:00:00"></th>
<th><input type="text" id="mac_destino" name="mac_destino" placeholder="00:00:00:00:00:00"></th>
<th><input type="text" id="ip_origen" name="ip_origen" placeholder="192.168.0.0/24"></th>
<th><input type="text" id="ip_destino" name="ip_destino" placeholder="192.168.0.0/24"></th>
<th><input type="number" id="puerto_origen" name="puerto_origen" placeholder="8080"></th>
<th><input type="number" id="puerto_destino" name="puerto_destino" placeholder="8080"></th>
<tr>
<br>
<tr>
<th></th>
<th>PROTOCOLO</th>
<th>
<select id="protocolo" name="protocolo">
<option value=""></option>
<option value="TCP">TCP</option>
<option value="UDP">UDP</option>
<option value="ICMP">ICMP</option>
<option value="ICMPv6">ICMPv6</option>
</select>
</th>
<th>ACCION</th>
<th>
<select id="accion" name="accion">
<option value="ALLOW">ACEPTAR</option>
<option value="DENY">RECHAZAR</option>
</select>
</th>
<th></th>
<tr>
</table>
<br><br>
<!--- BOTON ENVIAR--->
<input type="submit" value="MODIFICAR">
</form>
</center>
</div>
```

```
<?php
$dispositivo = $_GET["dispositivo"];
$habilitar = $_GET["habilitar"];
$deshabilitar = $_GET["deshabilitar"];
$mac_origen= htmlspecialchars($_GET["mac_origen"]);
$mac_destino=htmlspecialchars($_GET["mac_destino"]);
$ip_origen=htmlspecialchars($_GET["ip_origen"]);
$ip_destino=htmlspecialchars($_GET["ip_destino"]);
$puerto_origen=$_GET["puerto_origen"];
$puerto_destino=$_GET["puerto_destino"];
$protocolo=$_GET["protocolo"];
$accion = $_GET["accion"];
//habilitacion
if(isset($habilitar))
{
$url = "http://localhost:8080/firewall/module/enable/".$dispositivo;
$output = put_query($url);
$arr1 = json_decode($output, true); //decodificar la respuesta JSON de la pagina
//var_dump($output);
}
if(isset($deshabilitar))
{
$url = "http://localhost:8080/firewall/module/disable/".$dispositivo;
$output = put_query($url);
$arr1 = json_decode($output, true); //decodificar la respuesta JSON de la pagina
//var_dump($output);
}
/*
<td>".llenar($rules["dl_src"])."</td>
<td>".llenar($rules["dl_dst"])."</td>
<td>".llenar($rules["nw_src"])."</td>
<td>".llenar($rules["nw_dst"])."</td>
<td>".llenar($rules["nw_proto"])."</td>
<td>".llenar($rules["tp_src"])."</td>
<td>".llenar($rules["tp_dst"])."</td>
<td>".llenar($rules["actions"])."</td>
*/
$camposFw = array();
if(isset($mac_origen))
{
if(strlen($mac_origen)>15)
$camposFw["dl_src"] = strval($mac_origen);
//else
// echo "LA MAC DE ORIGEN: $mac_origen NO ES CORRECTA, POR FAVOR CORRIJALA E INTENTE
NUEVAMENTE";
}
```

```
if(isset($mac_destino))
{
if(strlen($mac_destino)>15)
$camposFw["dl_dst"] = strval($mac_destino);
//else
// echo "LA MAC DE ORIGEN: $mac_origen NO ES CORRECTA, POR FAVOR CORRIJALA E INTENTE NUEVAMENTE";
}
if(isset($ip_origen))
{
if(strlen($ip_origen)>7)
$camposFw["nw_src"] = strval($ip_origen);
//else
// echo "LA MAC DE ORIGEN: $mac_origen NO ES CORRECTA, POR FAVOR CORRIJALA E INTENTE NUEVAMENTE";
}
if(isset($ip_destino))
{
if(strlen($ip_destino)>7)
$camposFw["nw_dst"] = strval($ip_destino);
//else
// echo "LA MAC DE ORIGEN: $mac_origen NO ES CORRECTA, POR FAVOR CORRIJALA E INTENTE NUEVAMENTE";
}
if(isset($puerto_origen))
{
if(strlen($puerto_origen)>7)
$camposFw["tp_src"] = $puerto_origen;
}
if(isset($puerto_destino))
{
if(strlen($puerto_destino)>7)
$camposFw["tp_dst"] = $puerto_destino;
}
if(!empty($protocolo))
{
$camposFw["nw_proto"] = strval($protocolo); //agregar el tipo de protocolo
}
if(!empty($camposFw)) //si hay reglas agregadas al vector de reglas
{
//agregar accion en base a https://osrg.github.io/ryu-book/en/html/rest_firewall.html adding rules
$camposFw["actions"] = $accion;
//echo "PARA FIREWAL: ".print_r(json_encode($camposFw));
```

```
$url = "http://localhost:8080/firewall/rules/".$dispositivo;
$output = post_query($url,$camposFw);
$arr2 = json_decode($output, true); //decodificar la respuesta JSON de la pagina
//var_dump($output);
}
?>
<!--- LISTAR INFORMACION DEL CONTROL DE QOS-->
<div>
<center>
<h2>POLITICAS DE FIREWALL</h2>
</center>
<br>
<BR>
<form method="GET">
<label for="fws">DISPOSITIVOS:</label>
<select name="fws" id="fws">
<?php
foreach($arr as $key=>$value)
{
while(strlen($value)<16)
$value = '0'.$value;
echo "<option value='".$value."'>".$value."</option>";
}
//echo "<option value='all'>TODOS</option>";
?>
</select>
<input type="submit" value="CONSULTAR">
<br>
<?php
$fws = $_GET["fws"];
if($fws)
{
$url = "http://localhost:8080/firewall/rules/".$fws;
$output = get_query($url); //solicitar al servidor
$arr = json_decode($output, true); //decodificar la respuesta JSON de la pagina
$result = printValues($arr);
$n_reglas = repeticiones("rule_id",$result["keys"]);//obtener cuantas reglas existen en la cadena de retorno
/*
echo implode(",", $result["keys"]);
echo "<br>";
echo implode(",", $result["values"]);
echo "<br>";
*/
/*
$index = 0;
$n_reglas;
foreach($result["keys"] as $dato)
{
```

```
if($dato == "rule_id")
$n_reglas[] = $index;
$index++;
}
*/
}
?>
<!--- tabla de datos del firewall -->
<br>
<center>
<label for="reglasfw"><?php echo "SW_ID: ".$fws;?></label>
</center>
<table width="100%" id="reglasfw">
<tr>
<th>ID</th>
<th>MAC FUENTE</th>
<th>MAC DESTINO</th>
<th>IP FUENTE</th>
<th>IP DESTINO</th>
<th>PROTOCOLO</th>
<th>PUERTO FUENTE</th>
<th>PUERTO DESTINO</th>
<th>ACCION</th>
</tr>
<?php
if(empty($n_reglas))
{
for($i = 0; $i < 1; $i++)
{
echo "<tr> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td><td>-</td>
<td>-</td> <td>-</td> </tr>";
}
}
else
{
for($i = 0; $i < sizeof($n_reglas); $i++)
{
$rules = fw_rules($i, $n_reglas, $result["keys"], $result["values"]);
echo "<tr>
<td>".llenar($rules["rule_id"])."</td>
<td>".llenar($rules["dl_src"])."</td>
<td>".llenar($rules["dl_dst"])."</td>
<td>".llenar($rules["nw_src"])."</td>
<td>".llenar($rules["nw_dst"])."</td>
<td>".llenar($rules["nw_proto"])."</td>
<td>".llenar($rules["tp_src"])."</td>
<td>".llenar($rules["tp_dst"])."</td>
<td>".llenar($rules["actions"])."</td> </tr>";}}?></table></form></div>
```

```
<center>
<form method="GET">
<?php
$regla = $_GET["regla"];
$sw = $_GET["sw"];
$data = array();
if(isset($fws))
{
echo "<input type='number' id='regla' name='regla' placeholder='0'>";
echo "<input type='hidden' id='sw' name='sw' value=$fws>";
echo "<input type='submit' value='ELIMINAR REGLA'>";
}
if(isset($regla))
{
$data["rule_id"] = $regla;
$url = "http://localhost:8080/firewall/rules/".$sw;
$output = delete_query($url,$data);
$arr3 = json_decode($output, true); //decodificar la respuesta JSON de la pagina
echo "BORRANDO REGLA <b>$regla</b> DEL SWITCH <b>$sw</b>";
}
?>
</form>
</center>
</body>
```

Anexo 6. Algoritmo de configuración de dispositivo de alarma IoT

```
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"
/************************ WiFi Access Point *********************************/
#define WLAN_SSID "SSID_RED_SDN"
#define WLAN_PASS "PASSWORD"
/************************ Adafruit.io ***************************************/
#define AIO_SERVER "IP_SERVER"
#define AIO_SERVERPORT 1883
// use 8883 for SSL
#define AIO_USERNAME "" #define AIO_KEY ""
/************************ LATITUD Y LONGITUD DEL NODO ***********************/
const double lat = -1.256380526910583;
const double lng = -78.63791361197998;
uint16_t identificador = 2;
WiFiClient client;
//CONFIGURAR CLIENTE MQTT
Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);
/*****************************    TOPICOS    MQTT    ***************************************/
Adafruit_MQTT_Publish panico = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/panico");
Adafruit_MQTT_Subscribe onoffbutton = Adafruit_MQTT_Subscribe(&mqtt, "test");
/************************** PREDEFINIR METODOS ***********************************/
void MQTT_connect();
String cadena = "";
/*********************************************************************/
uint8_t GPIO_Pin = D2;
uint8_t ALARMA = D5;
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(115200);
  delay(10);
  //boton de panico
  pinMode(GPIO_Pin, INPUT_PULLUP);
  //alarma pinMode(ALARMA, OUTPUT);
  digitalWrite(ALARMA, HIGH);
// Connect to WiFi access point.
  Serial.println();
  Serial.println();
  Serial.print("CONECTANDO ... ");
  Serial.println(WLAN_SSID);
  WiFi.begin(WLAN_SSID, WLAN_PASS);
  while (WiFi.status() != WL_CONNECTED) { delay(500); Serial.print("."); }
  Serial.println();
  Serial.println("CONECTADO WiFi");
  Serial.println("IP: ");
  Serial.println(WiFi.localIP());
```

```
//CREAR SUSCRIPCIONES
mqtt.subscribe(&onoffbutton);
}
uint32_t x=0; uint32_t y=0;
void loop() {
//GARANTIZAR CONEXION
MQTT_connect();
Adafruit_MQTT_Subscribe *subscription;
while ((subscription = mqtt.readSubscription(1000))) {
if (subscription == &onoffbutton) {
cadena = (char *)onoffbutton.lastread;
Serial.print(F("test cadena: "));
Serial.print(cadena);
Serial.print(F(" Longitud: "));
Serial.print(cadena.length());
Serial.print(F(" Got: "));
Serial.println((char *)onoffbutton.lastread);
if(cadena== "on") {
  digitalWrite(LED_BUILTIN, LOW);
  digitalWrite(ALARMA, LOW);
}
else if(cadena == "off") {
  digitalWrite(LED_BUILTIN, HIGH);
  digitalWrite(ALARMA, HIGH); }
}
}
x = identificador << 1; x = x | digitalRead(GPIO_Pin);
// Now we can publish stuff!
Serial.print(F("\nSending panico val "));
Serial.print(x, BIN);
Serial.print("...");
if (!panico.publish(x)) {
  Serial.println(F("FALLA"));
}
else {
  Serial.println(F("OK!")); }
}
//FUNCION DE CONEXION CON BROKER
void MQTT_connect() {
int8_t ret;
if (mqtt.connected()) { return; }
Serial.print("CONECTANDO A MQTT... ");
uint8_t retries = 3;
```

```
while ((ret = mqtt.connect()) != 0) {
  Serial.println(mqtt.connectErrorString(ret));
  Serial.println("REINTENTANDO EN 5 SEGUNDOS...");
  mqtt.disconnect();
  delay(5000);
  retries--;
if (retries == 0) {
  while (1);
}
}
Serial.println("MQTT Connected!");
}
```

Anexo 7. Código basado en PHP y Javascript para la gestión de alertas generadas por dispositivos IoT sobre mapa mundial

```
<!DOCTYPE html>
<html>
<head>
<title>MAPA NODOS IOT</title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<!----------DEPENDENCIAS LEAFLET MAPA---------->
<link rel="shortcut icon" type="image/x-icon" href="http://localhost/leaflet/docs/images/favicon.ico" />
<link rel="stylesheet" href="http://localhost/leaflet/leaflet.css" />
<script src="http://localhost/leaflet/leaflet.js"></script>
<link rel="stylesheet" href="https://unpkg.com/leaflet.markercluster@1.3.0/dist/MarkerCluster.css" />
<link rel="stylesheet" href="https://unpkg.com/leaflet.markercluster@1.3.0/dist/MarkerCluster.Default.css" />
<script src="https://unpkg.com/leaflet.markercluster@1.3.0/dist/leaflet.markercluster.js"></script>
<!----------DEPENDENCIAS IOT---------->
<link rel="stylesheet" href="http://localhost/iot/nucliot/css/normalize.css">
<link rel="stylesheet" href="http://localhost/iot/nucliot/css/style.css">
<link rel="stylesheet" href="http://localhost/iot/nucliot/css/foundation.css">
<link rel="stylesheet" href="http://localhost/iot/nucliot/css/jquery.minicolors.css">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/fancybox/2.1.5/jquery.fancybox.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<style>
/**************ESTILO DEL MAPA LEAFLET*****************/
html, body {
height: 100%;
margin: 0;
}
#map {
width: 100%;
height: 75%;
}
/**************ESTILO DE LOS COMPONENTES IOT*****************/
.switch {
position: relative;
display: inline-block;
width: 60px;
height: 34px;
}
.switch input {
opacity: 0;
width: 0;
height: 0;
}
```

```
.slider {
position: absolute;
cursor: pointer;
top: 0;
left: 0;
right: 0;
bottom: 0;
background-color: #ccc;
-webkit-transition: .4s;
transition: .4s;
}
.slider:before {
position: absolute;
content: "";
height: 26px;
width: 26px;
left: 4px;
bottom: 4px;
background-color: white;
-webkit-transition: .4s;
transition: .4s;
}
input:checked + .slider {
background-color: #2196F3;
}
input:focus + .slider {
box-shadow: 0 0 1px #2196F3;
}
input:checked + .slider:before {
-webkit-transform: translateX(26px);
-ms-transform: translateX(26px);
transform: translateX(26px);
}
/* Rounded sliders */
.slider.round {
border-radius: 34px;
}
.slider.round:before {
border-radius: 50%;
}
</style>
</head>
```

```
<!-------------------------------------------------------------------->
<script type="text/javascript">
function obtenerPosicion(identidad)
{
var cadena =null;
$.ajax({url: 'consultaPos.php',
async: false,
type: 'POST',
data: {id:identidad},
success: function(data) {
cadena = data;
}
});
return cadena;
}
</script>
<!-------------------------------------------------------------------->
<script>
const checkboxPan = document.getElementById('panico');
var valFalso = new Boolean(0);
console.log("variable: " + valFalso);
checkboxPan.addEventListener('change', (event) => {
if (event.target.checked) {
console.log("BOTON DE PANICO ACTIVADO");
websocketclient.publish('/panico','0',1,false);
}
else
{
console.log("BOTON DE PANICO DESACTIVADO");
websocketclient.publish('/panico','1',1,false);
}
})
function suscripciontotal() {
console.log("suscribiendose a todo");
//if(websocketclient.subscribe("test",1,$('#colorChooser').val().substring(1))){$.fancybox.close();}
if(websocketclient.subscribe("/panico",0,$('#colorChooser').val().substring(1))){$.fancybox.close();}
}
</script>
</body>
</html>
```