PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

DISEÑO DE UNA RED CORPORATIVA PARA UNA MYPE GESTIONADA MEDIANTE EL PARADIGMA IBNM (INTENT BASED NETWORKING MANAGEMENT)

Tesis para obtener el título profesional de Ingeniero de las Telecomunicaciones

AUTOR:

Jose Luis Grande Zúñiga

ASESOR:

Pastor David Chávez Muñoz

Lima, Octubre, 2021

Resumen

El presente trabajo de tesis se plantea el diseño de una red corporativa para una MYPE que es gestionada mediante intenciones o intents basado en software de código abierto y plataformas de libre uso como alternativa de gestión de red al método tradicional. Durante el capítulo 1, se describe la como es el comportamiento de la red en la actualidad, el estado actual de las redes de las MYPES , como es la gestión de la red y los beneficios que se obtienen cuando se implementa la automatización. Por otro lado, se especifican los objetivos a cumplir en el desarrollo la presente tesis y la motivación respecto al problema a resolver. En el capítulo 2, se describe la red basada en la intención(IBN); sus características, consideraciones y otras propuestas de implementación. Por otro lado, se desarrolla como es la arquitectura de una red definida por software (SDN), el rol que cumple un controlador y su entorno de trabajo desarrollado para el manejo de intenciones, el comportamiento de la REST API para el trabajo con aplicaciones externas y Mininet como emulador de red. Además, se explica el rol de la plataforma de comprensión de lenguaje natural. Finalmente, se especifica que es una MYPE y la importancia de las TIC's en estas empresas. En el capítulo 3, se describe la arquitectura del diseño propuesto para esta tesis, las acciones necesarias para el entrenamiento de la plataforma de comprensión de lenguaje natural, la lógica aplicada al algoritmo que procesa la intención, las configuraciones necesarias en el controlador y, finalmente, el despliegue de la topología propuesta en Mininet. En el capítulo 4, se muestran los resultados de las pruebas del ingreso de dos intenciones en una topología full mesh y el ingreso de varias otras intenciones en una topología de dos niveles, diferente de la utilizada anteriormente.

Dedicatoria

A mis padres y familia

Agradecimientos

A los profesores y los compañeros de la especialidad de Ingeniería de las Telecomunicaciones que compartimos clases, laboratorios y eventos durante toda la duración de la etapa de facultad.

A mi asesor, Mg. Pastor David Chavez Muñoz, por la orientación, enseñanzas y motivación brindada durante el presente trabajo.

Índice

Índice de Tablas

Índice de Figuras

Glosario de Términos y Abreviaturas

3GPP: 3rd Generation Partnership Project

API: Interfaz de Programacion de Aplicaciones

ARP: Protocolo de Resolucion de Direcciones

CLI: Interfaz de Linea de Comando

CLN: Comprension de Lenguaje Natural

ETSI: European Telecommunications Standards Institute

GUI: Interfaz Grafica de Usuario

HTTP: Protocolo de Transferencia de Hipertexto

IA: Inteligencia Artificial

IBN: Redes Basadas en la Intencion

IBNM: Gestion de Redes Basadas en la Intencion

ICMP: Protocolo de Mensajes de Control de Internet

IDL: Lenguaje de Definicion de la Intencion

IEEE: Instituto de Ingenieros Eléctricos y Electrónicos

INEI: Instituto Nacional de Estadistica e Informatica

ITU: International Telecommunication Union

JSON: Formato para Intercambio de Datos

M2M: Intercambio de Informacion entre dos maquinas

MAC: Media Access Control

ML: Machine Learning

MYPES: Micro y Pequeñas Empresas

NBI: Interfaz NorthBound

NETCONF: Protocolo de Configuracion de Red

NetOps: Operadores de Red

NFV: Virtualizacion de Funciones de Red

NLU: Comprension de Lenguaje Natural

ONF: Open Networking Foundation

ONOS: Sistema Operativo de Red Definido por Software

OpenDaylight: Plataforma para la creación de Redes Definidas por Software

Openflow: Tecnologia de Switching

OPEX: Gastos de Operacion

OpFLex: Politica de Transferencia de XML y JSON entre Controlador y Equipo

OVS: Switch Virtual de Codigo Abierto

OVSDB: Protocolo de Administracion en un Ambiente SDN

P4: lenguaje de programación para controlar los planos de reenvío de paquetes en dispositivos de red, como enrutadores y conmutadores

PBI: Producto Bruto Interno

REST: Transferencia de Estado Representacional

SBI: Interfaz SouthBound

SDN: Redes definidas por software

TCP: Protocolo de Control de Transmision

TIC: Tecnologias de la Informacion y Comunicacion

UI: Interfaz de Usuario

URI: Identificador de Recursos Uniforme

URL: Localizador de Recursos Uniforme

W3C: World Wide Web Consortium

XML: Extensible Markup Language

Introducción

Durante muchos años, la gestión de la red ha sido la misma a pesar de los diversos avances que se han venido dando en tecnologías de automatización. La red, propiamente, está en constante transformación volviéndose más compleja y de mayor dimensión año tras año debido a las nuevas aplicaciones que han surgido, a la gran cantidad de equipos que se están conectando y a la gran cantidad de tráfico que estos generan. Por tal motivo, crece cada vez más la necesidad de buscar una alternativa que pueda gestionar la red de forma escalable, rápida y confiable. Existen diversos esfuerzos en el área de la gestión de redes que buscan hacer uso de software de código abierto para lograr tal cometido. Por esta razón, se busca proponer en este trabajo una alternativa mediante la implementación de gestión de redes basándose en el paradigma IBNM sobre una red SDN. Al hacer uso de software de código abierto y de libre uso se busca reducir el OPEX y hacerlo accesible para todas las empresas. De esta manera, en los siguientes capítulos se describirán de forma profunda la problemática que se desea solucionar, los beneficios a lograr y se describen los objetivos del presente trabajo; en el segundo capítulo, se describirán los conceptos claves del diseño propuesto; en el tercer capítulo, se presenta el desarrollo de la solución propuesta y; finalmente, se presentan las pruebas realizadas y los resultados obtenidos según lo desarrollado.

Capítulo 1. Problemática y Objetivos

En el presente capitulo se define y justifica el problema a tratar, se definen los objetivos a alcanzar y se describen las razones que motivaron el trabajo.

1.1. Definición del problema y justificación

La red actual está viviendo una constante transformación debido al constante crecimiento de dispositivos que van conectando a esta y a las nuevas aplicaciones que surgen, las cuales exigen que se necesite realizar cambios en la red de forma más constante. Al aumentar la cantidad de dispositivos aumenta, como consecuencia, la cantidad de equipamiento de red necesario para satisfacer las necesidades de estos dispositivos. Por tal motivo, los cambios necesarios son cada vez mayores los cuales son realizados en su mayoría de forma manual a través del uso de CLI como indica Garner [1]. Los cambios mencionados deben ser realizados por personal especializado de forma manual e individual generando una predisposición a cometer errores y generar

inconsistencias en la red. Por otro lado, existe un costo involucrado en el uso de personal considerado dentro de los gastos de operación (OPEX) siendo este el mayor egreso de capital que una empresa asume el cual es más crítico si se trata de una MYPE que posee recursos muy limitados. El impacto en operación y económico es grande para estas empresas ya que un problema en la conexión debido a un error de configuración o por una inconsistencia puede afectar en gran medida. Además, esta posee recursos limitados y al sumársele costos de operación de la red reduciría su capacidad de inversión en otros ámbitos.

Durante este año debido a la pandemia y la paralización de la actividad económica por las medidas sanitarias tomadas se generó un gran impacto negativo en las MYPES. Como resultado de esto, muchas de estas empresas tuvieron que cesar sus actividades y despedir gran cantidad de su personal generando desempleo. La necesidad que tuvieron casi en su totalidad las MYPES del sector servicios para realizar actividades no presenciales, como teletrabajo, y/o cambiar de rubro para sobrevivir fue mediante al uso de TIC's. Estas plataformas por mucho tiempo fueron consideradas un lujo y costosas para ser implementadas por las MYPES, pero ahora gracias a tecnologías como la nube y las herramientas gratuitas a disposición, las TIC's son un gran aliado para las empresas en esta época. La gran cantidad de emprendimientos que se han sostenido y han iniciado son propiamente las que hicieron y hacen uso de la tecnología como redes sociales, correo electrónico, sitios web y aplicaciones móviles. Las soluciones más recientes basadas en el entorno cloud hacen posible que las MYPES combinen funciones de acuerdo a sus necesidades específicas y puedan competir en las mismas condiciones y con los mismos beneficios que las grandes empresas. Muchas de las empresas que están produciendo actualmente en esta coyuntura lo realizan con trabajo remoto y usando el dispositivo de su preferencia logrando mantener su productividad en todo momento. Por todo lo expuesto anteriormente, se demuestra que las TIC's

se han vuelto una necesidad esencial porque hace posible que las empresas estén a la vanguardia de los procesos competitivos dentro del mercado nacional e internacional[2] y en contacto con sus similares. Dentro de las ventajas otorgadas por la tecnología las más importantes expuestas por [2]son las siguientes:

- Establecer conexión con otras mypes.
- Conseguir el incremento de su mercado de venta.
- Obtener nuevas oportunidades comerciales.
- Tener un mejor acceso a la información.
- Aumentar la transparencia con el sector público y privado.
- Facilita acuerdos comerciales

1.1.1. La red en la Actualidad

Las redes actualmente están volviéndose más complejas debido la cantidad aplicaciones nuevas que surgen y a la masiva cantidad de dispositivos conectados. En el reporte anual de Sandvine del 2019 [3] sobre el internet muestra lo siguiente: en primer lugar, el 60% del tráfico se debe a aplicaciones de video streaming como Netflix, Youtube, Http Media Stream, IPTV,etc; en segundo lugar, la búsqueda web alcanza un 13%, cada vez más a través de plataformas de redes sociales; en tercer lugar, esta los videojuegos en línea que representan un 8% del tráfico de Internet; por último, las redes sociales, son el 6% del tráfico total. Se tiene que mencionar que este reporte es hacia finales del 2019 antes que se genere el incremento global de los patrones de tráfico del año 2020 debido al confinamiento.

Tabla 1-1 Categorización de Aplicaciones Generadoras de Tráfico en la Red

Categoría de Aplicaciones	Downstram	Upstream
Video Streaming	60.6%	22.2%
Web	13.1%	10.3%
Video Juegos en Línea	8.0%	4.9%
Redes Sociales	6.1%	7.6%

Fuente:[3]

La cantidad de tráfico sigue teniendo una tendencia al alza, lo cual se predice que seguirá sucediendo en los próximos años. Según el reporte realizado en el Cisco Anual Internet Report (2018-2023), se dará un incremento de cerca de 800 millones de nuevos usuarios de internet para el año 2023 en el mundo.

Figura 1-1 Proyección de Billones de usuarios de Internet para el año 2023

Fuente:[4]

La tendencia del incremento de usuarios será similar en esta región. Latinoamérica no será la excepción ya que se espera que cerca del 70% de la población estará conectada a internet ese mismo año, mostrado en la figura 1-2.

Tabla 1-2 Proyección sobre el Total de Usuarios en Internet para el año 2023

Región	2018	2023
Global	51%	66%
Latinoamérica	60%	70%

Fuente:[4]

Por otro lado, los dispositivos y conexiones están creciendo de forma más rápida que la población y la cantidad de usuarios de internet, siendo las conexiones M2M las que más crecerán abarcando un 50 % del total de las conexiones y dispositivos.

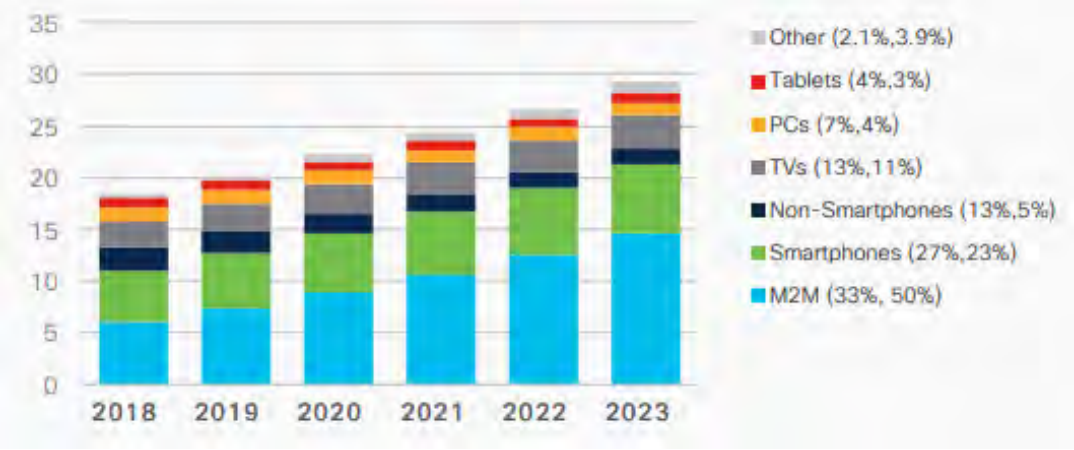


Figura 1-2 Proyección del Crecimiento de Dispositivos para el año 2023

Fuente:[4]

Se estima que el promedio de dispositivos y conexiones per cápita globalmente se incremente llegando a un valor de 3.6. De la misma forma, en Latinoamérica sucederá el mismo efecto llegando a obtenerse valores cercanos al promedio mundial.

Tabla 1-3 Proyección del Numero Promedio de Dispositivos Per-Cápita para el año 2023

Región	2018	2023
Global	2.4	3.6
Latinoamérica	2.2	3.1

Fuente:[4]

1.1.2. Estado de las redes en las MYPEs

Las redes de las mypes no difieren mucho en los desafíos que afrontan las redes de mayores dimensiones que se usan en las grandes empresas. Estas, por lo general, son más simples en su diseño y hacen uso de equipamiento de capacidades y costos menores. En el país, no todas las mypes tienen implementado una red para realizar sus actividades, ya que no las necesitan. Sin embargo, muchas de estas empresas desempeñan actividades en ciertos sectores como el tecnológico, servicios, salud, banca e industrial, donde se les hace necesario la implementación y uso de una. Las redes desplegadas en este tipo de empresas son muy similares entre sí, varían en las aplicaciones críticas que ofrecen según los servicios que prestan.

El diseño de las redes para la mypes a grandes rasgos por lo general consiste en diseños modulares, como se puede observar en la figura 1-3, donde el modulo central también llamado *Main Site* es donde las aplicaciones críticas residen en su mayoría. Otro de los componentes modulares son los llamados *remote sites*, figura1-4, los cuales son redes que se conectan al *main site* para poder hacer uso de las aplicaciones mencionadas, comunicarse con otros *remotes sites* o hacer uso de internet. En el país, por lo general los *remotes sites* implementados para una mype son ubicados en distintas plantas de un edificio, figura 1-5, o en una ubicación geográfica dentro de la ciudad o el país, siendo este el caso de menor ocurrencia.

Existen guías de referencia como la proporcionada por Cisco[5] donde dan recomendaciones para la implementación de redes para empresas de esta envergadura. La información proporcionada refleja que las recomendaciones son similares a las usadas en la actualidad en el país por diferentes empresas nacionales. La sugerencia proporcionada es la preferible ya que garantiza flexibilidad, modularidad y escalabilidad de la red para su correcto funcionamiento y pensando en los cambios a futuro. Por este motivo, gran cantidad de mypes que poseen una implementación de red con equipos de este *vendor* siguen el modelo descrito previamente (*Main site to Remote Sites*).

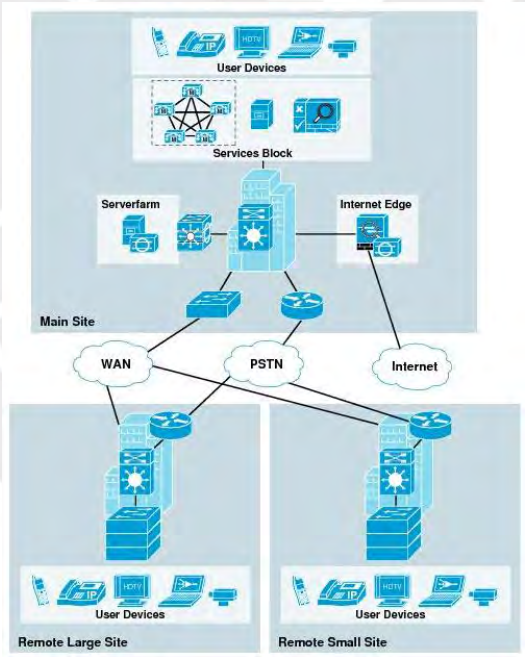


Figura 1-3 Recomendación de Cisco del diseño de una red para una pequeña empresa

Fuente:[5]

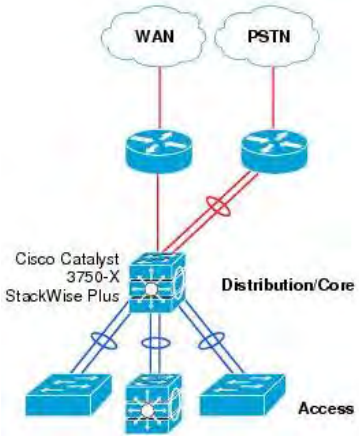


Figura 1- 4 Diseño propuesto de un "remote site"

Fuente:[5]

La arquitectura implementada, siguiendo el modelo propuesto, usa en la mayoría de los casos una estructura de dos niveles o también llamada de "core colapsado", donde se hace uso solamente de switches de acceso y distribución, y se evita el uso de switches core, esto se da principalmente para la reducción de costos y debido al tamaño mismo de la red.

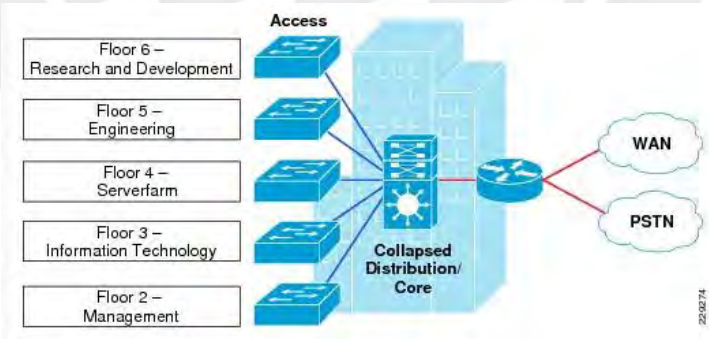


Figura 1- 5 Diseño propuesto e implementada en una misma ubicación

Fuente:[5]

1.1.3. Gestión de la red

Con el pasar de los años la tecnología en networking ha evolucionado con nuevos y mejores equipos, se han incrementado capacidades de procesamiento y han surgido nuevas tecnologías para satisfacer las crecientes demandas de los usuarios de la red. Una cosa que se ha mantenido largamente por muchos años de misma forma es la gestión de la red. Los operadores de la red aún se conectan a los routers, switches, balanceadores de carga y firewalls para realizar cambios de configuraciones a mano. Según el reporte de la empresa Red Hat en [6] existen numerosas razones por las que no se ha dado un cambio a pesar de los avances en el sector y de tener la tecnología para realizarlo:

- Frecuentemente, los equipos de NetOps están especializado en algunos pocos dominios y plataformas
- Proveedores(Vendors) están más enfocados en la venta de productos que en mejoras operacionales.
- La dependencia de interfaces de línea de comando(CLI) para los equipos de red impide la automatización
- Las plataformas propietarias monolíticas existentes carecen de capacidades para la automatización.
- Las prácticas operativas basadas en papel heredadas son difíciles de actualizar y cambiar.

En un estudio realizado por Gartner en 2018 [1] se revela que al menos un 70 % de sus clientes usan como método primario para realizar cambios en la red el uso del CLI en equipos individuales, seguido de lejos por el sistema de gestión de red de un vendor especifico con 10% y con un 8% está el uso de GUI también de forma individual en equipos.

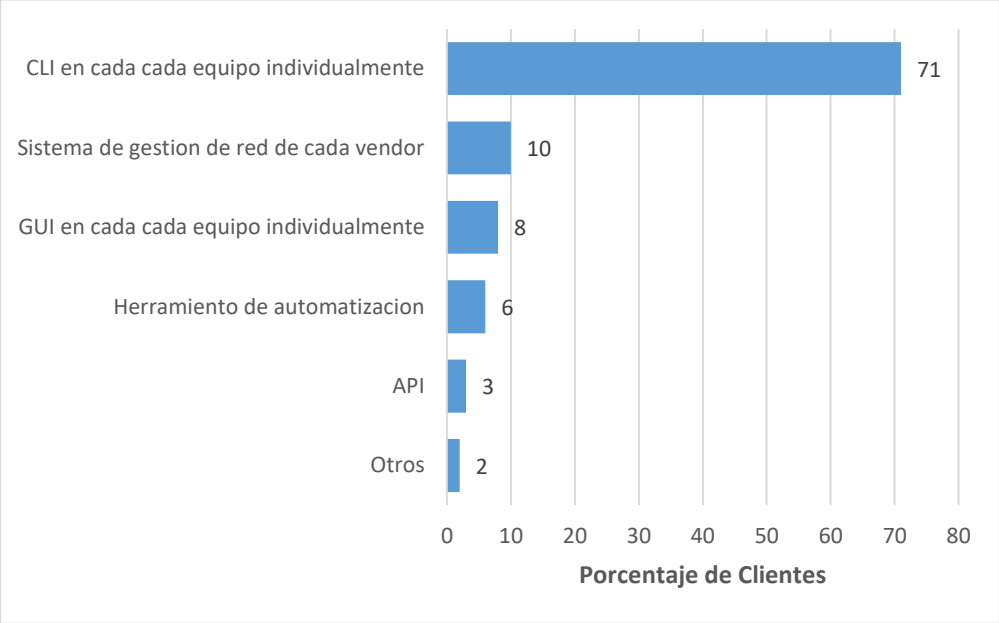


Figura 1- 6 Proporción de usuarios utilizando las distintas opciones de Gestionar una Red

Fuente:[1]

1.1.4. Beneficios de la Automatización de la red

La automatización de la red permite a los operadores de red configurar, escalar, asegurar e integrar de forma rápida la infraestructura de red y los servicios que existen sobre estos. Esta capacidad de realizar cambios de forma automática es muy importante para los proveedores de servicio debido al crecimiento acelerado de sus redes. Sin embargo, estos cambios en la red son cada vez más necesarios en todas las organizaciones, debido al dinamismo que posee la misma red en la actualidad. En consecuencia, la automatización permite a los operadores de red convertirse en activos agiles y flexibles que permiten cumplir con las demandas de la red actual. Tal como menciona en ACG Research [7] existen tres beneficios de alto impacto de la automatización de la red:

- Reducción de los costos de operación (OPEX).
- Disminución del tiempo para la realización de operaciones en la red.
- Velocidad de implementación de nuevos servicios en la red.

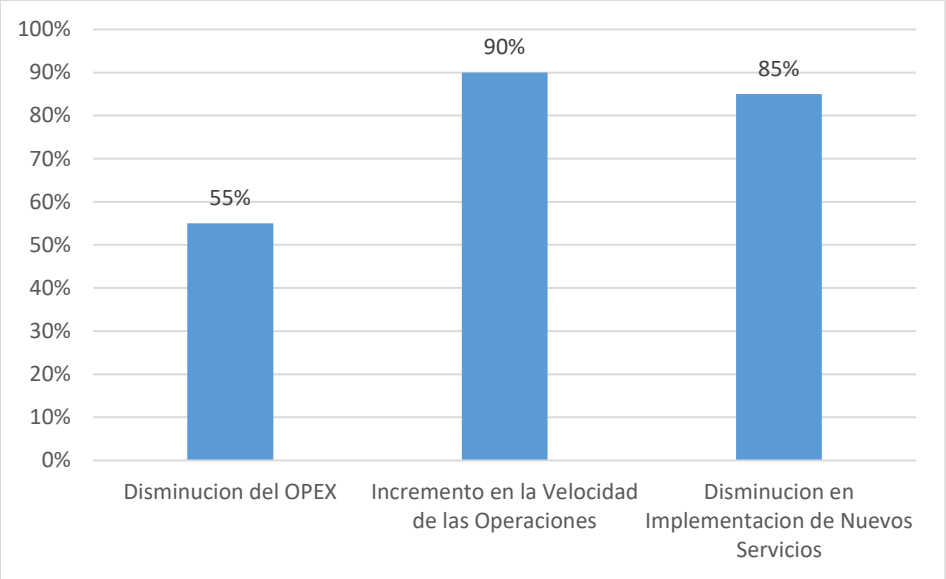


Figura 1- 7 Beneficios de la Automatización de Redes

Fuente: Elaboración propia basado en [7]

1.2. Objetivos de la tesis

1.2.1. Objetivos Generales

Es necesario el diseño de una alternativa a lo que se sigue utilizando desde hace muchos años hasta la actualidad. Se plantea el diseño de una red corporativa para una MYPE basada en el paradigma de una red basada en la intención (o IBN por sus siglas en inglés, *Intent-Based Network*), la cual no requiere que el operador de la red detalle ninguna línea de comando o tenga algún conocimiento técnico, basta que exprese una intención sobre cómo quiere que opere la red a muy alto nivel (lenguaje natural). La implementación de esta red será sobre una topología de SDN virtualizada en Mininet.

1.2.2. Objetivos específicos

Con la finalidad de poder realizar lo establecido anteriormente, se plantean los siguientes objetivos específicos para el desarrollo de la presente tesis:

- Establecer una comparación entre diversas implementaciones de redes basadas en la intención.
- Proponer un tipo de implementación basada en una aplicación con un algoritmo que interactua con una red SDN(virtualizada) y con una plataforma de comprensión de lenguaje natural basada en machine learning(ML).
- Validar el funcionamiento de la red basada en la intención mediante el ingreso de instrucciones básicas luego de implementada la topología.

1.3. Motivación respecto al problema enfocado

Las MYPES son de mucha importancia en la economía del Perú y de Latinoamérica, en su mayoría países emergentes. Según información oficial del Ministerio de Trabajo y Promoción del Empleo, las MYPES aportan en gran forma al sector económico del país con un valor cercano al 40 % del PBI nacional y con un 80% de la oferta laboral[8]. Caso similar sucede en los países emergentes, ya que estas empresas que son producto de la actividad emprendedora constituyen una de las principales fuerzas de crecimiento económico, desarrollo social y generación de empleo[9]. Tanto en el Perú como en otras economías la MYPE destaca por su flexibilidad que les permite adaptarse a cambios en el mercado y su contribución al mejoramiento de la competitividad. Por tal motivo, durante los últimos años y como resultado de la globalización, el uso cada vez mayor de nuevas tecnologías y la masiva interconexión se ha hecho cada vez más necesario que

estas empresas estén conectadas para incrementar su dinamismo, competitividad y accesibilidad en el escenario económico mundial.

La motivación para desarrollar la presente tesis es la de evidenciar que en la actualidad se tiene la tecnología suficiente para lograr la automatización de las redes utilizando plataformas de libre uso y software de código abierto. El uso de estas plataformas y software hace que sea una alternativa real para las micro y pequeñas empresas(MYPES), ya que se puede reducir los gastos en infraestructura y operación(OpEx), siendo esto importante, debido a que estas empresas cuentan con un limitado presupuesto. Este trabajo desea demostrar que se puede cambiar la forma tradicional de gestionar las redes basadas comúnmente en el uso del CLI y GUI, los cuales requieren de un operador con experiencia y conocimientos técnicos. Como consecuencia, la red es susceptible a una deficiente aplicación de políticas y errores de configuración.

Capítulo 2. Marco Teórico

En el presente capítulo se describe los conceptos necesarios para la realización de este trabajo, se presentarán conceptos, software y tecnologías a alto nivel.

2.1 Red Basada en la Intención(IBN)

El termino intención(*intent*) ha estado rondando por el mundo de las redes ya por algunos años. En el contexto de Redes Autonómicas, *intent* está definido como "una política abstracta de alto nivel utilizada para operar la red", según lo definido en [10]. Por otro lado, the Network Working Group en [11] define al *intent* como una ¨declaración de los objetivos operativos que una red debe cumplir sin que los usuarios se preocupen con configuraciones de bajo nivel de equipos de red o de especificar detalles técnicos¨. En otras palabras, un *intent* es un tipo de política declarativa de alto nivel que opera a la categoría de la red como un todo, no sobre equipos individuales. Se utiliza sin la necesidad de enumerar eventos, condiciones y acciones

específicas[11]. Sin embargo, para no confundirlos es necesario establecer una distinción clara. Las políticas son un conjunto de reglas que son usadas para administrar y controlar el estado de uno o varios recursos de red, son directrices que gobiernan como un sistema se debe comportar. Estas reglas pueden ser definidas para que sean condicionales, en el sentido de que, si una condición A es dada, entonces un grupo de políticas son implementadas en algunos dispositivos de red[12]. Por lo general, un evento o una serie de condiciones desencadenan una regla que fue predefinida, lo cual genera que una o más acciones se lleven a cabo mientras la condición se siga dando. Para poder tener una idea más clara entre las diferencias entre *intent* y política, se provee un cuadro comparativo entre estos en la tabla 2-1.

Tabla 2-1 Comparación entre el Intent y la Politica

Intent	Política
Conectar el area A con el area B	Deny all Inbound traffic from IP
Limitar en bw entre el area By C a 50 mbps	access-list 100 permit tcp any eq 80 10.10.10.0 0.0.0.255 (router)
Asignar la interfaz 3 a la vlan 10	permit tcp 192.168.1.33/32 0.0.0.0/32
Dejar abierto el puerto http solo a la ip 10.0.0.20	all of 172.17.0.0/16 except 172.17.1.0/24

Fuente: Elaboración propia

El año 2017 Gartner en [13] acuño el termino de Red-Basada en la Intención o (IBN, por sus siglas en inglés) llamándolo la próxima gran avance en el mundo de las redes. Este nuevo paradigma es descrito como un "pieza de software de redes que ayuda a planificar, diseñar, implementar y operar las redes generando una mejora en la disponibilidad y la agilidad de estas"[13]. Según Gartner el sistema de una IBN debe incorporar cuatro características claves:

- Traducción y Validación: El sistema toma una intención (que) de los usuarios finales y la convierte en la configuración de red necesaria (cómo).
- Implementación automatizada: El sistema puede configurar los cambios de red apropiados (cómo) en la infraestructura de red existente de forma automatizada.

- Conciencia del estado de la red: El sistema conoce el estado de la red en tiempo real para los sistemas bajo su control administrativo.
- Garantía y optimización / corrección dinámica: El sistema valida continuamente que se está cumpliendo la intención original y puede tomar acciones correctivas (como bloquear el tráfico, modificar la capacidad de la red o notificar) cuando no se cumple la intención deseada.

La red basada en la intención (IBN) tiene como objetivo principal la especificación de objetivos en alto nivel en forma declarativa sin tener conocimiento de los detalles subyacentes de implementación[14]. La concepción de realizar cambios en la red mediante la configuración de dispositivo por dispositivo ya no es sostenible debido a las transformaciones que se vienen dando en la red en los últimos años. Los cambios en la red actual requieren muchas veces que sean de forma automática e inmediata con un alto nivel de seguridad de que son los adecuados.

Una red basada en la intención se basa principalmente, como se mencionó anteriormente, en la "Intención" (o "Intent" por su traducción en inglés), la cual se refiere al deseo o declaración de objetivos de operación que la red debe alcanzar sin que se especifique como lograrlo. Existe un consenso en que la intensión debe ser expresada al más alto nivel posible, es decir, los usuarios u operadores de la red no se deben preocupar por tener conocimientos previos técnicos sobre la configuración de dispositivos[11] o de la red. Los usuarios y operadores no se deben preocupar sobre el cómo se debe realizar la intención expresada, basta con especificar qué es lo que quiere lograr y el sistema transforma las intenciones en políticas de bajo nivel que los dispositivos pueden entender y transformar en acciones o configuraciones.

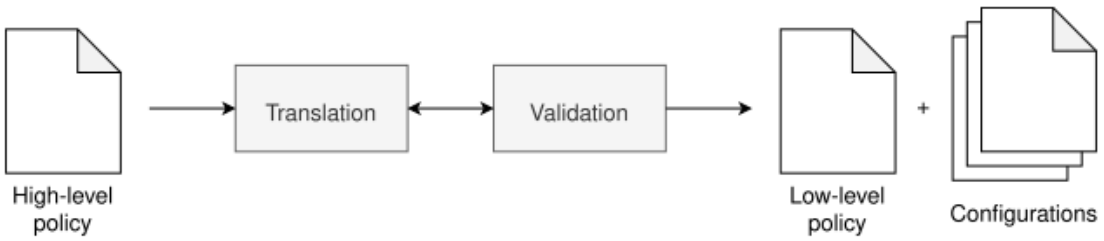


Figura 2-1 Proceso de una Intención

Fuente:[14]

Existen diversas aproximaciones sobre la realización de una red basada en la intención (IBN) sin llegar a un acuerdo general. Una de estas concepciones es la utilización de programabilidad en la red brindado por SDN para poder realizar los cambios deseados. Este método es el que se utilizara para la presente tesis para lograr desarrollar una red basada en la intención(IBN). Esta propuesta consiste en que el usuario establece una intención la cual es interpretada por una segunda capa llamada “Intent Engine”. La siguiente capa de esta jerarquía consume la intención y la transforma en acciones deseadas basándose en lo que se desea en la intención. Finalmente, en la última capa los dispositivos de la red realizan la configuración y acciones dirigidos a ellos por el “Intent Engine”[12]. Lo mencionado puede ser observado en la siguiente figura:

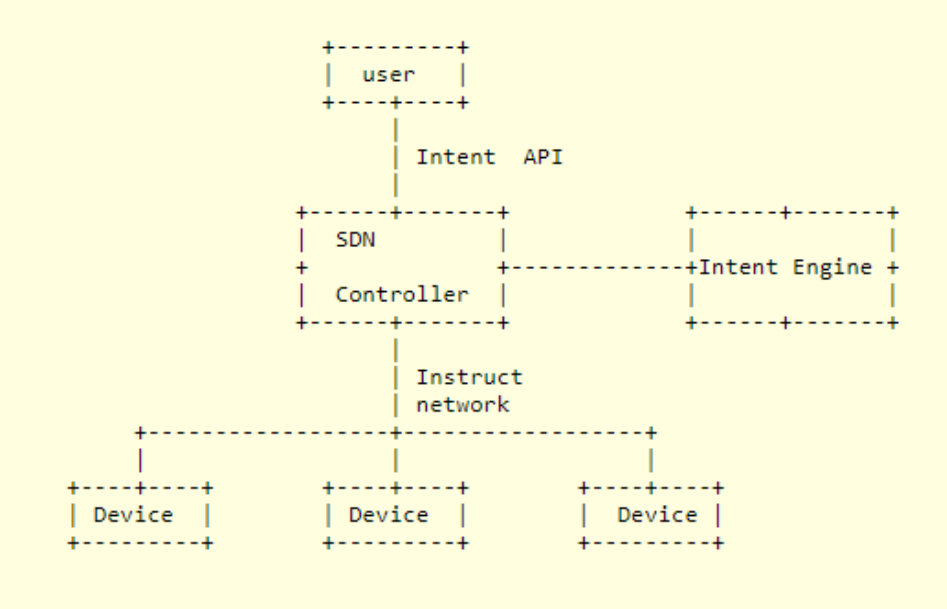


Figura 2-2 Jerarquía lógica en la gestión de una IBN basada en SDN

Fuente:[12]

Como se mencionó anteriormente, la intención tiene que ser expresada de forma general sin llegar a detallar acciones específicas. Teniendo en cuenta lo recién expuesto, existen algunas consideraciones a tomar en cuenta al momento de diseñar una red basada en la intención (IBN):

- Primero, puede haber muchas alternativas para satisfacer una Intención especifica en la red.
- Segundo, existe la posibilidad de no realización de ciertas Intenciones debido a la no disponibilidad de dispositivos de red o a la limitada capacidad de funcionalidad que la red posea.
- Tercero, algunas Intenciones pueden ocasionar conflictos con el estado actual de la red o pueden afectar Intenciones satisfechas previamente.

Actualmente, las plataformas y entornos de trabajo que proporcionan redes basadas en la intención se centran en el ámbito académico y no en la industria. Por tal motivo, no se avanzado en una estandarización a pesar de la introducción y desarrollo de SDN, Openflow y avances en inteligencia artificial especialmente en comprensión y procesamiento de lenguaje natural. Organizaciones como 3GPP (3rd Generation Partnership Project), ETSI (European Telecommunications Standards Institute), ONF (Open Networking Foundation) y la ITU (International Telecommunication Union) han desarrollado independientemente grupos de estudio referentes al tema buscando establecer un punto de partida hacia la normalización de IBN.

El primer esfuerzo fue realizado por ONF en 2016 cuya principal idea era crear una herramienta que pueda manipular la interfaz NorthBound (NBI) de intención que esté integrado o sea externo al controlador de red[15]. Si bien, la acuñación del termino es reciente como se menciona en [13], ya han habido varias implementaciones como la realizada en [16] presentada en el IEEE Netsoft 2019 donde el enfoque que realizan es la utilización del lenguaje de programación de redes P4 para instalar y remover programas de estos. Para la realización de lo mencionado vieron necesario describir un *Intent Definition Language* (IDL), crear un repositorio de plantillas en P4 basándose en los *intents,* proporcionar una técnica para realizar el programa P4 resultante en un switch programable, al tiempo que se acomodan las modificaciones de intención en cualquier momento y, finalmente, implementar una prueba de concepto para demostrar que las modificaciones de la intención se pueden realizar sobre la marcha. Por otro lado, otra implementación es la realizada en [17] también presentada el año 2019 propone un sistema de red basada en la intención acoplada con un asistente de voz capaz de realizar modificaciones en una red SDN. La propuesta utiliza un asistente de voz se basa en la utilización del servicio de voz Alexa de Amazon, además, realizan uso de un *intent engine* en python y el controlador Floodlight

encargado de configurar los OVS swithes usando openflow. Todos los componentes de esta implementación fueron virtualizados en equipos físicos utilizando el hipervisor VMware ESXi.

Asimismo, existe proyectos como el descrito en [18] donde proponen la creación de un framework de programación de red basado en SDN llamado OSDF(Open Software Defined Framework). Este marco provee una API a alto nivel donde se especifica los requisitos de red para aplicaciones y políticas para múltiples dominios. OSDF está implementado en onos donde hace uso de las APIs de topología y *flow rule* de este sistema operativo.

2.2 Redes definidas por Software (SDN)

Durante ya varios años surgió un paradigma diferente para lograr la programabilidad de la red. Como mencionan en [19], uno de los conceptos de SDN es tratar a la red como una todo en lugar de un conjunto de cajas independientes, ya que posee una entidad con lógica centralizada con una percepción global de la red Este nuevo enfoque proponía un cambio sustancial en cómo opera de la red la cual se ha mantenido de la misma forma hasta el día de hoy. El cambio propuesto se basa en la separación del plano de datos (impulsa el tráfico de red de acuerdo a las decisiones del plano de control) y el plano de control (que decide cómo manejar el tráfico de red) removiendo la integración vertical dentro de un dispositivo de red. Como resultado, se logra obtener el control global de la red como un todo mediante el operador central llamado Controlador. Este es un cambio total con respecto a la red tradicional la cual con el pasar de los años se ha incrementado en escala, se ha vuelto más compleja y difícil de operar.

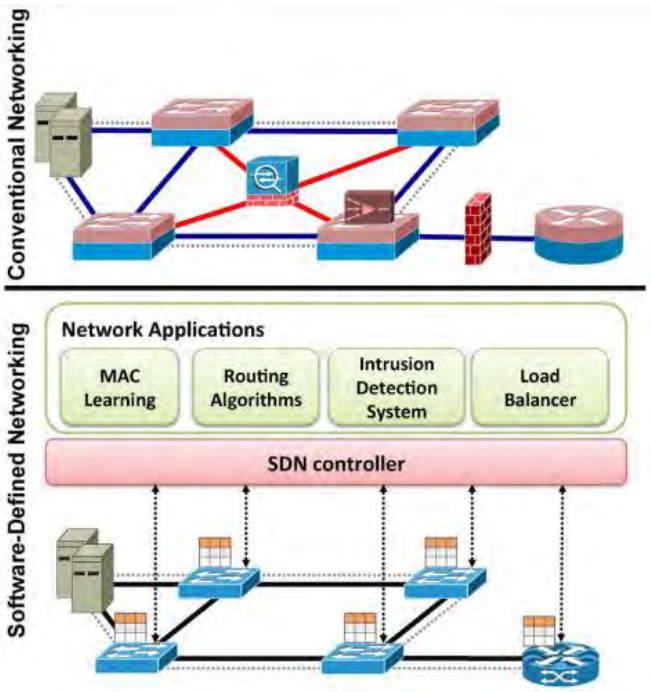


Figura 2-3 Contraste entre una red tradicional y una red SDN

Fuente:[20]

Tabla 2-2Comparación entre Red tradicional, SDN y IBN

Criterio	Red Tradicional	Red SDN	Red Basada en la Intención
Gestión de red	Difícil, Necesita ser implementados de forma individual	Hay una mejora con la ayuda del controlador	Fácil, la red lo realiza sin intervención de algún usuario
Vista global de la red	Difícil	Vista centralizada en el controlador	Se conoce el estado de la red en tiempo real
Costo mantenimiento	Alto	Bajo	Bajo
Tiempo requerido para actualización / manejo de errores	Puede Tomar meses	Se reduce el tiempo debido al uso del controlador central	Reducción mayor al de una red SDN
Utilización de recursos	Poco	Alto	Alto
Conocimientos técnicos de gestión de red	Importante	Es necesario conocer el funcionamiento del controlador	No es necesario

Fuente: Elaboración propia

2.2.1 Arquitectura SDN

La arquitectura SDN en forma general está compuesta de tres componentes: Capa de Aplicación, Capa de control y Capa de datos. Cada una de estos componentes cumplen un rol fundamental en el funcionamiento de una red SDN; en primer lugar, la capa de aplicación es el lugar donde programas y/o aplicaciones transmiten las acciones deseadas al controlador SDN a través de interfaces NBI (NorthBound Interface). La cantidad de interfaces NBI es relativa a la cantidad de aplicaciones SDN que se está utilizando con un controlador, por este motivo puede haber muchas de estas interfaces. En segundo lugar, en la capa de control encontramos al controlador SDN, el cual se encarga de traducir los requerimientos entregados por la capa de aplicación en instrucciones de configuración que deben ser aplicadas en los dispositivos de red pertenecientes a la capa de infraestructura mediante interfaces SBI (SouthBound Interface) por lo general, switches. Finalmente, en la capa de datos encontramos a los dispositivos de red que son los encargados de direccionar el trafico según las instrucciones brindadas por el controlador.

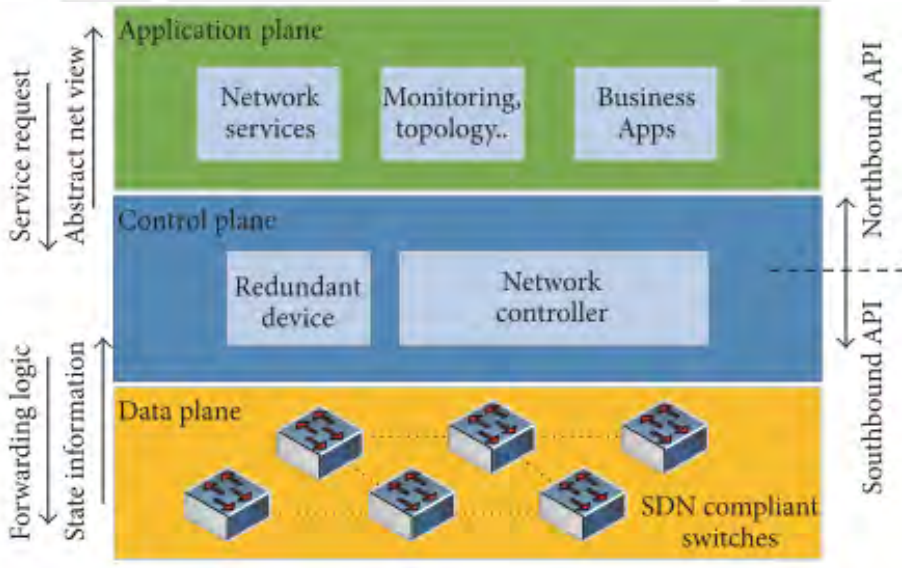


Figura 2-4 Arquitectura SDN

Fuente:[21]

2.2.2 Controlador SDN

El controlador reside en el plano de control de la arquitectura SDN, este es responsable de dar al plano de datos con información sobre su plano de control, es decir, los switches reciben instrucciones del controlador para poder realizar direccionamiento del tráfico. El controlador es el lugar donde toda la lógica es realizada, aquí se tiene el control sobre switching, routing, reglas de seguridad de firewall, clustering, etc.[22]. Este dispositivo coordina las APIs del NorthBound y SouthBound, y permite que las aplicaciones realizar métodos de devolución de llamadas (callback methods, en inglés) cada vez que una intención es actualizada o modificada. Actualmente existen dos controladores SDN de código abierto importantes, ONOS y OpenDaylight, que poseen cada uno entornos de trabajo (framework, en inglés) dedicado a la implementación de las intenciones además poseen características similares las cuales podemos observar en la tabla 2-3.

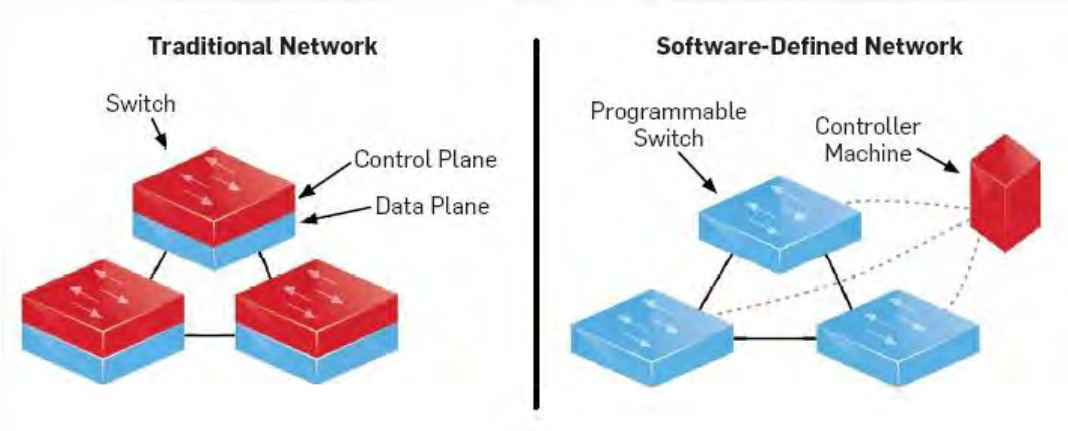


Figura 2-5 Operación del controlador SDN

Fuente: [23]

Tabla 2-3Comparación entre Controladores Considerados

Criterio	OpenDayLight	ONOS
Concepto	Proyecto de código abierto para adopción de SDN y NFV	SO de código abierto SDN
Características	Escalable, Alta disponibilidad, modular	Escalable, Alta disponibilidad y performance
Soporta tecnologías	Openflow, NETCONF,OVSDB, OpFLex	Openflow, NETCONF, OVSDB
Desarrollo	Escrito en Java	Escrito en Java
Importancia	Soporta variedad de protocolos SBI	Soporta variedad de protocolos SBI
Requerimientos del sistema	Mínimos: CPU: 2 Cores RAM: 2 GB Almacenamiento: 16 GB Recomendado: CPU: 8 Cores RAM: 8 GB Almacenamiento: 64 GB	CPU :2 core RAM : 2 GB Almacenamiento =10 GB 1 NIC (any speed)

Fuente: Elaboración propia

2.2.2.1 *OpenDayLight*

OpenDaylight (ODL) es un proyecto colaborativo open source fundado por *The Linux Foundation*(NF). El proyecto sirve como plataforma para SDN para monitoreo de equipamiento de red de forma centralizada. Su objetivo es facilitar la implantación de las nuevas tecnologías SDN y Network *Functions Virtualization* (NFV) en las redes actuales.

Este proyecto está escrito en Java, posee una infraestructura modular, extensible, escalable y ofrece alta disponibilidad. El soporte de múltiples protocolos *southbound* como openflow. Netconf, BGP y ovsdb lo hace muy atractivo para su uso y desarrollo de aplicaciones. Posee una

característica resaltante donde hace uso del lenguaje YANG para el modelado de datos, además de la REST API para la comunicación con distintas aplicaciones.

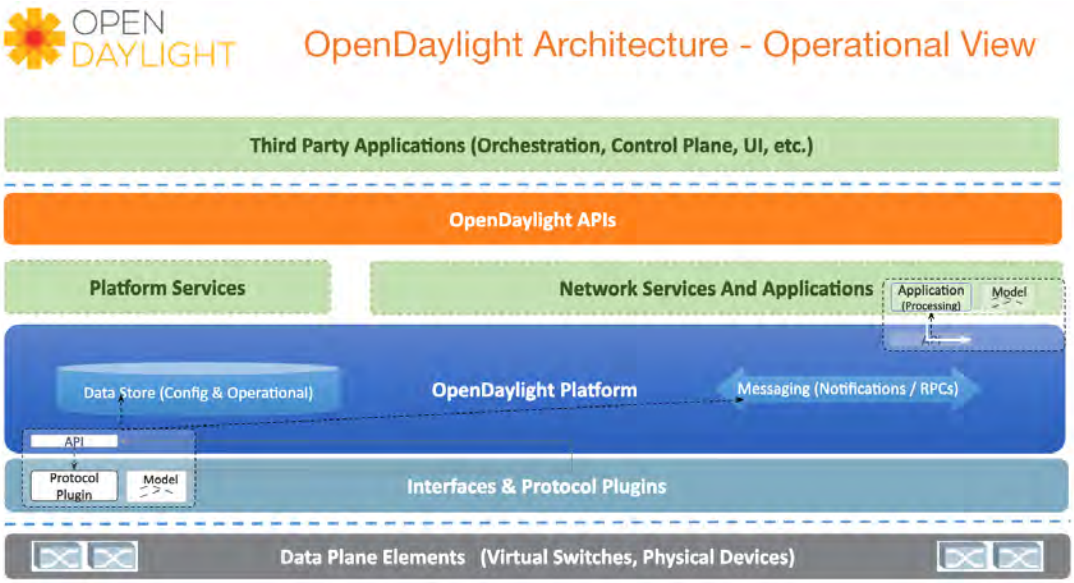


Figura 2-6 Estructura del controlador OpenDaylight

Fuente:[24]

2.2.2.2 *Controlador ONOS*

El controlador ONOS es un controlador SDN de código abierto que fue desarrollado por la Open Network Foundation (ONF) que provee de alto rendimiento, escalabilidad y disponibilidad. ONOS está distribuido bajo la licencia Apache 2.0 que utiliza un lenguaje de programación basado en Java y utiliza Apache Karav para dar vida a todas las funciones de ONOS[25]. Para mantener el estado de la red, interactua con dispositivos de red mediante los SouthBound APIs y ofrece servicios a aplicaciones mediante los NorthBound APIs..

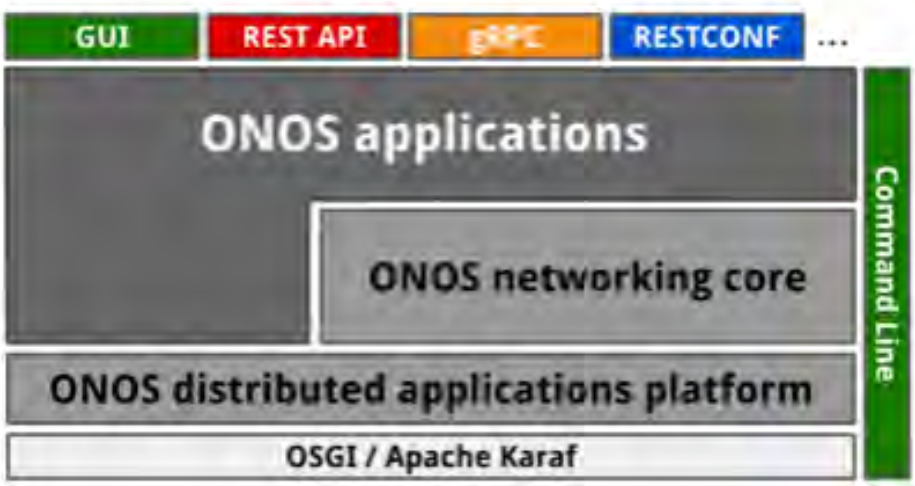


Figura 2-7 Estructura del controlador ONOS a alto nivel

Fuente:[26]

ONOS brinda muchas capacidades y funcionalidades como el uso de APIs y abstracciones, además de software para la interacción con los usuarios atraves de CLI, GUI y REST API, siendo esta la que tendrá una gran importancia en la presente tesis. La REST API permite instalar diferentes tipos de intención que ya fueron implementadas previamente en el controlador.Otras de las capacidades de ONOS es la de administrar toda la red no solo algunos dispositivos llegando a simplificar la gestión, configuración y despliegue de nuevo software, hardware y servicios.[26]

2.2.2.2.1 *Intent Framework*

Este subsistema de ONOS permite que aplicaciones especifiquen las necesidades que desean que la red cumpla. Las especificaciones dadas por las aplicaciones son transformadas mediante el uso de compilación en intenciones instalables

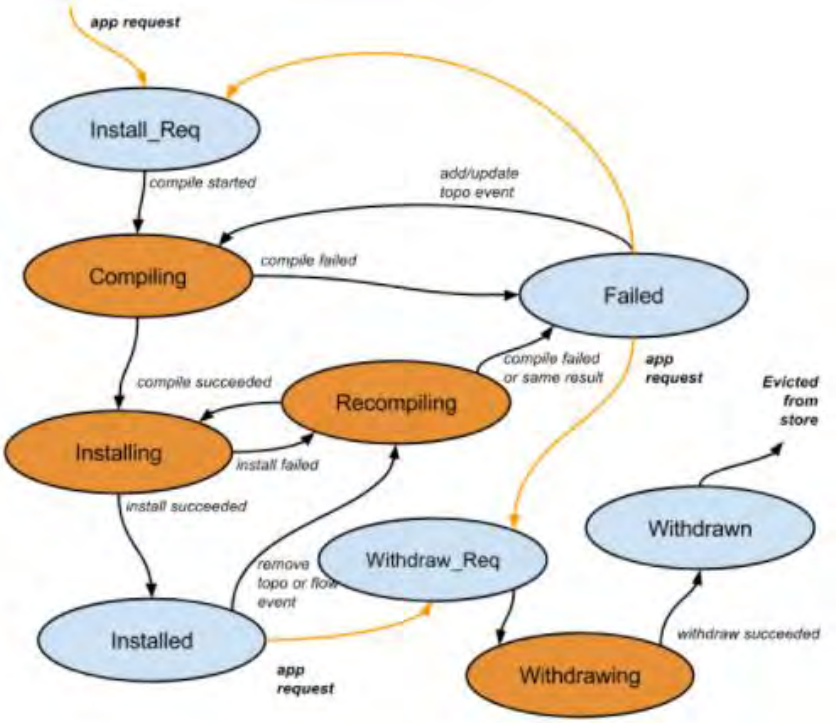


Figura 2-8 Procesamiento de una Intención según el framework

Fuente:[27]

Luego que una intención es introducida, se enviara a la fase de compilación, después a la fase de instalación, y finalmente, al llegar al estado de instalado[28]. Una de las características interesantes de este entorno de trabajo es la capacidad de cumplir los objetivos expresados en la intención a pesar de que surjan fallas en la red. Entre las intenciones más importantes implementadas en ONOS se tiene las siguientes:

- Intención “Host to Host” (permite realizar conexiones entre dos dispositivos).
- Intención “Point to Point”, “Single Point to Multi Point” y “Multi Point to Single Point” (conexiones entre distintos puntos de la red).
- Intención “Protected” (permite conectividad con un camino primario y un camino de respaldo en caso falle el primer camino).

Dentro del controlador se puede acceder a la lista de instaladores y compiladores mediante el CLI. En la figura 2-9 se puede observar la presencia de los compiladores mencionados previamente *HostToHostIntent, PointToPointIntent* así como muchos otros más. Cabe mencionar que el *intent framework* aún está en desarrollo y no todos satisfacen lo que se quiere realizar en este trabajo de tesis.

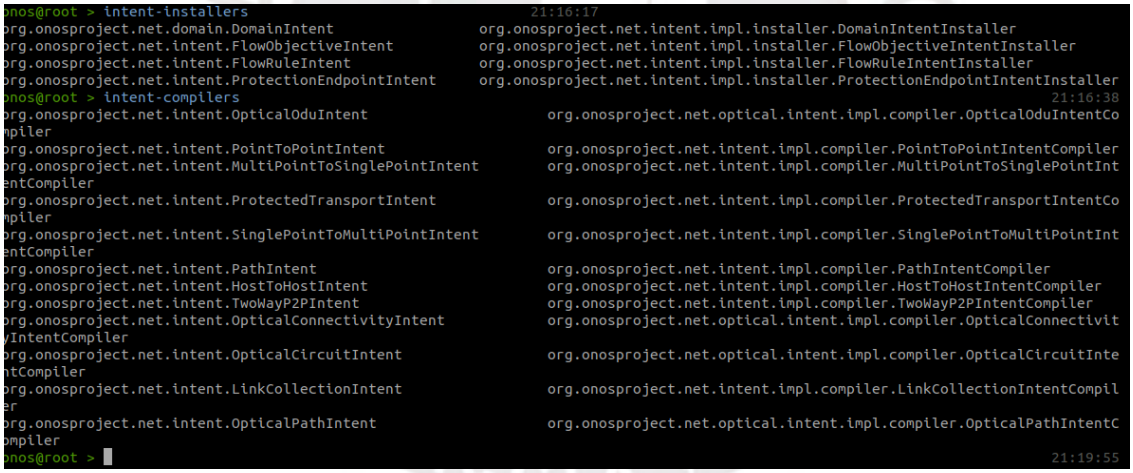


Figura 2-9 Onos Intent framework en el controlador

Fuente: Elaboración propia

2.2.3 REST API

Este tipo arquitectura fue desarrollado por el W3C (*World Wide Web Consortium*), tiene como objetivo ofrecer escalabilidad, generalidad e independencia, y permite la inclusión de componentes intermedios entre un modelo cliente y servidor[21]. REST utiliza el protocolo HTTP para el desarrollo de APIs permitiendo obtención de información o ejecución de operaciones sobre los datos utilizando los formatos XML, JSON o texto plano. Posee operaciones bien definidas como POST, GET, PUT y DELETE, las cuales son utilizadas para acceder y/o modificar información de los recursos disponibles a través de consultas a una URI.

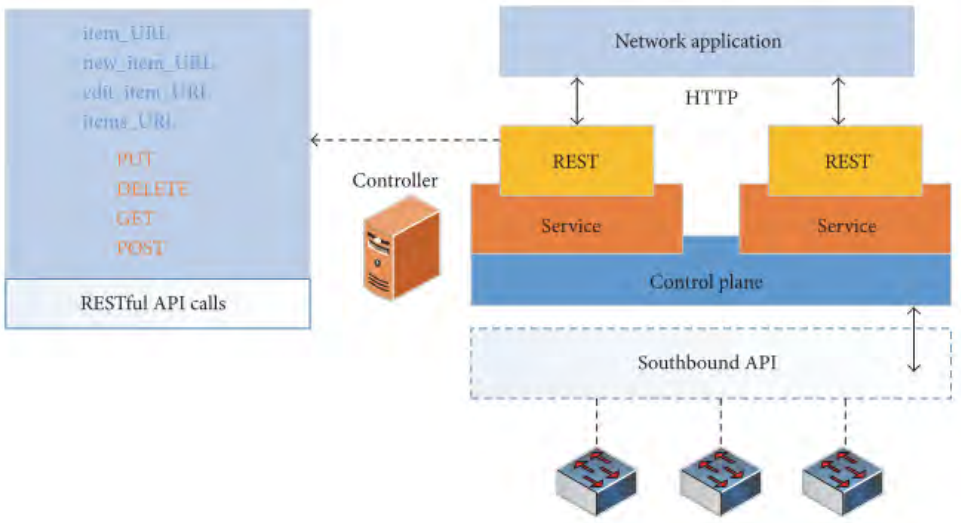


Figura 2-10 Operación de la REST API

Fuente:[21]

Una de las diversas formas de interacción de ONOS con los usuarios es mediante el uso de la REST API. ONOS cuenta con diversas entidades que soportan las operaciones ya mencionadas para la obtención y modificación de datos sobre la red y sus dispositivos como se puede observar en la siguiente tabla.

Tabla 2-4 Operaciones Soportadas por las Entidades de la REST API de ONOS

Entidades	GET	POST	PUT	DELETE
Device/Dispositivos	X	X	X	X
Link/Enlaces	X	X	X	X
Host/PCs	X	X	X	X
Topology/Topología	X			
Path/Camino	X			
Flow/Flujo	X	X		X
Flow Objectives /Objetivos del Flujo	X	X		
Group/Grupo	X	X		X
Meter/Medidor	X	X		X
Intent/Intención	X	X		
Application/Aplicación	X	X		

Fuente: [29]

La documentación sobre el REST API puede ser accedida desde cualquier instancia ejecutada de ONOS. La información desplegada muestra las operaciones soportadas por cada entidad de la tabla anterior y sus URIs correspondientes además de la estructura de la data en formato JSON como se observa en la siguiente figura.

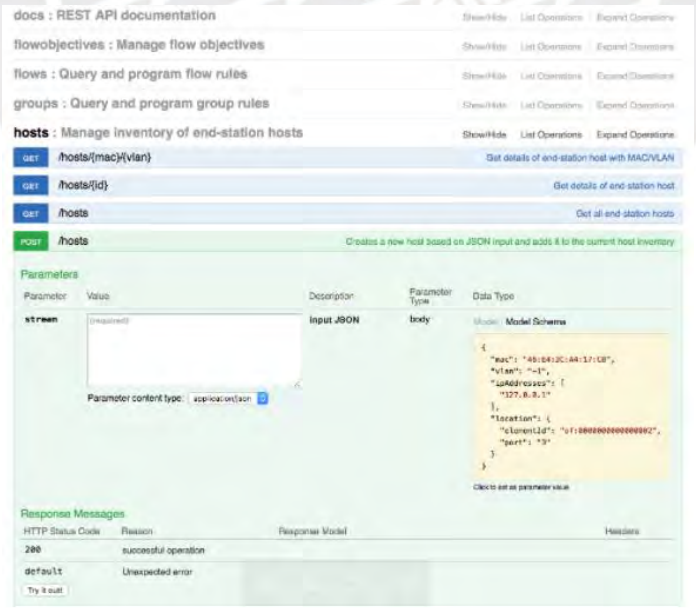
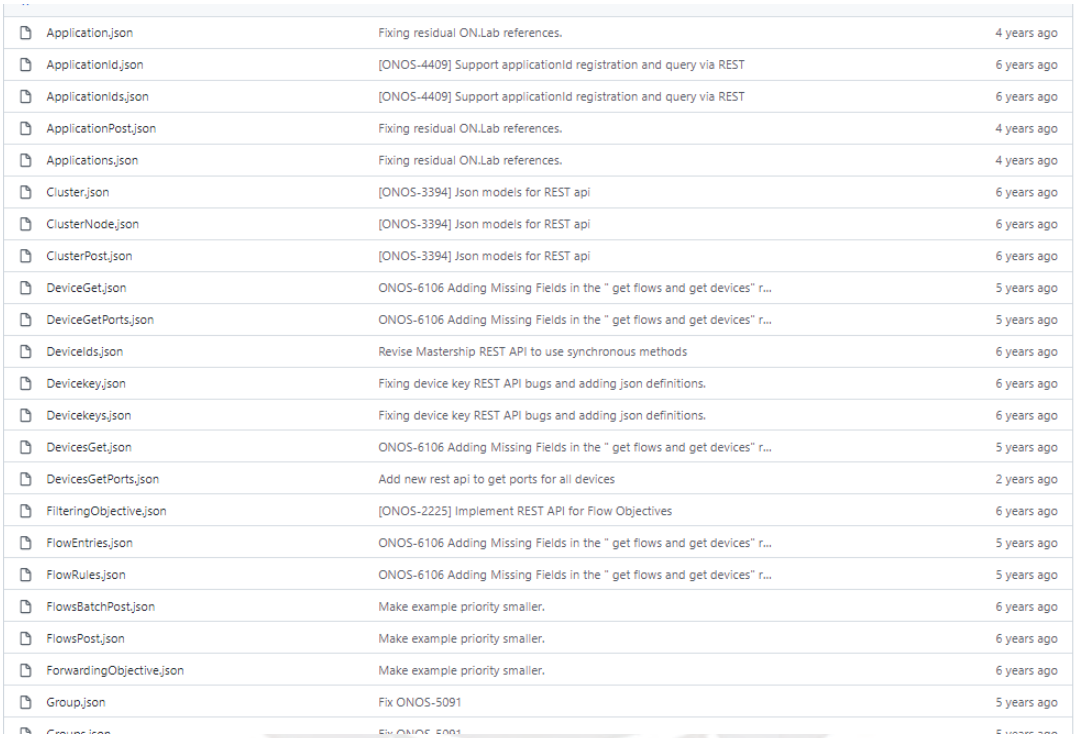


Figura 2-11 Documentación REST API de ONOS desplegada en la instancia ejecutada

Fuente:[29]

A pesar de que en la tabla 2-4[29] se listan cierta cantidad de entidades que pueden ser accedidas mediante la REST API para su configuración, es posible tener una mejor concepción de las capacidades con las que se cuenta accediendo a la documentación del controlador en cuestión observada en la figura 2-11 (los métodos soportados pueden cambiar según la versión de controlador que se esté utilizando) o accediendo al repositorio en GitHub [30].



Application.json	Fixing residual ON.Lab references.	4 years ago
ApplicationId.json	[ONOS-4409] Support applicationId registration and query via REST	6 years ago
ApplicationIds.json	[ONOS-4409] Support applicationId registration and query via REST	6 years ago
ApplicationPost.json	Fixing residual ON.Lab references.	4 years ago
Applications.json	Fixing residual ON.Lab references.	4 years ago
Cluster.json	[ONOS-3394] Json models for REST api	6 years ago
ClusterNode.json	[ONOS-3394] Json models for REST api	6 years ago
ClusterPost.json	[ONOS-3394] Json models for REST api	6 years ago
DeviceGet.json	ONOS-6106 Adding Missing Fields in the " get flows and get devices" r...	5 years ago
DeviceGetPorts.json	ONOS-6106 Adding Missing Fields in the " get flows and get devices" r...	5 years ago
DeviceIds.json	Revise Mastership REST API to use synchronous methods	6 years ago
Devicekey.json	Fixing device key REST API bugs and adding json definitions.	6 years ago
Devicekeys.json	Fixing device key REST API bugs and adding json definitions.	6 years ago
DevicesGet.json	ONOS-6106 Adding Missing Fields in the " get flows and get devices" r...	5 years ago
DevicesGetPorts.json	Add new rest api to get ports for all devices	2 years ago
FilteringObjective.json	[ONOS-2225] Implement REST API for Flow Objectives	6 years ago
FlowEntries.json	ONOS-6106 Adding Missing Fields in the " get flows and get devices" r...	5 years ago
FlowRules.json	ONOS-6106 Adding Missing Fields in the " get flows and get devices" r...	5 years ago
FlowsBatchPost.json	Make example priority smaller.	6 years ago
FlowsPost.json	Make example priority smaller.	6 years ago
ForwardingObjective.json	Make example priority smaller.	6 years ago
Group.json	Fix ONOS-5091	5 years ago

Figura 2-12 Repositorio de procesos de para REST API

Fuente: [30]

2.2.4 Mininet

Este emulador de red nos permite implementaciones de redes virtuales compuestas de hosts, switches, controladores y enlaces habilitando la capacidad de desarrollo SDN en cualquier laptop o PC. Debido a que los switches virtualizados dentro de Mininet como el OVS soportan OpenFlow, es una perfecta herramienta para el desarrollo, puesta en prueba y ensayos para redes SDN. Toda interacción con Mininet se realiza a través de su CLI, con simples comandos se puede desplegar redes conectadas con controladores externos (como ONOS) y la creación de topologías personalizadas haciendo uso de una API de Python.

2.3 Comprensión del Lenguaje Natural

La Comprensión del Lenguaje Natural (NLU), por sus siglas en inglés, es un sub-campo del procesamiento del lenguaje natural en inteligencia artificial(IA) que se centra en la habilidad que tiene una máquina de aprender, comprender y producir el lenguaje humano. Existen diversas plataformas basadas en la nube desarrolladas por las principales empresas de TI como Google, Facebook, IBM, Microsoft y Apple. De estas empresas Apple con Siri, Google con su Asistente, Amazon con Alexa y Microsoft con Cortana son las que son más populares ya que sus plataformas de comprensión de lenguaje son pre-entrenadas para los usuarios finales [31]y ofrecidas como producto final. Por otro lado, se tiene plataformas de NLU basadas en la nube desarrolladas también por las empresas mencionadas impulsadas por algoritmos de machine learning(ML), que pueden ser personalizadas y entrenadas por desarrolladores para poder integrarlos en proyectos y diversas aplicaciones. Las características de las plataformas más reconocidas, usadas por la industria y evaluadas para su uso en la presente tesis están descritas en la tabla 2-5, la cual se realizó basándose en experiencia propia y de información obtenida de [32] y [33].

Tabla 2-5 Plataformas de comprensión de lenguaje natural evaluadas

Aspectos	Dialogflow	IBM Watson
Canales	Voz y Texto	Voz y Texto
Curva de aprendizaje	Rápido de aprender gracias a tutoriales	No tan rápido de aprender
Facilidad de uso	Posee un UI que facilita la creación de chatbots	Posee un UI fácil de navegar
Integraciones	Diversas integraciones con un solo click en varios canales de mensajería	Posee integraciones menos desarrolladas
Integración Web	Posee facilidad de integración sin mucho uso de código mediante webhook	Posee un UI básico para páginas web
Lenguajes	Gestión incorporada de diálogos multi-idioma	Etapa temprana de implementación de idiomas
Costo	Gratis plan standard(usado)	Gratis con restricciones
Uso	Recomendable para diálogos y chatbots sencillos	Recomendable diálogos y chatbots complejos

Fuente: Elaboración propia[32]

2.3.1 DialogFlow

DialogFlow es una plataforma basada en la nube de libre uso perteneciente a Google que posee integración con diferentes idiomas y lenguajes de programación. Al igual que las otras plataformas de comprensión de lenguaje natural(CLN) tiene como base dos conceptos: Intents y Entidad.

2.3.1.1 *Intents*

El *intent* en Dialogflow sirven para referirse a la intención del usuario u operador en un turno de conversación. Cuando se procede a combinar diversos *intents* se puede manejar una conversación completa. Se busca que la intención expresada por el usuario en esta plataforma coincida con *intent* previamente ingresado[8]. Un *intent* posee los siguientes elementos:

- Frases de entrenamiento
- Acción
- Parámetros
- Respuestas

2.3.1.2 *Entidad*

La entidad permite extraer valores de los parámetros del lenguaje natural ingresado[31]. En la expresión "Que hora es en Lima" el intento se mapea a una pregunta por la hora y Lima seria la entidad Ciudad de la expresión. DialogFlow posee entidades predefinidas para los tipos comunes de datos como fechas, horas, emails, números, etc; sin embargo, existe la posibilidad de crear tus propias entidades personalizadas según los requerimientos del usuario.

2.3.1.3 *Contextos*

Los contextos en DialogFlow nos permiten referirnos a entidades previamente mencionadas y relacionarla con otra *intent* lo cual genera que se dé un comportamiento más cercano a una conversación en lenguaje natural.

Por otro lado, DialogFlow tiene la capacidad de integrarse con otras plataformas mediante el uso de APIs, o hacer usos de sus capacidades e interactuar con otras aplicaciones mediante el uso del mecanismo de Webhook para la obtención de las entidades procesadas por la plataforma.

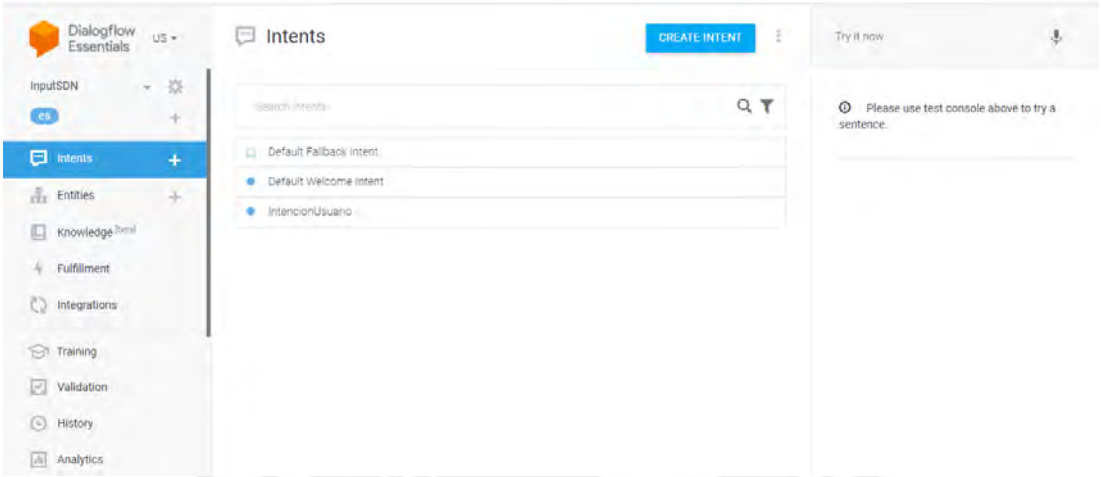


Figura 2-13 GUI de DialogFlow

Fuente: Elaboración propia

2.4 MYPE

Existe una clasificación de empresas según su tamaño dentro de las cuales está la micro y pequeña empresa (MYPE), que según la legislación peruana[34], son organizaciones que tienen como objetivo desarrollar actividades de extracción, transformación, producción, comercialización de bienes o prestación de servicios. Las diferencias entre la micro y pequeña empresa se detallan en la tabla 2-6. De acuerdo con las últimas cifras de la Encuesta Nacional de Hogares (Enaho), elaborada por el Instituto Nacional de Estadística e Informática (INEI), en 2019, las MYPES representan el 95% de las empresas peruanas[35].

Tabla 2-6 Características de las MYPE

Tipo de Empresa	Número de Trabajadores	Ventas anuales
Microempresa	< 10	máx. 150 UIT
Pequeña Empresa	< 100	máx. 1700 UIT

Fuente:[36]

En el mundo empresarial las TICs juegan un papel muy importante en el desarrollo de todo tipo de empresas. En contraste con las MYPEs, en el resto de empresas la necesidad de trabajar con las TICs ya es una realidad, especialmente en las medianas y pequeñas empresas ya que aportan mucho y brindan grandes beneficios a estas. Otra diferencia muy marcada e importante en el empleo de las TICS y por qué estaban fuera del alcance de las MYPEs eran el gasto que se debía realizar para su uso. Gracias a las nuevas tecnologías y los servicios disponibles de forma gratuita, ya no es necesario que se adquieran software y hardware que era lo necesario para implementar plataformas de TICs. Ahora es posible utilizar herramientas gratuitas de desde cualquier dispositivo que cuente con conexión a internet y poder acceder a la información requerida. Esto hace que las TICs se vuelvan cada vez más atractivas para las MYPEs.

Capítulo 3. Diseño de una Red IBNM para una MYPE del Sector Servicios

En el desarrollo de este capítulo se describen las características, consideraciones y alcances del diseño propuesto que constata la gestión de una red SDN corporativa mediante el paradigma IBNM.

3.1. Entorno de trabajo

La propuesta de solución consta de varias etapas implementadas en un entorno local, a excepción del procesamiento de comprensión del lenguaje natural, el cual es realizado en una plataforma de la nube. En la siguiente tabla se detalla las características del servidor local donde se realizaron la implementación del resto de etapas de esta propuesta de solución.

Tabla 3-1 Configuración del servidor local

Componente	Descripción
CPU	Intel® Core™ i7-4500U CPU 1.80GHz
RAM	6 GB
Red	100 Mbps
SO(Versión)	Ubuntu 18.04 LTS
Almacenamiento	20 GB

Fuente: Elaboración propia

3.2. Arquitectura de la Red Basada en la Intención(IBN)

Como se mencionó en el capítulo anterior, específicamente en la sección 2.1, en la actualidad aún no existe una estandarización sobre cómo debería ser estar compuesta de una red basada en la intención. Por tal motivo, para el diseño de la presente Tesis se propone desarrollar una implementación de una red gestionada según el paradigma IBNM que se componga de diversas etapas, una de las cuales será el desarrollo de un *Intent engine*. El total de etapas pueden ser observadas en un gráfico de alto nivel en la figura 3-1. Las etapas mencionadas en la solución propuesta son las siguientes:

- Selección y uso de la plataforma de comprensión de lenguaje natural
- Procesamiento y traslado de la intención a instrucciones para el controlador (*Intent engine*).
- Acciones en el controlador SDN seleccionado.
- Despliegue de una red virtualizada

Figura 3-1 Diseño de red basada en la intención(IBN)

Fuente: Elaboración propia

Se puede observar en esta arquitectura a alto nivel que las secciones que tienen más importancia son la Plataforma de Comprensión de Lenguaje Natural (NPU), por sus siglas en inglés, que se encargada de procesar el lenguaje natural, y el *Intent engine*, que se encargará de procesar las palabras claves obtenidas por el procesamiento de la intención y las traducirá en acciones enviadas dentro de solicitudes mediante el REST API del controlador ONOS. Durante la primera etapa, el usuario final ingresa la intención deseada mediante un chatbot hacia nuestra plataforma de NPU. En la segunda etapa se realiza la extracción de palabras claves del lenguaje natural ingresado por el usuario final. Las palabras claves son los valores de los parámetros de las entidades definidas en la plataforma dialogflow. La tercera etapa, comienza cuando los valores extraídos son enviados a nuestro servicio web implementado localmente y se dan las validaciones necesarias para

identificar las acciones y configuraciones a realizar en la red. Las configuraciones mencionadas son construidas según el formato de la REST API del controlador ONOS que administra la red. En tabla 3-2 se describen las versiones de los sistemas empleados en las diferentes etapas de la implementación propuesta.

Tabla 3-2 Versiones de los sistemas utilizados

Sistema	Version
Python	3.6.9
Onos	3.6.9
Ubuntu	18.04
Flask	1.1.2
Ngrok	2.3.35
Openflow	2.9.5
Mininet	2.3.0d6

Fuente: Elaboración propia

3.3. Plataforma de Comprensión de Lenguaje Natural

Existen diversas plataformas de comprensión de lenguaje natural que son gratuitas y están disponibles para desarrolladores. Los sistemas que se evaluaron y probaron para esta tesis fueron Watson de IBM y DialogFlow de Google. La selección de la plataforma se realizó en base a la curva de aprendizaje, a la facilidad de uso, disponibilidad y a las características de la plataforma detallados en la tabla 2-5 del capítulo anterior, que cumplían con lo requerido para el desarrollo de lo propuesto. La plataforma de comprensión del lenguaje natural establecida en el entorno cloud debió ser entrenada con las expresiones que puedan asemejar a las intenciones que un usuario final o administrador de red desee realizar en la red desplegada. En primer lugar, para realizar lo mencionado fue necesario crear un *intent*, en el cual se añadió las expresiones deseadas o que se

esperaban serían posibles de realizar. Por otro lado, se estableció las entidades personalizadas que nos permitirían reconocer cuáles son las palabras claves de cada intent. Finalmente, luego de tener los intents y las entidades establecidas fue necesario relacionarlos. Para esto, se identificó en cada intent las entidades personalizadas creadas (acciones, objetos, nombres de nodos, etc) y las preestablecidas por la plataforma (numeros), como se puede observar en la figura 3-2, con la finalidad de usar los valores de los parámetros reconocidos, figura 3-3, enviados al servicio web para que pueda ser procesados luego por el algoritmo desarrollado en la siguiente etapa.

Figura 3-2 Expresiones donde se entrena a Dialoglfow a reconocer las palabras claves de la intención.

Fuente: Elaboración propia

Los valores de los parámetros de las entidades obtenidos por la plataforma de procesamiento de lenguaje natural (Dialogfow) son muy importantes en esta solución propuesta, ya que con estos se pudo reconocer que era lo que el usuario desea que se realice en la red. Los valores detectados en los *intents* fueron enviados, para este caso específico, a nuestro servicio web y procesados por el algoritmo desarrollado.

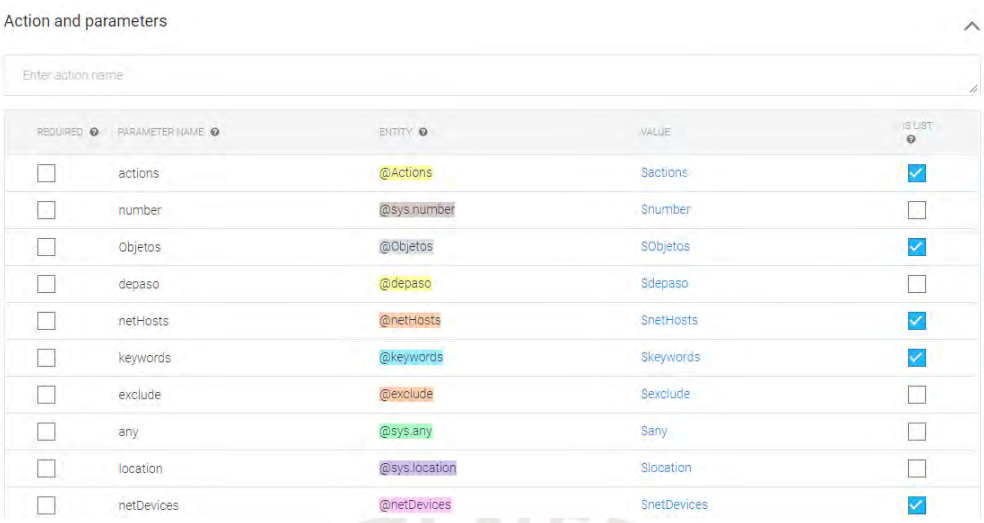


Figura 3-3 Lista de Entidades reconocidas en las expresiones(intents).

Fuente: Elaboración propia

Por último, se busca que los valores de las entidades obtenidas de la intención sean procesados por el algoritmo ubicado en el servidor local. Para esto, en la plataforma de lenguaje natural se habilita dentro de las funciones de *fulfillment* el servicio de *Webhook* brindando la URL de nuestro servicio web. El servicio web recibirá una solicitud POST con los valores de los parámetros de las entidades reconocidas en las intenciones ingresadas por los usuarios.

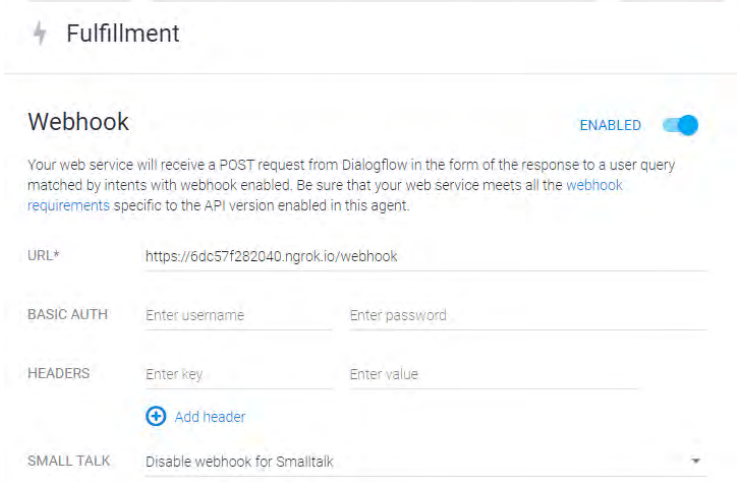


Figura 3-4 Servicio de webhook

Fuente: Elaboración propia

Es posible integrar el chatbot de la plataforma de lenguaje natural a nuestro servicio web a través de la habilitación del apartado “web demo” dentro de la sección integraciones. Solo bastara con agregar el widget dentro del servicio. Para habilitar el uso del chatbot de la plataforma de comprensión de lenguaje natural (DialogFlow) fuera de su página de desarrollo fue necesario utilizar la integración mencionada para el servicio web utilizado localmente.

Figura 3-5 Chatbot añadido en el servicio web

Fuente: Elaboración propia

3.4. Procesamiento de la Intención

El procesamiento de la intención se inicia desde la recepción de los valores de las entidades enviados al servicio web que se habilitó localmente. Luego de tener la información se procede a la evaluación, validación y procesamiento de la intención mediante un algoritmo desarrollado usando el lenguaje de programación python en su versión 3.6. Este algoritmo, que define el comportamiento del *Intent Engine*, permite, en principio, verificar el estado de los elementos de la red (hosts, switches) y según las palabras claves obtenidos de los parámetros enviados por el NLU

poder corroborar que la intención se puede cumplir. Para que el diseño de este tipo de red sea óptimo para mejorar la gestión de la red de una MYPE, debe satisfacer las configuraciones más comunes en una red de esta envergadura. Por tal motivo se prevé que el diseño propuesto en este trabajo de tesis satisfaga la siguiente lista de intenciones, sus variaciones presentadas en la tabla 3-3.

Tabla 3-3 Lista de Intenciones para una MYPE

Intenciones
• Asignar un host X a una Vlan Y.
• cambiar a la vlan Z al host X.
• Conectar un host X a un switch Z.
• cambiar la ubicación de un host X a otro SW Y.
• Cambiar la ip de un host X por una nueva ip.
• Bloquear todo el tráfico desde un host X a un host Y.
• Restringir el ingreso de un tipo de tráfico (TCP) a un puerto X desde un host Y.
• Conectar host X y host Y.
• Conectar host X atraves del nodo Z con el host Y
• Priorizar un tipo de trafico(UDP, TCP) entre host X y host Y
• Borrar el enlace directo entre nodo X y nodo Y.

Fuente: Elaboración propia

3.4.1. Servicio Web

Para el desarrollo de un servicio web se buscó que este sea el más sencillo posible debido a que este servicio se encargara de realizar solamente las consultas con DialoFlow. El servicio web se realizó utilizando flask, el cual es un framework de python que nos facilita la creación de aplicaciones web y el desarrollo de APIs de una forma sencilla, además posee un servidor web incluido. La selección de este framework fue muy importante para esta parte del desarrollo debido a que facilito la implementación del servicio web y género que las líneas de código fuesen mucho menores. Por otro lado, la necesidad de que nuestro servicio web tenga una URL para interactuar con la API de DialogFlow género que se busque una herramienta que nos permita tener un dominio

en internet. Se tenía conocimiento que la obtención de un dominio para nuestro servicio web necesitaría de una inversión, por lo que se buscó una alternativa sin costo y de fácil implementación en linux. En consecuencia, para poder exponer el servicio web se hizo uso de la herramienta ngrok, la cual nos permite hacer publica nuestro servidor local hacia internet facilitando su implementación y eliminando gastos al ser este una herramienta gratuita. Ngrok permite que se pueda interactuar con la API de Dialogflow para obtener los valores de los parámetros de la identidad y puedan ser procesados. La exposición pública de nuestro servicio web utilizando ngrok se realizó digitando tan solo un comando: *./ngrok http 5000,* donde el numero 5000 representa el puerto del servidor web que utiliza flask. Con esta herramienta es posible obtener una URL publica, la cual es la que nos permite ingresar en el apartado del Webhook de dialogflow mostrado en la figura 3-4. Se debe tener en cuenta que la URL generada por ngrok es la misma mientras la sesión no sea cerrada, en caso se inicie otra sesión esta cambiara.

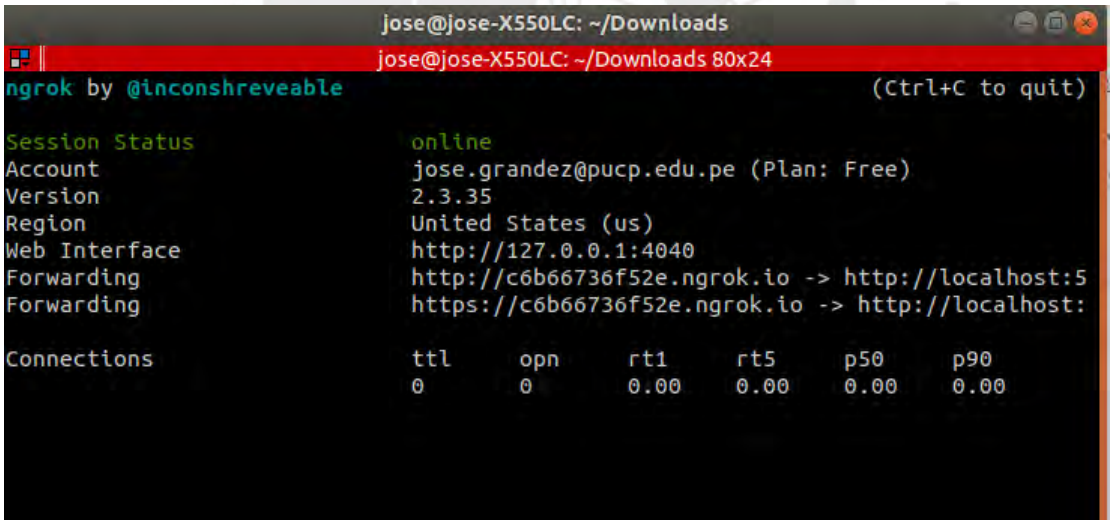


Figura 3-6 Herramienta ngrok activa

Fuente: Elaboración propia

3.4.2. Procesamiento de la data por el Intent Engine

Para el procesamiento de la intención por parte de un algoritmo se propuso, como se indicó anteriormente, utilizar el lenguaje de programación python en su versión 3.6. La implementación empieza con la obtención de los valores de los parámetros de las entidades obtenidas por Dialogflow y enviadas mediante un método POST a nuestro servidor web. Luego de la validación sobre la recepción de la información, se confirmó que los datos de los valores de los parámetros de las entidades recibidos estaban en formato JSON. Esta data obtenida tuvo que ser seleccionada y contrastada con información obtenida de la red (equipos de red, enlaces, hosts, intents, flows, topología, etc) para establecer que la intención expresada por el usuario es realizable. Por lo tanto, al confirmarse la factibilidad de realización de la intención se procede a hacer validaciones sobre los objetos de red definidos en las entidades, los cuales se refieren a switches, hosts, enlaces, routers. Esta información es importante, al igual que su cantidad, ya que si se debe especificar en qué y cuantos lugares de la red se debe realizar un cambio. De ser positiva la validación sobre los objetos de red presentes en la intención se procede a evaluar si existe una acción a realizar y cual es esta. La importancia de la acción es similar a la de los objetos de red, por este motivo es necesario su presencia en la intención. Finalmente, por cada validación mencionada se define las solicitudes y la data en estas (MAC, IDs, etc) que se envían al controlador mediante la REST API para la aplicación de las configuraciones necesarias en los equipos de red. El código del procesamiento de la intención esta adjunta en los anexos A continuación, se describe en la siguiente figura la lógica del procesamiento de la intención por el algoritmo descrito en esta sección.

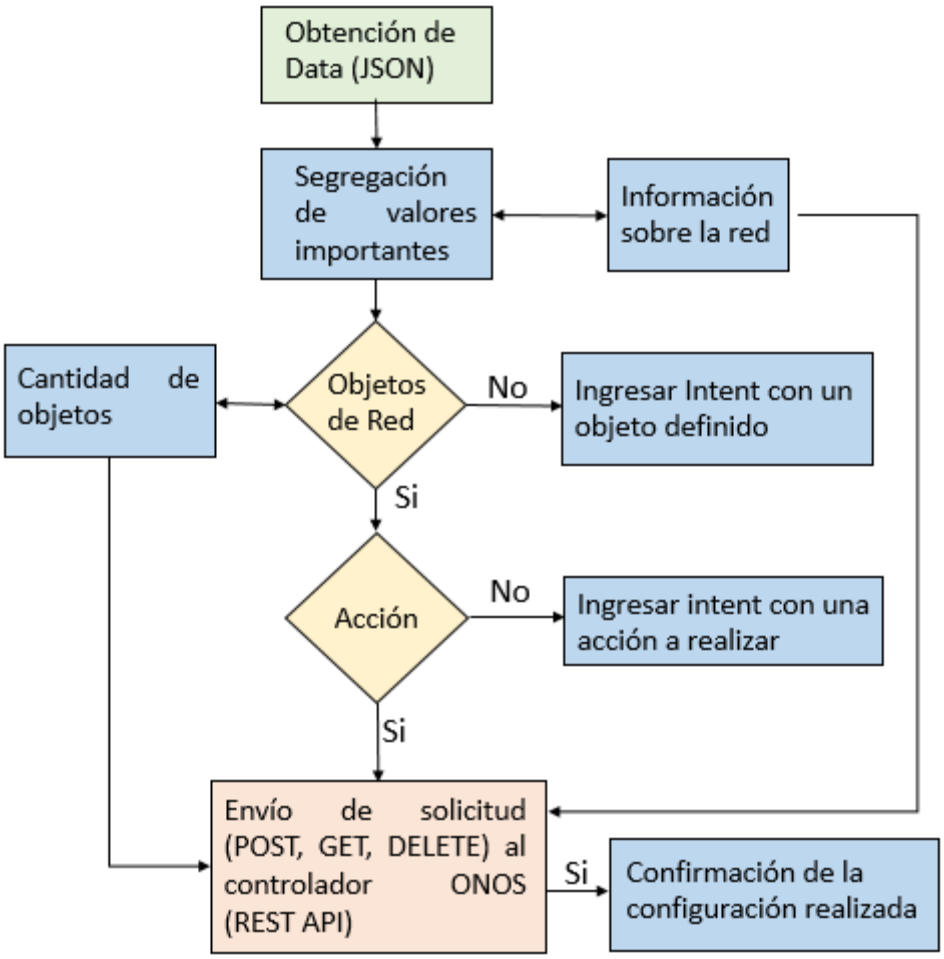


Figura 3-7 Diagrama de flujo del procesamiento de la intención

Fuente: Elaboración propia

3.5. Acciones en el controlador SDN

La selección del controlador se basó en lo descrito en el capítulo anterior y lo expuesto en la tabla 2-3. El controlador SDN que se utilizó es ONOS en su versión 2.3.0 llamado también Toucan. Esta versión es una de las más recientes liberada en enero del 2020, y como se describió en el capítulo anterior posee un intent frameweork encargado de procesar las intenciones. Para poder iniciar el servicio del controlador onos, tenemos que tener en cuenta lo mencionado en el inicio de este capítulo, que se está trabajando en un entorno Linux. Por consiguiente, bastara con ingresar

dentro del directorio de onos 2.3.0 e ingresar el siguiente comando: bin/onos-service start, como se puede observar en la siguiente figura.

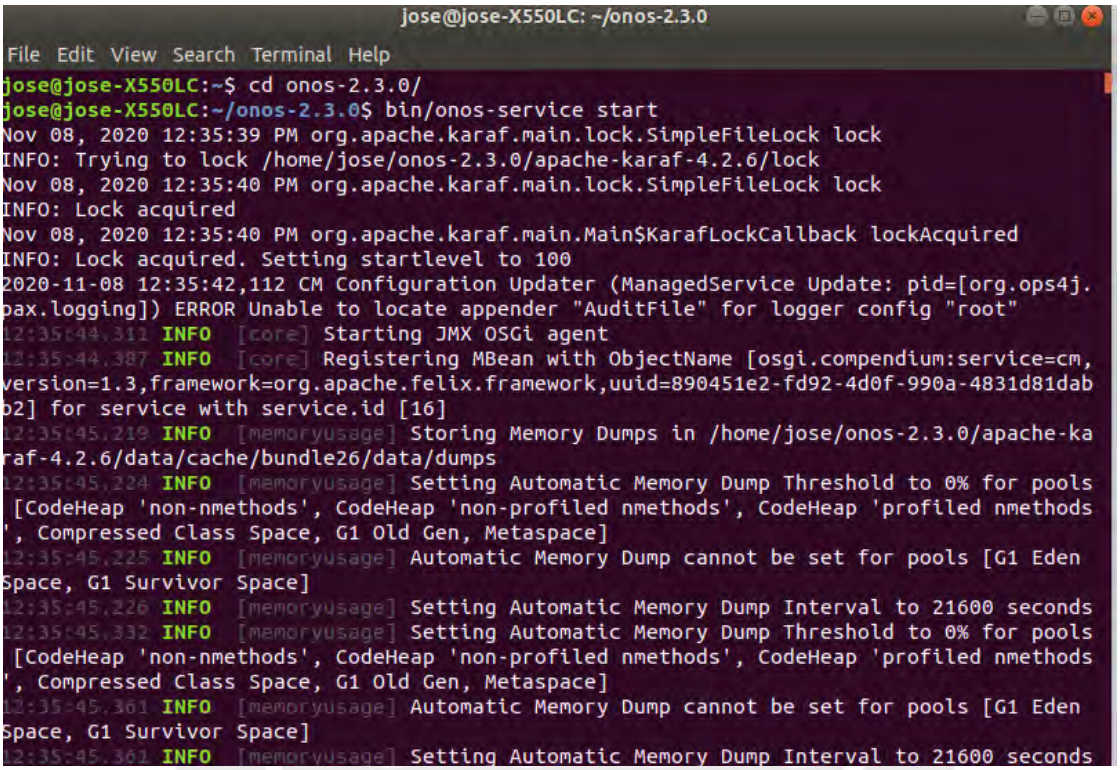


Figura 3-8 Activación del Controlador ONOS en linux

Fuente: Elaboración propia

Luego de tener el controlador en funcionamiento, para esta tesis se vio necesaria la activación de ciertas aplicaciones propias de ONOS para poder lograr un funcionamiento deseado de este controlador en conjunto con la topología desplegada en Mininet sin ningún contratiempo. Para esto se ingresó al controlador haciendo uso del protocolo SSH mediante el siguiente comando en linux: ssh -p 8101 onos@192.168.123.1 donde la dirección ip es la del controlador y el puerto 8181 es propio de ONOS. Luego de ingresar al controlador en la interface del CLI se deberá ingresar los siguientes comandos:

- app activate org.onosproject.openflow : Habilitar openflow
- app activate org.onosproject.linkdiscovery : Habilitar descubrimiento de enlaces en la topologia
- app activate org.onosproject.fwd: Habilitar el envió de paquetes
- app activate org.onosproject.proxyarp : Habilitamos proxy arp para el uso de los intents

Se verifico que las aplicaciones estén activadas mediante el ingreso del siguiente comando en el CLI de ONOS: app –a –s.

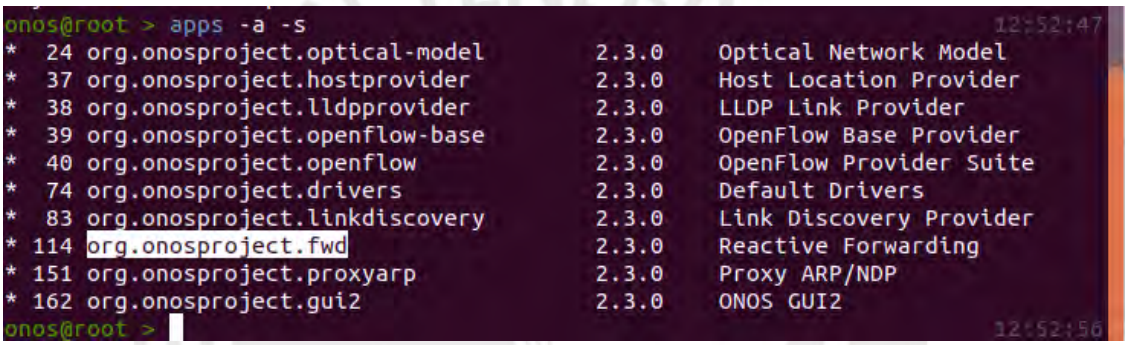


Figura 3-9 Lista de aplicaciones activas en el controlador ONOS

Fuente: Elaboración propia

Por otro lado, el controlar posee una GUI con la que es posible tener una referencia grafica de la topología que administra el controlador ONOS. Un ejemplo es la figura 3-9 donde está desplegada una topología full mesh de 3 switches. Esta topología se muestra en en la GUI ya que está reconocida y bajo el control de ONOS. La URL necesaria para el acceso al GUI es la siguiente: http://192.168.123.1:8181/onos/ui/index.html. Dentro del GUI también es posible administrar la red, sin embargo, para los objetivos establecidos dentro del desarrollo de este trabajo no es necesario, ya que el GUI nos brindara en este caso una referencia grafica de la topología.

Figura 3-10 GUI del controlador ONOS mostrando topología bajo su control

Fuente: Elaboración propia

3.6. Despliegue de red SDN para una MYPE virtualizada

La selección de Mininet como emulador de la red SDN se basó en la practicidad que esta posee, la documentación disponible y las experiencias académicas de uso que se tuvieron. Para el despliegue de una topología que pueda satisfacer las necesidades de una MYPE se tuvo en consideración lo descrito en el capítulo anterior y también en el estado de las redes de las MYPES descrito en la sección. En la sección 2.4 se especifica la cantidad de usuarios promedio y nos permite estimar la cantidad de dispositivos que estarán conectados en la red. Por tal motivo, se decidió implementar una topología full mesh de tres switches SDN en caso las distintas áreas de la empresa no estén muy apartas unas de otras, si este no es el caso, se prefiere que se pueda utilizar la topología que actualmente se está usando y que se recomienda, descrita en la sección 1.1.2, para empresas de esta envergadura.

Mininet posee algunas topologías por defecto, pero en este caso ninguna satisface lo que se desea para esta implementación. En consecuencia, la topología necesitó ser implementarla de forma personalizada mediante un script de python. Una característica de este emulador de redes SDN es la de poder utilizar una API de python donde se modifica el código para la creación de la topología fullmesh. El código de esta topología es un archivo en python ubicado dentro de la carpeta donde Mininet está instalado. Se puede visualizar el contenido de este archivo en los anexos. Por otro lado, la capacidad más importante que Mininet posee es de poder generar una red SDN con una simple línea de código, la cual varía de acuerdo a las características deseadas de la topología, controlador, etc. Para el presente despliegue se realizó mediante el siguiente comando: sudo mn --custom prueba.py --topo=mytopo --controller=remote,ip=192.168.123.1,port=6633 , donde prueba.py es el código en python de la topología fullmesh y la ip es la del controlador ONOS haciendo uso del puerto 6633 para que pueda administrar esta red.

```python
from mininet.topo import Topo

class MyTopo( Topo ):
    "Simple topology example."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        H1 = self.addHost( 'h1' )
        H2 = self.addHost( 'h2' )
        H3 = self.addHost( 'h3' )
        S1 = self.addSwitch( 's1' )
        S2 = self.addSwitch( 's2' )
        S3 = self.addSwitch( 's3' )
        S4 = self.addSwitch( 'S4' )

        # Add links
        self.addLink( H1, S1 )
        self.addLink( H2, S2 )
        self.addLink( H3, S3 )
        self.addLink( S1, S4 )
        self.addLink( S2, S4 )
        self.addLink( S3, S4 )


topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Figura 3- 11 Archivo de configuración de topología deseada en mininet

.

Fuente: Elaboración propia

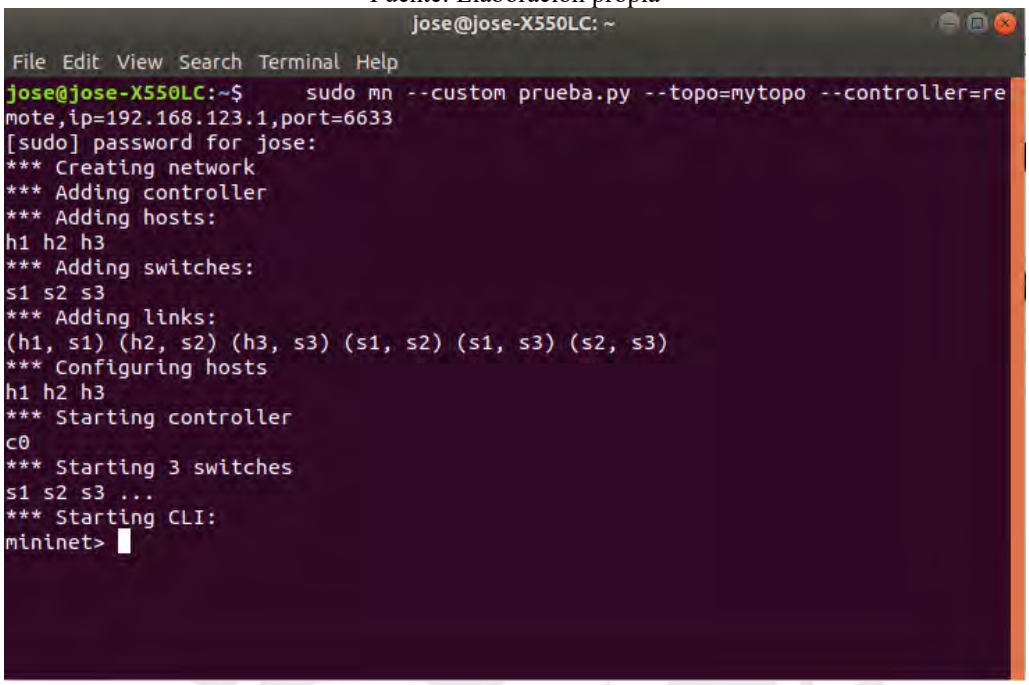


Figura 3- 12 Creación de la red SDN full mesh utilizando el controlador ONOS

Fuente: Elaboración propia

Es necesario habilitar el protocolo ARP en Mininet de la siguiente forma para cada host: nombrehost arp –an

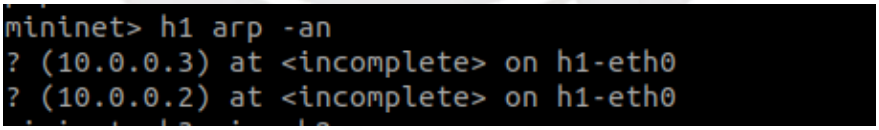


Figura 3- 13 Habilitación de ARP en mininet

Fuente: Elaboración propia

Capítulo 4. Análisis por Simulación de la Red IBNM para una MYPE del sector servicios

Durante el desarrollo de este capítulo se mostrarán los resultados al ingreso de los *intents* descritos en la tabla 3-3 en el diseño propuesto. Asimismo, se incluirán pruebas de conectividad y análisis de las conexiones generadas durante las pruebas realizadas.

4.1 Pruebas con la topología full mesh

Como parte de la demostración del funcionamiento de la gestión de una red corporativa para una MYPE mediante el paradigma IBNM se procedió a ingresar *intents* en el chatbot de Dialogflow sobre la topología full mesh de tres switches dondese expresa un deseo de conectar dos hosts ubicados detrás de distintos nodos sin dar más detalles técnicos sobre su ubicación

4.1.1 Conexión entre hosts de forma directa

Para la realización de esta prueba de desactivo el *forwarding* de paquetes realizado por la aplicación org.onosproject.fwd el cual genera de forma automática *flows* donde establece conexiones entre los hosts de la topología.

La prueba se inicia con el ingreso del *intent* en la plataforma NLU desplegada en el servidor web del localhost observado en la figura 4-1.

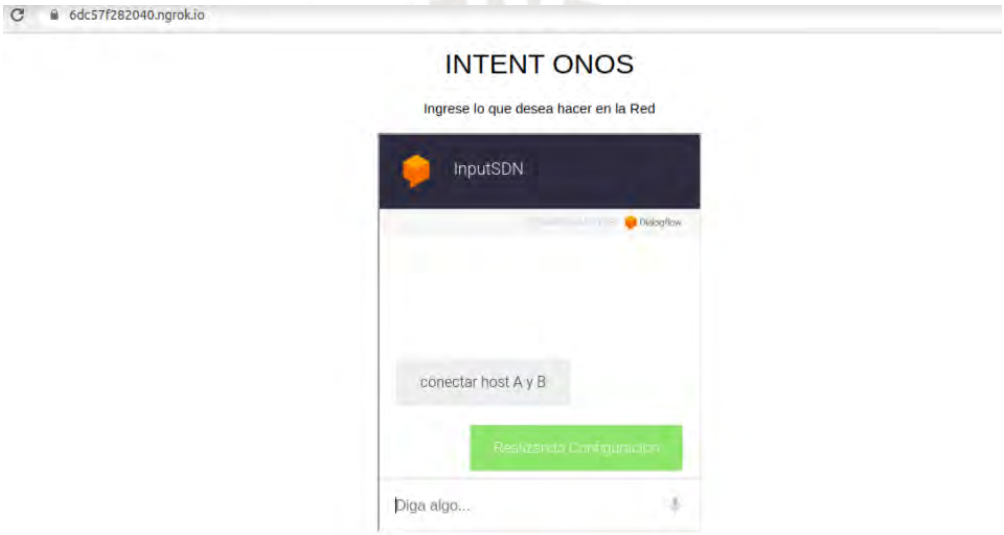


Figura 4-1 Ingreso de un intent en el chatbot de DialogFlow

Fuente: Elaboración propia

Asimismo, al observar el GUI de onos, figura 4-2, podemos ver que el flujo de datos entre los hosts especificados se establece por el camino más corto entre ellos. Esto significa que se dio una correcta interpretación de la intención y una correcta aplicación de las reglas y configuraciones necesarias en el controlador.

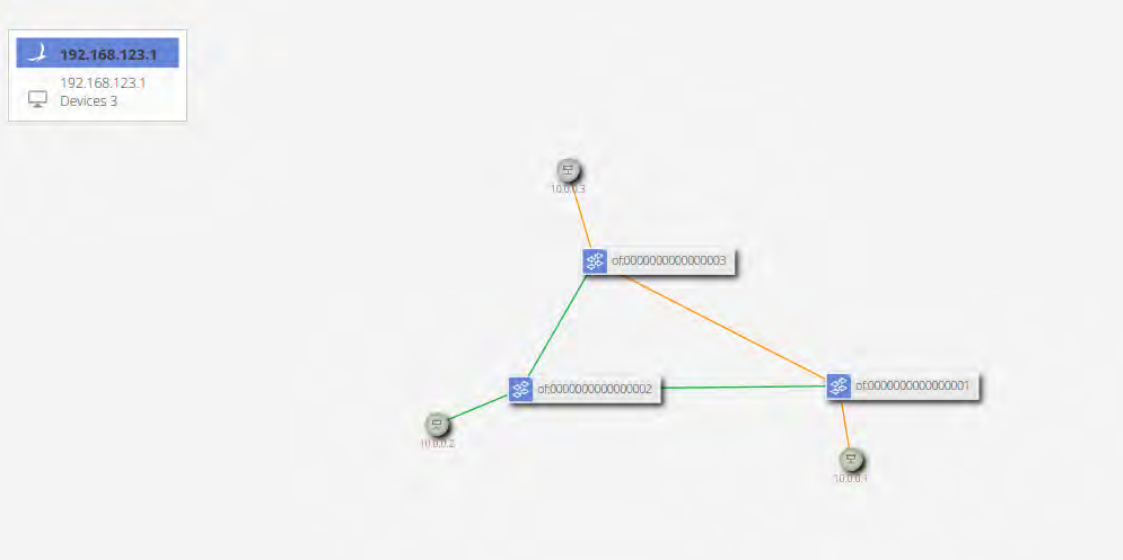


Figura 4-2 Flujo de datos entre los hosts

Fuente: Elaboración propia

La demostración del correcto funcionamiento de la plataforma de comprensión de lenguaje natural(DialogFlow) en el tratamiento de la intención y la obtención de los valores de los parámetros de las entidades definidas previamente puede ser observada en la figura 4-3. Los valores mostrados en la imagen: conectar, host, A y B, fueron asignados correctamente a las entidades acciones, objetos y nombreRedes logrando que la plataforma pueda reconocer el lenguaje natural del *intent* y extraer los parámetros claves que permiten demostrar el deseo del usuario.

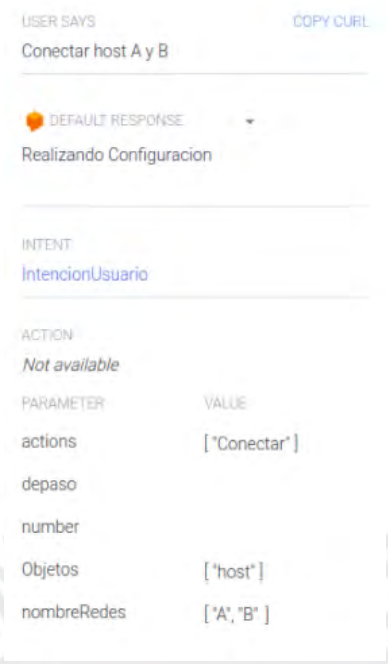


Figura 4-3 Resultado de la operación de DialogFlow

Fuente: Elaboración propia

Los valores extraídos por la plataforma serán enviados mediante la API de DialogFlow a nuestro servicio web en formato JSON como se muestra en la figura 4-4. Esta información luego será filtrada para hacer el procesamiento necesario mediante el algoritmo desarrollado.


```
{
  "responseId": "e87ab3c9-ef89-444b-8460-9586c6350996-a02f1a09",
  "queryResult": {
    "queryText": "Conectar host A y B",
    "parameters": {
      "actions": [
        "Conectar"
      ],
      "depaso": "",
      "number": "",
      "Objetos": [
        "host"
      ],
      "nombreRedes": [
        "A",
        "B"
      ]
    },
```


Figura 4-4 POST en formato JSON de la API de DialogFlow

Fuente: Elaboración propia

4.1.1.1 *Instalación de reglas en el controlador*

Para comprobar el correcto funcionamiento del procesamiento del algoritmo y de la posterior instalación correcta de la regla en el controlador, se tuvo que ingresar al controlador mediante ssh y verificar haciendo uso del CLI si, efectivamente, se instaló una regla, ya que se utilizó el intent framework de onos. La instalación mostro que la regla tuvo el estado de instalado luego de ingresar el intent, y esta fue del tipo de intent host to host tal como se puede observar en la figura 4-5.

Figura 4-5 Verificación de la instalación de la regla en el CLI de onos

Fuente: Elaboración propia

Asimismo, se verifico que al instalar una regla en una red SDN se generarían flows propios de esta tecnología generado por el controlador hacia los equipos que están bajo su control. Por tal motivo en la figura 4-6 se observa que después del ingreso del *intent* se generaron 2 flows los cuales tienen como fuente y destino los hosts localizados en los switches 1 y 2. Estos flows muestran las MAC los hosts debido a que en la intención se detalló que se deseaba una conexión entre hosts y esta se volvió a verificar en imagen anterior debido a que la regla instalada fue un *intent* del tipo host to host. Como se mostró en el apartado 4.1 en la imagen del GUI estos flows son los suficientes para lograr que los hosts puedan lograr una conexión entre sí.

Figura 4-6 Flows instalados en el controlador onos debido al intent

Fuente: Elaboración propia

4.1.1.2 *Análisis del enlace*

Como se pudo observar en los resultados expuestos en la secciones anteriores, todos estos tuvieron resultados positivos segun lo que se buscaba realizar. Por tal motivo, solo quedo comprobar que existe una conexión entre los hosts especificados, para esto se realizo pruebas de conectividad haciendo uso del ping entre los hosts de la topologia a traves de miniet, figura 4-7, y tambien ingresando a cada host mediante el uso de xterm.

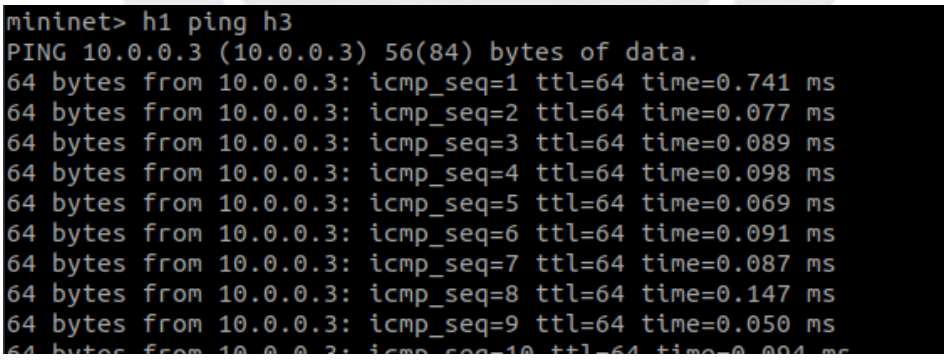


Figura 4-7 Conectividad de los hosts debido al intent

Fuente: Elaboración propia

Por otro lado, otra de las pruebas realizadas fue de evaluar la latencia del enlace entre los hosts en diversas ocasiones. Los resultados de las pruebas se muestran gráficamente en la figura 4-8. Los datos obtenidos muestran una latencia promedio de 0.08 ms lo cual refleja un valor esperado debido a la topología implementada.

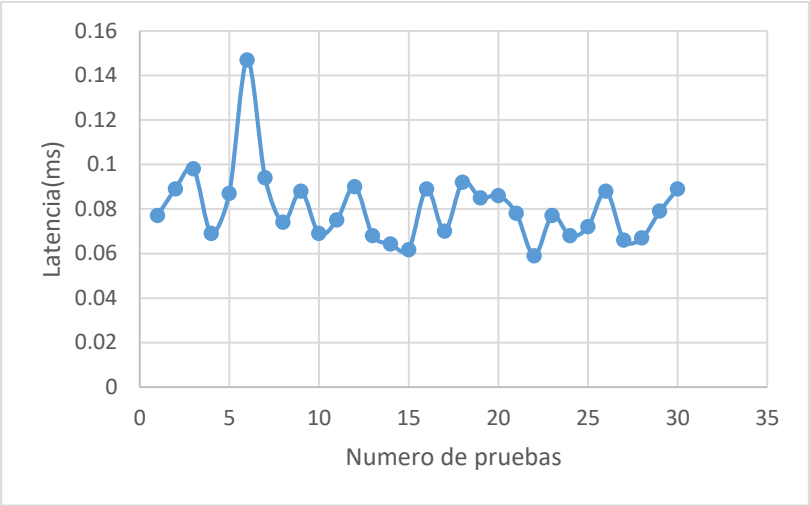


Figura 4-8 Testeo de la latencia en el enlace de la primera prueba

Fuente: Elaboración propia

Finalmente, se realizó una evaluación del throughput usando la herramienta de iperf. El flujo datos determinado en diversos ensayos observado en la tabla 4-1 fue realizado haciendo uso de paquetes TCP enviados entre los hosts donde se levantó la conexión. El resultado a destacar en este test es que existe una media de 37.2 Gbits/s.

Tabla 4-1 Throughput en el enlace directo entre hosts

Test	Throughput
Test 1	36.1 Gbits/s
Test 2	38.0 Gbits/s
Test 3	39.4 Gbits/s
Test 4	35.3 Gbits/s

Fuente: Elaboración propia

4.1.2 Conexión entre hosts mediante un nodo de paso

De forma similar a lo realizado en la evaluación 4.1.1 para demostrar que se puede lograr la gestión de una red mediante IBNM se desactivo el *forwarding* de paquetes realizado por la aplicación org.onosproject.fwd el cual genera de forma automática *flows* donde establece conexiones entre los hosts de la topología. Luego de la desactivación se ingresó otro *intent* en DialogFlow, como se observa en la figura 4-9. Esta intención tuvo el objetivo de lograr otra conexión entre los hosts con la diferencia de que se utilice otro camino, es decir, el flujo de datos se debería dar a través del switch del host restante en la topología propuesta.

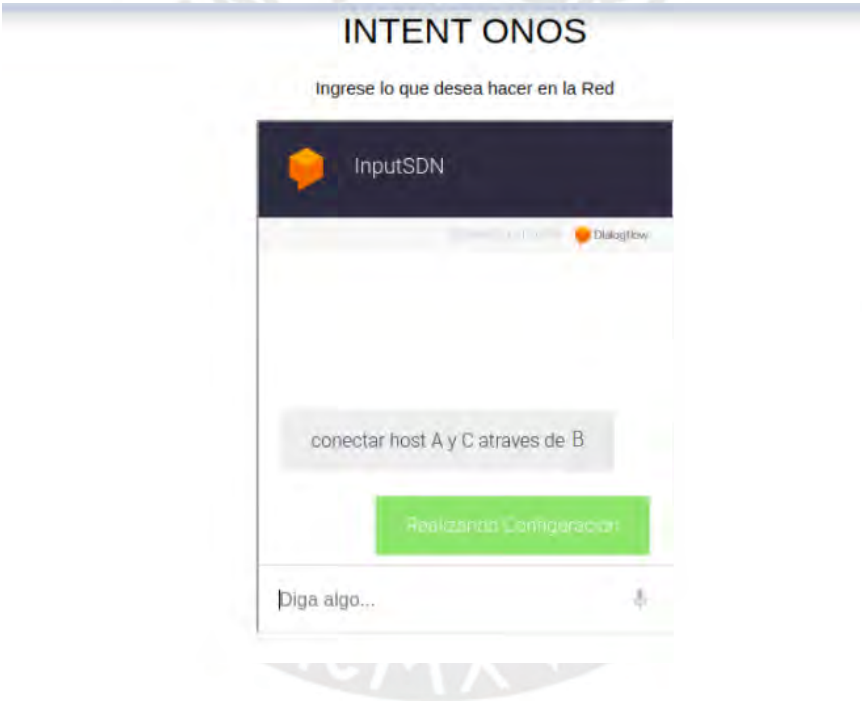


Figura 4-9 Ingreso de otro intent en el chatbot de DialogFlow

Fuente: Elaboración propia

El resultado de esta prueba puede ser observado en la figura 4-10 donde se observa que se logró lo expresado en el intent sin especificar algún detalle técnico. Tal como muestra el GUI de onos la conexión entre los hosts se realizó por medio del switch B, mientras que lo realizado en la prueba 4.1 fue efectuado mediante la conexión directa entre los hosts A y C.

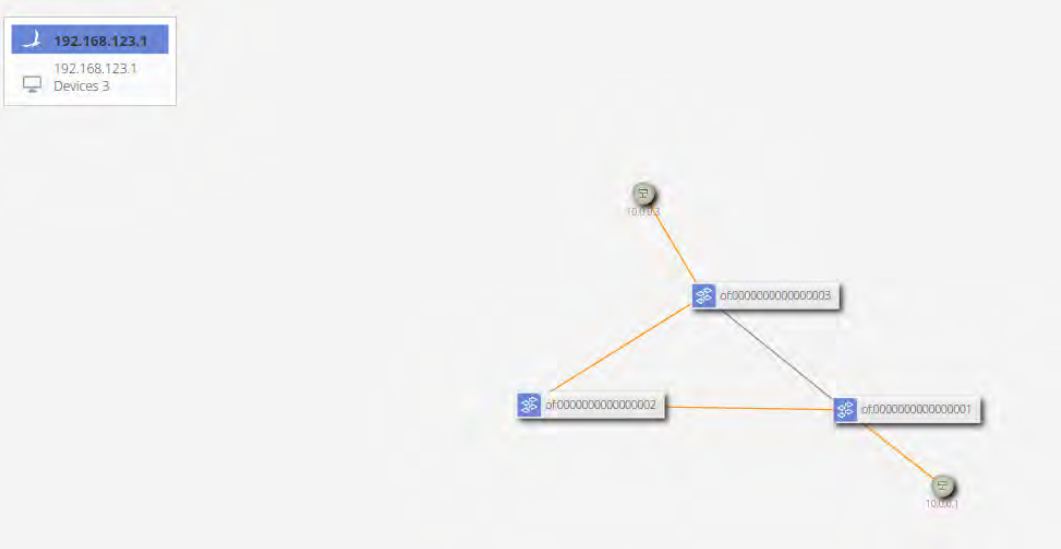


Figura 4-10 Flujo de datos entre hosts A y C a través de B

Fuente: Elaboración propia

Se muestra el resultado de la operación de la plataforma de comprensión del lenguaje natural (DialogFlow) debido al ingreso del segundo intent. Según se puede observar en la figura 4-11 al ingresar el intent la plataforma detectó según su entrenamiento las palabras: conectar, a través, host, A, B y C, los cuales son los valores de los parámetros de las entidades creadas. Debido a esto, se puede esclarecer que es lo que deseo realizar el usuario en la red. La imagen siguiente se extrajo de la plataforma de DialogFlow donde se realiza la configuración y entrenamiento de esta.

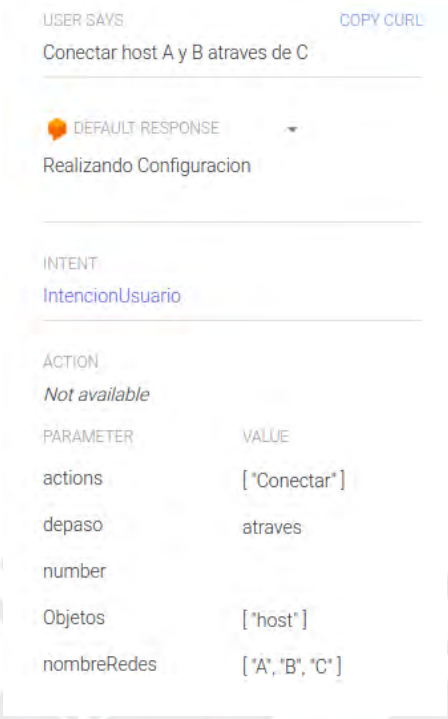


Figura 4-11 Resultado de la operación de DialogFlow con el segundo intent

Fuente: Elaboración propia

Como se mostró anteriormente, se extrajo los valores que corresponden a los parámetros de las entidades creadas en la configuración de la plataforma para este trabajo de tesis. Los valores tuvieron que ser enviados a nuestro servicio web que es parte del *intent engine*. El envió de estos datos según el funcionamiento de la API de DialogFlow es mediante el formato JSON. Para comprobar que se realizó correctamente el envio hacia nuestro servidor local se accedió al contenido de la data como se muestra en la figura 4-12. En la imagen se puede observar que los valores tienen el formato esperado lo cual nos demuestra que al hacer uso de otro intent se comprobó el funcionamiento correcto de la plataforma.

```
{
  "responseId": "e87ab3c9-ef89-444b-8460-9586c6350996-b87e6a39",
  "queryResult": {
    "queryText": "Conectar host A y C atraves de B",
    "parameters": {
      "actions": [
        "Conectar"
      ],
      "depaso": "atraves",
      "number": "",
      "Objetos": [
        "host"
      ],
      "nombreRedes": [
        "A",
        "B",
        "C"
      ]
    },
```


Figura 4-12 POST en formato JSON de la API de DialogFlow según el segundo intent

Fuente: Elaboración propia

4.1.2.1 *Instalación de reglas en el controlador*

La instalación de las reglas necesarias en el controlador debido al ingreso de otro *intent* es otra etapa a verificar. La revisión de los resultados de esta etapa nos refleja de forma indirecta el correcto funcionamiento del algoritmo implementado. Para revisar el resultado de la correcta instalación de las reglas en el controlador se tuvo que ingresar al CLI de este mediante ssh. Dentro se ingresó a revisar los intentos instalados debido a que la intención ingresada por el usuario, en este caso, puede ser satisfecha con el intent framework. Lo mencionado puede ser verificado en la siguiente imagen.

Figura 4-13 Verificación de la instalación de las reglas en el CLI de onos

Fuente: Elaboración propia

Asimismo, se necesitó verificar si se instalaron nuevas reglas en los switches que nos indiquen que se realizaron cambios en el comportamiento de la red. Por tal motivo, se ingresó nuevamente en el controlador y se verifico los flujos o flows instalados en estos dispositivos de red encontrándose, como se puede observar la seleccionen la figura 4-14, dos nuevas reglas en cada switch. Estas reglas nos permiten verificar que lo deseado por el usuario en su *intent* de conectar los hosts mediante el switch B fue satisfecho.

Figura 4-14 Flows instalados en el controlador onos debido al segundo intent

Fuente: Elaboración propia

4.1.2.2 *Análisis del enlace*

Se comprobó el correcto funcionamiento de todas las etapas anteriores de esta segunda prueba. Por tal motivo, en esta etapa se realizaron los análisis de conectividad entre los hosts A y C. La primera prueba realiza fue un test de ping, el cual realiza un envío de paquetes ICMP de solicitud y respuesta. Con esta prueba se comprobó que existe una comunicación entre los hosts.

```
mininet> h1 ping h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=0.588 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=0.104 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=0.097 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=0.087 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=0.123 ms
64 bytes from 10.0.0.3: icmp_seq=6 ttl=64 time=0.052 ms
64 bytes from 10.0.0.3: icmp_seq=7 ttl=64 time=0.057 ms
64 bytes from 10.0.0.3: icmp_seq=8 ttl=64 time=0.072 ms
^C
--- 10.0.0.3 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7166ms
rtt min/avg/max/mdev = 0.052/0.147/0.588/0.168 ms
```

Figura 4-15 Conectividad entre los hosts A y C según lo descrito en el intent

Fuente: Elaboración propia

Otro de los análisis realizados en el nuevo enlace fue el testeo de la latencia. Para encontrar un valor promedio se realizaron varias pruebas donde el resultado fue de 0.094 ms. Este valor comparado al resultado obtenido en la sección anterior de la primera prueba correspondiente al primer intent es mayor. No obstante, es un valor esperado debido a que ya no es una conexión directa entre los switches A y C, sino que en esta segunda prueba existió un paso por el switch B. La tendencia del comportamiento de la latencia en este testeo puede verse en la figura 4-16.

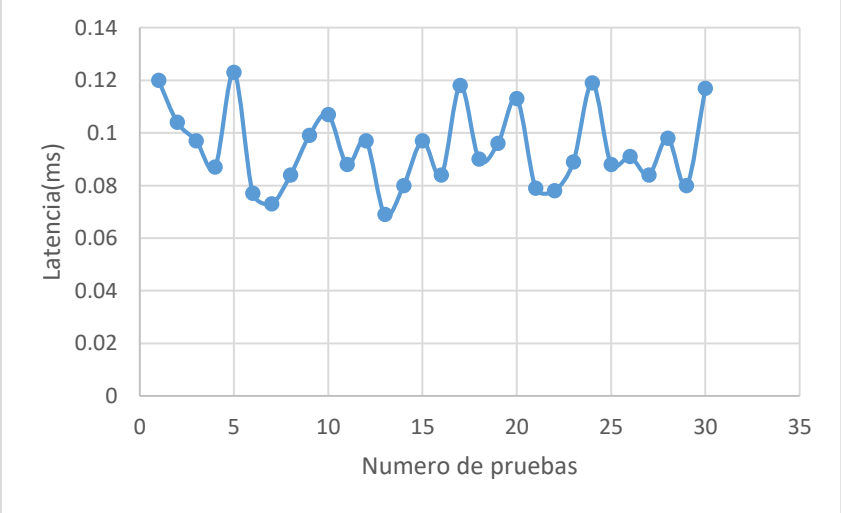


Figura 4-16 Testeo de latencia en el enlace de la segunda prueba

Fuente: Elaboración propia

Finalmente, se realizó el análisis del throughput del enlace usando la herramienta de iperf como en la sección anterior correspondiente a la primera prueba. Para obtener los resultados se ingresó al host A y C mediante xterm y se procedió a que uno de ellos realice el rol de servidor y el otro de cliente para el envío de paquetes TCP. Se realizaron tres tests donde se concluyó que el enlace de esta segunda prueba tiene un promedio de 37.8 Gbits/s. Este resultado es coherente con lo encontrado en la sección 4.1.3 donde también se encontró un valor cercano de throughput debido a que se trata de un mismo tipo de enlace sin ninguna variación

Tabla 4-2 Throughput en el enlace entre hosts mediante nodo de paso

Test	Troughput
Test 1	37.8 Gbits/s
Test 2	37.6 Gbits/s
Test 3	38.0 Gbits/s

Fuente: Elaboración propia

4.2 Prueba con la topología a dos niveles

Como parte de lo propuesto como una topología recomendada y que se utiliza actualmente por empresas MYPE, se desplego una red de dos niveles donde en cada switch de acceso tienen conectado un host que puede asemejar un usuario final o un servidor con un servicio específico.

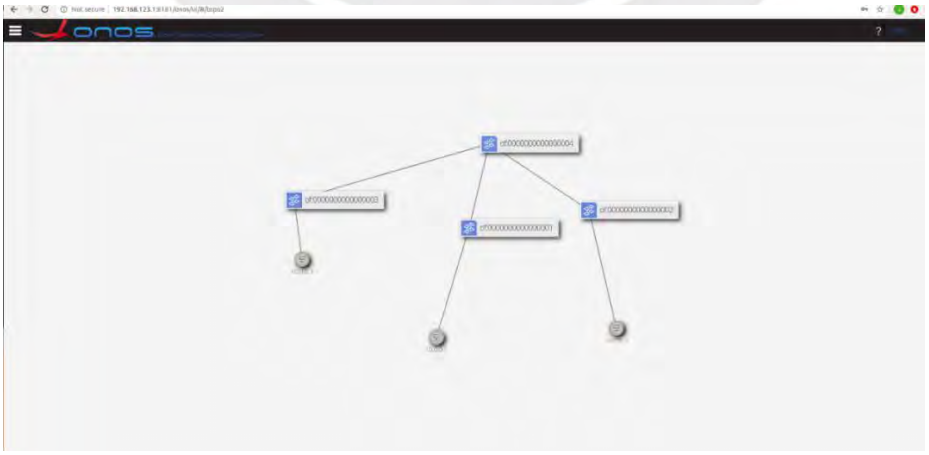


Figura 4-17 Despliegue de topología de dos niveles en ONOS

Fuente: Elaboración propia

4.2.1 Bloqueo de tráfico entre dos hosts

Durante esta prueba se realizó el ingreso de un *intent* con el objetivo que se restrinja el flujo de tráfico entre dos hosts. En el siguiente grafico se puede observar que está activado el forwarding de trafico org.onosproject.fwd donde permite que exista conexión entre los hosts.

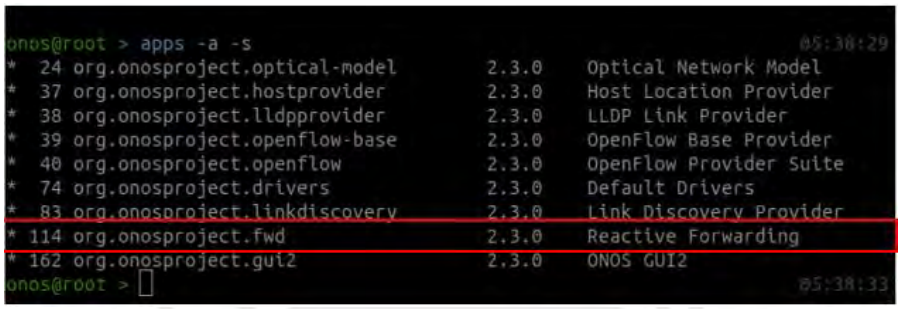


Figura 4-18 Aplicaciones de ONOS activas

Fuente: Elaboración propia

Para verificar que existe conexión entre los hosts se realizó prueba de conexión ICMP entre los hosts, los resultados positivos de este test se ven reflejados en la figura 4-19.

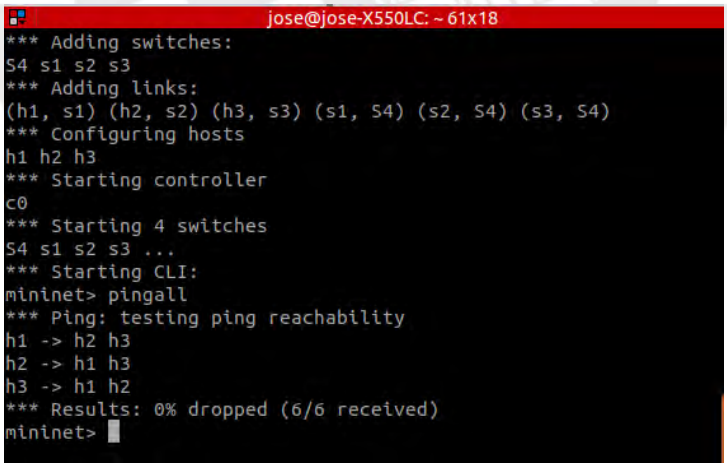


Figura 4-19 Prueba de conexión entre hosts

Fuente: Elaboración propia

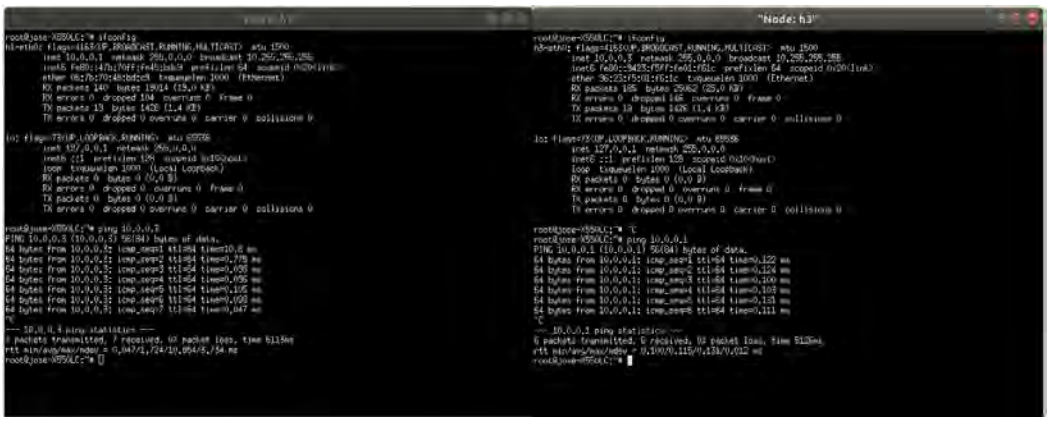


Figura 4-20 Verificación de ping entre hosts

Fuente: Elaboración propia

Luego de verificar que existe conexión entre los hosts se procede a el ingreso del *intent* (Figura 4-20) cuyo objetivo es la restricción de flujo de tráfico entre los hosts mencionados. La intención expresada es procesada, en principio, por la plataforma de NLU (Dialogflow), la cual envía los parámetros detectados en el *intent* a el *intent engine* que se encargara de hacer las validaciones necesarias para hacer los cambios en la red SDN.

Figura 4-21 Ingreso del Intent en la plataforma del servidor web

Fuente: Elaboración propia

Finalmente, los cambios podrán ser realizados o se verificaran que no pueden realizar. En la figura 4-22 podemos observar que luego de que existiera conexión entre los hosts esta se pierde debido a que el *intent engine* interpreto lo que quería realizar.

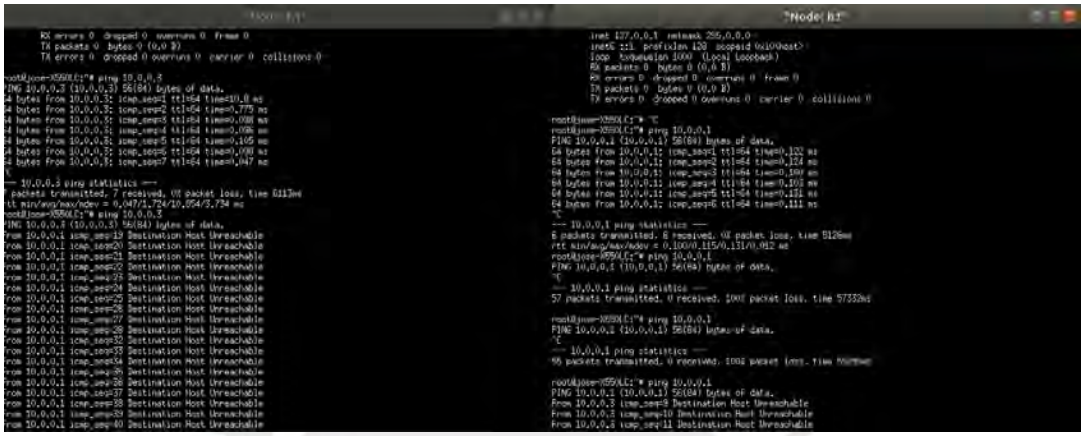


Figura 4-22 Verificación de perdida de conexión entre hosts

Fuente: Elaboración propia

4.2.2 Asignación/cambio de vlan a un host

Durante esta prueba se realizó el ingreso de un *intent* con el objetivo que se asigne la vlan a la que pertenece un host. En el siguiente grafico se puede observar que el host al que se le desea asignar la nueva vlan no posee ninguna

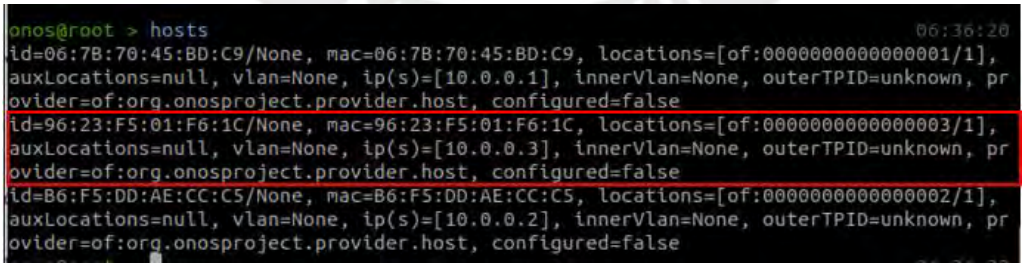


Figura 4-23 Estado inicial de la vlan de los hosts.

Fuente: Elaboración propia

Luego de verificar el estado del host objetivo, se procede a el ingreso del *intent* (Figura 4-24) cuyo objetivo es la asignación del usuario a la vlan 100. La intención expresada es procesada,

en principio, por la plataforma de NLU (Dialogflow), la cual envía los parámetros detectados en el *intent* a el *intent engine* que se encargara de hacer las validaciones necesarias para hacer los cambios en la red SDN.

Figura 4-24 Ingreso del Intent en la plataforma del servidor web.

Fuente: Elaboración propia

Finalmente, los cambios podrán ser realizados o se verificaran que no pueden realizar. En la figura 4-25 podemos observar que luego de que el host no estuviera asignado a una vlan (None), el valor actual es 100.

```
onos@root > hosts                                                    06:48:42
id=06:7B:70:45:BD:C9/None, mac=06:7B:70:45:BD:C9, locations=[of:0000000000000001/1],
auxLocations=null, vlan=None, ip(s)=[10.0.0.1], innerVlan=None, outerTPID=unknown, pr
ovider=of:org.onosproject.provider.host, configured=false
id=96:23:F5:01:F6:1C/100, mac=96:23:F5:01:F6:1C, locations=[of:0000000000000003/1], a
uxLocations=null, vlan=100, ip(s)=[10.0.0.3], innerVlan=None, outerTPID=unknown, prov
ider=host:org.onosproject.rest, configured=true
id=B6:F5:DD:AE:CC:C5/None, mac=B6:F5:DD:AE:CC:C5, locations=[of:0000000000000002/1],
auxLocations=null, vlan=None, ip(s)=[10.0.0.2], innerVlan=None, outerTPID=unknown, pr
ovider=of:org.onosproject.provider.host, configured=false
onos@root >                                                          06:50:09
```

Figura 4-25 Estado final de la vlan del host en cuestión.

Fuente: Elaboración propia

4.2.3 Asignación/cambio de ip a un host

Durante esta prueba se realizó el ingreso de un *intent* con el objetivo que se asigne una nueva ip un host. En el siguiente grafico se puede observar que el host al que se le desea asignar la nueva ip ya posee ninguna.

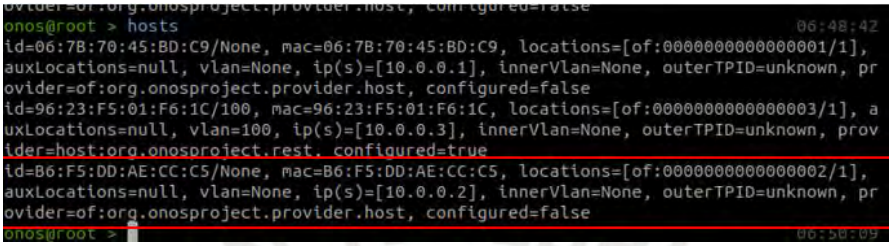


Figura 4-26 Estado inicial de la ip en el host.
Fuente: Elaboración propia

Luego de verificar el estado del host objetivo, se procede a el ingreso del *intent* (Figura 4-27) cuyo objetivo es la asignación de una nueva ip a un host. La intención expresada es procesada, en principio, por la plataforma de NLU (Dialogflow), la cual envía los parámetros detectados en el *intent* a el *intent engine* que se encargara de hacer las validaciones necesarias para hacer los cambios en la red SDN.

Figura 4-27 Ingreso del Intent en la plataforma del servidor web.
Fuente: Elaboración propia

Finalmente, los cambios podrán ser realizados o se verificaran que no pueden realizar. En la figura 4-28 podemos observar que luego de que el host tuviera una ip de 10.0.0.3 el valor actual es 10.0.015.

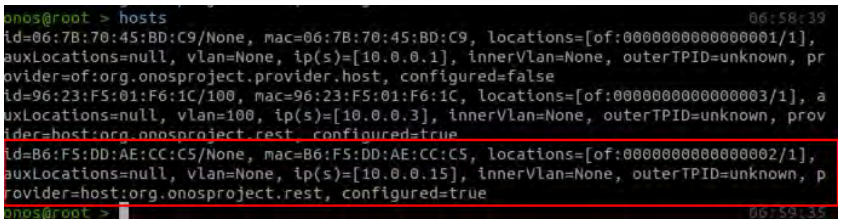


Figura 4-28 Estado final de la vlan del host en cuestión.
Fuente: Elaboración propia

4.2.4 Asignación/cambio de un host a un switch

Durante esta prueba se realizó el ingreso de un *intent* con el objetivo que se asigne el host a otro switch. En el siguiente grafico se puede observar que el host al que se le desea asignar la nueva ip ya posee ninguna.

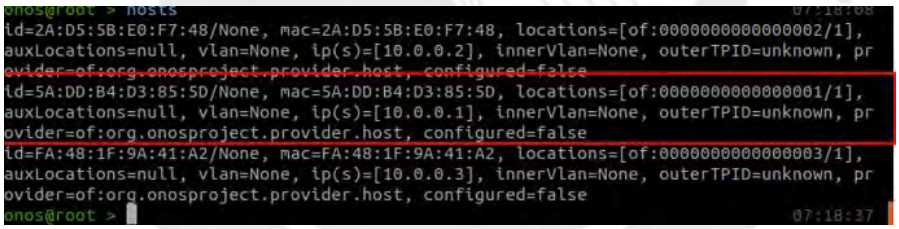


Figura 4-29 Ubicación actual del host

Fuente: Elaboración propia

Luego de verificar el estado del host objetivo, se procede a el ingreso del *intent* (Figura 4-30) cuyo objetivo es el cambio de ubicación del host a un nuevo switch. La intención expresada es procesada, en principio, por la plataforma de NLU (Dialogflow), la cual envía los parámetros detectados en el *intent* a el *intent engine* que se encargara de hacer las validaciones necesarias para hacer los cambios en la red SDN.

Figura 4-30 Ingreso del Intent en la plataforma del servidor web.

Fuente: Elaboración propia

Finalmente, los cambios podrán ser realizados o se verificaran que no pueden realizar. En la figura 4-31 podemos observar que luego de que el host estuviera en el switch of:0000000000000001 se encuentra en of:00000000000000004. Tambien se puede observar de forma gráfica en la figura 4-32.

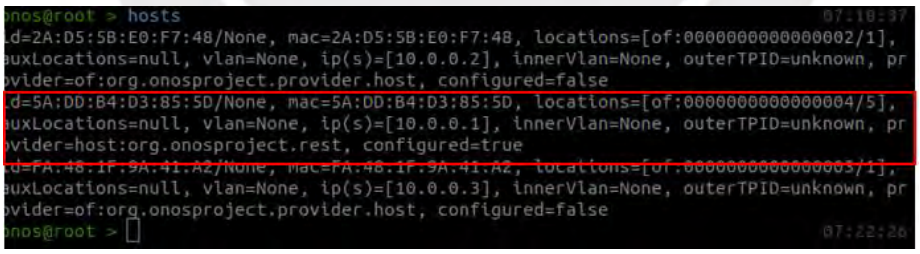


Figura 4-3125 Ubicación final del host

Fuente: Elaboración propia

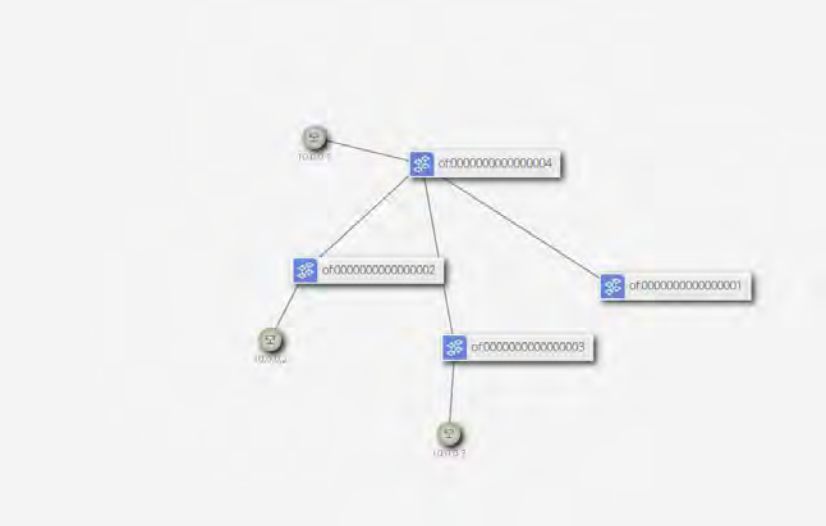


Figura 4-3226 Ubicación final del host(Grafica de la topología)

Fuente: Elaboración propia

Conclusiones

- La solución propuesta haciendo uso de un *intent engine* en la presente tesis se siguió debido a que no existe un consenso en la topología de una red IBNM de las diversas implementaciones descritas en el documento.
- La implementación de el algoritmo desarrollado para el *intent engine* podrá trabajar correctamente para la topología propuesta. Es posible que se necesite ajustes para otros casos y para futuros versiones de software.
- Las pruebas realizadas llegaron a satisfacer lo requerido para el entorno presentado. Si se desea aplicar intenciones sobre servicios y configuraciones más avanzados no se podrán realizar ya que el sistema funciona solo para servicios básicos y descrito en la lista de intenciones.
- En el desarrollo de la solución implementada se tuvo en consideración que la topología de la red SDN propuestas son la adecuada para satisfacer las necesidades de una MYPE. Además, se pudo comprobar que los cambios realizados para la implementación de la red basada en la intención fueron inmediatos y no hubo conflictos que afectaron el rendimiento de la red.
- La configuración realizada en los equipos de la topología virtualizada fueron realizados sin inconvenientes, esto puede cambiar si se desea hacer uso del sistema con infraestructura real ya que se tendrá que tener en cuenta la implementación del *southbound* entre el controlador y el equipamiento.
- Existen diversas aplicaciones desarrolladas para satisfacer los diversos usos que se quiere dar a la red haciendo uso del controlador ONOS. Estas aplicaciones deberán ser instaladas y usadas de forma manual no de la forma desarrollada en este trabajo de tesis.

Recomendaciones

- Si existe un deseo de implementación de una red basada en la intención(IBN) basado en el diseño propuesto para futuros proyectos de mayor magnitud, es recomendable poder contar con un servidor que posea características mayores de software y hardware para evitar inconvenientes con el procesamiento realizado.
- La implementación del proyecto en un sistema de mayor envergadura podría mejorar su eficiencia en proyectos más particulares.

Observaciones

- Durante el desarrollo de la presente tesis se comprobó el funcionamiento de la gestión mediante IBNM mediante el ingreso de diversos *intents*. De darse el caso que se desee procesar otro tipo de intenciones o acciones en la red se tendrá que entrenar la plataforma de comprensión de lenguaje natural y agregar líneas de código en el algoritmo implementado para satisfacer lo deseado.

Bibliografía

[1] Gartner, “Look Beyond Network Vendors for Network Innovation,” 2018.

[2] “Uso de Las TICs en Las MYPES, lo que debes saber,” *Ideas y Tecnologías*, 2019. [Online]. Available: https://www.ideasytecnologias.com/blog/uso-de-las-tics-en-las-mypes-lo-que-debes-saber/. [Accessed: 11-Nov-2020].

[3] Sandvine, “The Global Internet Phenomena Report COVID-19 Spotlight,” 2020.

[4] Cisco, “Cisco Annual Internet Report (2018–2023),” 2018.

[5] S. Pueblas, Martin; Gyurindak, Steve; Strika, John; Kachalia, Rahul; Hamilton, Dan; Tenneti, “Small Enterprise Design Profile Reference Guide,” *Cisco*, pp. 16–39, 2010.

[6] Red Hat, “Modernize Your Network With Red Hat Ansible Automation Platform,” 2019.

[7] M. Mortensen, “ECONOMIC BENEFITS OF NETWORK AUTOMATION EXECUTIVE SUMMARY,” 2020.

[8] Google Cloud, “Conceptos básicos de Dialogflow | Documentación de Dialogflow.” [Online]. Available: https://cloud.google.com/dialogflow/docs/basics?hl=es-419. [Accessed: 22-Oct-2020].

[9] Paul M. Muchinsky, “Las Pyme en economía emergentes,” *Psychol. Appl. to Work An Introd. to Ind. Organ. Psychol. Tenth Ed. Paul*, vol. 53, no. 9, pp. 1689–1699, 2012.

[10] L. Behringer, M; Pritikin, M; Bjarnason, S; Clemm, A; Carpenter, B; Jiang, S; Ciavaglia, “RFC 7575 - Autonomic Networking: Definitions and Design Goals,” *IETF Network Working Group Internet-Draft*, 2015. [Online]. Available:

https://tools.ietf.org/html/rfc7575. [Accessed: 12-Nov-2020].

[11] J. Clemm, A., Ciavagliam, L., Granville, L., Tantsura, "Intent-Based Networking - Concepts and Overview," *IETF Network Working Group Internet-Draft*, 2019. [Online]. Available: https://tools.ietf.org/id/draft-clemm-nmrg-dist-intent-03.html. [Accessed: 07-Oct-2020].

[12] M. Sivakumar, K., Chandramouli, "Concepts of Network Intent," *IETF Network Working Group Internet-Draft*, 2017. [Online]. Available: https://tools.ietf.org/id/draft-moulchan-nmrg-network-intent-concepts-00.html. [Accessed: 07-Oct-2020].

[13] A. Lerner, "Intent-based Networking," *Gartner*, 2017. [Online]. Available: https://blogs.gartner.com/andrew-lerner/2017/02/07/intent-based-networking/. [Accessed: 12-Nov-2020].

[14] P. Widmer Zürich, E. J. Scheid, B. B. Rodrigues, and B. Stiller, "Design and Implementation of an Intent-based Blockchain Selection Framework," 2020.

[15] E. Zeydan and Y. Turk, "Recent Advances in Intent-Based Networking: A Survey," in *IEEE Vehicular Technology Conference*, 2020, vol. 2020-May, pp. 1–5, doi: 10.1109/VTC2020-Spring48590.2020.9128422.

[16] M. Riftadi and F. Kuipers, "P4I/O: Intent-Based Networking with P4," in *Proceedings of the 2019 IEEE Conference on Network Softwarization: Unleashing the Power of Network Softwarization, NetSoft 2019*, 2019, pp. 438–443, doi: 10.1109/NETSOFT.2019.8806662.

[17] A. Chaudhari *et al.*, "VIVoNet: Visually-represented, Intent-based, Voice-assisted Networking," *Int. J. Comput. Networks Commun.*, vol. 11, no. 2, pp. 1–13, Apr. 2019, doi:

10.5121/ijcnc.2019.11201.

[18] D. Comer and A. Rastegatnia, "OSDF: An Intent-based Software Defined Network Programming Framework," in *Proceedings - Conference on Local Computer Networks, LCN*, 2019, vol. 2018-Octob, pp. 527–535, doi: 10.1109/LCN.2018.8638149.

[19] M. Jain *et al.*, "Intent-Based, Voice-Assisted, Self-Healing SDN Framework," *J. Netw. Commun. Emerg. Technol.*, vol. 10, no. 2, 2020.

[20] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015, doi: 10.1109/JPROC.2014.2371999.

[21] T. Bakhshi, "State of the art and recent research advances in software defined networking," *Hindawi*, vol. 2017, 2017, doi: 10.1155/2017/7191647.

[22] A. Prajapati, A. Sakadasariya, and J. Patel, "Software defined network: Future of networking," in *Proceedings of the 2nd International Conference on Inventive Systems and Control, ICISC 2018*, 2018, pp. 1351–1354, doi: 10.1109/ICISC.2018.8399028.

[23] "SDN - Datacomm Diangraha." [Online]. Available: https://www.datacomm.co.id/en/telco/sdn/. [Accessed: 21-Oct-2020].

[24] "Home - OpenDaylight." [Online]. Available: https://www.opendaylight.org/. [Accessed: 26-Sep-2021].

[25] P. Monika, R. M. Negara, and D. D. Sanjoyo, "Performance analysis of software defined network using intent monitor and reroute method on ONOS controller," *Bull. Electr. Eng. Informatics*, vol. 9, no. 5, pp. 2065–2073, Oct. 2020, doi: 10.11591/eei.v9i5.2413.

[26] "ONOS - ONOS - Wiki." [Online]. Available: https://wiki.onosproject.org/display/ONOS/ONOS. [Accessed: 13-Oct-2020].

[27] "Intent Framework - ONOS - Wiki." [Online]. Available: https://wiki.onosproject.org/display/ONOS/Intent+Framework. [Accessed: 14-Oct-2020].

[28] A. Campanella, "Intent Based Network Operations - IEEE Conference Publication," *IEEE Access*, 2019. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8696962. [Accessed: 13-Mar-2020].

[29] "Appendix B: REST API - ONOS - Wiki." [Online]. Available: https://wiki.onosproject.org/display/ONOS/Appendix+B%3A+REST+API. [Accessed: 14-Oct-2020].

[30] "onos/web/api/src/main/resources/definitions at master · opennetworkinglab/onos · GitHub." [Online]. Available: https://github.com/opennetworkinglab/onos/tree/master/web/api/src/main/resources/definitions. [Accessed: 26-Sep-2021].

[31] M. Canonico and L. De Russis, "A Comparison and Critique of Natural Language Understanding Tools," 2018.

[32] D. Mamgain, "Dialogflow vs Lex vs Watson vs Wit vs Azure Bot | What to Choose?," *kommunicate*, 2019. [Online]. Available: https://www.kommunicate.io/blog/dialogflow-vs-lex-vs-watson-vs-wit-vs-azure-bot/. [Accessed: 24-Nov-2020].

[33] A. Listo, "Experiencias y recomendaciones de uso de IBM Watson y Google DialogFlow," *enzyme advising group*, 2020. [Online]. Available:

https://blog.enzymeadvisinggroup.com/uso-ibm-watson-google-dialogflow. [Accessed: 15-Dec-2020].

[34] "Ley de Promoción y Formalización de la Micro y Pequeña Empresa," *Diario El Peruano*, 2003.

[35] ComexPerú, "ComexPerú - Sociedad de Comercio Exterior del Perú," *ComexPerú*, 2020. [Online]. Available: https://www.comexperu.org.pe/articulo/las-mype-peruanas-en-2019-y-su-realidad-ante-la-crisis. [Accessed: 18-Oct-2020].

[36] "Ley de Promocion de la Competitividad, Formalizacion y Desarrollo de la Micro y Pequeña Empresa y del Acceso al Empleo Decente," *Diario El Peruano*, 2008.

Anexos

Código para la generación de la topología full mesh en Mininet

```
from mininet.topo import Topo

class MyTopo( Topo ):
   "Simple topology example."

   def __init__( self ):
      "Create custom topo."
      Topo.__init__( self )

      # Add hosts and switches
      H1 = self.addHost( 'h1' )
      H2 = self.addHost( 'h2' )
      H3 = self.addHost( 'h3' )
      S1 = self.addSwitch( 's1' )
      S2 = self.addSwitch( 's2' )
      S3 = self.addSwitch( 's3' )

      # Add links
      self.addLink( H1, S1 )
      self.addLink( H2, S2 )
      self.addlink( H3, S3 )
      self.addLink( S1, S2 )
      self.addLink( S1, S3 )
      self.addLink( S2, S3 )


topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Código para la generación de la topología full de dos niveles en Mininet

```
from mininet.topo import Topo

class MyTopo( Topo ):
   "Simple topology example."

   def __init__( self ):
      "Create custom topo."
      Topo.__init__( self )

      # Add hosts and switches
      H1 = self.addHost( 'h1' )
      H2 = self.addHost( 'h2' )
      H3 = self.addHost( 'h3' )
      S1 = self.addSwitch( 's1' )
      S2 = self.addSwitch( 's2' )
      S3 = self.addSwitch( 's3' )
      S4 = self.addSwitch( 's4' )

      # Add links
      self.addLink( H1, S1 )
      self.addLink( H2, S2 )
      self.addlink( H3, S3 )
      self.addLink( S1, S4 )
      self.addLink( S2, S4 )
      self.addLink( S3, S4 )


topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Código de el Algoritmo del Intent engine desarrollado para el procesamiento del Intent

```
import json
import random
from typing import Match
import requests
from flask import Flask, render_template, request, make_response


app = Flask(__name__)

URL = 'http://192.168.123.1:8181/onos/v1/'


@app.route('/')
def index():
  return render_template('index.html')


@app.route('/webhook', methods=['POST'])
def webhook():
  if request.method == "POST":
    req = request.get_json(silent=True, force=True)
    res = processRequest(req)
    res = json.dumps(res, indent=4)
    r = make_response(res)
    r.headers['Content-Type'] = 'application/json'
    return r


@app.route('/hello')
def HelloWorld():
  return "hello world"


def processRequest(req):
  query_response = req["queryResult"] #query enviado por dialogflow es dict type
  '''print(query_response)'''
  text = query_response.get('queryText', None) #.get son operaciones en diccionarios
  parameters = query_response.get('parameters', None)
  accion = parameters.get('actions', None) #
  hostnames = parameters.get('netHosts', None)
  switches = parameters.get('netDevices', None) # hostnames es una lista revisar
  keyword = parameters.get('keywords', None) #keywords es una Lista
  num = parameters.get('number', None) # num es un int
  exclude = parameters.get('exlude', None)
  objeto = parameters.get('Objetos', None) #objeto es una Lista
  desvio = parameters.get('depaso', None)
  ipadd= parameters.get('any', None)
                    #'actions' para obtener los valores de la lista se necesita un for
```

```
  if (hostnames[0] != '') and (accion[0] != ''):       #### # creo que las listas vacias deberian ser
len(nombre) != 0

      json_hosts = requests.get(url=URL+'hosts', auth=('onos', 'rocks'))# json_hosts value type is
<class 'requests.models.Response'>
      hosts_list = json.loads(json_hosts.text)       # pasando a json la respuesta, json_hosts.text
obtiene el str que envio el controlador
      #info_hosts = hosts_list['hosts']

      dictHostMac= {}   #diccionario para mapear los hosts con sus macs
      dictHostId= {}   #diccionario para mapear los hosts con sus ids

      for item in hosts_list['hosts']:       #recorremos cada item de la lista de hosts

         mac = item['mac']
         hostid = item['id']
         for x in item['locations']:
            swid=x['elementId']
         #llenando los diccionarios h1 -> id y h1 -> mac
         if(swid == 'of:0000000000000001'):
            dictHostMac['h1'] = mac
            dictHostId['h1'] = hostid
         elif (swid == 'of:0000000000000002'):
            dictHostMac['h2'] = mac
            dictHostId['h2'] = hostid
         elif (swid == 'of:0000000000000003'):
             dictHostMac['h3'] = mac
             dictHostId['h3'] = hostid

      if (accion[0] == 'bloquear') and (keyword == 'trafico'): # validacion si se va a bloquear todo
el trafico de un host a otro host
         hid = {}
         hmc = {}
         for i in range(0, len(hostnames)):
            hid['idHost' +  str(i+1)]=dictHostId[hostnames[i]]
            hmc['macHost'+  str(i+1)]=dictHostMac[hostnames[i]]

         getpath(hid['idHost1'], hid['idHost2'], hmc['macHost1'], hmc['macHost1'])
         del hid  #Se borra los dict para futuros ejecuciones
         del hmc  #Se borra los dict para futuros ejecuciones

      elif ((accion[0] == 'modificar') or (accion[0] == 'asignar')) and (keyword == 'vlan'):
#Validacion para modificacion/asignacion de vlan de los hosts
         hostmac = dictHostMac[hostnames[0]]

         vlanhostops(hosts_list,hostmac,num )

      elif ((accion[0] == 'modificar') or (accion[0] == 'asignar'))  and (keyword == 'ip'):
         hostmac = dictHostMac[hostnames[0]]
         iphostops(hosts_list,hostmac,ipadd[0])
```

```
        elif (accion[0] == ('asignar' or 'unir' or 'cambiar')) and (switches[0] != ''): #### # creo que las
listas vacias deberian ser len(nombre) != 0
            hostmac = dictHostMac[hostnames[0]]
            dictSw = dictsw()
            swhostops(hosts_list, switches[0],hostmac, dictSw)

        elif (accion[0] == 'conectar'):
            if (desvio == ''):
                hostIntent(hostnames[0], dictHostId ) # funcion para conectar directamente 2 hosts

        elif (accion[0] == 'conectar') and (desvio == 'atraves'):
            pointIntent(hosts_list, hostnames[0], dictHostMac, switches[0])


    elif (switches[0] != '') and (accion[0] != ''):
        accion_sw(accion[0], switches[0])

    res = get_data()
    return res




# funcion para realizar una conexion entre hosts sin definir flows.
def hostIntent(intenthosts, dicthostsId):

    numHostIntent = len(intenthosts)
    if  (numHostIntent == 2):  # crear intencion de conectar 2 hosts con el camino mas corto.
        dictVar= dict()
        for i in range(0 , numHostIntent):
            dictVar['id'+str(i+1)] =dicthostsId[intenthosts[i]]

        data = {"type": "HostToHostIntent",  "appId": "org.onosproject.ovsdb", "priority": 40001,
"one": dictVar['id1'],
                "two": dictVar['id2']}
        stream = json.dumps(data)
        requests.post(url=URL+'intents', data=stream, auth=('onos', 'rocks'))
        del dictVar #Borra el dict para futuros ejecuciones

    # crear intencion de conectar 2 hosts atraves de otro camino.
def pointIntent(ListaHosts, intentHosts, dictHostsMac, intentSw):
    dictSwitch = dictsw()
    intentSWId= dictSwitch[intentSw]
    intentswport= {}

    for i in range(0,len(intentHosts)):

        mac= dictHostsMac[intentHosts[i]]

        for item in ListaHosts['hosts']:
            if(mac == item['mac']):
```

```
            for locat in item['locations']:
                device = locat['elementId']
                devport = locat['port']

            listalinksdev= getlinksDev(device)
            for itemlink in listalinksdev['links']:
                if (intentSWId == itemlink['src']['device']):
                    devport2= itemlink['dst']['port']
                    intentswport['x'+str(i+1)]= itemlink['src']['port']

            datosida = {"type": "PointToPointIntent",  "appId": "org.onosproject.ovsdb", "priority":
5000, "constraints": [{
            "inclusive": False, "types": ["OPTICAL"], "type": "LinkTypeConstraint"}], "ingressPoint":
{
            "port": devport, "device": device},  "egressPoint": {"port": devport2, "device": device}}

            datosvuelta = {"type": "PointToPointIntent",  "appId": "org.onosproject.ovsdb", "priority":
5000, "constraints": [{
            "inclusive": False, "types": ["OPTICAL"], "type": "LinkTypeConstraint"}], "ingressPoint":
{
            "port": devport2, "device": device},  "egressPoint": {"port": devport, "device": device}}

            streamida = json.dumps(datosida)
            streamvuelta = json.dumps(datosvuelta)
            requests.post(url=URL+'intents', data=streamida, auth=('onos', 'rocks'))
            requests.post(url=URL+'intents', data=streamvuelta, auth=('onos', 'rocks'))


   datosidasw = {"type": "PointToPointIntent",  "appId": "org.onosproject.ovsdb", "priority": 5000,
"constraints": [{
            "inclusive": False, "types": ["OPTICAL"], "type": "LinkTypeConstraint"}], "ingressPoint":
{
            "port": intentswport['x1'], "device": intentSWId},  "egressPoint": {"port":
intentswport['x2'], "device": intentSWId}}

   datosvuelsw = {"type": "PointToPointIntent",  "appId": "org.onosproject.ovsdb", "priority": 5000,
"constraints": [{
            "inclusive": False, "types": ["OPTICAL"], "type": "LinkTypeConstraint"}], "ingressPoint":
{
            "port": intentswport['x2'], "device": intentSWId},  "egressPoint": {"port":
intentswport['x1'], "device": intentSWId}}

   streamidasw = json.dumps(datosidasw)
   streamvueltasw = json.dumps(datosvuelsw)
   requests.post(url=URL+'intents', data=streamidasw, auth=('onos', 'rocks'))
   requests.post(url=URL+'intents', data=streamvueltasw, auth=('onos', 'rocks'))


def accion_sw(action, equipo):
   dictdev = dictsw()
   for i in range(0,len(action)):
```

```
    if (action == 'borrar'):
        idequipo=dictdev[equipo]
        id = hexformater(idequipo)
        requests.delete(url=URL+'devices/'+ id, auth=('onos', 'rocks'))

#funcion que se encarga de hace la validacion de vlan y cambiar/asignar vlans a los hosts
def vlanhostops(hosts_list, hmac, numvlan):

  for item in hosts_list['hosts']:
    if (hmac == item['mac']) and (numvlan != item['vlan']):
        vlan = item['vlan']
        mac = item['mac']
        ip = item['ipAddresses'][0]
        for listlocat in item['locations']:
            deviceId= listlocat['elementId']
            port = listlocat['port']

        data = {"mac": mac,  "vlan": numvlan,  "ipAddresses": ip,  "locations": [{"elementId":
deviceId,
                "port": port}]}

        stream = json.dumps(data)
        # hacemos el post  del host con en la nueva vlan
        requests.post(url=URL+'hosts', data=stream, auth=('onos', 'rocks'))

        # hacemos el delete para cambiar la vlan
        requests.delete(url=URL+'hosts/'+mac + '/' +
                    vlan, auth=('onos', 'rocks'))

    elif (hmac == item['mac']) and (numvlan == item['vlan']):
        print('Ya existe el host con esa Vlan')

    else:
        print('El host no existe')

#funcion para cambiar la ip de un host(op similar a un DhCP)
def iphostops(info_hosts, hmac, newip):

   for item in info_hosts['hosts']:
    if (hmac == item['mac']) and (newip != item['ipAddresses'][0]):
        for listlocat in item['locations']:
            deviceId= listlocat['elementId']
            port = listlocat['port']

        data={"mac": item['mac'],  "vlan": item['vlan'],  "ipAddresses": newip,  "locations":
[{"elementId": deviceId,
        "port": port}]}

        stream = json.dumps(data)
        # hacemos el post  del host con en la nueva vlan
        requests.post(url=URL+'hosts', data=stream, auth=('onos', 'rocks'))
```

```
            # hacemos el delete para cambiar la vlan
            requests.delete(url=URL+'hosts/'+ item['mac'] + '/' +
                        item['vlan'], auth=('onos', 'rocks'))

        elif (hmac == item['mac']) and (newip == item['ipAddresses'][0]):
            print('Ya existe el host con esa ip')
        else:
            print('El host no existe')

def dictsw(): #funcion para obtener un diccionario de switches de la topologia(sw -> id)
    json_dev = requests.get(url=URL+'devices', auth=('onos', 'rocks'))# json_hosts value type is
<class 'requests.models.Response'>
    dev_list = json.loads(json_dev.text)
    dictSwId={}   #diccionario para mapear los hosts con sus macs

    for item in dev_list['devices']:        #recorremos cada item de la lista de hosts

        id = item['id']
        numsw = item['chassisId']

        dictSwId['sw'+numsw] = id
    return dictSwId

def getlinks(): # obtencion de la lista de links
    json_links = requests.get(url=URL+'links', auth=('onos', 'rocks'))# json_hosts value type is
<class 'requests.models.Response'>
    links_list = json.loads(json_links.text)

    return links_list

def getlinksDev(devId):
    id = hexformater(devId)
    json_linksdev = requests.get(url=URL+'links'+'?device='+id, auth=('onos', 'rocks'))# json_hosts
value type is <class 'requests.models.Response'>
    links_listdev = json.loads(json_linksdev.text)
    return links_listdev

def swhostops(info_hosts, IntentSw, intenthostmac, dicDevices):
    newSwID = dicDevices[+[0]]  # ya que intentsw es una lista
    usedport=0
    linkport=0
    for item in info_hosts['hosts']:
        for listlocat in item['locations']:
            deviceId = listlocat['elementId']

        if (intenthostmac == item['mac']) and (newSwID != deviceId ):
            for x in info_hosts['hosts']: # para revisar si hay hosts conectados en el mismo device
                for xport in x['locations']: #para saber si otro host esta en el mismo device  que puerto
usa
                    if (newSwID == xport['elementId']):
```

```
                usedport = int(xport['port'])
        linksNewDevice = getlinksDev(newSwID)#tenemos que verificar que puertos estan
usando los links(device a device)
        for itemlink in linksNewDevice['links']:
            if (newSwID == itemlink['src']['device']):
                linkport=int(itemlink['src']['port'])

        desport= random.randint(0,32)
        if (usedport != desport):
            if(linkport != desport):
                port =desport

        #eliminamos el host anteiro
        requests.delete(url=URL+'hosts/'+ item['mac'] + '/' + item['vlan'], auth=('onos', 'rocks'))

        data={"mac": item['mac'],  "vlan": item['vlan'], "configured":item['configured'],
"ipAddresses": item['ipAddresses'][0],  "locations": [{"elementId": newSwID,
        "port": port}]}

        stream = json.dumps(data)
        # hacemos el post  del host con en la nueva vlan
        requests.post(url=URL+'hosts', data=stream, auth=('onos', 'rocks'))


    elif (intenthostmac == item['mac']) and (newSwID == deviceId ):
        print('el host ya esta en el device')


# funcion para formaterar a hexadecimal una mac or host id
def hexformater(var):
  var1 = var.replace(':','%3A')
  resvar= var1.replace('/','%2F')
  return resvar

#funcion que obtiene la lista de links desde un source(hostid) a un destination(hostid)
def getpath(srcid, dstid,srcmac, dstmac): #id de cada host
  hexsrcid= hexformater(srcid)
  hexdstid= hexformater(dstid)

  reqpath= requests.get(url=URL+'paths/'+hexsrcid+'/'+hexdstid , auth=('onos', 'rocks'))
  path_list = json.loads(reqpath.text)
  #info_paths= jsonpath['paths']
  for item in path_list['paths']:
    numlinks= int(item['cost'])
    numPathlinks = len(item['links'])
    count=1
    for itemlinks in item['links']:

      while (count < numPathlinks):
        device=itemlinks['dst']['device']
```

```
            droptrafico(srcmac, dstmac, device ) # cada device encontrado(menos el ultimo) se
envia a la funcion
            count+=1                        #para generar los flows


#funcion que generara los flows para en cada enlace para que se dropee trafico que vaya de un
srcmac a un dstmac
def droptrafico(src, dst, device):
    #falta agregar el appid(no necesario)
   data={"flows": [
      {
      "priority": 50000,
      "timeout": 0,
      "isPermanent": True,
      "deviceId": device    , #sacamos de getpath
      "treatment": { # si dropeamos todo el trafico el 'treatment' debe estar vacio{}
      },
      "selector": {
         "criteria": [
         {
            "type": "ETH_DST",
            "mac": src       #no obtenemos del getpath
         },
         {
            "type": "ETH_SRC",
            "mac": dst #no obtenemos del get path
         }
         ]
      }
      }
      ]
   }
   stream= json.dumps(data)
   requests.post(url=URL+'flows',data=stream, auth=('onos', 'rocks'))
#http://192.168.123.1:8181/onos/v1/flows?appId=my.nice.app%20 (request url con el appId)

def get_data():
   speech = "Realizando Configuracion"
   return{
      "fulfillmentText": speech,
   }


if __name__ == '__main__':
   app.debug = True
   app.run(host='0.0.0.0', port=5000)
```