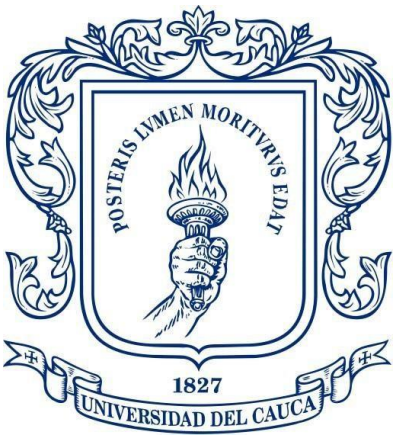
MÉTODO DE CONTROL COGNITIVO APLICADO AL ENRUTAMIENTO DE UNA RED BAJO ARQUITECTURA SDN/NFV

José Luis Rivera Hurtado

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones
Programa de Maestría en Electrónica y Telecomunicaciones
GNTT- Grupo I+D Nuevas Tecnologías en Telecomunicaciones
Popayán, Diciembre 2018

MÉTODO DE CONTROL COGNITIVO APLICADO AL ENRUTAMIENTO DE UNA RED BAJO ARQUITECTURA SDN/NFV



Trabajo de Grado para optar al título de Magister en Electrónica y Telecomunicaciones

José Luis Rivera Hurtado

Director: PhD. Ing. José Giovanny López Perafán

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones
Programa de Maestría en Electrónica y Telecomunicaciones
GNTT- Grupo I+D Nuevas Tecnologías en Telecomunicaciones
Popayán, Diciembre 2018

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ACRÓNIMOS

ACLs	*Access Control List,* (Lista de Control de Acceso)
ACO	*Ant Colony Optimization*, (Optimización por Colonia de Hormigas)
AI	*Artificial Intelligence*, (Inteligencia Artificial).
APIs	*Applications*, (Aplicaciones).
ATM	*Asynchronous Transfer Mode,* (Modo de Transferencia Asincrónico).
BCP	*Burst Control Packet*, (Paquete de Control de la Ráfaga).
BGP	*Border Gateway Protocol*, (Protocolo de Puerta de Enlace de Frontera)
BP	*Blocking Probability*, (Probabilidad de Bloqueo).
CAPEX	*Capital Expenses,* (Costos de Capital)
CDS	*Cognitive Decisión System,* (sistema de decision cognitiva)
CMS	*Control Mananger System,* (Sistema de administracion de control)
CRN	*Cognitive Routing Network*, (Red con Enrrutamiento Cognitivo).
EA	*Evolutionary Algorithms*, (Algoritmos Evolutivos)
EDA	*Estimation Distribution Algorithms,* (Estimación de los Algoritmos de Distribución)
GRASP	*Randomized and Adaptive Processes of Voracious Search*, (Procesos Aleatorizados y Adaptativos de Búsqueda Voraz)
HTTPS	*Hypertext Transfer Protocol Secure*, (Protocolo Seguro de Transferencia de hipertexto)
MPLS	*Multiprocol Label Switching*, (Conmutacion de Etiquetas Multiprotocolo).
NCP	*Network Control Program,* (Programa de Control de Red)
NFV	*Network function virtualization*, (Virtualización de Funciones de Red)
NFV M&O	*Network Functions Virtualization Management and Orchestration,* (Orquestacion y administracion de virtualizacion de funciones de red)
NFVI	*Network Functions Virtualization Infrastructure,* (infraestructura de virtualizacion de funciones de red)
NOS	*Network Operating System*, (Sistema Operativo de Redes).
NSFNET	*National science foundation network*, (Red de fundacion cientificas nacionales)
IEEE	*Institute of Electrical and Electronics Engineers*, (Instituto de Ingeniería Eléctrica y Electrónica).
ILS	*Iterated Local Search*, (Búsqueda Local Iterada)
IP	*Internet Protocol*, (Protocolo de Internet).
ISP	*Internet Service Provider,* (Proveedor de Servicio de Internet)
IT	*Information Technology,* (Tecnologias de la informacion)
OPEX	Operational Expenses, (Costos Operativos)
OSI	*Open System Interconnection*, (Modelo de Interconexion de Sistemas Abiertos).
PHP	*Personal Hypertext processor,* (Procesador de Hipertexto Personal)
PSO	*Particle Swarm Optimization,* (Optimización por Nubes de Partículas).
QoS	*Quality of Service*, (Calidad de Servicio).
RCP	*Rate Control Protocol, (*Protocolo de control de velocidad*)*
SA	*Simulated Annealing*, (Recocido Simulado)
SDN	*Software Defined Network*, (Redes Definidas por Software)
SS	*Scattered Search,* (Búsqueda Dispersa)
TCP	*Transmission Control Protocol*, (Protocolo de Control de Transmisión)

TS	*Taboo Search*, (Búsqueda Tabú)
VL	*Virtual Link,* (Enlace Virtual)
VN	*Virtual Node,* (Nodo Virtual)
VNF	*Virtualized Network Function,* (Funciones de Red Virtualizadas)
VNS	*Variable neighborhood search,* (Búsqueda de vecindad variable)
VTN	*Virtual Tenant Network*, (Redes Virtuales Multi-vecino)

Introducción

Durante la última década, las redes de telecomunicaciones han tenido un crecimiento de manera exponencial trayendo consigo nuevos retos en servicios, usuarios, infraestructura, cobertura, ancho de banda, seguridad, entre otros. Esto sin duda representa una gran dificultad en la actualidad debido a que la arquitectura de la red tradicional no está preparada para los requerimientos de los usuarios del momento y a futuro. En este contexto, el concepto SDN/NFV (*Software Defined Networking/Network Function Virtualization*) surge como una tecnología que promete cambiar radicalmente la administración de las redes (LAN, MAN y WAN) aportando beneficios y soluciones a las mencionadas dificultades, introduciendo inteligencia, flexibilidad y escalabilidad, convirtiendo la red en una herramienta de negocio que a través del desarrollo de API, (*Application Programming Interfaces*) que modificarán dinámicamente los servicios proporcionados por la red y los procesos para su prestación[86].

Es así como la principal característica de las redes definidas por software se encuentra en la división de la infraestructura de la inteligencia. Con el fin de programar a través de un entorno de programación abierto. SDN desagrega la arquitectura de las redes tradicionales para mejorar las características de la red, en otras palabras, realizar una separación en planos de datos y control, lo cual puede emplearse para “personalizar” el funcionamiento de la red en entornos especializados [1].

En SDN el sistema operativo es el cerebro de la red y controla todo el estado de la misma y su comportamiento, observando y actuando de acuerdo a los requerimientos configurados en el plano de control, además, SDN centraliza la inteligencia de la red e introduce la posibilidad de programación de los dispositivos de la infraestructura de red explotando la opción de nuevos conceptos cognitivos que permitan un dinamismo en las redes inalámbricas y cableadas presentes y futuras [1].

Igualmente, las redes cognitivas en el futuro van a tener el potencial de proporcionar una red adaptable, dinámica, robusta y de alto ancho de banda mediante la centralización del control, permitiéndole observar el estado actual de la red para su análisis y adaptación de una manera más eficiente a los cambios detectados[2], por lo tanto la red cognitiva propone automatizar las redes mediante la detección de dichos cambios en tiempo real adaptándola a estos y realizando acciones para la aplicación de sistemas de control de bucle para que aprenda y pueda actualizarse a sí misma para realizar acciones futuras sin intervención humana [2].

Bajo estas premisas, se propone el presente trabajo de grado con el cual se busca emplear la tecnología SND/NFV para diseñar, programar y probar mediante simulación los elementos de red de la capa de infraestructura del modelo SDN, añadiendo puntualmente al proceso de enrutamiento de la red SDN/NFV algunas funciones cognitivas (diseño de un método cognitivo) y empleando la meta heurística ACO *(Ant Colony Optimization*) Mono-Objetivo para realizar dicho proceso, igualmente, al realizar la prueba del método cognitivo mediante simulación, se lleva a cabo el análisis de desempeño del sistema de red mediante la evaluación de la probabilidad de bloqueo BP (*Blocking Probability*) de cada enlace.

De esta forma, en el nivel de Aplicación de la arquitectura SDN, se incluiran las funciones cognitivas tomadas del modelo de estados del ciclo cognitivo básico (observar, decidir, aprender y actuar) cuyo aprendizaje alimenta una base de datos con cada una de las rutas seleccionadas mediante el empleo de la meta heurística ACO.

En ese sentido, para llegar a su mejor término, se define un objetivo general “Analizar el desempeño de una red SDN/NFV mediante la aplicación de un método cognitivo basado en ACO” el cual a su vez estará dividido en 3 objetivos específicos “Identificar las diferentes características fundamentales en la literatura seleccionada de un control cognitivo (CHRON) basado en la meta heurística ACO mono-objetivo en el proceso de enrutamiento para una red bajo arquitectura SDN/NFV bajo el parámetro de probabilidad de bloqueo con flujo de tráfico Elefante y/o Ratón”, “Desarrollar un algoritmo cognitivo basado en CHRON con meta heurística ACO mono-objetivo aplicado al proceso de enrutamiento de la Red SDN/NFV de acuerdo a un volumen de flujo de tráfico Elefante y/o Ratón” y “Evaluar el método cognitivo del ítem anterior mediante simulación en las topologías NSNET e IRIS con flujo de tráfico Elefante y/o Ratón para analizar la probabilidad de bloqueo en la Red inicial SDN/NFV y la red con el método cognitivo en la función de enrutamiento” los cuales se alcanzan mediante el desarrollo de una serie de actividades.

De esta manera, en cada uno de los capítulos que se describen a continuación se expone el desarrollo de las actividades que permitieron dar cumplimiento a los objetivos planteados en el anteproyecto:

En el primer capítulo se da una visión general de las tecnologías que se utilizan en la caracterización de la red a simular, sobre la cual se implementará el algoritmo cognitivo, todo esto con el propósito de establecer una base conceptual clara de la literatura científica existente al respecto. Seguidamente en el segundo capítulo se da a conocer los

conceptos, en relación a la configuración e implementación del modelo cognitivo baja la metaherisitica ACO a emplear, por último el tercer capítulo consiste en evaluar el parámetro de desempeño en las topologías de red configuradas para cada caso estudio.

Capítulo 1. Caracterización de la red bajo arquitectura SDN/NFV

En el presente capítulo se expone una base conceptual sobre el estado actual de SDN, NFV y VTN. Se relacionan conceptos, arquitecturas, características, entre otros con el propósito de dar a conocer las tecnologías a utilizar, las cuales permitirán extraer los elementos necesarios para diseñar la red de prueba donde se implementa el modelo cognitivo. En primer lugar se hace referencia a la evolución que han tenido las redes tradicionales a través del tiempo en sus diferentes modelos, así como también en la problemática actual que ha llevado a la investigación de nuevas alternativas que permitirán mitigar los inconvenientes de las redes actuales.

1.1. Redes tradicionales

Las redes tradicionales surgen de la necesidad de trasmitir información entre dos puntos a través de una red de comunicaciones, dando como resultado el diseño de protocolos (TCP/IP, HTTPS (*Hypertext Transfer Protocol Secure*), entre otros), además de la creación de dispositivos especializados en la trasmisión de la información (*Routers, Switch*, entre otros) los cuales han tenido una evolución debido al aumento de manera exponencial de peticiones de ancho de banda y servicios en línea [86].

A continuación se presenta una breve descripción, así como la historia, el funcionamiento, los modelos de referencia y la problemática actual de las redes tradicionales.

1.1.1. Historia

En la década de los años 90 fueron estandarizadas las redes ATM (*Asynchronous Transfer Mode*) cuyo fin era resolver el gran inconveniente de controlar la congestión causada por el aumento masivo del tráfico de voz, datos, video, entre otros. En la década de los años 90 nace una nueva tecnología de enrutamiento denominada IP-QoS la cual se vuelve más influyente sobre ATM, debido a que es más fácil de configurar en redes de datos, como consecuencia de lo anterior, IP-QoS aumentó dramáticamente la popularidad de los servicios proporcionados sobre el internet de uso público. A finales de los años 90 nace la tecnología MPLS (*Multiprocol Label Switching*) la cual trabaja sobre IP y fue un intento de ingeniería de tráfico más simple en internet enfocando en los *Backbones* de Internet. Sin embargo, la ingeniería de tráfico para MPLS se enfoca en el control y gestión de internet en virtud de los mecanismos actuales y los elementos de red [3].

1.1.2. Definición

Las redes tradicionales se pueden definir como un conjunto de elementos Hardware y Software interconectados, por medio de los cuales se envía y recibe información para compartir recursos y servicios de red [4].

1.1.3. Modelos de referencia

En la actualidad existen varios modelos de referencia para redes, los más conocidos son: el modelo OSI (*Open System Interconnection*) y el modelo TCP/IP *(Transmission Control Protocol*). Aunque los protocolos asociados al modelo OSI no se utilizan, es un es válido desde el punto de vista académico [10], mientras que con el modelo TCP/IP pasa lo contrario el modelo en si casi no se utiliza, pero los protocolos asociados son de amplio uso y conocimiento.

1.1.3.1. Modelo de referencia OSI

En los inicios de la década de los 80 ISO (Organización Internacional para la Normalización) da inicio a la estandarización de un protocolo que utilice varias capas y en 1985 lo denomina con el nombre de modelo de referencia OSI [5].

El modelo de referencia OSI está dividido en siete capas (ver Figura 1.1) y fue diseñado principalmente para solucionar los problemas de compatibilidad de interconexión encontrados en su momento. La más grande ventaja está en ser multicapa ya que al tener una falla se puede localizar rápidamente el error [6]. Las únicas capas que pueden interactuar con el usuario son la capa física y la capa de aplicación.

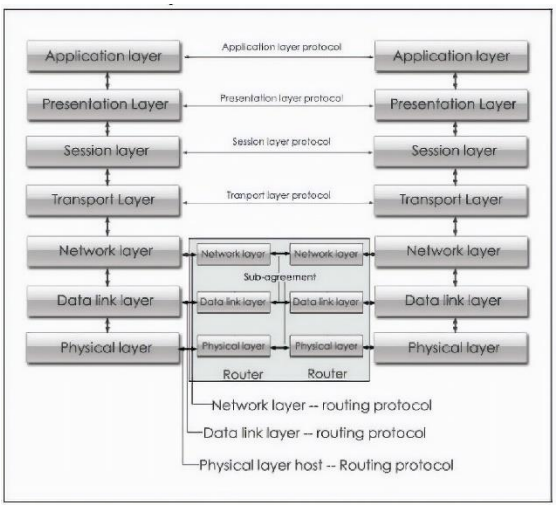


Figura. 1.1. Modelo OSI en siete capas [7]

1.1.3.2. Modelo de referencia TCP/IP

El desarrollo de TCP/IP fue un proyecto iniciado en la década de los 60 por la US DARPA (*Defense Advanced Research Project Agency*) con el fin de conectar varios dispositivos anfitrion *(Hosts)* en diferentes centros de investigación dando como resultado la creación de una red de área amplia denominada ARPANET la cual resultó siendo un precursor a lo que años después surgió como *Internet* [7]. En los años 90 el modelo de referencia TCP/IP se convirtió en el protocolo estándar dominante en internet [7].

Como se puede observar en la Figura 1.2, el modelo es muy similar a la mayoría de los protocolos de comunicación [8], donde gran parte del tiempo de ejecución se utiliza en bucles internos ya que los cálculos de copia de datos y comprobación se realizan en bucles.

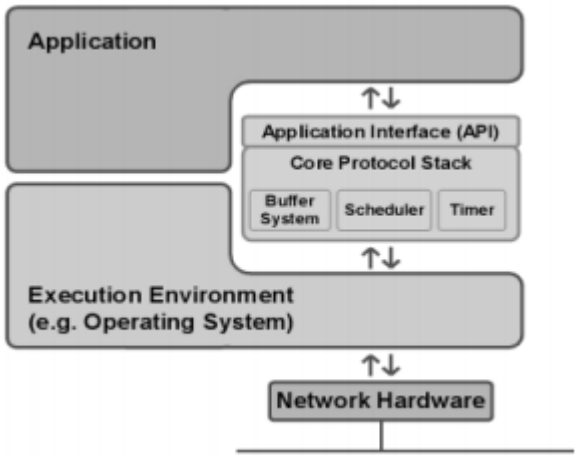


Figura. 1.2 Estructura Protocolo TCP/IP [8]

En el modelo TCP/IP cada capa tiene su funcionalidad y servicio. La capa de aplicación es donde la mayoría de los programas realizan la comunicación con la red. La capa de transporte es la responsable de la transferencia de mensajes de extremo a extremo, en consecuencia, cada capa tiene sus propios problemas y desafíos [9].

1.1.3.3. Comparación modelo de referencia OSI y TCP/IP

Los modelos de referencia OSI y TCP/IP se basan en el concepto de pila de protocolos independientes donde el modelo OSI tiene siete capas mientras que el modelo TCP/IP tiene cuatro capas como se muestra en la figura 1.3, los dos modelos tienen diferencias significativas, aunque son muy similares en las capas de aplicación y transporte [10].

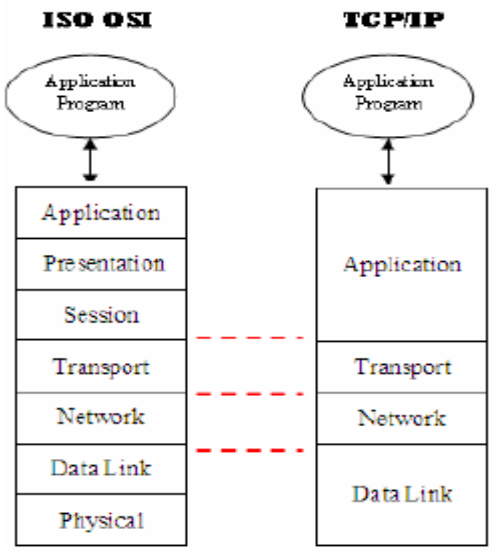


Figura. 1.3. Comparacion del Modelo TCP/IP y el Modelo OSI [10]

TCP/IP combina las funciones de la capa de presentación y sesión en la capa de aplicación, combina la capa física y la capa enlace de datos del modelo OSI en una misma capa, además TCP/IP contiene protocolos con los cuales fue creado internet [11].

1.1.4. Inconvenientes redes tradicionales

En la primer década del siglo XXI, nace una nueva perspectiva en relación con la gestión de redes en la industria de las Tecnologías de la Información (TI) y se empieza a disfrutar de una innovación en el campo de la virtualización y el almacenamiento [12]. Por el contrario, las redes han tenido un estancamiento por más de 20 años lo que ha llevado a redes excesivamente complejas e inflexibles que ya no satisfacen los requerimientos actuales.

A continuación, se trae a consideración los principales inconvenientes de las redes actuales o redes heredadas:

- No son flexibles: Se comportan con base en las reglas (protocolos) que incorporen los fabricantes de dispositivos. Lógicamente, cuanto más configurables sean éstos, más costosos serán [13].
- Políticas inconsistentes: Para implementar una política de toda la red, se tendría que configurar miles de dispositivos y mecanismos. Por ejemplo, cada vez que una nueva máquina virtual es creada, puede tardar horas, en algunos casos días, para que las ACLs (*Access Control List*) sean reconfiguradas en toda la red. La complejidad de las redes de hoy en día hace que sea muy difícil aplicar un control de acceso seguro, calidad de servicio, y otras políticas que cada vez

más usuarios móviles demandan, lo que deja a las redes vulnerables a violaciones de la seguridad, el incumplimiento de las regulaciones con consecuencias negativas [14].

- Incapacidad para escalar: Debido a que las demandas en los centros de datos crecen rápidamente, la red también crece. Sin embargo, se vuelve muy compleja con la adición de cientos o miles de dispositivos nuevos que deben ser configurados y gestionados. Por lo cual las empresas se han basado en patrones de tráfico previsibles; sin embargo, en los centros de datos virtualizados de hoy, los patrones de tráfico son dinámicos y por lo tanto impredecibles [14].
- Consumo de las tecnologías de información: Los usuarios están empleando cada vez más dispositivos personales móviles, como Smartphones, Tabletas y computadores portátiles para el acceso a la red. Las tecnologías de información se encuentran bajo presión para dar cabida a todos estos dispositivos personales de una manera adecuada y al mismo tiempo proteger los datos corporativos y la propiedad intelectual [15].
- No se prestan a la investigación y el desarrollo: Evidentemente, los fabricantes y administradores de redes, son reacios a experimentar y desarrollar nuevos paradigmas en redes que ya funcionan de una forma “satisfactoria”. Es el célebre “si funciona no lo toques” [13].

Las redes tradicionales han evolucionado con nuevos y complejos dispositivos Hardware (switch, router, firewall, IDS (*Intrusion Detection System*) y filtros), los cuales en sus características de diseño y fabricación con bastate similares, donde hay un Hardware especializado para cada elemento de red y sobre el funciona un sistema operativo, además, de haber miles de líneas de código lo que genera una tecnología de fabricación bastante compleja y cerrada al administrador de red además significa que no se puede tener una visión global de la red como un todo [16].

A continuación en la Tabla 1 se presentan las diferencias más relevantes entre la arquitectura de red tradicional y la arquitectura de red SDN [17].

Tabla 1.1. Comparación Arquitectura de Red Tradicional y Arquitectura de Red SDN

ARQUITECTURA DE RED ACTUAL	**ARQUITECTURA DE RED SDN**
Envío de paquetes dependiente a las tablas de enrutamiento.	Tablas de enrutamiento abiertas
Las tablas de flujo están cerradas en los dispositivos	Formato y acciones de las tablas claramente especificadas
	APIs bien definidas

Protocolos totalmente distribuidos	APIs abiertas para el acceso y manipulación del plano de Datos por ejemplo *OPENFLOW*
Interfazs Propietarias	Control lógicamente centralizado en el Software controlador
	mayor tasa de innovación
	Aumento de la fiabilidad y seguridad de la red
Automatización posible pero tediosa	Una solo interfaz (API) para todos los dispositivos como por ejemplo ODL (*OpenDayLight)*

1.2. Software Defined Network (SDN)

Las redes definidas por software son una tecnología emergente estandarizada que desagrega la arquitectura de las redes tradicionales para mejorar las características de la red, en otras palabras realiza una separación en plano de datos y control, lo cual puede emplearse para "personalizar" el funcionamiento de la red en entornos especializados.

1.2.1. Historia.

A mediados de los años 90 surge una nueva tecnología denominada redes activas (*Active Networks)* cuyo aporte principal fue introducir funciones programables a la red [19] lo que ayudo a tomar un nuevo enfoque surgiendo un suceso de innovación a principios del año 2000 la cual consiste en separar el control del plano de datos permitiendo desarrollar interfaces abiertas y en el año 2010 se crean los primeros sistemas operativos además de la API *OpenFlow* [18] lo que generó la primera interfaz abierta y el desarrollo de iniciativas para hacer escalable la separación del plano de control del plano de datos. La virtualización de la red jugó un papel muy importante en la naciente historia del concepto SDN.

El concepto de SDN nace en la Universidad de Stanford y proviene del laboratorio de investigación en redes del cual forma parte el estudiante Martin Casado quien introdujo además un par de conceptos nuevos en la arquitectura SDN, el primero es el reenvío de datos basado en tablas de flujos y el segundo corresponde a un controlador centralizado [20].

1.2.2. Definición.

El concepto SDN (*Software Defined Network*) es bastante nuevo y nace debido al aumento masivo en las peticiones de red y a la complejidad de las redes actuales las

cuales funcionan ejecutando tanto la red de transporte como el control en el mismo dispositivo haciéndolas difíciles de gestionar, además cada dispositivo se configura de manera independiente y de acuerdo con el fabricante complicando el tratamiento de las fallas [21].

SDN en un paradigma en las tecnologías emergentes el cual puede ayudar a cambiar las limitaciones de las actuales infraestructuras de red [22**]**, rompiendo el prototipo que consiste en separar la lógica del control de la red de la infraestructura física. SDN estuvo precedido por trabajos realizados en redes programables tales como (*Active Networks*) [23], redes ATM programables [24], [25] y en separación de plano de datos NCP (*Network Control Program*) [26] y el RCP (*Rate Control Protocol*) [27] con el fin de presentar una nueva evolución en el desarrollo de la administración centralizada de la infraestructura de Red.

1.2.3. Características SDN

Como se ha Evidenciado hasta el momento, SDN es una tecnología actual que busca mejorar las problemáticas evidenciadas en las redes tradicionales con el aumento masivo de dispositivos conectados a la red y los diferentes servicios que han ido surgiendo en la actualidad. A continuación se presentan las principales características de SDN [28]:

- Directamente Programable: El control de la red es directamente programable debido a que las funciones de reenvió (*forwarding*) (en el plano de datos) fueron desacopladas.
- Ágil: Al tener monitoreo constante sobre la red de telecomunicaciones, permite al administrador de red o a las aplicaciones hacer cambios dinámicos en los flujos de red según los requerimientos del servicio.
- Gestión centralizada: La inteligencia de la red está centralizada en un software llamado controlador SDN, el cual mantiene una visión global de la red, haciendo parecer una gran red de *Switches* como un único conmutador lógico.
- Programación configurada: SDN permite a los administradores de red configurar, administrar, asegurar y optimizar los recursos de red muy rápidamente a través de los programas de SDN dinámicos, que pueden ser escritos por ellos mismos, ya que los programas no dependen de software propietario.
- Basados en estándares abiertos y de proveedor neutral: Como se implementa a través de estándares abiertos, SDN simplifica el diseño de la red y la operación porque integra un solo protocolo para todos los fabricantes [28].

1.2.4. Controlador SDN

Como se ha mencionado, SDN es una arquitectura de red emergente cuyo principal objetivo es desacoplar el control existente en la infraestructura ubicada en el plano de datos de los diferentes nodos de la red y de una forma lógica centralizar dicho control y hacerlo completamente programable, de esta manera el administrador puede definir completamente el comportamiento de la red sin depender de los fabricantes de distintos elementos de red.

1.2.5. Arquitectura SDN

A continuación se describe la arquitectura SDN además de sus principales características según la ONF (*Open Networking Fundation*) y ODL (*OpenDayLight*).

1.2.5.1. Arquitectura según referencia ONF

La ONF es una organización creada en el año 2011 por seis multinacionales tecnológicas (Deutsche Telekom, Facebook, Google, Microsoft, Verizon y Yahoo) dedicada al desarrollo y estandarización de SDN y su protocolo *OpenFlow c*uyo objetivo central es mejorar los servicios de telecomunicaciones [29].

La arquitectura SDN, especificada por la ONF es ilustrada en la Figura 1.4, donde se aprecia que está compuesta por 3 capas, las cuales se describe en seguida.

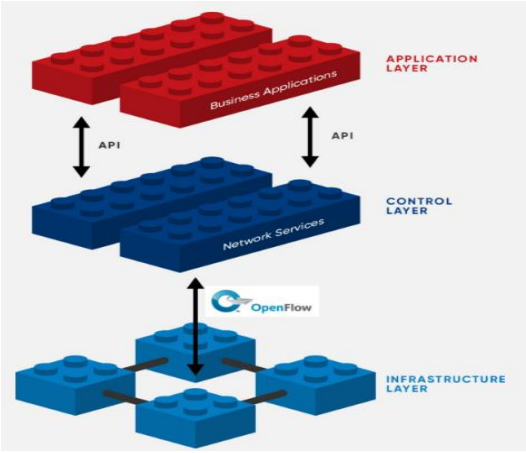


Figura. 1.4. Arquitectura SDN según ONF [29]

Capa de aplicación: Es la encargada de permitir la creación de las aplicaciones para automatizar tareas de configuración, provisión y despliegues de nuevos servicios sobre la red. La capa de aplicación se comunica con la capa de control por medio de un conjunto de API's (*Application Programming Interface*), con el fin de administrar las condiciones actuales de la toda la red.

Capa de control: Esta capa es la encargada de crear una vista centralizada de la topología de red, la cual permite gestionar el flujo de datos en la capa de infraestructura por medio del protocolo OpenFlow, es en esta capa donde se encuentra la API controladora (por ejemplo, *OpenDayLight).*

Capa de infraestructura: Está conformada específicamente por el hardware de la Red, sin ninguna inteligencia la cual es gestionada por medio del protocolo *OpenFlow,* el cual facilita la configuración y la administración de la Red.

1.2.5.2. Arquitectura según referencia *OpenDayLight*

ODL lanzó en 2018 su versión llamada Carbon, S3P *OpenDayLight* Release, cuya arquitectura se muestra en la Figura 1.5.

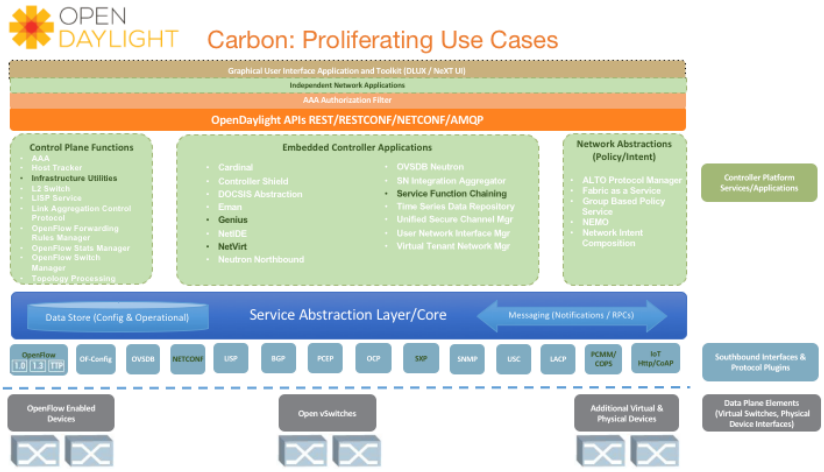


Figura. 1.5. Arquitectura SDN según ODL [30]

Si bien el conjunto básico de servicios ODL que se muestra en la Figura 1.5 es suficiente para automatizar los servicios de la red estándar, los usuarios pueden implementar otras características y servicios adicionales ya que la arquitectura ODL lo permite, además cuenta con una mayor integración con *OpenStack* permitiendo gestionar las redes mediante capacidades avanzadas como grupos de seguridad, *router* virtuales o *switch* virtuales distribuidos. Por otro lado, al soportar *Apache Karaf* (contenedor ligero sobre el que se puede desplegar aplicaciones y componentes) los usuarios pueden personalizar la instalación de ODL solo con las características que necesite una red en particular [30].

1.3. Virtualización de funciones de red, *Network Function Virtualization* (NFV)

1.3.1. Historia

El concepto NFV se origina gracias a los ISP (*Internet Service Provider*) que buscaban la aceleración en la implementación de nuevos servicios de red con el propósito de respaldar su crecimiento. Debido a las grandes limitantes de los dispositivos basados en Hardware los llevaron a la aplicación de las tecnologías de virtualización de TI. Los ISP con el fin de acelerar el progreso hacia este objetivo común se unieron y crearon el Instituto Europeo de Estándares de Telecomunicaciones (ETSI).

El grupo de especificación industrial ETSI para la virtualización de funciones de red, fundado en noviembre de 2012 por siete de los principales proveedores de redes de telecomunicaciones del mundo, es el encargado de desarrollar nuevas arquitecturas para la virtualización de diversas funciones dentro de las redes de telecomunicaciones.

1.3.2. Definición

NFV es una arquitectura basada en la separación entre la lógica de red y la infraestructura donde se ejecutan, tiene como principal objetivo implementar funciones de red en un entorno de virtualización de IT (*Information Technology*) abierto y estandarizado, contrario al Hardware específico de los proveedores. En esencia NFV busca desacoplar funciones de red, mejorar su eficiencia operacional y controlar sus costos. Algunas funciones de red que pueden ser virtualizadas son la traducción de direcciones (NAT), resolución de nombres de dominio (DNS) o *Firewalls*.

NFV busca transformar toda la arquitectura de red mediante la evolución de la tecnología estándar de virtualización de TI para consolidar routers, balanceadores de carga y otros dispositivos hardware a máquinas virtuales como se muestra en la Figura. 1.6.

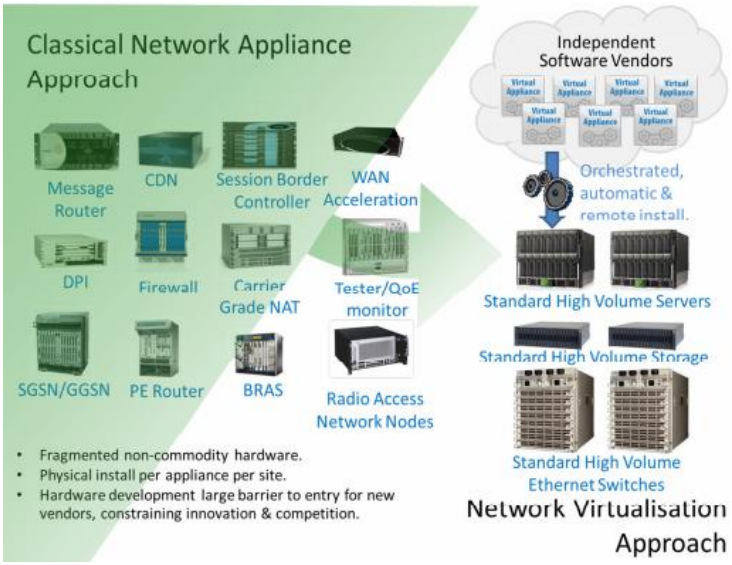


Figura. 1.6. Concepto NFV [31]

En relación con SDN, la virtualización de funciones de red como se muestra en la Figura. 1.6. no depende de las redes definidas por software por lo contrario son complementarias.

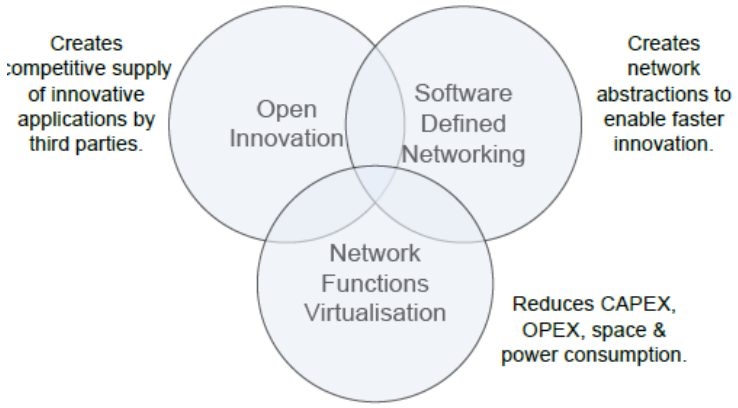


Figura. 1.7. Relación SDN/NFV [31]

1.3.3. Características.

Dentro de la tecnología NFV existen muchas características, las cuales la han llevado a ser una de las soluciones más óptimas en el crecimiento expansivo que se ha tenido en los últimos tiempos en servicios dispuestos en la red. Las principales características son:

- Segregación entre el hardware y el software: NFV sustituye los dispositivos de red físicos por el software ejecutado en servidores, eliminando la necesidad de Routers, *Switches* y *Firewalls*.[32].

- Flexibilidad de las funciones de red: Permite que las operaciones de negocio y de red adquieran las características de agilidad, fiabilidad y economía que ofrece la nube (*Cloud)*. Asimismo, una mayor flexibilidad para escalar la capacidad de la red, adaptarse dinámicamente a los cambios y desplegar soluciones a medida de cada cliente mediante una configuración más abierta de los equipos [32].

1.3.4. Arquitectura.

Como se observa en la Figura. 1.8. los componentes principales de las funciones de red son:

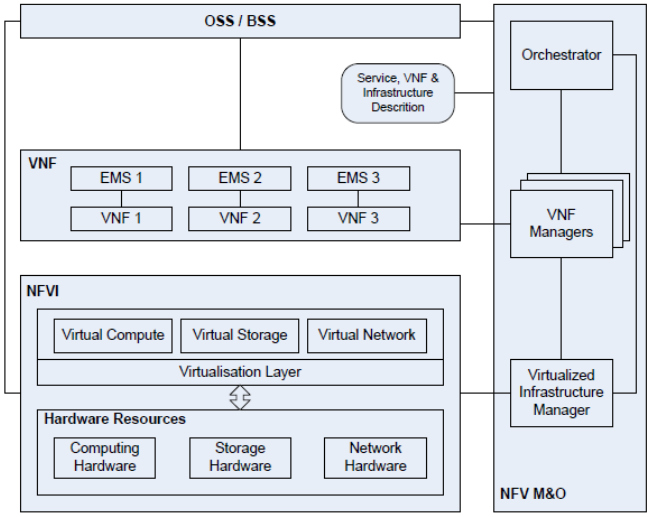


Figura. 1.8. Arquitectura NFV [32]

- NFVI (*Network Functions Virtualisation Infrastructure*): es la base fundamental de la arquitectura contiene el hardware para alojar las máquinas virtuales, el almacenamiento, el software que hace posible la virtualización y los recursos de red a través de la virtualización y los proporciona como recursos virtuales.
- VNF (*Virtualized Network Function*): utiliza las máquinas virtuales que brinda NFVI construyendo sobre ellas las funciones de red virtualizadas además de incluir el software necesario.
- NFV M&O (*Network Functions Virtualisation Management and Orchestration*): es el bloque separado de la arquitecta, el cual es el encargado de interactuar con el NFVI y el VNF donde se Implica la orquestación y la gestión del ciclo de vida de los recursos físicos y del software incluidos los VNF, además este marco arquitectónico fue construido para gestionar los recursos de la capa de infraestructura.

1.4. Comparación SDN/NFV

Tabla 1.2. Comparación SDN/NFV

ITEM	SDN	NFV
INICIO	Creado por investigadores y arquitectos de centros de datos	Creado por un consorcio de proveedores de servicios
OBJETIVO	Enfoque basado en software para la creación de redes	Enfoque basado en software para la creación de redes
PROPÓSITO	Separación de control y datos, centralización del control y capacidad de programación de la red	Extraer las funciones de red de dispositivos dedicados a servidores genéricos
APLICABILIDAD	Red donde los planos de datos y de control estén separados	Red tradicional basada en hardware, una red SDN, o una combinación de ambas
ARQUITECTURA	Tres planos: Plano de infraestructura "hardware sin inteligencia", Plano de Control "controladora administración del plano de infraestructura", Plano de Aplicaciones "APPs"	Tres componentes principales: NFVI "hardware para alojar las máquinas virtuales", VNF "virtualiza las funciones de red sobre las máquinas que brinda NFVI", NFV M&O "gestiona los recursos de la infraestructura "
CARACTERÍSTICAS	Directamente Programable, Ágil, Gestión Centralizada, Programación configurada, Basada en estándares abiertos.	Segregación entre hardware y software, Flexibilidad de las funciones de red
HARDWARE DESTINO	Servidores y *Switches* básicos	Servidores y *Switches* básicos
PROTOCOLO	OpenFlow	Ninguno
BENEFICIOS	Reduce CAPEX y OPEX, proporciona agilidad y flexibilidad y facilita la innovación	Reduce CAPEX y OPEX, proporciona agilidad y flexibilidad y acelera el tiempo de salida al mercado
NFV no depende de SDN por lo contrario es complementario		

A pesar que las tecnologías SDN y NFV (*Network Functions Virtualization*) son diferentes y pueden existir sin dependencia una de la otra, son conceptos muy relacionados y pueden ser complementarias. Es así que la combinación de ambas tecnologías consigue un mayor potencial y adaptabilidad a los nuevos requerimientos tecnológicos.

1.5. Redes virtuales multi-vecino, *Virtual Tenant Network* (VTN)

1.5.1. Definición

VTN es una aplicación que permite múltiples redes virtuales en un controlador SDN [32], donde la principal característica de VTN consiste en mapear la red física en una red virtual, permitiendo que la red virtual deseada o creada por el usuario involucre tanto el mapeo de nodos como el mapeo de enlaces.

1.5.2. Arquitectura VTN

La arquitectura VTN como se muestra en la figura.1.9 consta de dos componentes principales VTN *Coordinator* y VTN *manage*r donde VTN manager se implementa dentro de VTN *Coordinator* en forma de complemento con el propósito de gestionar la información del red virtual incluyendo la topología de red y la información de asignación.

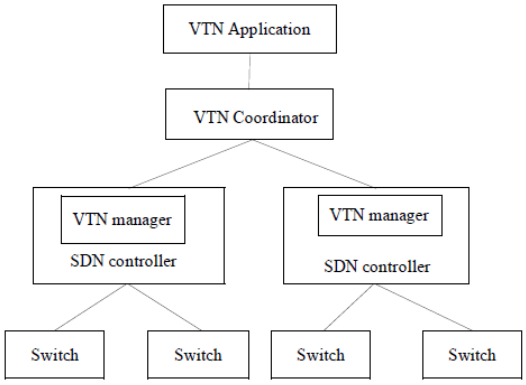


Figura 1.9. Arquitectura VTN [33]

El *Coordinator* VTN administra múltiples controladores SDN y proporciona API REST para la configuración de VTN [34]. Dentro de la implementación de una la red virtual la arquitectura VTN consiste en Nodos Virtuales (VN, *Virtual Node*), Enlaces Virtuales (VL, *Virtual Link*) e Interfazs Virtuales, donde los nodos virtuales incluyen *Switch* Virtuales, Router Virtuales Túneles Virtuales, entre otros [35].

1.6. VTN según referencia *OpenDayLight*

OpenDayLight VTN surge como un plano de abstracción lógica, permitiendo la separación completa del plano lógico del plano físico [36], donde el usuario no tendrá restricciones de ancho de banda o desconocimiento de la red física para el diseño o implementación de una red evitando así, la complejidad de la red subyacente con una mejor gestión de los recursos de la red.

Las redes VTN son una tecnología de extensión en el controlador *OpenDayLight* la cual se puede utilizar en múltiples escenarios como *Data Center*, computación en la nube, entre otros [37]. VTN tiene la posibilidad de construir una red virtual la cual automáticamente se puede asignar a la red física con el propósito de mapear los nodos y enlaces por medio de virtualizaciones soportadas por *OpenDayLight* VTN denominadas vBright, vRouter y vLink.

La arquitectura ODL *Virtual Tenant Network* hace parte abstracta de la arquitectura SDN como se muestra en la Figura 1.10. El módulo VTN manager se implementa como un complemento de *OpenDayLight* para interactuar con otros módulos de la arquitectura VTN, lo cual proporciona una interfaz REST para crear, actualizar y eliminar componentes de VTN.

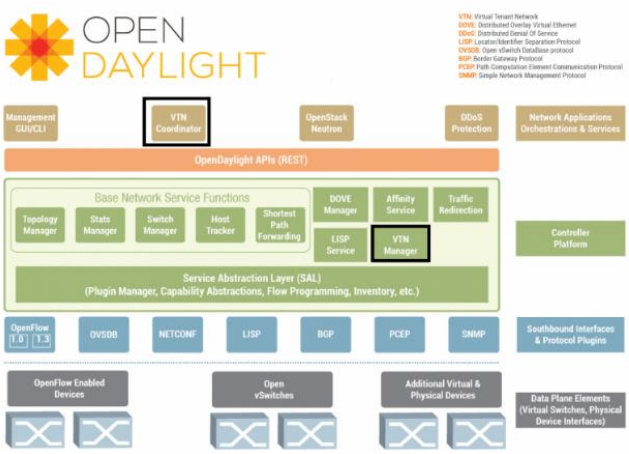


Figura 1.10. Arquitectura VTN según *OpenDayLight* [38]

1.7. Protocolo OPENFLOW.

Es una interfaz de comunicaciones estándar, definida entre la capa de control y la capa de infraestructura de la arquitectura SDN permitiendo el acceso directo a la manipulación de plano de datos para escribir los flujos en las tablas de flujos de cada dispositivo de red como enrutadores y conmutadores tanto físicos como virtuales [40].

El OpenFlow *switch* está compuesto por un conjunto de tablas de flujo y el canal del controlador y consta de tres partes como se muestra en la figura 1.11 [40].

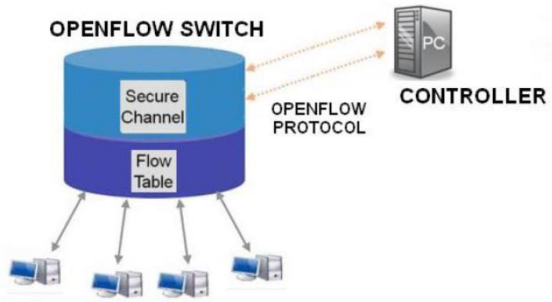


Figura. 1.11. OpenFlow Switch [40].

- Una tabla de flujos, indica cómo debe procesar el switch los flujos que lleguen. Un *switch OpenFlow* puede tener diferentes tablas de flujos las cuales analizan la coincidencia entre los paquetes entrantes los cuales constan de 2 campos:
  - ✓ Cabecera, se define en el flujo.
  - ✓ Contadores, contabilizan la coincidencia de paquetes.
- Canal para comunicar el switch con el controlador y esta comunicación se genera por medio del protocolo *OpenFlow*.
- El protocolo *OpenFlow*, consiste en un estándar abierto mediante el cual se especifican las entradas en la tabla de flujos. *OpenFlow* es una API ubicada hacia el sur de modelo (*Southbound API*), o un protocolo de comunicación entre la capa de control y la capa de infraestructura de red como lo muestra la Figura 1.11.

Mediante el desarrollo del primer capítulo, se describe de manera general las funcionalidades, características, arquitecturas entre otros de las tecnologías emergentes de SDN, NFV y VTN objeto de estudio de este trabajo de investigación, para su posterior análisis de desempeño.

Así mismo se logra establecer la funcionalidad de las redes tradicionales en relación a las nuevas tecnologías, en búsqueda de la mejora continua a los requerimientos que surgen con el crecimiento masivo de las peticiones sobre la red.

En capítulo siguiente, se continua forjando la base conceptual con el propósito de cumplir con el objeto de este trabajo, caracterizando el modelo cognitivo a evaluar en relación al factor de desempeño.

Capítulo 2. Caracterización del modelo cognitivo

La rápida evolución de las redes de telecomunicaciones, dan la oportunidad de afrontar los principales retos que han venido provocando un crecimiento sin precedente del tráfico de datos en la red, además en los recientes y rápidos adelantos de las diferentes tecnologías. Hacen que las redes cognitivas sean una alternativa que ayude en la búsqueda de soluciones optimas que logren afrontar el mundo de las telecomunicaciones, es así que la investigación enfocada a la generación de nuevas alternativas en relación a algoritmos inteligentes ha logrado demostrar ser una opción optima en las diferentes problemáticas de la actualidad [51].

Teniendo en cuenta lo anterior, en el presente capítulo se hace una descripción de los conceptos a emplear, para lograr la configuración e implantación de un modelo cognitivo basado en metaheuristica con el propósito de mejorar la probabilidad de bloqueo de los enlaces y de la red. Asimismo, al proponer el desarrollo de un algoritmo cognitivo aplicado al proceso de enrutamiento, es necesario construir una base conceptual, por tanto en el presente capítulo se describe una base conceptual de las metaheuristicas basadas en población (ACO, Ant Colony Optimization, PSO, Particle Swarm Optimization) con el fin de lograr la mejor alternativa en la búsqueda de rutas.

2.1. Metaheurísticas

Dentro del mundo de las telecomunicaciones se abordan múltiples desafíos en razón de los problemas de optimización. Dicha optimización se pueden clasificar en función de diferentes factores como su carácter dinámico o estático, lineal o no lineal, mono-objetivo multi-objetivo, entre otros [41]. Por lo tanto una estrategia de solución han sido las metaheuristicas que se han desarrollado como una estrategia que usa diferentes métodos para afrontar problemas con espacios de búsqueda de gran tamaño por medio de técnicas ya establecidas.

2.1.1. Definición

El concepto de metaheuristica fue creado por Fred.Glover en el año 1986 [41] donde lo definió como "*Un procedimiento maestro de alto nivel que guía y modifica otras heurísticas para explorar soluciones más allá de la simple optimalidad actual*" *"heurística es cualquier enfoque de resolución de problemas o autodescubrimiento que emplea un método practico, que no garantiza que sea optimo, perfecto, lógico o racional, si no que es suficiete para alcanzar una meta inmediata* [wiki]*"* las cuales se conocieron como técnicas heurísticas modernas y consisten en procedimientos generalizados de una determinada heurística para cualquier tipo de problema y así

generar soluciones aproximadas.

Así mismo en las técnicas heurísticas modernas se incluyen algoritmos como colonia de hormigas, algoritmos evolutivos, búsqueda local iterada, búsqueda tabú, entre otros, las cuales poseen ciertas propiedades fundamentales [42] que se nombran a continuación:

- Las metaheurísticas son estrategias o plantillas generales que guían el proceso de búsqueda.
- El objetivo es una exploración eficiente del espacio de búsqueda para encontrar soluciones (casi) óptimas.
- Las metaheurísticas son algoritmos no exactos y generalmente son no deterministas.
- Pueden incorporar mecanismos para evitar regiones no prometedoras del espacio de búsqueda.
- El esquema básico de cualquier metaheurística tiene una estructura predefinida.
- Las metaheurísticas pueden hacer uso de conocimiento del problema que se trata de resolver en forma de heurísticos específicos que son controlados por la estrategia de más alto nivel.

2.1.2. Clasificación de metaheurísticas

Existen diferentes maneras de clasificar las técnicas metaheurísticas de las cuales se van a seleccionar las técnicas con más usabilidad, las metaherísticas basadas en trayectoria y las metaheurísticas basadas en población. Las primeras trabajan con un único elemento en cada espacio de búsqueda mientras que las segundas afectan varios de ellos (población) [43]. En la figura. 2.1. se hace una clasificación de la taxonomía y las principales metaheurísticas.

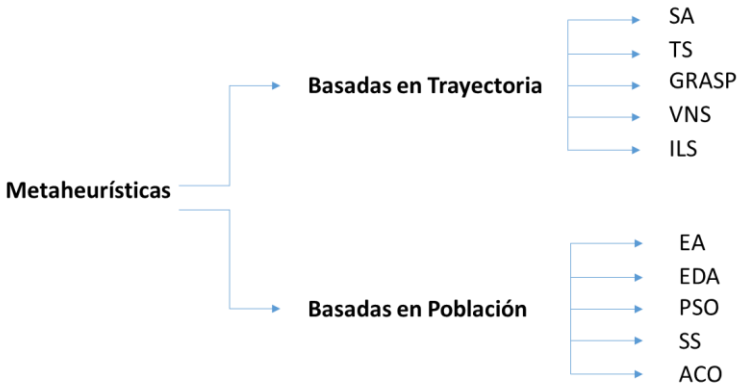


Figura. 2.1. Clasificación de Metaheurísticas.

2.1.2.1. Metaheurísticas basadas en trayectoria

Las metaheurísticas basadas en trayectoria son aquellas que parten de una solución previa y, mediante la exploración del campo van actualizando la solución actual formando un índice de trayectoria. Lo que implica que normalmente la búsqueda termina cuando se alcanza un número máximo predefinido de iteraciones, se detecte una detención del proceso o se encuentra una solución mediante la prueba a través de una función *fitness* o calidad [42].

2.1.2.2. Metaheurísticas basadas en población

Los métodos basados en población a diferencia de los métodos basados en trayectoria no manipulan una única solución μ del espacio de búsqueda por iteración, que se caracterizan por trabajar con un conjunto de soluciones llamados población, en conclusión ($\mu > 1\ y/o\ \lambda > 1$) [42]. En seguida se realiza una breve descripción de PSO y ACO.

$\mu = Numero\ de\ soluciones\ en\ un\ paso.$

$\lambda = Nuevo\ numero\ de\ soluciones\ generadas\ en\ cada\ iteración.$

2.1.2.3. PSO

Los algoritmos basados en partículas (PSO) están inspirados en el comportamiento de muchos seres vivos y se caracteriza por presentar movimiento cooperativo coordinado (Microorganismos, Cardúmenes de peces o bandadas de aves). El algoritmo PSO consiste en mantener un conjunto de soluciones denominadas (Partículas) que parten de la aleatoriedad en el espacio de búsqueda donde cada partícula posee una posición y velocidad que cambia en el tiempo de acuerdo a la búsqueda.

Básicamente la Metaheuristica PSO consiste en un algoritmo iterativo basado en un enjambre (población de individuos), en la que cada partícula (individuo) sobrevuela el espacio en búsqueda de soluciones óptimas. PSO ha mostrado ser muy eficiente en problemas de optimización mono-objetivo.

2.1.2.4. ACO

La optimización por colonia de hormigas ha sido formalizada en una metaheurística de optimización aleatoria por Marco Dorigo [44], la cual se propone para simular el comportamiento natural de las hormigas en búsqueda de comida. Este proceso consiste inicialmente en la exploración aleatoria del área cercana, una vez la hormiga encuentra comida esta se comunica a través de un medio biológico denominado feromona, el cual al paso del tiempo ayuda a más y más hormigas a encontrar el camino más corto entre el nido y la comida.

Inspirada en el comportamiento natural de la alimentación de las hormigas se propone

la optimización por colonia de hormigas para resolver el problema de optimización en un sistema discreto. Este algoritmo en la actualidad se ha utilizado para resolver diferentes problemas de optimización como la planificación de rutas de los robots móviles [45], el problema de asignación [46], el problema de selección de características [47], el problema de viajero frecuente, entre otros. El algoritmo de colonia de hormigas sin conocimiento previo es un método de optimización estocástica donde las hormigas seleccionan aleatoriamente el nodo, optimizan gradualmente el recorrido con la ayuda de las feromonas y finalmente obtienen un viaje óptimo.

En la actualidad existen varios algoritmos mono-objetivo como Max-Min Ant System (MMAS) [48], Ómicrom ACO (OA) [49], entre otros y algoritmos multi-objetivo como Ant Colony System (MOACS) [50], Multiobjetive OA (MOA) [49], entre otros. Estos se diferencian en el manejo de varios objetivos en lugar de un solo objetivo, en el manejo varios espacios de búsqueda en lugar de uno y en el uso de nuevas metodologías para obtener resultados ideales. En la tabla 2.1. Se presenta algunas características de los algoritmos mencionados.

Tabla 2.1. Comparación Algoritmos Mono-Objetivo y Multi-Objetivo

Algoritmos Mono-Objetivo	
MMAS	OA
Optimización Mono-Objetivo	Optimización Mono-Objetivo
Búsqueda de la mejor solución global	Búsqueda de la mejor solución global
Niveles máximos y mínimos de feromonas	Niveles máximos y mínimos de feromonas
Elitista - Actualización de feromonas con la mejor solución hallada	Elitista - Actualización de feromonas con el conjunto ordenado de soluciones
Heurística de una visibilidad	Heurística de una visibilidad
	Mantiene un conjunto ordenado de mejores soluciones
Algoritmos Multi-Objetivo	
MOACS	MOA
Optimización Multi-objetivo 4 (objetivos).	Optimización Multi.objetivo 4 (objetivos).
Búsqueda del frente paralelo.	Búsqueda del frente paralelo.
Selección pseudo-aleatoria y evaporación de feromonas en línea de los enlaces durante la construcción de una solución.	Mantiene un conjunto ordenado de mejores soluciones
Heurística de 3 visibilidades (costo, retardo y Tráfico actual)	Heurística de 3 visibilidades (costo, retardo y Tráfico actual)
	Elitista - Actualización de feromonas con el conjunto ordenado de soluciones
	Sin imposición de niveles de feromonas

El diseño del algoritmo de enrutamiento es basado en la metaheuristicas ACO por cuanto entre las metaheuristicas basadas en población son las de mejores

resultados para dar soluciones de enrutamiento centradas en QoE, sujetas a las demandas de tráfico y restricciones de red [44]. También se elige ACO, ya que ofrece una compensación favorable entre la diversificación y la intensificación del espacio de soluciones, logrando encontrar la solución más apropiada de acuerdo al parámetro de desempeño a evaluar [44].

2.2. Cognición

Diferentes estudios han profundizado durante siglos en los misterios de la cognición humana intentando comprender la manera como el desarrollo mental y la capacidad de aprendizaje se relaciona con la toma de decisiones autónomas de los nuevos algoritmos de aprendizaje en la ingeniería [51].

En ese mismo sentido las ciencias de la cognición han venido evolucionando a tal punto que ha pasado de manipular unidades simples de la ingeniería, a un tratamiento eficiente de modelado de la información. A través del tiempo se han venido proponiendo y desarrollando distintos modelos para diferentes propósitos. Sin embargo muchos modelos se construyen en áreas lingüísticas [51]. Ball propone establecer un modelo de comprensión del lenguaje cognitivo, que se construye desde el punto de vista funcionalista, el cual podría ser muy adoptable para el modelado computacional.

2.2.1. Definición

Cognición del Latín *Cognitio*, se entiende como conocimiento alcanzado mediante el ejercicio de las facultades mentales, la cognición está correlacionada con el aprendizaje desde la experiencia, a medida que los seres humanos viven nuevas experiencias llegan nuevas apreciaciones y conceptos, para producir un cambio en su comportamiento [wiki].

Así mismo la cognición está relacionada con el conocimiento que es el hecho de conocer algo por la experiencia o la asociación, es decir, a medida que se llega a ser consciente de nueva información y aprendemos de esta, ya haría parte de nuestro nuevo conocimiento.

Por último el concepto de cognición enfocada en la rama de telecomunicaciones nace de diferentes investigaciones. Mitola [52] bajo el enfoque de redes inalámbricas hace una breve mención de cómo sus radios cognitivos (CR, *Cognitive Radio*) podrían interactuar dentro del alcance de nivel del sistema de un (CN, *Cognitive Network*). Saracco [53] se refiere a (CNs) en su investigación sobre el futuro de las tecnologías de la información. También hace una postulación que el movimiento de la inteligencia

de red desde el control de los recursos hasta el entendimiento de las necesidades del usuario, ayuda a la red a mover la inteligencia más hacia los bordes o fronteras de la misma.

2.2.2. Método Cognitivo

El método de red cognitiva se puede definir desde la perspectiva de un proceso cognitivo, ayudado con el desarrollo del proyecto CHRON, como la capacidad de observar, aprender y planear por medio de un algoritmo, el cual asume la responsabilidad de administrar la red. Con estos elementos, el presente trabajo de grado se enfoca hacia la implantación de un método cognitivo o proceso cognitivo que interactúe en la capa más elevada del modelo SDN donde a modo de aplicación se puede observar, planear y actuar de acuerdo al ítem establecido, en busca de mejorar el desempeño de la red por medio de la evaluación de la probabilidad de bloqueo PB.

2.2.3. Ciclo Cognitivo

Todos los sistemas que pueden ajustar su funcionamiento de acuerdo con los cambios en su entorno se basan en información de retroalimentación. Las redes cognitivas no son una excepción a este respecto, por lo que también usarán un ciclo de control, también denominado ciclo de cognición que aplicado a las telecomunicación nace como concepto por Mitola [54] el cual enfoca el desarrollo y la investigación a las redes inalámbricas proponiendo aplicar el ciclo cognitivo a las capas 1 y 2 del modelo OSI. La aplicación del Ciclo cognitivo o ciclo de cognición permitirá a la Red tomar decisiones inteligentes por medio del seguimiento del lazo cognitivo, el cual consta de seis procesos, véase en la Figura 2.2 (observar, orientar, planear, aprender, tomar decisiones y actuar) [55] y así poder optimizar su desempeño.

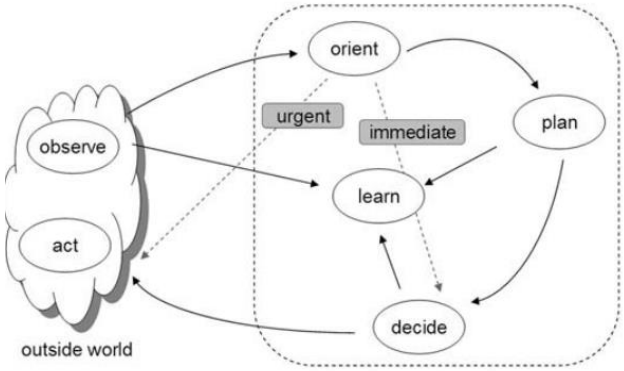


Figura. 2.2. Ciclo Cognitivo Mahmound 2007, p.207 [55].

2.2.4. Redes Cognitivas

El crecimiento masivo de grandes centros de datos, el aumento exponencial del uso de los servicios expuestos en la nube, esto sumado a la cantidad y variedad de protocolos de enrutamiento asociados a las redes heredadas facilitan el nacimiento de nuevos paradigmas en la evolución de algoritmos distribuidos [56] o de algoritmos centralizados [57] a algoritmos cognitivos.

Igualmente, las redes cognitivas en el futuro van a tener el potencial de proporcionar una red adaptable, dinámica, robusta y de alto ancho de banda mediante la centralización del control, permitiéndole observar el estado actual de la red para su análisis y adaptación de una manera más eficiente a los cambios detectados. Por lo tanto la red cognitiva propone automatizar las redes mediante la detección de dichos cambios en tiempo real adaptándola a estos y realizando acciones para la aplicación de sistemas de control de bucle que aprenda y pueda actualizarse a sí misma con el propósito de realizar acciones futuras sin intervención humana [58].

De otro lado, las denominadas redes radio cognitivas se están integrando al modelo SDN evolucionando con nuevas arquitecturas [59]. Las propuestas van dirigidas a una mejor gestión del espectro compartido en el espacio blanco de television TV para redes heterogéneas usando el protocolo *OpenFlow* OF. En la literatura al respecto se evidencia entre otros un concepto SDN [6] el cual asigna algunas responsabilidades de la gestión de las estaciones base (BS, *Base Station*) a los usuarios finales en un marco de virtualización de funciones de red.

2.2.4.1. Definición

La primera definición de red cognitiva (CN, *cognitive network*) fue presentada por primera vez en la conferencia IEEE DySPAN en 2005 [60]. Una red cognitiva es una red con un proceso cognitivo que puede percibir las condiciones actuales de la red, planificar, decidir y actuar en esas condiciones. La red puede aprender de esas adaptaciones y utilizarlas para decidir, lo anterior teniendo en cuenta las metas de extremo-extremo (*end to end goals*).

El concepto de redes cognitivas (CN, *Cognitive Network*) ha tenido mayores esfuerzos en investigación y desarrollo especialmente en las redes inalámbricas, en la literatura se encuentran diferentes menciones como Radio Cognitiva [61], radios inteligentes [62], antenas inteligentes [63], paquetes cognitivos [64], generando varias discusiones con respecto a su significado. Mitola [66] propone interactuar en la capa física del modelo OSI y ahí aplicar el ciclo cognitivo sobre los elementos de red y Saraco [65] propone que la inteligencia de la red sea controlada de acuerdo con las necesidades

del usuario moviendo la inteligencia de la red más hacia los bordes de la red.

2.2.4.2. Elementos básicos de una red cognitiva

Los elementos básicos de una red cognitiva son la Observación, el análisis y la toma de decisiones. Thomas [67] representa por medio de un sistema piramidal estos elementos, donde en la base de la pirámide se encuentran los elementos de observación y actuación como funciones distribuidas a través de la red. En el centro y cima de la pirámide se encuentran los elementos de análisis y toma de decisiones, los cuales son de mayor inteligencia y pueden presentar la administración de la red como se muestra en la Figura 2.3.

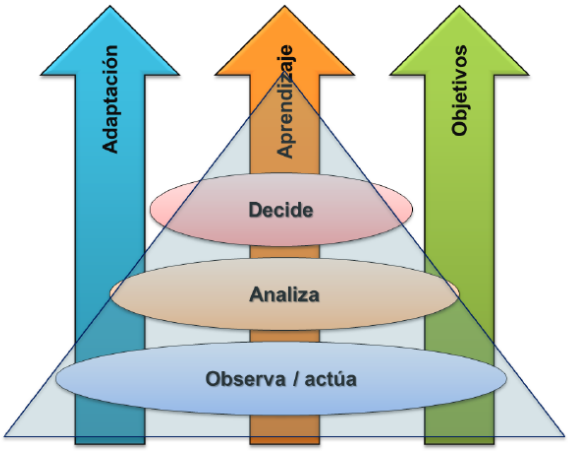


Figura 2.3. Procesos Cognitivos Por Planos [67]

Este modelo se caracteriza por tener tres planos. El plano de adaptación, el plano de aprendizaje y el plano de objetivos. El plano de adaptación es el encargado de adaptarse a los cambios que se den en el entorno y así reportarlo al elemento de aprendizaje. El plano de aprendizaje es el core o núcleo del sistema cognitivo y es el encargado de administrar la red para llevar a cabo diferentes acciones de acuerdo a las experiencias pasadas. El plano de objetivos genera una gran variedad de metas para ser alcanzadas mediante los procesos cognitivos. En conclusión, los planos de objetivos y adaptación están directamente relacionados con la observación, el análisis y la toma de decisiones, con el fin de elegir las acciones adecuadas de acuerdo a los objetivos traslados al inicio del proceso cognitivo.

2.2.4.3. Proyectos Sobre Redes Cognitivas

En contexto de redes cognitivas se han propuesto varias arquitecturas cognitivas que pueden ser aplicadas a redes cableadas [68], redes inalámbricas o redes ópticas [69]. A continuación se presentan algunos proyectos que muestran arquitecturas cognitivas que se pueden implementar en diferentes dimensiones, en términos de dispositivos y capas de protocolo.

2.2.4.4. FOCALE

FOCALE (*Fundation- Observe- Compare- Act- Learn- Reazon*), es una arquitectura de red autónoma impulsada por modelos que pueden generar código de forma dinámica para reconfigurar elementos gestionados [70] como se muestra en la figura 2.4.

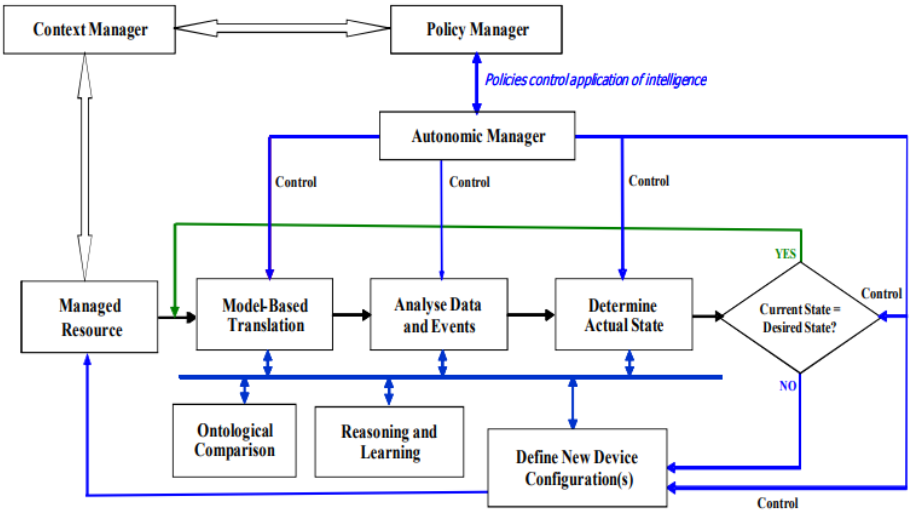
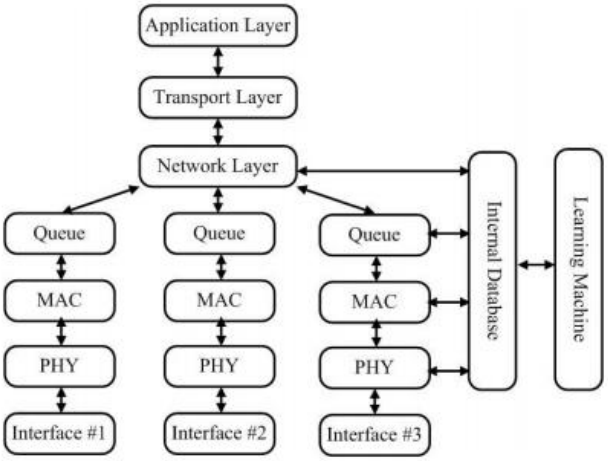


Figura 2.4. Modelo Knowledge in DEN-ng [70]

FOCALE funciona como un módulo de aprendizaje y razonamiento donde los datos de red se envían a un proceso de traducción basado en modelos el cual traduce los datos de red específicos del proveedor y del dispositivo en una forma normalizada utilizando el modelo de información DEN-ng [71]. Luego estos datos se analizan para determinar el estado actual de la red gestionada donde el estado actual se compara con el estado deseado de la red para actualizar los dispositivos a partir de la aplicación de nuevas configuraciones. Las políticas se utilizan para luego definir la funcionalidad permitida. A medida que cambia el contexto, las políticas cambian y la funcionalidad del sistema se ajusta en consecuencia.

2.2.4.5. CogNet

CogNet (*Cognitive Complete Knowledge Network*) propuesto por la NSF (*National Software Foundation*) es una técnica que se centra en la aplicación de técnicas cognitivas para mejorar el rendimiento de la una red de telecomunicaciones, la arquitectura CogNet se utiliza principalmente en redes inalámbricas para la función de enrutamiento en la capa de red.

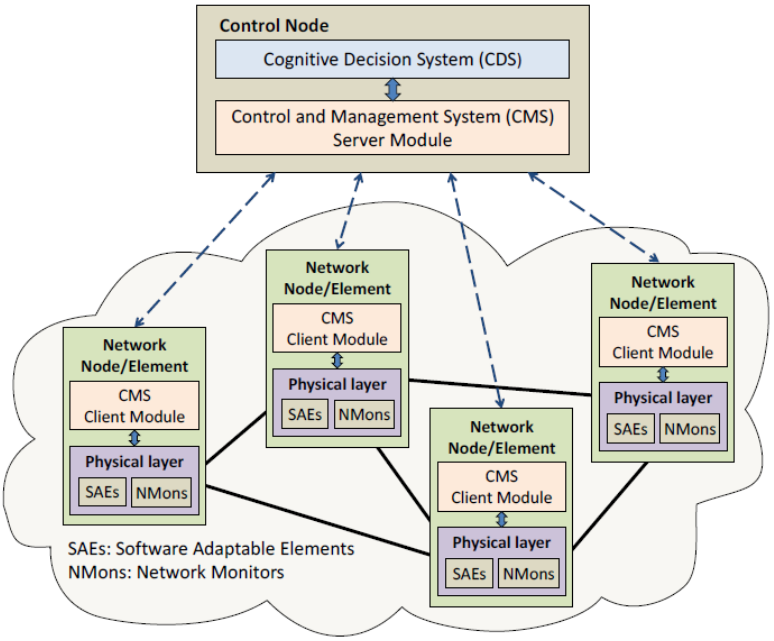


2.5. Arquitectura CogNet [72]

La arquitectura CogNet propuesta [72] muestra el intercambio de información entre capas mejorando el rendimiento de la red para redes cableadas, la retroalimentación optimizada permite que se comparta la información entre capas es decir, una base de datos interna se adjunta a las tres capas inferiores de la Figura 2.2.3. Permitiendo que los CR realicen funciones optimizadas de capa cruzada con la información de red compartida, adjuntado una máquina de aprendizaje que sirve como un motor cognitivo en la base de datos interna. La máquina de aprendizaje realiza las funciones cognitivas procesando la información de red compartida almacenada en la base de datos interna.

2.2.4.6. CHRON

CHRON (*Cognitive Heterogeneous Reconfigurable Optical Network*) corresponde a un proyecto europeo de redes heterogéneas que tiene como objetivo desarrollar una estrategia para controlar de manera eficiente la red mediante la toma de decisiones cognitivas sobre el uso eficiente de los recursos en un escenario heterogéneo, además de cumplir con los requisitos QoS de cada tipo de aplicación y servicio compatible con la red [73]. CHRON depende de la cognición, por lo que las decisiones de control deben tomarse con un conocimiento apropiado del estado actual y respaldado por un proceso de aprendizaje para mejorar el rendimiento con la experiencia adquirida.



2.6. Arquitectura CHRON [73]

La arquitectura del modelo CHRON como se muestra en la Figura 2.6. está compuesta por un (CDS, *Cognitive Decisión System*) que determina como manejar las demandas de tráfico o eventos de red, así mismo de optimizar el uso y rendimiento de la red, teniendo en cuenta tanto el estado actual de la red como la experiencia adquirida. (CMS, *Control Mananger System*) son los encargados de las actualizaciones sobre el estado de la red y la disponibilidad de los recursos, además recibe las instrucciones tomadas por los CDS para entregarlas a todos los nodos interesados y vigila el proceso de configuración del dispositivo, notificando a los CDS de cualquier mal funcionamiento o anomalía en la red [73].

La arquitectura de CHRON también incluye SAEs (*Software-Adaptable elements*) los cuales se configuran de acuerdo con las decisiones tomadas por los CDS y por lo tanto permiten la adaptación de la red a las condiciones actuales. Los NMons (*Network Monitors)* son los encargados de proporcionar el estado actual del tráfico de la red a los CDS. Las funcionalidades de adaptabilidad y monitoreo se manejan en cada elemento de red a través de un administrador de capa física, que funciona como una interfaz común hacia el sistema de control y administración [85].

2.2.5. Redes Cognitivas Definidas Por Software

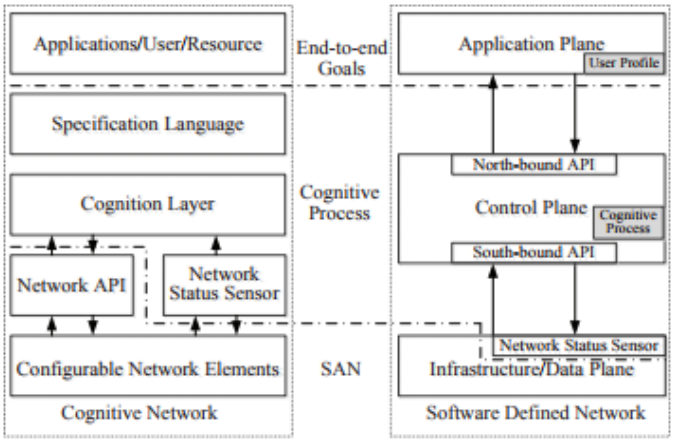
Las arquitecturas de redes han tenido una evolución en los últimos años tanto es así que compañías como Google han implementado *OpenFlow* en su red interna aumentado el uso promedio de Google WAN link en aproximadamente un 99%,

proporcionando nuevas experiencias para las próximas generaciones de redes donde se busca que sean más inteligentes, flexibles y reconfigurables para el manejo de grandes velocidades y las adaptaciones de nuevos formatos de modulación proponiendo nuevos desafíos para posteriores investigaciones.

En ese mismo sentido se ha podido probar y demostrar que por medio de la orquestación cognitiva SDN se pueden gestionar eficientemente los servicios de extremo a extremo (E2E) en escenarios de multi-tecnologia y multi-dominio utilizando el ciclo cognitivo [74]. También se ha podido exponer la utilización de el algoritmo LARAC [75] para calcular valores de predicción para obtener la ruta más rápida de extremo a extremo donde la red SDN cognitiva aprovecha la respuesta para reacomodar el flujo de servicio.

2.2.5.1. Arquitectura

El principal objetivo de las redes cognitivas definidas por software está en la adaptación dinámica de los recursos de red disponibles, para proporcionar soluciones novedosas, inteligentes y adaptativas a los problemas de enrutamiento, asignación de espectro, entre otros, introduciendo la programación a los elementos de Red y centralizando lógicamente el plano de control [76]. En la figura 2.7. se presenta la correlación entre los componentes de las arquitecturas de redes cognitivas y redes definidas por software donde su meta es que los usuarios, aplicaciones o recursos especifiquen los objetivos de la red.



2.7. Correlación Entre Redes Cognitivas y SDN[77]

El aspecto referido como SAN [77] comprende elementos de red que son ajustables en tiempo de ejecución y proporcionan espacio de acción para el proceso cognitivo. El proceso cognitivo es el aprendizaje real y la decisión es la respuesta adecuada basada en el comportamiento observado de la red [77]. Los objetivos extremo a extremo son realizados en forma de aplicaciones del modelo SDN, de forma similar el motor

cognitivo se implementa en el plano de aplicación de SDN el cual recibe la información del estado de la red desde un modelo del proceso cognitivo en el controlador [77].

En el capítulo anterior se presentó la evaluación de los modelos cognitivos que se encargan principalmente del estado actual de la red, y con esta información tomar decisiones y actuar sin descuidar el parámetro de desempeño. Seguidamente la red aprende, guardar la información y tomar decisiones de acuerdo a esta experiencia adquirida.

Así mismo, se dio a conocer las diferentes metaheuristicas basadas en población, con el fin de dar solución al problema de la escogencia de la ruta con menos probabilidad de bloqueo en el flujo de enrutamiento de las redes definidas por software.

Igualmente en el capítulo 3 se realiza el análisis de resultados de la probabilidad de bloqueo en dos escenarios diferentes, la red definida por software con el algoritmo de enrutamiento por defecto y la red definida por software bajo el método cognitivo CHRON con el algoritmo de enrutamiento ACO, por medio de simulaciones y pruebas establecidas previamente.

Capítulo 3. Simulación, pruebas y análisis de resultados

La simulación es un procedimiento que se refiere a la operación de un modelo numérico que representa la estructura de un proceso dinámico con el propósito de representar la conducta del proceso a través del tiempo y así tener una mejor compresión de los conceptos estudiados. En ese mismo sentido simular es la representación de la realidad mediante el empleo de un modelo u otro mecanismo que reaccionara del mismo modo que la realidad bajo una serie de condiciones dadas [wiki].

En el presente capítulo se desarrolla el procedimiento de diseño de los escenarios de simulación en diferentes herramientas (MININET, *OpenDayLight*, *Virtual Tenant Network*) bajo una arquitectura SDN, enfocado en el proceso de enrutamiento, con el fin de evaluar y analizar el parámetro de probabilidad de bloqueo de los enlaces en la red SDN.

Teniendo en cuenta la base conceptual se realiza la simulación de las topologías de red Prueba TEST, NSFNET e IRIS10 bajo la arquitectura SDN con flujo de tráfico Ratón y/o Elefante (ITEM 3.2.1.5), donde su variación esta tanto en el modelo cognitivo como en el algoritmo de enrutamiento. Todo esto con base en los lineamientos de una metodología de simulación.

- Primero se hace una descripción de las herramientas de simulación requeridas (MININET, *OpenDayLight* y *Virtual Tenant Network*) para la implementación de las redes de prueba.
- Segundo se da a conocer la metodología de simulación escogida en el presente documento para evaluar la probabilidad de bloqueo de los enlaces en las redes de prueba simuladas.
- Tercero se realiza el análisis comparativo de la red bajo la arquitectura SDN con el algoritmo de enrutamiento por defecto (*Simple Forwarding*) y la red SDN cognitiva (CHRON) bajo el algoritmo de enrutamiento ACO, con flujo de tráfico Elefante y/o ratón para determinar el impacto en el parámetro de desempeño evaluado.

con los desarrollos hechos en este capítulo, se realiza un gran aporte al cumplimiento del objetivo específico número tres expuesto en el anteproyecto: ***"Evaluar el método cognitivo del ítem anterior mediante simulación en las topologías de red Prueba TEST, NSFNET e IRIS10 con flujo de tráfico Elefante y/o Ratón para analizar la probabilidad de bloqueo de los enlaces en la red inicial SDN/NFV y la red con el***

método cognitivo CHRON en la función de enrutamiento" y por ende, al objetivo general del trabajo de grado: ***"Analizar el desempeño de una red SDN/NFV mediante la aplicación de un método cognitivo basado en ACO"***.

3.1. Herramientas de simulación

3.1.1. VMWARE

El software de virtualización VMWARE corresponde a una compañía de software enfocada en el campo de la virtualización de sistemas y computación en la nube. VMWARE permite a los usuarios crear múltiples entornos virtuales o sistemas informáticos virtuales en una sola computadora o servidor. Es decir, básicamente una computadora o servidor podría usarse para alojar o administrar muchos sistemas informáticos virtuales. El software virtualiza los componentes de hardware como adaptadores de red, discos duros, tarjetas de memoria RAM, entre otros. Para las empresas, esto es especialmente útil para configurar múltiples sistemas de servidor sin tener que comprar hardware por separado para cada uno de ellos.

VMWARE ESXI, como se muestra en la figura 3.1 es la plataforma encargada de alojar los servidores virtuales los cuales están configurados bajo el sistema operativo Linux (*Ubuntu 14 y Centos 7*). Dentro del trabajo de investigación se opta por la creación de tres máquinas virtuales en similitud con el modelo SDN, donde la primera máquina virtual bajo el sistema operativo *Ubuntu 14* tiene alojada la herramienta *MININET*, la segunda máquina virtual bajo el sistemas operativo *Ubuntu 14* tiene alojado el aplicativo *Carbón de OpenDayLight* y por ultimo una máquina virtual bajo el sistema operativo *Centos 7* para alojar la herramienta *VTN Coordinator de OpenDayLight*.

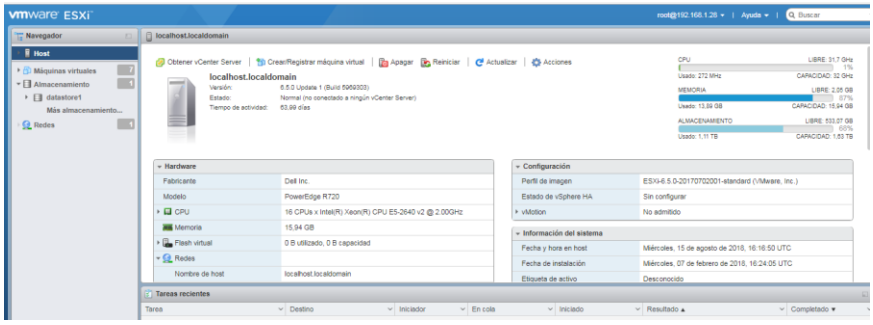


Figura 3.1. Software de virtualización vmware ESXI.

3.1.2. MININET

La herramienta MININET es una plataforma de emulación de red que soporta un rápido desarrollo en SDN mediante el protocolo *OpenFlow*. Es la plataforma de simulación de infraestructura para SDN más utilizada por los investigadores debido a su simplicidad, disponibilidad y flexibilidad. Además, MININET está totalmente dedicado a la arquitectura *OpenFlow*. *MININET* utiliza las extensiones de *Linux* junto con S*cripts* de lenguaje *Python*, para construir una red virtual de gran número de anfitriones (*Host*), conmutadores (*Switch*) *OpenFlow* y controladores en cualquier topología de red que el investigador emplea en una única estación de trabajo [87].

MININET podría usar sus herramientas de software incorporadas, para desarrollar topología de redes a través de la interfaz de línea de comandos (CLI), o se adapta a herramientas de software de terceros que implementan otros controladores o motores de interfazs gráficas.

MININET cuenta con un simulador de interfaz gráfico llamado MINIEDIT como se muestra en la figura 3.2. MINIEDIT es un complemento de MININET que presenta un lienzo con iconos y herramientas por medio del cual se pueden crear topologías de rede en modo gráfico.

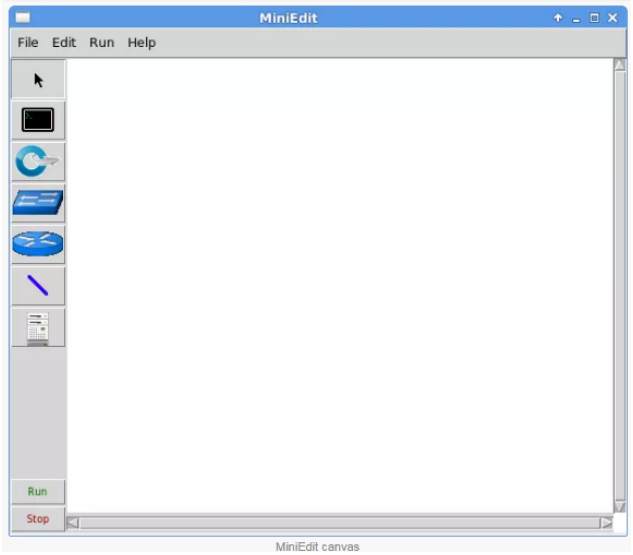


Figura 3.2. MINIEDIT.

Dentro del complemento MINIEDIT se desarrolla la simulación de la topología de red Prueba TEST, NSFNET e IRIS10 para los casos de uso de este trabajo de investigación como se muestra en la tabla 3.1.

3.1.3. OpenDayLight Nitrogen

(ODL, *OpenDayLight*) es un proyecto creado en 2013 por *Linux Foundation* cuyo principal objetivo es la creación de una plataforma SDN basada en código abierto, para mitigar la complejidad de la infraestructura de red, además de desplegar nuevos servicios y capacidades. La plataforma abierta SDN de ODL utiliza el protocolo *OpenFlow* y proporciona un controlador modular y flexible que se implementa en software y contiene su propia máquina virtual en *Java*.

ODL Nitrógeno como se muestra en la figura 3.3 es un proyecto de simulación enfocado en SDN donde su principal característica está en el contenedor *Apache Karaf* que se utiliza para coordinar micro servicios ODL, proporcionando una infraestructura de registro común, habilitando la administración remota a través de *JMX y Shell Unix*, facilitar la configuración dinámica, la implementación en caliente y proporcionar un conjunto común de recursos base en todo el producto. El proyecto de *Apache Karaf* tiene varias responsabilidades y ha servido como una pieza crucial de infraestructura para la arquitectura general de *OpenDayLight* desde su inclusión en el lanzamiento de Helium.

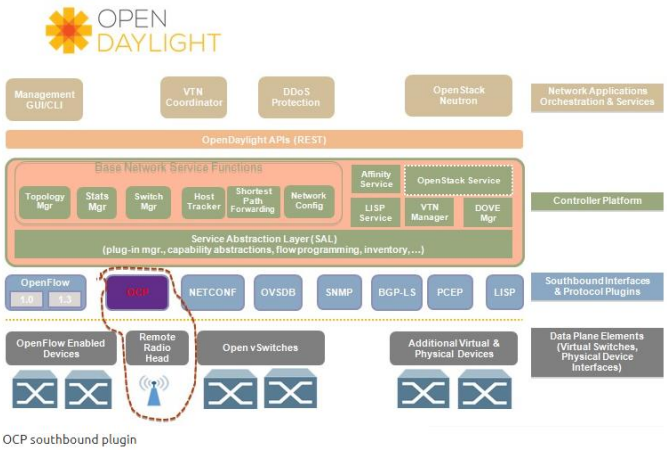


Figura 3.3. Arquitectura *OpenDayLight* Nitrógeno.

ODL Nitrógeno funciona en la capa de control del modelo SDN y dentro del trabajo de investigación a implementar, será el encargado de administrar la infraestructura de la red, brindar reportes de funcionamiento, generar la comunicación entre la capa de infraestructura y la capa de aplicación con el propósito de integrar la VTN con los dispositivos de red físicos.

3.1.4. Virtual Tenant Network

VTN Coordinador está en la capa más elevada del modelo SDN haciendo parte de la

capa de aplicación, orquestación y servicios, funciona como una aplicación externa que proporciona una interfaz REST para que el usuario pueda usar la virtualización de *OpenDayLight* [78]. VTN *Coordinator* usa la interfas REST expuesta por VTN *Manager* para generar la red virtual usando *OpenDayLight*.

La función vBrigde de VTN proporciona un puente para la transferencia de paquetes a un puerto virtual de acuerdo a las direcciones MAC de destino como se muestra en la Figura 3.4. El aporte de VTN a este trabajo de investigación está en la implementación de la función de enrutamiento bajo el algoritmo ACO dentro de la capa más elevada del modelo SDN, y así cumplir con el modelo cognitivo CHRON.

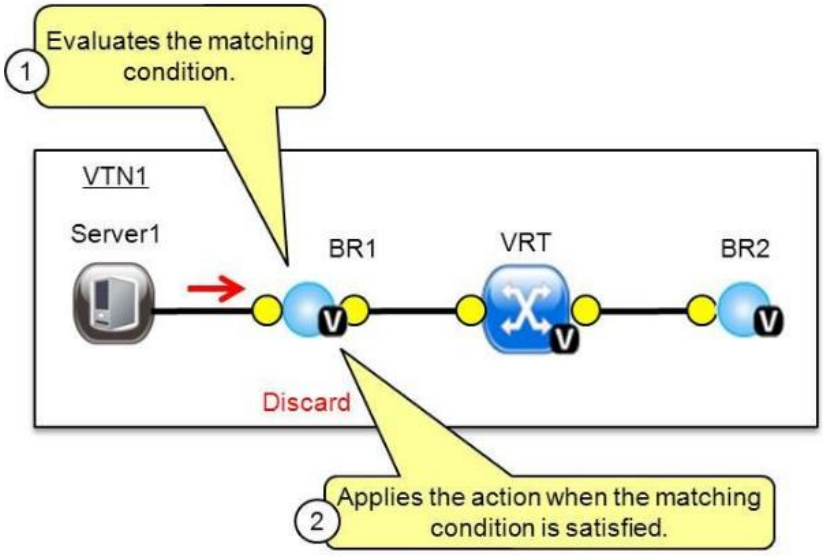


Figura 3.4. Función vBrigde [78]

3.2. Metodología de simulación

Para el desarrollo de este proyecto y en busca de responder a la pregunta de investigación, además de alcanzar los objetivos propuestos es de vital importancia planificar y determinar la metodología a utilizar. Por lo anterior, la metodología seleccionada para la simulacion es la Perspectiva - Práctica como se muestra en el siguiente diagrama de flujo figura 3.5.

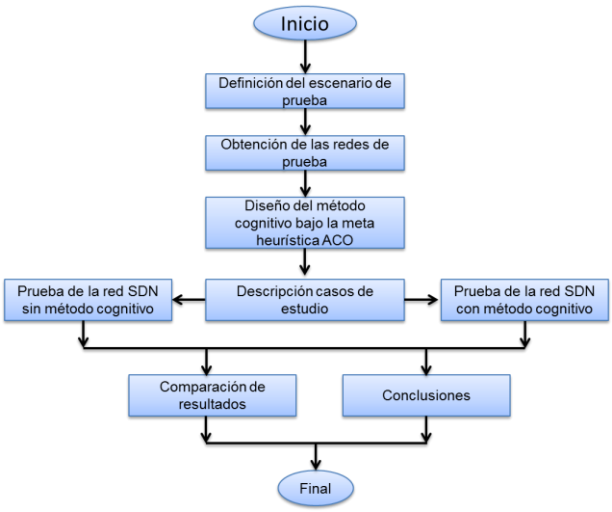


Figura 3.5. Diagrama de flujo metodología de simulación.

Aunque estas fases se aplican generalmente en secuencia, es posible que se requiera volver a las fases anteriores debido a los cambios en el ámbito. En resumen el proceso puede repetirse para cada alternativa estudiada o desarrollada como parte del proyecto.

3.2.1. Fase 1: definición del escenario de prueba

Con el propósito de cumplir con el objetivo general de este trabajo de investigación: "Analizar el desempeño de una red SDN/NFV mediante la aplicación de un método cognitivo basado en ACO", se traza como el inicio, la definición de los espacios del escenario de simulación, donde posteriormente se plantean temas de estudio relevantes, en la búsqueda de obtener los resultados esperados en las redes de prueba. A continuación, se definen las topologías de red para este trabajo de investigación.

3.2.1.1. Diseño de las topologías de red

El espacio del escenario es la topología de red, donde se piensa ubicar la red para almacenar todos los elementos del escenario, conectar nodos y establecer las asociaciones entre los dispositivos. Un aspecto relevante es definir la topología de la red. La topología de red se puede definir como la interconexión de diferentes dispositivos de red indicando nodos, enlaces de interconexión, sitios geográficos entre otros ubicados dentro de un mapa. Desde el punto de vista del control se cuenta con dos tipos de topología las jerárquicas y las planas. La topología jerárquica dependen de una raíz o nodo principal que controla la operación de la red mientras que la topología plana tiene un carácter más distribuido y es principalmente usada en redes

tipo WAN. Además las topologías pueden ser irregulares o regulares dependiendo de la distribución de la arquitectura de red. Debido a las características de este trabajo de investigación se ha seleccionado topologías tipo WAN irregulares que no tienen similitud a formas (estrella, bus, malla, anillo, árbol, entre otras) y una topología tipo LAN de forma Malla completamente conectada, como se muestra en la siguiente tabla resumen 3.1.

Tabla 3.1: Resumen datos técnicos topologías de red

ITEM	Red NSFNET	Red IRIS10	Red de prueba TEST
Ubicación geográfica	Estados unidos	Europa	Colombia
Topología	Irregular	Irregular	Malla
Numero de nodos	14	19	4
Numero de enlaces	21	29	6
Longitud enlace (Km)	1086	256	300

De acuerdo a la selección de estas topologías se procede a la descripción de las mimas.

3.2.1.2. Topología de red prueba TEST

La topología de red prueba TEST, toma el *Backbone* de la antigua red de Movistar Colombia e interconecta las principales ciudades, Bogotá, Medellín, Barranquilla y Santiago de Cali. La topología de red Prueba TEST tiene una configuración de red malla completamente conectada como se muestra en la figura 3.6.

Figura 3.6. Topología de red de prueba TEST.

3.2.1.3. Topología de red NSFNET

La topología de red (NSFNET*, National science foundation network*) se fundó en 1986, donde fue creado con el fin de coordinar, promover, investigar y educar de forma avanzada en trabajos de redes en los estados unidos, conecta 14 ciudades como se muestra en la Figura 3.7, posee una topología irregular y es de gran importancia debido a que la gran mayoría de pruebas se llevan a cabo sobre esta topología. Así mismo es una de las topologías más utilizadas para el análisis de desempeño en diferentes trabajos de investigación.

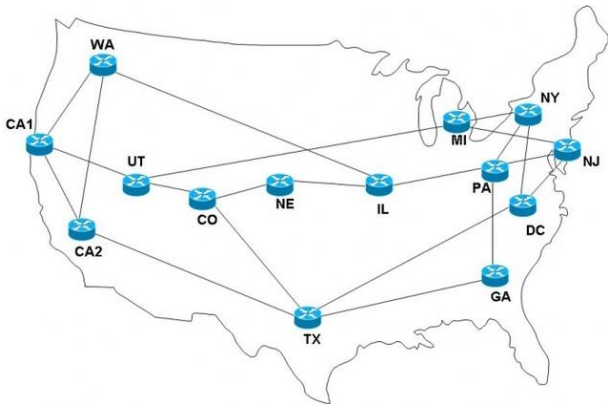


Figura 3.7. Topología de Red NSFNET [79]

3.2.1.4. Topología de red IRIS10

Red IRIS10 es una red de investigación española, fundada en 1988, que proporciona servicios avanzados de comunicaciones para la comunidad científica y universitaria. Interconecta 19 universidades y centros de investigación como se muestra en la figura 3.8 por medio de una red hibrida donde se soportan servicios de conmutación de circuitos y conmutación de paquetes, además es una de las topologías como mayores desafíos en la investigación.

Figura 3.8. Topología de red IRIS10[80]

3.2.1.5. Flujo elefante y ratón

El fenómeno de los flujos de tráfico elefante y ratón nos contribuye en la búsqueda de la mejor opción para inundar la red con diferentes tipos de tráfico, y así poder determinar el comportamiento y rendimiento de las redes de prueba objeto de este estudio. Una característica notable de los flujos elefante, es que contribuyen a una gran parte del volumen total de tráfico de las redes, a pesar de ser relativamente pocos en el número de flujos, mientras que los flujos ratón tienen una pequeña cantidad de paquetes. Por lo tanto, el impacto de los flujos elefante en el rendimiento de la red es significativo [88].

En conclusión, los flujos elefante pueden transferir mucho más tráfico en la red aunque el número de ellos sea pequeño como por ejemplo copias de seguridad, movimiento de máquinas virtuales, FTP, entre otros. Y los flujos ratón pueden transferir mucho menos tráfico en la red aunque el número de ellos sea grande como tráfico web, solicitudes HTTP y HTTPS en los puertos 80 y 443, entre otros. En comparación los flujos elefante tienen más probabilidad de causar una congestión en la red que afecte el rendimiento de la misma.

Es así, que el manejo del flujo de tráfico en este trabajo de investigación se realizara tomando tanto flujo ratón como flujo elefante para las redes de prueba, y así conocer el comportamiento del parámetro de desempeño a medir.

3.2.1.6. Parámetro de desempeño: probabilidad de bloqueo

La probabilidad de bloqueo para este trabajo de investigación la definimos como la evaluación del bloqueo de cada conexión física, suponiendo independencia entre la probabilidad de bloqueo de los enlaces, y la probabilidad de bloqueo de la red. Es decir, la relación entre el número de paquetes recibidos y el número de paquetes enviados por cada enlace físico, donde él envio de flujo ratón y elefante de un nodo a otro es cuantificada tanto en la transferencia como en la recepción. En otras palabras, la evaluación de la probabilidad de bloqueo de los enlaces físicos es independiente a la probabilidad de bloqueo de la red. Dado esto como un proceso importante para controlar la congestión del tráfico en la red.

El parámetro de desempeño para la evaluación de la red, sin método cognitivo y la red con método cognitivo está dado por la probabilidad de bloqueo que está determinado por la ecuación (1) [83].

$$1 - \left(\frac{Pr}{Pe}\right) \qquad (1)$$

$$donde\ Pr = Paquetes\ recibidos; \qquad Pe = Paquetes\ enviados$$

3.2.2. Fase 2: Obtención de las redes de prueba

En esta sección se describe la configuración realizada para cada topología de red de prueba en arquitectura SDN para llevar a cabo el proceso de simulación, además se implantan dentro de las redes de prueba los casos de estudio a evaluar.

Con el fin de evaluar el parámetro de desempeño se implementan, 3 redes de prueba con topologías diferentes, las cuales se configuran con inyección de flujo de tráfico ratón y elefante uniformemente entre los puertos físicos, seguída sobre las topologías de red se generan 3 variaciones de tráfico entre host con distintos niveles de flujo tanto ratón como elefante, con el propósito de separar los tráficos y tener un análisis especifico del comportamiento de la red y así, tener una idea del comportamiento de un sistema real, como se muestra en la Figura 3.9.

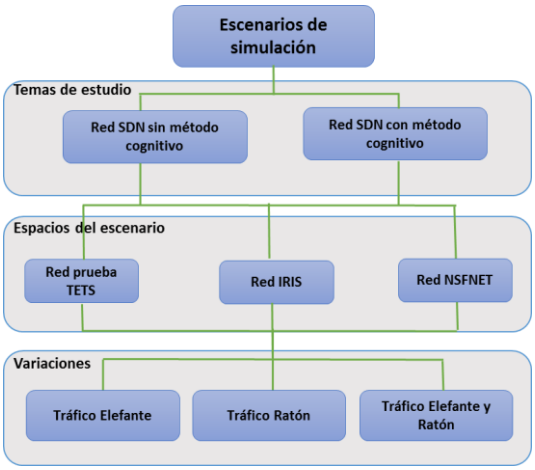


Figura 3.9. Escenarios y casos de simulación

Las variaciones del escenario son las encargadas de cambiar el funcionamiento del sistema con el propósito de analizar el comportamiento de las redes de prueba con diferentes cargas. Los espacios del escenario son las topologías de red escogidas para la implantación de los temas de estudio. Los temas de estudio son las configuraciones de la capa más alta del escenario de simulación para la evaluación de la red SDN sin método cognitivo y la red SDN con método cognitivo.

La implementación del escenario de simulación consta de componentes virtuales. Estos componentes virtuales incluyen servidores, características y aplicaciones que se utiliza para ejecutar las máquinas virtuales del ambiente de simulación, las cuales se muestran en la tabla 3.2.

Tabla 3.2. Servidores, características y aplicaciones.

	VMWARE ESXi, 6.0.0						
	NOMBRE	**CPU**	**RAM**	**SISTEMA OPERATIVO**	**HERRAMIENTA**	**FUNCIONALIDAD**	**APLICACIONES**
SERVIDORES	ServMININET	2 core	2 Gbts	Ubuntu 14	MININET	Simular la red NSFNET, programación en PYTHON	MININET PYTHON
	ServODLNITROGENO	2 core	2 Gbts	Ubuntu 14	*OPENDAYLIGHT*	Controlador de infraestructura de red simulada en MININET	SDN *OPENDAYLIGHT* VTN CONTROLADOR
	ServVTN	2 core	2 Gbts	Centos 7	VIRTUAL TENANT NETWORK	Aplicación, Vrouter implementación del método cognitivo basado en la meta heurística ACO.	NFV VTN CORDINADOR

De acuerdo a los diferentes componentes virtuales que requiere el escenario de simulación se hace necesaria la inclusión de un *hypervisor*, que se utiliza para ejecutar las 3 máquinas virtuales que harán parte del ambiente de simulación como se muestra en la figura 3.10.

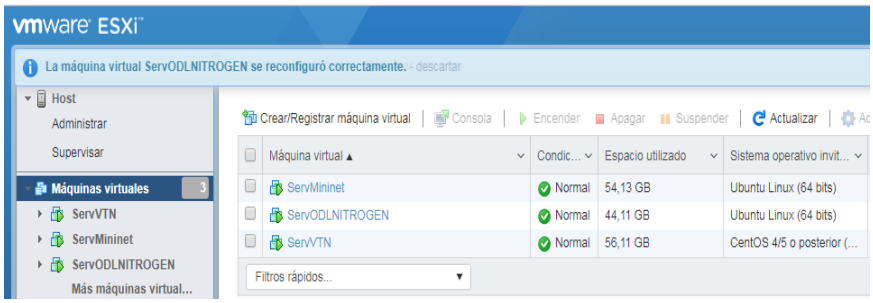


Figura 3.10. *Hypervisor* Ambiente de Simulación

3.2.2.1. Simulación Topología de red

El proceso de simulación de la topologías de red (Red prueba TEST, NSFNET e IRIS10), se configuran en la herramienta MININET por medio de un paquete integrado denominado MINIEDIT el cual permite crear las topologías de red de manera gráfica. Como se muestra en las figura 3.11, 3.12 y 3.13 respectivamente.

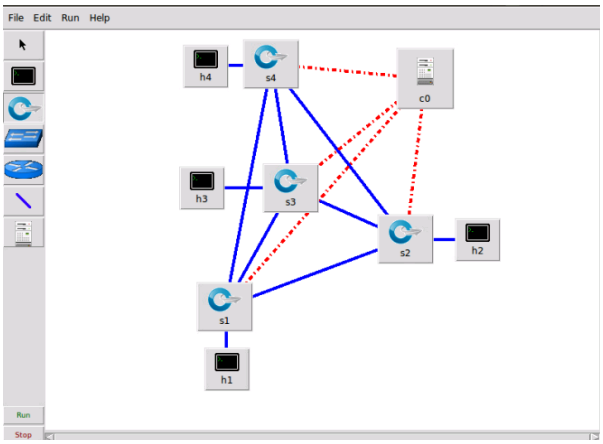


Figura 3.11. Topología de red prueba TEST en MINIEDIT.

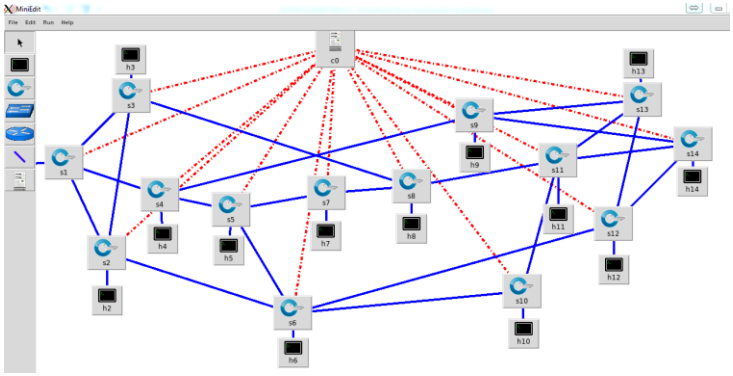


Figura 3.12. Topología de red NSFNET en MINIEDIT

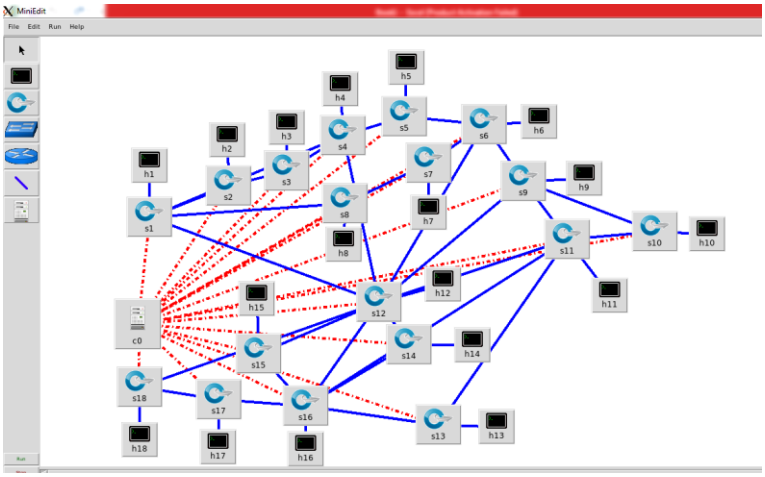


Figura 3.13. Topología de red IRIS10 en MINIEDIT

Para el proceso de simulación se deben establecer las topologías de red física TEST, NSFNET e IRIS10. Con el propósito de realizar la configuración de las topologías de red en las extensiones TEST.py, NSFNET.py e IRIS10.py como se describe en los anexos 1, 2 y 3. Cabe aclarar que el lenguaje de programación utilizado en la herramienta MININET es PYTHON.

Con el fin de tener la seguridad de que las redes de prueba estén configuradas de una manera óptima existe dentro de la herramienta MININET un comando "***net***" que permite dar a conocer la configuración de cada uno de los elementos que hacen parte de la red configurada.

Tabla 3.3. Comando ***Net*** en topologías de red simuladas MININET.

Comando net Red prueba TEST
MININET> net h3 h3-eth0:s3-eth4 h2 h2-eth0:s2-eth4 h1 h1-eth0:s1-eth4 h4 h4-eth0:s4-eth4 s4 lo: s4-eth1:s3-eth1 s4-eth2:s1-eth3 s4-eth3:s2-eth3 s4-eth4:h4-eth0 s3 lo: s3-eth1:s4-eth1 s3-eth2:s2-eth1 s3-eth3:s1-eth1 s3-eth4:h3-eth0 s1 lo: s1-eth1:s3-eth3 s1-eth2:s2-eth2 s1-eth3:s4-eth2 s1-eth4:h1-eth0 s2 lo: s2-eth1:s3-eth2 s2-eth2:s1-eth2 s2-eth3:s4-eth3 s2-eth4:h2-eth0 c0
Comando net Red NSFNET
MININET> net h5 h5-eth0:s5-eth4 h1 h1-eth0:s1-eth4 h11 h11-eth0:s11-eth5 h12 h12-eth0:s12-eth4 h14 h14-eth0:s14-eth4 h2 h2-eth0:s2-eth4 h13 h13-eth0:s13-eth4 h10 h10-eth0:s10-eth3 h8 h8-eth0:s8-eth4 h3 h3-eth0:s3-eth4 h6 h6-eth0:s6-eth5 h7 h7-eth0:s7-eth3 h9 h9-eth0:s9-eth4 h4 h4-eth0:s4-eth4 s2 lo: s2-eth1:s3-eth2 s2-eth2:s1-eth2 s2-eth3:s6-eth2 s2-eth4:h2-eth0 s3 lo: s3-eth1:s1-eth1 s3-eth2:s2-eth1 s3-eth3:s8-eth2 s3-eth4:h3-eth0 s7 lo: s7-eth1:s5-eth2 s7-eth2:s8-eth1 s7-eth3:h7-eth0 s4 lo: s4-eth1:s1-eth3 s4-eth2:s5-eth1 s4-eth3:s9-eth1 s4-eth4:h4-eth0 s14 lo: s14-eth1:s11-eth2 s14-eth2:s12-eth2 s14-eth3:s9-eth3 s14-eth4:h14-eth0 s6 lo: s6-eth1:s5-eth3 s6-eth2:s2-eth3 s6-eth3:s10-eth1 s6-eth4:s12-eth3 s6-eth5:h6-eth0 s5 lo: s5-eth1:s4-eth2 s5-eth2:s7-eth1 s5-eth3:s6-eth1 s5-eth4:h5-eth0 s9 lo: s9-eth1:s4-eth3 s9-eth2:s13-eth3 s9-eth3:s14-eth3 s9-eth4:h9-eth0 s1 lo: s1-eth1:s3-eth1 s1-eth2:s2-eth2 s1-eth3:s4-eth1 s1-eth4:h1-eth0 s13 lo: s13-eth1:s12-eth1 s13-eth2:s11-eth3 s13-eth3:s9-eth2 s13-eth4:h13-eth0 s8 lo: s8-eth1:s7-eth2 s8-eth2:s3-eth3 s8-eth3:s11-eth1 s8-eth4:h8-eth0 s12 lo: s12-eth1:s13-eth1 s12-eth2:s14-eth2 s12-eth3:s6-eth4 s12-eth4:h12-eth0
Comando net Red IRIS10
MININET> net h14 h14-eth0:s14-eth3 h1 h1-eth0:s1-eth1 h16 h16-eth0:s16-eth7 h10 h10-eth0:s10-eth3 h6 h6-eth0:s6-eth5 h15 h15-eth0:s15-eth3 h18 h18-eth0:s18-eth3 h2 h2-eth0:s2-eth1 h8 h8-eth0:s8-eth1 h11 h11-eth0:s11-eth6 h3 h3-eth0:s3-eth1 h13 h13-eth0:s13-eth3 h9 h9-eth0:s9-eth4 h5 h5-eth0:s5-eth1 h12 h12-eth0:s12-eth6 h7 h7-eth0:s7-eth1 h17 h17-eth0:s17-eth3 h4 h4-eth0:s4-eth1 s7 lo: s7-eth1:h7-eth0 s7-eth2:s6-eth2 s7-eth3:s8-eth2 s12 lo: s12-eth1:s9-eth2 s12-eth2:s11-eth3 s12-eth3:s14-eth1 s12-eth4:s15-eth1 s12-eth5:s16-eth5 s12-eth6:h12-eth0 s12-eth7:s18-eth1 s12-eth8:s1-eth3 s12-eth9:s4-eth4 s12-eth10:s6-eth3 s3 lo: s3-eth1:h3-eth0 s3-eth2:s2-eth3 s3-eth3:s4-eth2 s15 lo: s15-eth1:s12-eth4 s15-eth2:s16-eth4 s15-eth3:h15-eth0 s9 lo: s9-eth1:s10-eth1 s9-eth2:s12-eth1 s9-eth3:s11-eth2 s9-eth4:h9-eth0 s9-eth5:s6-eth4 s10 lo: s10-eth1:s9-eth1 s10-eth2:s11-eth1 s10-eth3:h10-eth0 s6 lo: s6-eth1:s5-eth3 s6-eth2:s7-eth2 s6-eth3:s12-eth10 s6-eth4:s9-eth5 s6-eth5:h6-eth0

Por otro lado el protocolo *OpenFlow*, crea la conexión entre la herramienta MININET (infraestructura de red) y el controlador *OpenDayLight*, logrando así, exportar lógicamente la infraestructura de las redes, del plano de datos al plano de control. A continuación se muestra las redes obtenidas en el controlador *OpenDayLight (*ver Figura 3.14, 3.15 y 3.16).

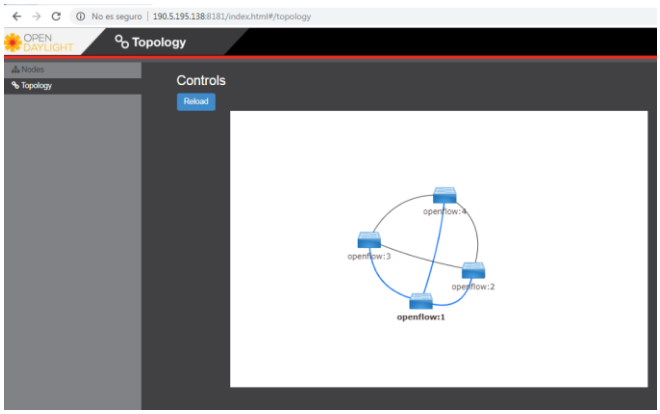


Figura 3.14. Topología de red prueba TEST en *OpenDayLight.*

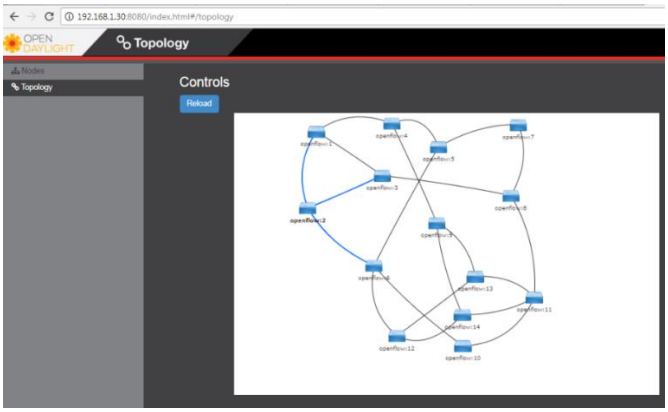


Figura 3.15. Topología de red NSFNET en *OpenDayLight*

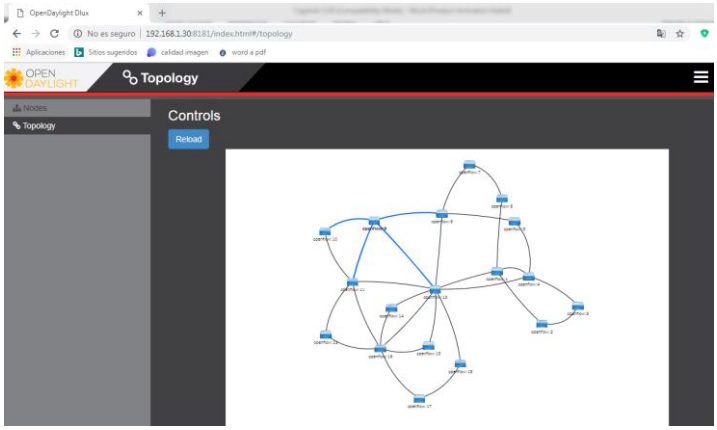


Figura 3.16.Topología de red IRIS10 en *OpenDayLight*.

OpenDayLight hace parte de la capa de control del modelo SDN, la cual gestiona la infraestructura de red, así como también brinda las estadísticas pertinentes para evaluar el parámetro de desempeño.

3.2.3. Fase 3: Diseño del método cognitivo bajo la metaheurística ACO

Para la obtención del método cognitivo, fue muy importante la información brindada en los documentos [73][81][82]. Aunque está relacionada para redes ópticas el modelo cognitivo CHRON es el que más se adapta a la implementación de este proyecto de acuerdo a la base científica especificada anteriormente.

De acuerdo a los capítulos 1 y 2, se procede a realizar el diseño del método cognitivo CHRON bajo la metaheurística ACO
En principio, se toma como referencia el sistema mostrado en la figura 2.6 (Capitulo 2). La arquitectura CHRON donde la red implementa un proceso que puede percibir las condiciones actuales de la red, y luego planificar y actuar en esas condiciones. La red puede aprender de esas adaptaciones y utilizarlas para tomar decisiones futuras de acuerdo a los objetivos propuestos.

En la figura 3.17 se crea una implementación basada en SDN y CHRON, con el propósito de generar una integración entre estos modelos, orientados al ciclo cognitivo y buscar una mejora en el parámetro de desempeño.

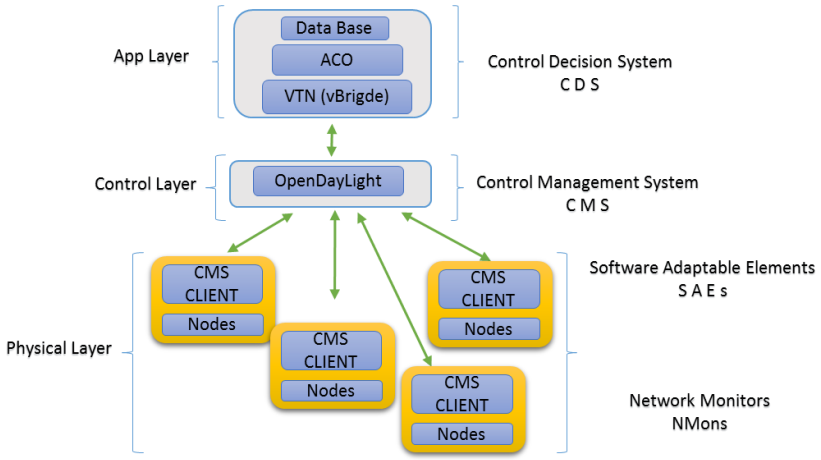


Figura 3.17. Método de control cognitivo trabajo de investigación.

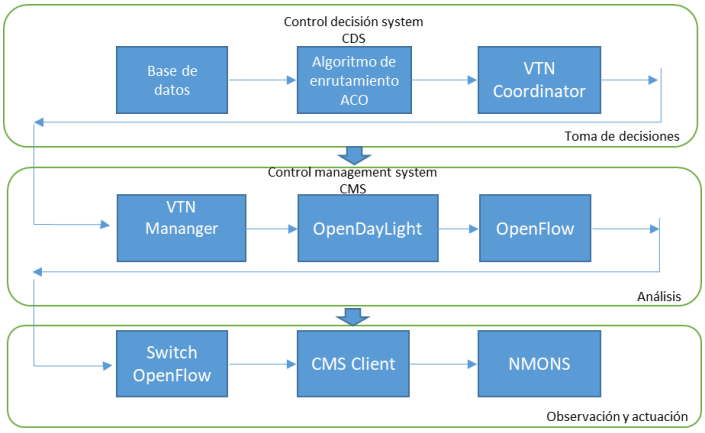
El desarrollo del método cognitivo se basa en la meta heurística ACO que dentro del modelo establecido se encuentra en la capa de aplicación del modelo SDN y hace parte de la CDS de CHRON. La arquitectura CDS se divide en diferentes módulos (*Data Base*, ACO, VTN) todas explotan la cognición y actúan de forma orquestada para lograr el comportamiento deseado. El algoritmo de enrutamiento basado en la meta heurística ACO. será implantado sobre los puertos lógicos importados por medio de la

característica VTN, cumpliendo así con las acciones del ciclo cognitivo, observación y toma de decisiones, lo cual está respaldado por un módulo de aprendizaje (*Data Base*) cuyo objetivo será almacenar las rutas previamente encontradas para en un futuro evitar enviar el algoritmo ACO y así mejorar el rendimiento.

Seguidamente con la integración entre el modelo SDN y la arquitectura CHRON, los CMS serán los encargados de entregar las decisiones tomadas por la capa más elevada del modelo, a los CMS CLIENT por medio del controlador *OpenDayLight* el cual recopila la información sobre el tráfico, los puertos físicos y el estado de la Red desde los NMONS. La capa de control dentro del modelo cognitivo puede verse como el sistema nervioso de la arquitectura y ha sido diseñado como el software de plano de control y gestión.

Finalmente la capa de infraestructura, es la capa donde se encuentran los elementos físicos vistos desde el modelo SDN. De acuerdo a la integración de SDN y CHRON esta capa pasa a tener cognición y de ahora en adelante se observa como el sistema sensorial del modelo el cual se compone por mudulos en primera instacia el modulo CMS CLIENT el cual es el encargado de procesar la información proveniente del controlador *OpenDayLight* por medio del protocolo *OpenFlow,* que genera la comunicaciones entre estas dos capas además de enviar la información proveniente de los monitores de red NMONS al controlador. El segundo son los Monitores de red NMONS, los cuales se encargan principalmente de proporcionar los puertos físicos y el estado del tráfico de la red a los CMS CLIENT como se muestra en el siguiente diagrama de flujo.

Diagrama de flujo funcionamiento método cognitivo basado en CHRON.



3.2.4. Fase 4: Descripción casos de estudio

A continuación se describen los casos de estudio que se determinan para analizar la probabilidad de bloqueo de los enlaces en cada una de las redes por medio de la configuración de 3 *Script*, que se encargan de distribuir el tráfico uniformemente entre

los pares de conmutadores físicos, los cuales de ahora en adelante serán llamados (TráficoTEST.sh, TráficoNSFNET.sh y TráficoIRIS10.sh). Los anexos 3, 4, 5, respectivamente, muestran la configuración del tráfico que inunda las diferentes topologías de red (Prueba TEST, NSFNET e IRIS) y serán utilizados en los dos casos de estudio.

El comando para el envio de tráfico tanto elefante como ratón se describe en la figura 3.18.

```
iperf -c IPaconectar -u -b AnchodebandaM -i 1 -t 99 -p Puerto -l Tamañodelpaquete -m
```

Figura 3.18. Comando envio de tráfico

Iperf: Comando utilizado como cliente para acceder a otro nodo de acuerdo a las configuraciones establecidas.
IPaconectar: IP receptora del tráfico
-b AnchodebandaM: Ancho de banda del canal.
-i 1 -t 999: Número de Iteraciones.
-p Puerto: Puerto de conexión.
Tamañodelpaquete –m: Tamaño del paquete enviado.

Así mismo el tráfico entre *Hosts* desde el punto de vista del análisis se separa con el propósito de tener un estudio efectivo sobre el tráfico y tener una mejor idea del comportamiento real de un sistema, el flujo de tráfico se envía de acuerdo a la configuración de los *Script* (hostratonTEST.sh, hostelefanteyratonTEST.sh, hostelefanteTEST.sh, hostratonNSFNET.sh, hostelefanteyratonNSFNET.sh, hostelefanteNSFNET.sh, hostratonIRIS10.sh, hostelefanteyratonIRIS10.sh y hostelefanteIRIS10.sh) nexos 7, 8, 9, 10, 11, 12, 13, 14 y 15 respectivamente, sobre cada una de las topologías de los casos de estudio. Con el propósito determinar las diferentes estadísticas, como por ejemplo (paquetes enviados, paquetes recibidos, puertos de conexión, entre otros).

Seguidamente, se realiza la configuración del protocolo *OpenFlow* por medio del dispositivo Controlador como se muestra en la figura 3.19.

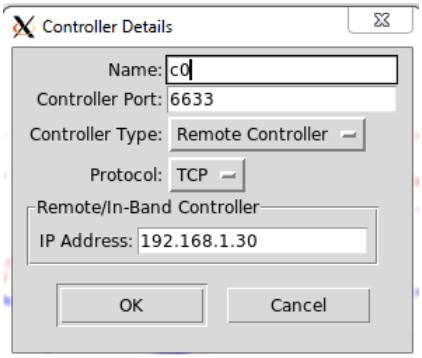


Figura 3.19. Conectividad capa física y capa de control.

Dentro de la configuración de la herramienta MININET existe el dispositivo Controlador, el cual aparte de realizar la conexión entre la capa física y la capa de control, exporta las configuraciones y estadísticas que se dan desde los diferentes dispositivos de red (*Host, Switch,* entre otros) al controlador *OpenDayLight*.

Para obtener todas las configuraciones y las estadísticas de los depósitos de la capa de infraestructura en el controlador *OpenDayLight* es necesario instalar las siguientes características como se muestra en la figura 3.20.

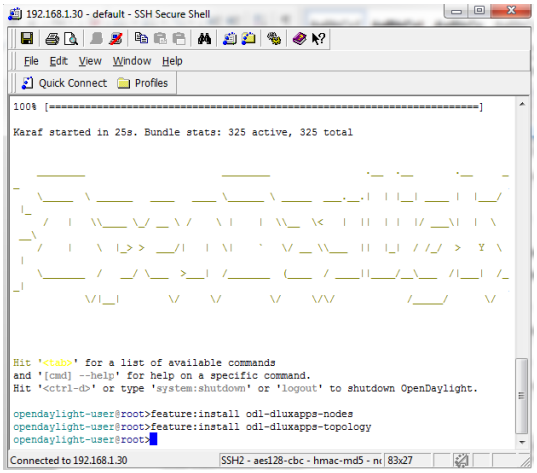


Figura 3.20. Características de configuración *OpenDayLight*.

Las características de *OpenDayLight feature:install*, *odl-dluxapps-nodes* y *feature:install odl-dluxapps-topology* son los encargados de instalar los módulos necesarios que logran extraer de la infraestructura de red, tanto la topología como los nodos interconectados.

Después de haber obtenido las topologías de red en el ítem 3.2.2.1 y condicionado el escenario se debe verificar que los nodos hayan sido exportados con éxito al controlador *OpenDayLight*.

Las figuras 3.21, 3.22 y 3.23 muestran los nodos en *OpenDayLight*, los cuales nos permiten analizar los *OpenFlow*, así como el número de conexiones de cada nodo *OpenFlow* y sus estadísticas. Así mismo en las tablas 3.4, 3.5 y 3.6 se identifican los puertos físicos interconectados entre los nodos de cada una de las topologías de red.

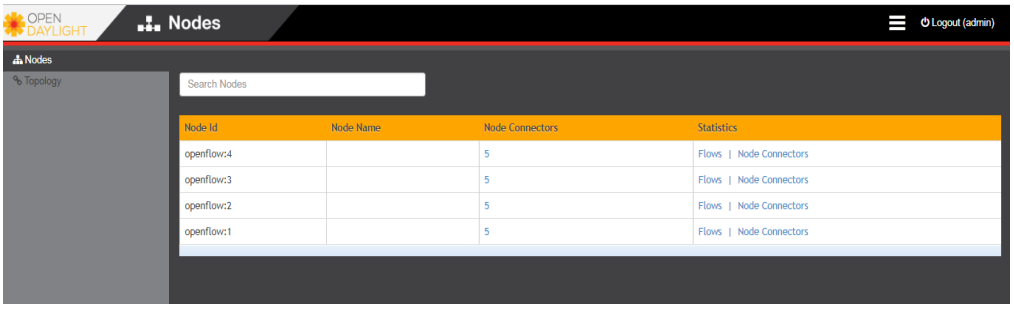


Figura 3.21. Nodos en *OpenDayLight* Prueba TEST.

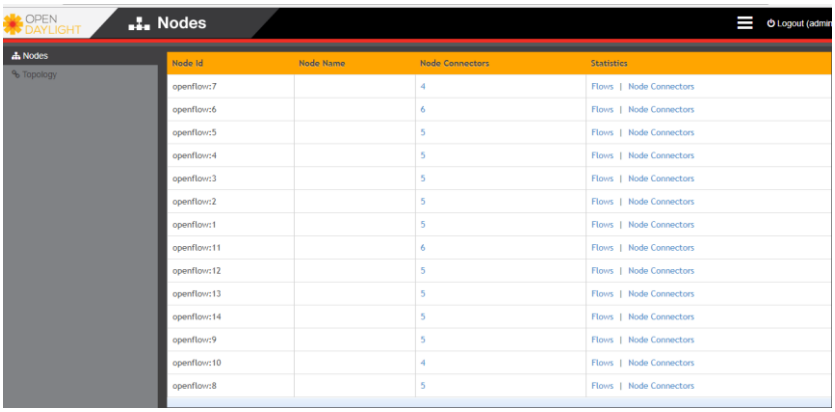


Figura 3.22. Nodos en *OpenDayLight* NSFNET.

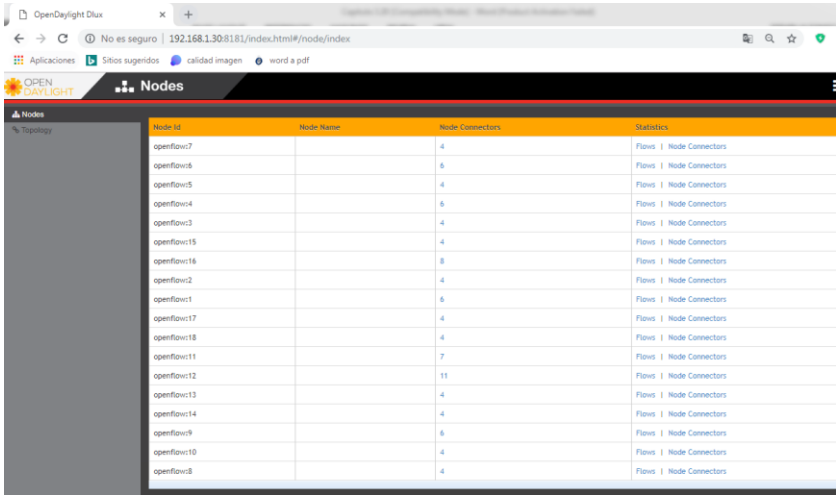


Figura 3.23. Nodos en *OpenDayLight* IRIS10.

Tabla 3.4. Puertos de conexión para cada nodo *OpenFlow* prueba TEST

	Host	**s1**	**s2**	**s3**	**s4**
s1	eth4	X	eth2	eth1	eth3
s2	eth4	eth2	X	eth1	eth3
s3	eth4	eth3	eth2	X	eth1
s4	eth4	eth2	eth3	eth1	X

Tabla 3.5. Puertos de conexión para cada nodo *OpenFlow* NSFNET

	Host	**s1**	**s2**	**s3**	**s4**	**s5**	**s6**	**s7**	**s8**	**s9**	**s10**	**s11**	**s12**	**s13**	**s14**
s1	eth4	X	eth2	eth1	eth3	X	X	X	X	X	X	X	X	X	X
s2	eth4	eth2	X	eth1	X	X	eth3	X	X	X	X	X	X	X	X
s3	eth4	eth1	eth2	X	X	X	X	X	eth3	X	X	X	X	X	X
s4	eth4	eth1	X	X	X	eth2	X	X	X	eth3	X	X	X	X	X
s5	eth4	X	X	X	eth1	X	eth3	eth2	X	X	X	X	X	X	X
s6	eth5	X	eth2	X	X	eth1	X	X	X	X	eth3	X	eth4	X	X
s7	eth3	X	X	X	X	eth1	X	X	eth2	X	X	X	X	X	X
s8	eth4	X	X	eth2	X	X	X	eth1	X	X	X	eth3	X	X	X
s9	eth4	X	X	X	eth1	X	X	X	X	X	X	X	X	eth2	eth3
s10	eth3	X	X	X	X	X	eth1	X	X	X	X	eth2	X	X	X
s11	eth5	X	X	X	X	X	X	X	eth1	X	eth4	X	X	eth3	eth2
s12	eth4	X	X	X	X	X	eth3	X	X	X	X	X	X	eth1	eth2
s13	eth4	X	X	X	X	X	X	X	X	X	X	eth2	eth1	eth3	X
s14	eth4	X	X	X	X	X	X	X	X	eth3	X	eth1	eth2	X	X

Tabla 3.6: Puertos de conexión para cada nodo *OpenFlow* IRIS10

	Host	**s1**	**s2**	**s3**	**s4**	**s5**	**s6**	**s7**	**s8**	**s9**	**s10**	**s11**	**s12**	**s13**	**s14**	**s15**	**s16**	**s17**	**s18**
s1	eth1	X	eth2	X	eth4	X	X	X	eth5	X	X	X	eth3	X	X	X	X	X	X
s2	eth1	eth2	X	eth3	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
s3	eth1	X	eth2	X	eth3	X	X	X	X	X	X	X	X	X	X	X	X	X	X
s4	eth1	eth3	X	eth2	X	eth5	X	X	X	X	X	X	eth4	X	X	X	X	X	X
s5	eth1	X	X	X	eth2	X	eth3	X	X	X	X	X	X	X	X	X	X	X	X
s6	eth5	X	X	X	X	eth1	X	eth2	X	eth4	X	X	eth3	X	X	X	X	X	X
s7	eth1	X	X	X	X	X	eth2	X	eth3	X	X	X	X	X	X	X	X	X	X
s8	eth1	eth3	X	X	X	X	X	eth2	X	X	X	X	X	X	X	X	X	X	X
s9	eth4	X	X	X	X	X	eth5	X	X	X	eth1	eth3	eth2	X	X	X	X	X	X
s10	eth3	X	X	X	X	X	X	X	X	eth1	X	eth2	X	X	X	X	X	X	X
s11	eth6	X	X	X	X	X	X	X	X	eth2	eth1	X	eth3	eth4	X	X	eth5	X	X
s12	eth6	eth8	X	X	eth9	X	eth10	X	X	eth1	X	eth2	X	X	eth3	eth4	eth5	X	eth7
s13	eth3	X	X	X	X	X	X	X	X	X	X	eth1	X	X	X	X	eth2	X	X
s14	eth3	X	X	X	X	X	X	X	X	X	X	X	eth1	X	X	X	eth2	X	X
s15	eth3	X	X	X	X	X	X	X	X	X	X	X	eth1	X	X	X	eth2	X	X

s16	eth7	X	X	X	X	X	X	X	X	X	X	eth1	eth5	eth2	eth3	eth4	X	eth6	X
s17	eth3	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	eth1	X	eth2
s18	eth3	X	X	X	X	X	X	X	X	X	X	X	eth1	X	X	X	X	eth2	X

s = Nodo OpenFlow.
eth = Puerto de conexión (tarjeta de red).

Se destaca igualmente la inclusión, dentro del controlador *OpenDayLight* de una Base de datos con motor MYSQL, que permite la administración de los datos. Para determinar la probabilidad de bloqueo se crea un nuevo campo "Bp" sobre las tablas "*OpenFlow:1…14*" de la base de datos "*OpenDayLight*data", como se muestra en la figura 3.24 con la siguiente cadena de conexión en lenguaje (PHP, *Personal Hypertext processor*) de la figura 3.25.

Node Connector Statistics for Node Id - openflow:6

Node Connector Id	Rx Pkts	Tx Pkts	Rx Bytes	Tx Bytes	Bp
openflow:6:1	185246	3351	53249974848	45794918959	0.02
openflow:6:2	17426	1920	27098215932	35236297315	0.05
openflow:6:3	15933	1670	56318920154	28823581419	0.03
openflow:6:LOCAL	8	1670	648	153640	1.00
openflow:6:4	15933	1670	56626839465	28783981950	0.06
openflow:6:5	54	186790	3780	35532240	1.00

Figura 3.24. Campo "Bp" sobre las tablas *OpenDayLight*.

```
<?php
$servername = "192.168.1.30";
$username = "root";
$database = "openDayLightdata";
$password = "mona1951nina";
//  cadena de conexion base de datos
$conn = new mysqli($servername, $username, $password);
alter table openflow:1 add Bp int(20) null;
alter table openflow:2 add Bp int(20) null;
alter table openflow:3 add Bp int(20) null;
alter table openflow:4 add Bp int(20) null;
alter table openflow:5 add Bp int(20) null;
alter table openflow:6 add Bp int(20) null;
alter table openflow:7 add Bp int(20) null;
alter table openflow:8 add Bp int(20) null;
alter table openflow:9 add Bp int(20) null;
alter table openflow:10 add Bp int(20) null;
alter table openflow:11 add Bp int(20) null;
alter table openflow:12 add Bp int(20) null;
alter table openflow:13 add Bp int(20) null;
alter table openflow:14 add Bp int(20) null;
?>
```

Figura 3.25. Cadena de conexión PHP "Bp".

Finalmente, en la tabla 3.7 se organiza el desarrollo de las pruebas tanto en las

topologías a utilizar como en los S*cripts* de configuración, enfocado para los dos casos de estudio.

Tabla 3.7. Configuración de pruebas casos de estudio.

Topologías	**Tráfico red física**	**Tráfico red entre Host**
Red Prueba TEST	TráficoTEST.sh	HostelefanteTEST.sh
		HostratonTEST.sh
		HostelefanteyratonTEST.sh
Red Prueba NSFNET	TráficoNSFNET.sh	HostelefanteNSFNET.sh
		HostratonNSFNET.sh
		HostelefanteyratonNSFNET.sh
Red Prueba IRIS10	TráficoIRIS10.sh	HostelefanteIRIS10.sh
		HostratonIRIS10.sh
		HostelefanteyratonIRIS10.sh

Se genera, sobre cada topología de red la inyección de tráfico sobre la red física por medio de scripts con flujo de trafico variado (raton y elefante), y sobre la red lógica se implantan 3 variaciones de tráfico entre *Hosts* con flujos separados (Ratón, Elefante y Ratón y Elefante), y así, obtener resultados más cercanos a la realidad.

Asi mismo tanto el flujo de trafico elefante como el flujo de trafico raton se destribuye sobre la red, de acuerdo a la distribución temporal de los usuarios de internet chino dado por el instituto de trafico Baidu, donde se establecen las proporciones de entre el 3% y 5% para flujo de trafico Elefante y el porcentaje restante para flujo de trafico raton. Por lo cual los intervalos se configuran para un total de 5 dias donde el flujo de tráfico raton se programa cada 2 segundos en 216.000 iteraciones y los flujos de tráfico elefante se establecen cada 60 segundos para 7.200 iteraciones.

A continuación se describen los dos casos de estudio objeto de este trabajo de investigación, los cuales hacen relevancia en la implentacion de una red SDN cognitiva y no cognitiva, con el propósito de análizar los diferentes resultados del parámetro de desempeño.

3.2.4.1. Caso de estudio 1

De acuerdo a la configuración del tráfico de flujo elefante y ratón sobre las topologías de red Prueba TEST, NSFNET e IRIS10 que previamente están configuradas como se muestra en la Figura 3.26, el primer caso de estudio consiste en analizar la probabilidad de bloqueo de los enlaces en la Red bajo arquitectura SDN sin método cognitivo en la función de enrutamiento, sobre cada uno de los espacios del escenario, por medio de inyección de flujo de tráfico, de acuerdo a los SHELL (TráficoTEST.sh, TráficoNSFNET.sh y TráficoIRIS10.sh). Donde posteriormente, se envía tráfico entre

los diferentes host (HostelefanteTEST.sh, HostratonTEST.sh, HostelefanteyratonTEST.sh, HostelefanteNSFNET.sh, HostratonNSFNET.sh, HostelefanteyratonNSFNET.sh, HostelefanteIRIS10.sh, HostratonIRIS10.sh y HostelefanteyratonIRIS10.sh),

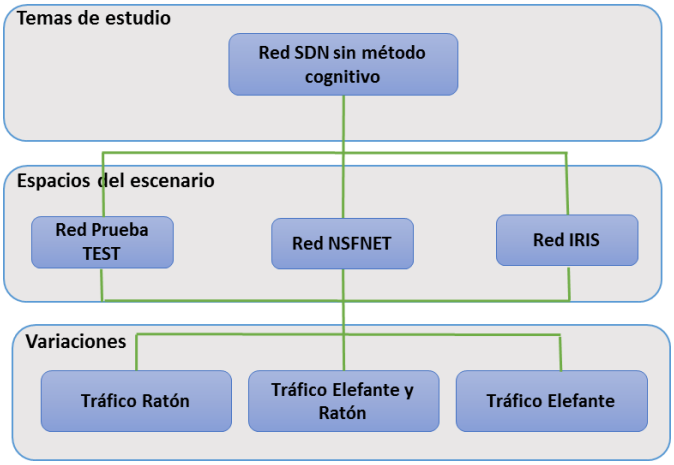


Figura 3.26. Caso de estudio 1.

Finalmente el análisis va enfocado en la probabilidad de bloqueo de los enlaces de los *Host*, y en la probabilidad de bloqueo de la red por medio de la media de los datos suministrados por las probabilidades de bloqueo de los enlaces, además se tomarán cada 12 horas durante 5 días muestras del comportamiento de la red, donde los datos suministrados serán importados a la tabla 3.8.

Tabla 3.8. Importación base de datos *OpenDayLightdata* caso de estudio1.

Protocol	port_connection	node_connector_id	ip_address	connection_route	rx_bytes	tx_bytes	bp	node_conector_id	ip_adress	rx_bytes	tx_bytes	bp

Los datos suministrados en esta tabla se encuentran alojados en la base de datos *OpenDayLight*data, la cual es exportada por los siguientes PHP (*consulta_bp_TEST.php, consulta_bp_NSFNET.php, consulta_bp_IRIS10.php*) anexo 16, 17 y 18 según la topología de red.

3.2.4.2. Caso de estudio 2

El segundo caso de estudio consiste en analizar la probabilidad de bloqueo, en los enlaces de la Red bajo arquitectura SDN, con el método cognitivo basado en CHRON en la función de enrutamiento ACO sobre los puertos lógicos importados por la función VTN, de la capa más elevada del modelo SDN, con inyección de flujo de tráfico, de acuerdo a los SHELL (*TráficoTEST.sh, TráficoNSFNET.sh y TráficoIRIS10.sh*). por

medio del envio de tráfico entre los diferentes host (HostelefanteTEST.sh, HostratonTEST.sh, HostelefanteyratonTEST.sh, HostelefanteNSFNET.sh, HostratonNSFNET.sh, HostelefanteyratonNSFNET.sh, HostelefanteIRIS10.sh, HostratonIRIS10.sh y HostelefanteyratonIRIS10.sh), de acuerdo a la configuración del tráfico de flujo elefante y ratón, sobre las topologías de red Prueba TEST, NSFNET e IRIS10, como se muestra en la Figura 3.27.

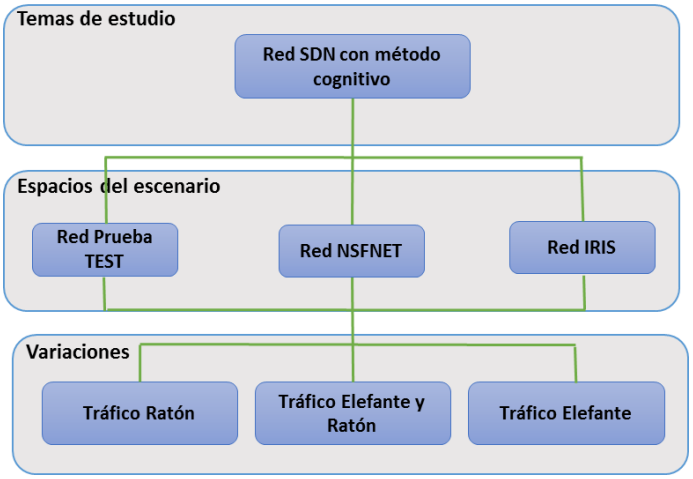


Figura 3.27. Caso de estudio 2.

Finalmente el análisis va enfocado en la probabilidad de bloqueo de los enlaces de los Host, y en la probabilidad de bloqueo de la red por medio de la media de los datos suministrados por las probabilidades de bloqueo de los enlaces, además se tomarán muestras cada 12 horas durante 5 días del comportamiento de la red por medio de la siguiente tabla 3.9.

Tabla 3.9. Importación base de datos *OpenDayLightdata* caso de estudio 2.

Protocol	port_connection	node_connector_id	ip_address	connection_route	rx_bytes	tx_bytes	bp	node_conector_id	ip_adress	rx_bytes	tx_bytes	bp

Los datos suministrados en esta tabla se encuentran alojados en la base de datos *OpenDayLight*data, la cual es exportada por el siguientes php (consulta_bp_TEST.php, consulta_bp_NSFNET.php, consulta_bp_IRIS10.php) anexo 16, 17 y 18 según la topología de red.

3.2.5. Fase 5: Caso de estudio 1 prueba de la red SDN sin método cognitivo

En este caso de estudio, se realiza la implementación de una red bajo arquitectura SDN con las topologías de red Prueba TEST, NSFNET e IRIS10, con inyección de flujo

de tráfico uniforme entre los puertos físicos, además del envió de tráfico entre *Host*, buscando obtener la probabilidad de bloqueo de los enlaces físicos y logicos y de la red.

Para obtener el algoritmo de enrutamiento por defecto del controlador *OpenDayLight* se procede a cargar la característica *feature:install odl-l2switch-switch* como se muestra en la figura 3.28.

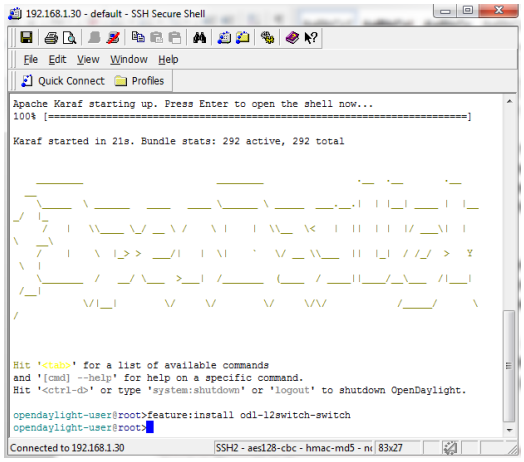


Figura 3.28. Instalación del algoritmo de enrutamiento Simple Forwarding.

El algoritmo de enrutamiento por defecto (*Simple Forwarding*) del controlador *OpenDayLight*, consiste en un rastreador de *Host*, donde los *Switch* obtendrán una regla de reenvio de paquetes de acuerdo a la dirección IP de destino.

Con el propósito de evaluar las funcionalidades así como de realizar pruebas de conectividad entre los nodos se inyecta tráfico a modo de ping entre el Host 1 y Host 7 como se muestra en la figura 3.29.

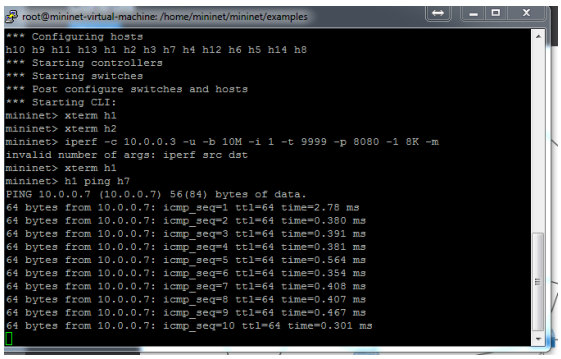


Figura 3.29. Conectividad entre nodos algoritmo *Simple Forwarding.*

Seguidamente desde la herramienta MININET se establece el comando *pingall* como se muestra en la figuras 3.30, 3.31 y 3.32 con el fin de verificar la conectividad entre *Host*, para ser exportados al aplicativo *OpenDayLight* y lograr la verificación del algoritmo de enrutamiento por defecto.

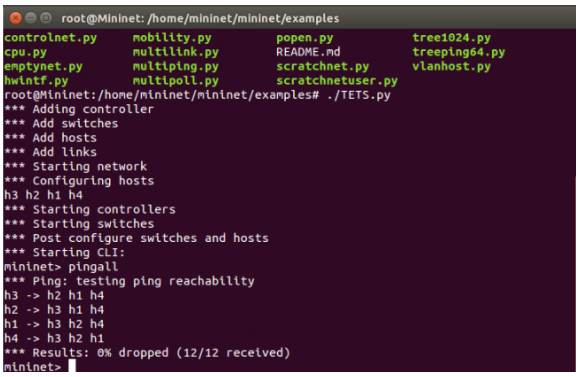


Figura 3.30. Comando pingall Prueba TEST.

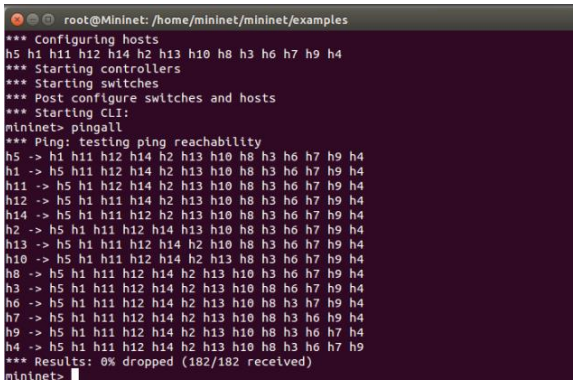


Figura 3.31. Comando pingall NSFNET.

Figura 3.32. Comando pingall IRIS10.

Para este caso puntual, se debe probar que la configuración de los Host en la herramienta MINIET haya sido importada con éxito por *OpenDayLight* como se muestra en las figuras 3.33, 3.34, 3.35.

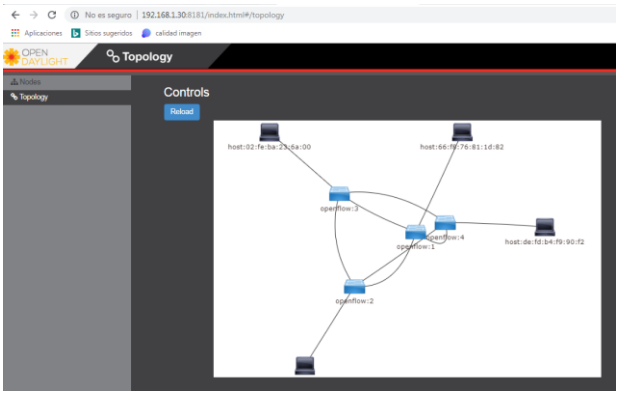


Figura 3.33. *Host* en *OpenDayLight* Prueba TEST

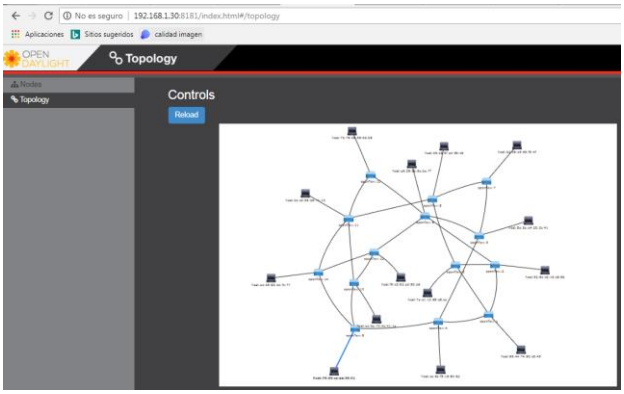


Figura 3.34. *Host* en *OpenDayLight* NSFNET

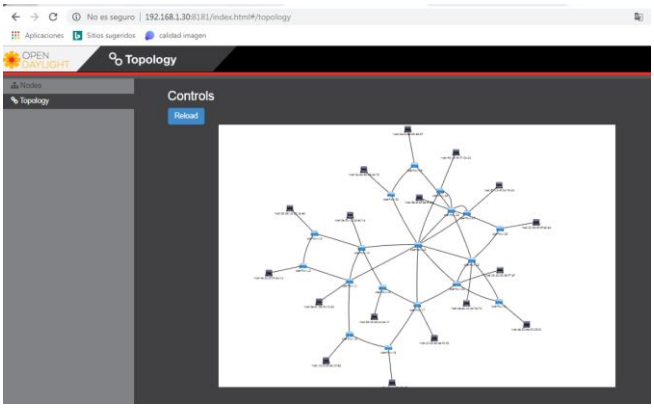


Figura 3.35. *Host* en *OpenDayLight* IRIS10

3.2.5.1. Resultados obtenidos desde la topología de red Prueba TEST

La preparación de la prueba en la topología de red Prueba TEST para realizar el análisis de resultados desde el parámetro de desempeño se determina de acuerdo a la siguiente lista de chequeo ver tabla 3.10.

Tabla 3.10. Lista de chequeo preparación de la red prueba TEST.

	Prueba TEST	**NSFNET**	**IRIS10**
Carga de Topología			
MININET	X		
OpenDayLight	X		
Virtual Tenant Network			
Método Cognitivo			
Sin	X		
Con			
Carga de tráfico, espacio de los escenarios			
TráficoPruebaTEST.sh	X		
TráficoNSFNET.sh			
TráficoIRIS10.sh			
Variaciones de Tráfico			
HostelefanteTEST.sh	X		
HostratonTEST.sh	X		
HostelefanteyratonTEST.sh	X		
HostelefanteNSFNET.sh			
HostratonNSFNET.sh			
HostelefanteyratonNSFNET.sh			
HostelefanteIRIS10.sh			
HostratonIRIS10.sh			
HostelefanteyratonIRIS10.sh			

Dentro de la lista de chequeo se carga la topología de red Prueba TEST tanto en la herramienta MININET como el aplicativo *OpenDayLight*. Seguidamente se hace la verificación del funcionamiento del algoritmo de enrutamiento por defecto de *OpenDayLight*, para así, cargar de tráfico al espacio del escenario, por medio del *Script*. TráficoTEST.sh uniformemente. Finalmente se cargan uno a uno las tres variaciones de tráfico HostratonTEST.sh, HostelefanteyratonTEST.sh y HostelefanteTEST.sh sobre la red Prueba TEST.

3.2.5.1.1. Variación 1: Tráfico ratón

La variación 1, consiste en inyectar tráfico (HostratonTEST.sh) ratón entre los diferentes *Host*, con intervalo de 2 segundos, durante 5 días tomando muestras cada 12 horas de acuerdo a la configuración realizada en el CRONTAB del servidor que aloja *OpenDayLight*. La probabilidad de bloqueo de los enlaces y la red es dada por medio del PHP, (consulta_bp_TEST.php), como se muestra en la tabla 3.11.

Tabla 3.11. Resultado probabilidad de bloqueo de la red Prueba TEST con tráfico ratón.

# Iteraciones	horas	Bp
21600	12	0.001108333
43200	24	0.002008333
64800	36	0.002825
86400	48	0.003466667
108000	60	0.004075
129600	72	0.004533333
151200	84	0.004758333
172800	96	0.004941667
194400	108	0.005058333
216000	120	0.005141667

En la Figura 3.36, se da a conocer el consolidado de las pruebas realizadas durante 5 días (120 Horas) de la probabilidad de bloqueo de la red Prueba TEST con tráfico Ratón en relación al numero de iteraciones dadas cada 12 horas.

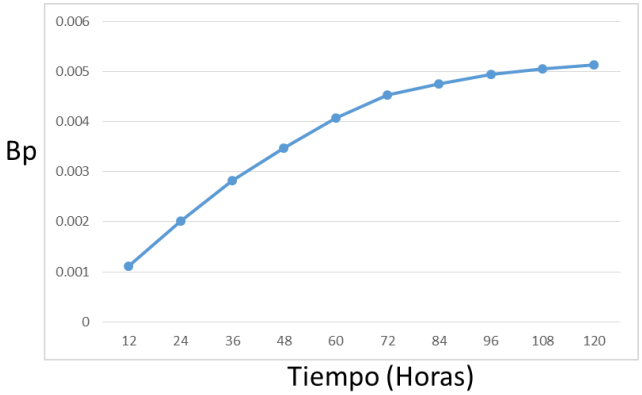


Figura 3.36. Inyeccion de tráfico ratón sobre la red Prueba TEST en función de la Bp.

3.2.5.1.2. Variación 2: Tráfico elefante y ratón.

La variación 2, consiste en inyectar tráfico (HostelefanteyratonTEST.sh) elefante y ratón entre los diferentes Host, con intervalo de 2 segundos, durante 5 días tomando muestras cada 12 horas del tráfico ratón y con intervalo de 60 segundos, durante 5 días tomando muestras cada 12 horas de tráfico elefante, de acuerdo a la configuración realizada en el CRONTAB del servidor que aloja *OpenDayLight.* La probabilidad de bloqueo de los enlaces y la red es dada por medio del PHP, (consulta_bp_TEST.php), como se muestra en la tabla 3.12.

Tabla 3.12. Resultado probabilidad de bloqueo de la red Prueba TEST con tráfico elefante y ratón.

horas	Bp
12	0.004675
24	0.006508333
36	0.007975
48	0.00925
60	0.010333333
72	0.011241667
84	0.012075
96	0.012691667
108	0.013158333
120	0.013383333

En la Figura 3.37 se da a conocer el consolidado de las pruebas realizadas durante 5 días (120 Horas) de la probabilidad de bloqueo de la red Prueba TEST con tráfico elefante y ratón en relación al numero de iteraciones dadas cada 12 horas.

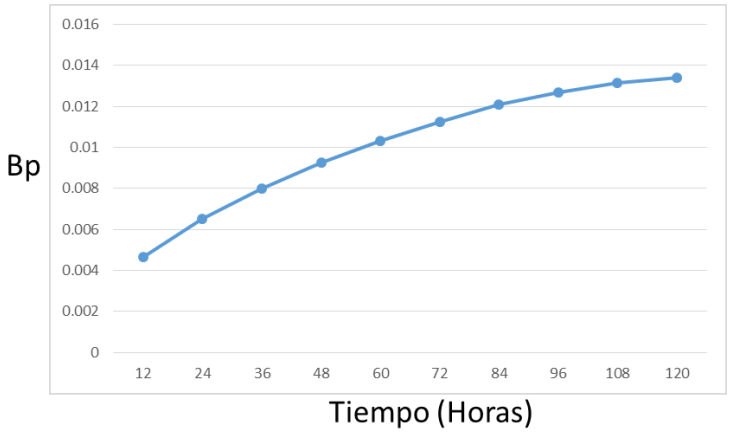


Figura 3.37. Inyeccion de tráfico elefante y ratón sobre la red Prueba TEST en función de la Bp.

3.2.5.1.3. Variación 3: Tráfico elefante.

La variación 3, consiste en inyectar tráfico (HostelefanteTEST.sh) ratón entre los diferentes Host, con intervalo de 60 segundos, durante 5 días tomando muestras cada 12 horas de acuerdo a la configuración realizada en el CRONTAB del servidor que aloja *OpenDayLight*. La probabilidad de bloqueo de los enlaces y la red es dada por medio del PHP, (consulta_bp_TEST.php), como se muestra en la tabla 3.13.

Tabla 3.13. Resultado probabilidad de bloqueo de la red Prueba TEST con tráfico elefante.

# Iteraciones	horas	Bp
720	12	0.009025
1440	24	0.013825
2160	36	0.017933333
2880	48	0.021408333
3600	60	0.02455
4320	72	0.027075
5040	84	0.02935
5760	96	0.030966667
6480	108	0.032241667
7200	120	0.032991667

En la Figura 3.38 se da a conocer el consolidado de las pruebas realizadas durante 5 días (120 Horas) de la probabilidad de bloqueo de la red Prueba TEST con tráfico Elefante en relación al numero de iteraciones dadas cada 12 horas.

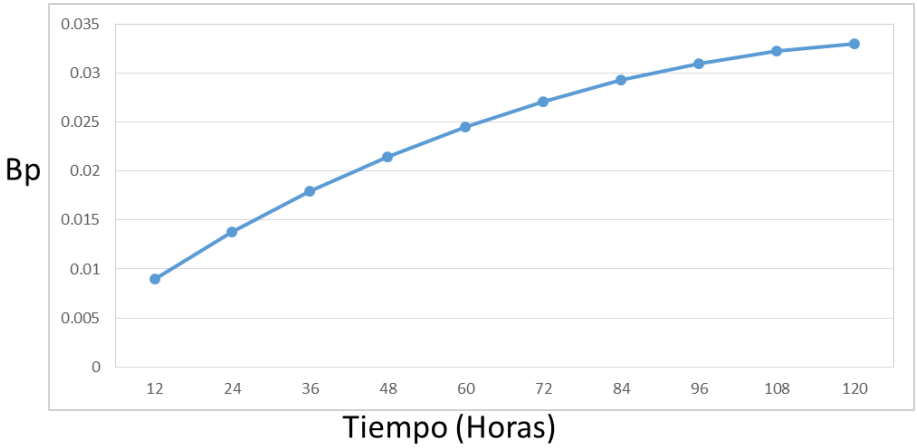


Figura 3.38. Inyeccion de tráfico elefante sobre la red Prueba TEST en función de la Bp.

3.2.5.2. Resultados obtenidos desde la topología de red NSFNET.

La preparación de la prueba en la topología de red NSFNET para realizar el análisis de resultados desde el parámetro de desempeño se determina de acuerdo a la siguiente lista de chequeo.

Tabla 3.14. Lista de chequeo preparación de la red NSFNET.

	Prueba TEST	NSFNET	IRIS10
Carga de Topología			
MININET		X	
OpenDayLight		X	
Virtual Tenant Network			
Método Cognitivo			
Sin		X	
Con			
Carga de tráfico, espacio de los escenarios			
TráficoPruebaTEST.sh			
TráficoNSFNET.sh		X	
TráficoIRIS10.sh			
Variaciones de Tráfico			
HostelefanteTEST.sh			
HostratonTEST.sh			
HostelefanteyratonTEST.sh			
HostelefanteNSFNET.sh		X	
HostratonNSFNET.sh		X	
HostelefanteyratonNSFNET.sh		X	
HostelefanteIRIS10.sh			
HostratonIRIS10.sh			
HostelefanteyratonIRIS10.sh			

Dentro de la lista de chequeo se carga la topología de red NSFNET tanto en la herramienta MININET como el aplicativo *OpenDayLight*. Seguidamente se hace la verificación del funcionamiento del algoritmo de enrutamiento por defecto de *OpenDayLight*, para así, cargar de tráfico el espacio del escenario por medio del *Script*. TráficoNSFNET.sh uniformemente. Finalmente se cargan uno a uno las tres variaciones de tráfico HostratonNSFNET.sh, HostelefanteyratonNSFNET.sh y HostelefanteNSFNET.sh sobre la red NSFNET.

3.2.5.2.1. Variación 1: Tráfico ratón.

La variación 1, consiste en inyectar tráfico (HostratonNSFNET.sh) ratón entre los diferentes Host, con intervalo de 2 segundos, durante 5 días tomando muestras cada 12 horas de acuerdo a la configuración realizada en el CRONTAB del servidor que aloja *OpenDayLight.* La probabilidad de bloqueo de los enlaces y la red es dada por medio del PHP, (consulta_bp_NSFNET.php), como se muestra en la tabla 3.15.

Tabla 3.15. Resultado probabilidad de bloqueo de la red NSFNET con Tráfico ratón.

# Iteraciones	horas	Bp
21600	12	0.002109
43200	24	0.002995
64800	36	0.003707
86400	48	0.004402
108000	60	0.004994
129600	72	0.005507
151200	84	0.005963
172800	96	0.006365
194400	108	0.006597
216000	120	0.006714

En la Figura 3.39 se da a conocer el consolidado de las pruebas realizadas durante 5 días (120 Horas) de la probabilidad de bloqueo de la red NSFNET con Tráfico Ratón en relación al numero de iteraciones dadas cada 12 horas.

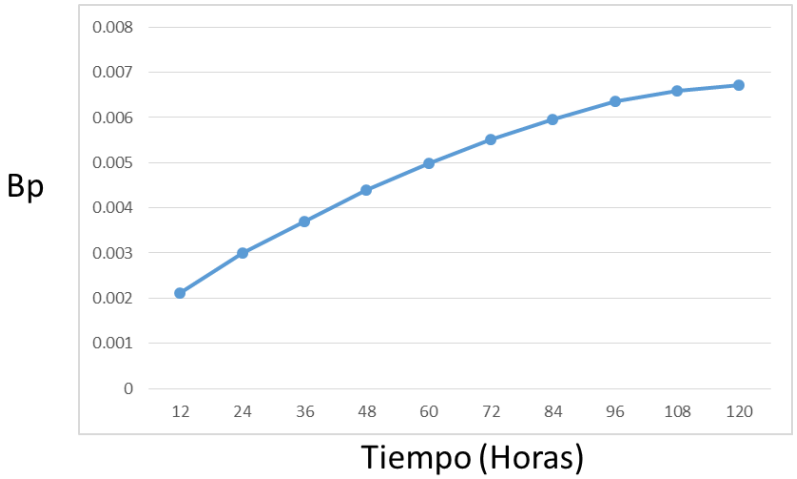


Figura 3.39. Inyeccion de tráfico ratón sobre la red NSFNET en función de la Bp.

3.2.5.2.2. Variación 2: Tráfico elefante y ratón.

La variación 2, consiste en inyectar tráfico (HostelefanteyratonNSFNET.sh) elefante y ratón entre los diferentes Host, con intervalo de 2 segundos, durante 5 días tomando muestras cada 12 horas del tráfico ratón y con intervalo de 60 segundos, durante 5 días tomando muestras cada 12 horas de acuerdo a la configuración realizada en el CRONTAB del servidor que aloja *OpenDayLight*. La probabilidad de bloqueo de los enlaces y la red es dada por medio del PHP, (consulta_bp_NSFNET.php), como se muestra en la tabla 3.16.

Tabla 3.16. Resultado probabilidad de bloqueo de la red NSFNET con Tráfico elefante y ratón.

horas	Bp
12	0.006714
24	0.008501
36	0.010141
48	0.011426
60	0.01259
72	0.013688
84	0.014507
96	0.015131
108	0.015637
120	0.015878

En la figura 3.40. se da a conocer el consolidado de las pruebas realizadas durante 5 días (120 Horas) de la probabilidad de bloqueo de la red NSFNET con tráfico elefante y ratón en relación al numero de iteraciones dadas cada 12 horas.

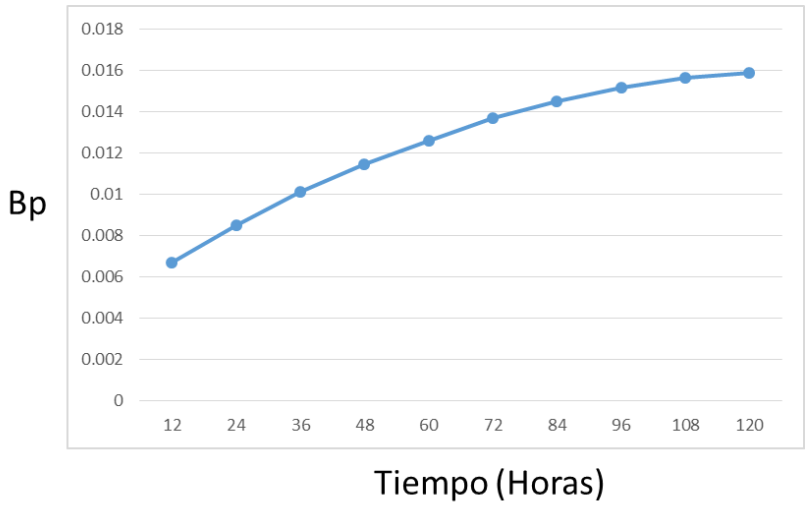


Figura 3.40. . Inyeccion de tráfico elefante y ratón sobre red NSFNET en función de la Bp.

3.2.5.2.3. Variación 3: Tráfico elefante.

La variación 3, consiste en inyectar tráfico (HostelefanteNSFNET.sh) elefante entre los diferentes Host, con intervalo de 60 segundos, durante 5 días tomando muestras cada 12 horas de acuerdo a la configuración realizada en el CRONTAB del servidor que aloja *OpenDayLight*, de la probabilidad de bloqueo de los enlaces y la red por medio del PHP, (consulta_bp_NSFNET.php), como se muestra en la tabla 3.17.

Tabla 3.17. Resultado probabilidad de bloqueo de la red NSFNET con Tráfico elefante.

# Iteraciones	horas	Bp
720	12	0.025113
1440	24	0.030401
2160	36	0.035092
2880	48	0.039037
3600	60	0.042454
4320	72	0.045478
5040	84	0.048045
5760	96	0.049918
6480	108	0.051571
7200	120	0.052592

En la figura 3.41 se da a conocer el consolidado de las pruebas realizadas durante 5 días (120 Horas) de la probabilidad de bloqueo de la red NSFNET con Tráfico Elefante en relación al número de iteraciones dadas cada 12 horas.

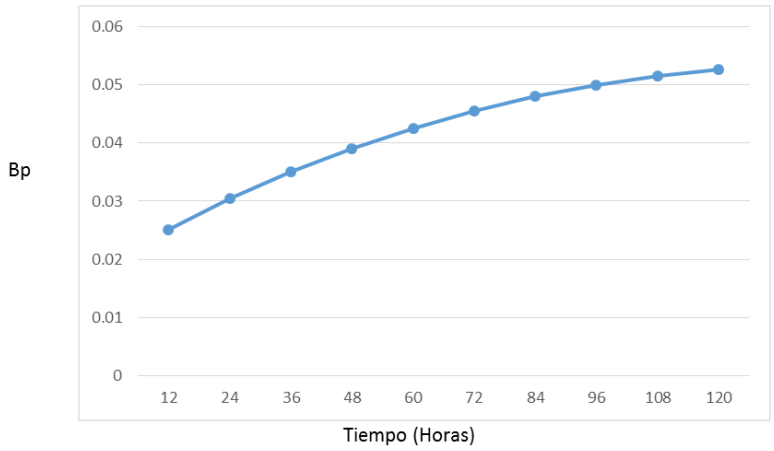


Figura 3.41. . Inyeccion de tráfico elefante sobre red NSFNET en función de la Bp.

3.2.5.3. Resultados obtenidos desde la topología de red IRIS10.

La preparación de la prueba en la topología de red IRIS10 para realizar el análisis de resultados desde el parámetro de desempeño se determina de acuerdo a la siguiente lista de chequeo.

Tabla 3.18. Lista de chequeo preparación de la red IRIS10.

	Prueba TEST	NSFNET	IRIS10
Carga de Topología			
MININET			X
OpenDayLight			X
Virtual Tenant Network			
Método Cognitivo			
Sin			X
Con			
Carga de tráfico, espacio de los escenarios			
TráficoPruebaTEST.sh			
TráficoNSFNET.sh			
TráficoIRIS10.sh			X
Variaciones de tráfico			
HostelefanteTEST.sh			
HostratonTEST.sh			
HostelefanteyratonTEST.sh			
HostelefanteNSFNET.sh			
HostratonNSFNET.sh			
HostelefanteyratonNSFNET.sh			
HostelefanteIRIS10.sh			X
HostratonIRIS10.sh			X
HostelefanteyratonIRIS10.sh			X

Dentro de la lista de chequeo se carga la topología de red IRIS10 tanto en la herramienta MININET como el aplicativo *OpenDayLight*. Seguidamente se hace la verificación del funcionamiento del algoritmo de enrutamiento por defecto de *OpenDayLight*, para así, cargar de tráfico el espacio del escenario por medio del SHELL. ***TráficoIRIS10.sh*** uniformemente. Finalmente se cargan uno a uno las tres variaciones de tráfico ***HostelefanteIRIS10.sh, HostratonIRIS10.sh, HostelefanteyratonIRIS10.sh*** sobre la red IRIS10.

3.2.5.3.1. Variación 1: Tráfico ratón.

La variación 1, consiste en inyectar tráfico (HostratonIRIS10.sh) ratón entre los diferentes Host, con intervalo de 2 segundos, durante 5 días tomando muestras cada 12 horas de acuerdo a la configuración realizada en el CRONTAB del servidor que aloja OpenDayLight, de la probabilidad de bloqueo de los enlaces y la red por medio del PHP, (consulta_bp_ IRIS10.php), como se muestra en la tabla 3.19.

Tabla 3.19. Resultado probabilidad de bloqueo de la red IRIS10 con Tráfico ratón.

# Iteraciones	horas	Bp
21600	12	0.003230592
43200	24	0.004104276
64800	36	0.004839145
86400	48	0.005524342
108000	60	0.006036842
129600	72	0.006426316
151200	84	0.006754276
172800	96	0.007001645
194400	108	0.007113816
216000	120	0.007187829

En la Figura 3.42 se da a conocer el consolidado de las pruebas realizadas durante 5 días (120 Horas) de la probabilidad de bloqueo de la red IRIS10 con Tráfico Ratón en relación al número de iteraciones dadas cada 12 horas.

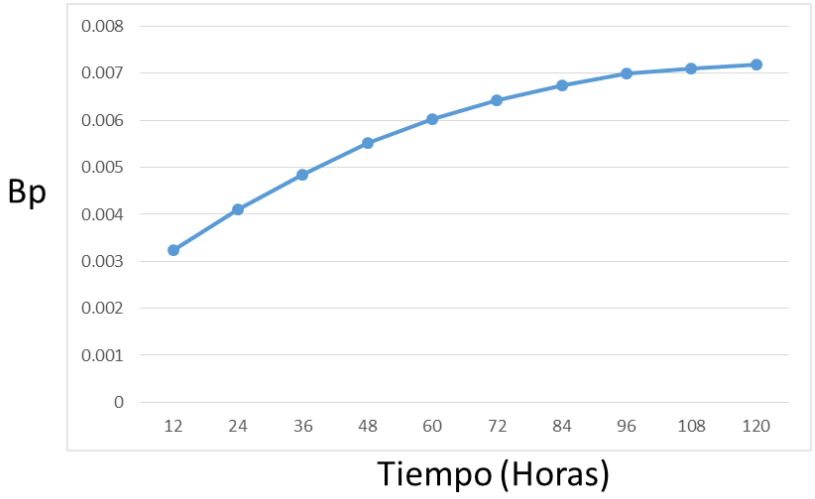


Figura 3.42. Inyeccion de tráfico ratón sobre red IRIS10 en función de la Bp.

3.2.5.3.2. Variación 2: Tráfico elefante y ratón.

La variación 2, consiste en inyectar tráfico (HostelefanteyratonIRIS10.sh) elefante y ratón entre los diferentes *Host*, con intervalo de 2 segundos, durante 5 días tomando muestras cada 12 horas del Tráfico ratón y con intervalo de 60 segundos, durante 5 días tomando muestras cada 12 horas de acuerdo a la configuración realizada en el CRONTAB del servidor que aloja *OpenDayLight*, de la probabilidad de bloqueo de los enlaces y la red por medio del PHP, (consulta_bp_IRIS10.php), como se muestra en la tabla 3.20.

Tabla 3.20. Resultado probabilidad de bloqueo de la red IRIS10 con Tráfico elefante y ratón.

horas	Bp
12	0.008589145
24	0.010408882
36	0.0119625
48	0.013308553
60	0.014550329
72	0.015486513
84	0.016325329
96	0.016974342
108	0.017407566
120	0.017662829

En la figura 3.43 se da a conocer el consolidado de las pruebas realizadas durante 5 días (120 Horas) de la probabilidad de bloqueo de la red IRIS10 con Tráfico elefante y ratón en relación al número de iteraciones dadas cada 12 horas.

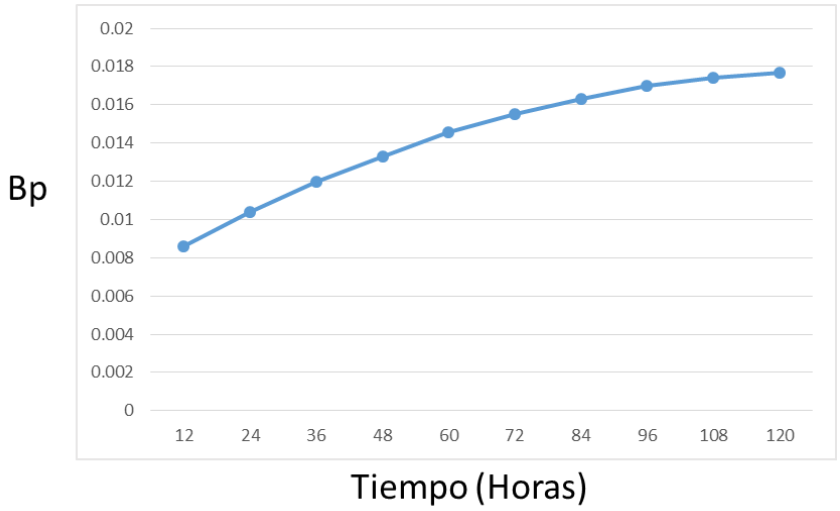


Figura 3.43. Inyeccion de tráfico elefante y ratón sobre red IRIS10 en función de la Bp.

3.2.5.3.3. Variación 3: Tráfico elefante.

La variación 3, consiste en inyectar tráfico (HostelefanteIRIS10.sh) elefante entre los diferentes Host, con intervalo de 60 segundos, durante 5 días tomando muestras cada 12 horas de acuerdo a la configuración realizada en el CRONTAB del servidor que aloja OpenDayLight, de la probabilidad de bloqueo de los enlaces y la red por medio del PHP, (consulta_bp_IRIS10.php), como se muestra en la tabla 3.21.

Tabla 3.21. Resultado probabilidad de bloqueo de la red IRIS10 con Tráfico elefante.

# Iteraciones	horas	Bp
720	12	0.0303
1440	24	0.0363
2160	36	0.0413
2880	48	0.0457
3600	60	0.0499
4320	72	0.0528
5040	84	0.0554
5760	96	0.0576
6480	108	0.0594
7200	120	0.0606

En la figura 3.44 se da a conocer el consolidado de las pruebas realizadas durante 5 días (120 Horas) de la probabilidad de bloqueo de la red IRIS10 con Tráfico Elefante en relación al número de iteraciones dadas cada 12 horas.

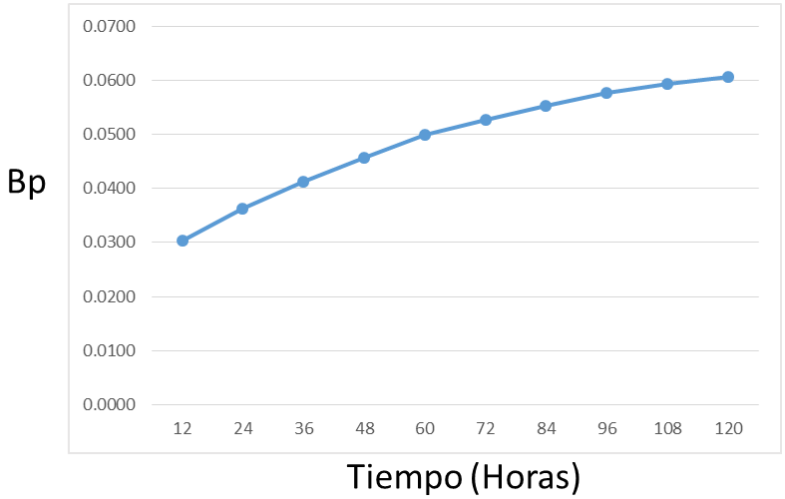


Figura 3.44. Inyeccion de tráfico elefante sobre red IRIS10 en función de la Bp.

Tras la obtención con éxito de los resultados en el caso de estudio 1 prueba de la red SDN sin método cognitivo, se espera que los resutados que se logren obtener en el caso de estudio 2 prueba de la red con método cognitivo sean convergentes a un numero acercándose a ser optimos en concencuencia a la utilización de la metahursiitca ACO, con valores de probabildiad de bloqueo de red menores o iguales al caso de estudio 1, debido a que el algoritmo de enrutamiento por defecto de OpenDayLight es un algoritmo configurado implantado y probado por un gran numero de redes con diferentes condiciones alrededor del mundo.

De no ser asi y los resultados obtenidos en el caso de estudio 2 no son los esperados se podría concluir que el método cognitivo no es competitivo para resolver el problema de enrutamiento en redes bajo arquitectura SDN.

3.2.6. Fase 6: Caso de estudio 2 prueba de la red SDN con método cognitivo.

En el segundo caso de estudio, se realiza la implementación del método cognitivo CHRON para las topologías de red Prueba TEST, NSFNET e IRIS10 bajo arquitectura SDN. La función de enrutamiento para el caso de estudio 2 está dada por la metaheuristica ACO aplicada a los puertos lógicos importados por la función VTN, de la capa más elevada del modelo SDN. La inyección de flujo de tráfico es la misma utilizada en el caso de estudio 1 y está dada, de acuerdo a los SHELL (TráficoTEST.sh, TráficoNSFNET.sh y TráficoIRIS10.sh).por medio del envió de tráfico entre los diferentes Host (HostelefanteTEST.sh, HostratonTEST.sh, HostelefanteyratonTEST.sh, HostelefanteNSFNET.sh, HostratonNSFNET.sh, HostelefanteyratonNSFNET.sh, HostelefanteIRIS10.sh, HostratonIRIS10.sh y HostelefanteyratonIRIS10.sh), de acuerdo a la configuración del tráfico de flujo elefante y ratón.

Para la implementación del método cognitivo basado en CHRON, se hace necesario utilizar el servidor llamado ServVTN según la Tabla 3.2, el cual trabaja sobre CDS en la capa de aplicación del modelo SDN y será el encargado del enrutamiento de las topologías utilizadas por medio de la metaheuristica ACO, así como del almacenamiento de las rutas de destino de acuerdo a la probabilidad de bloqueo, para la toma de decisiones.

El inicio de la implementación del método cognitivo basado en CHRON consiste en cargar las topologías de red en la herramienta MININET como se muestra en las Figuras 3.10, 3.11 y 3.12 adicionalmente desde *OpenDayLight* se verifica que las topologías de red Prueba TEST, NSFNET e IRIS10 hayan sido importadas de manera exitosa como se muestra en las Figuras 3.13, 3.14 y 3.15. Seguidamente se establece la conexión entre el controlador *OpenDayLight* y el API REST VTN por medio de la instalación de la característica "*feacture:install odl-vtn-manager-rest*" en el controlador *OpenDayLight*, además es indispensable la no utilización de la característica de enrutamiento que trae por defecto *OpenDayLight* "*feacture:odl-l2Switch-Switch*" con el propósito de no alterar el reenvió de paquetes que se administrara desde CDS. Por último con los siguientes comandos se genera la conexión entre API REST de VTN y el controlador *OpenDayLight* como se muestra en la Figura 3.45.

```
Create Controller
// curl -v --user admin:adminpass -H 'content-type: application/json' -X POST -d '{"vtn" : {"vtn_name":"vtn1","description":"test VTN" }}'
http://192.168.1.33:8083/vtn-webapi/vtns.json

Create VTN
// curl -v --user admin:adminpass -H 'content-type: application/json' -X POST -d '{"controller": {"controller_id": "controllerone",
["ipaddr":"192.168.1.30", "type": "odc", "version": "1.0", "auditstatus":"enable"}}' http://192.168.1.33:8083/vtn-webapi/controllers.json
```

Figura 3.45. Comandos de conexion VTN y *OpenDayLight.*

Además para poder acceder a las topologías de red implementadas en MININET así como también la identificación del estado de conexión entre el CDS y CMS se envía el siguiente comando Figura 3.46.

```
.//curl --user admin:adminpass -H 'content-type: application/json' -X GET http://192.168.1.33:8083/vtn-webapi/controllers/detail

//{"controllers":[{"auditstatus":"enable","operstatus":"up","controller_id":"controllerone","actual_version":"1.0.0.0","ipaddr":
"192.168.1.30","type":"odc","version":"1.0"}]}[root@localHost bin]#.
```

Figura 3.46. Comando estado de conexión **UP**.

De acuerdo al resultado del anterior comando se logra demostrar que el CDS es capaz de acceder a la topología de red implementada en la herramienta MININET, debido a la comunicación entre SAES y CMS del método cognitivo.

Asimismo para la importación de los puertos lógicos asignados de la red física de las topologías de red Prueba TEST, NSFNET e IRIS10, implementadas en la herramienta MININET a la capa más elevada del método cognitivo se establece el siguiente comando (Figura 3.47) y los resultados se dan en la tabla 3.22.

```
//curl -v --user admin:adminpass -H 'content-type: application/json' -X GET http://192.168.1.33:8083/vtn-webapi/controllers/controllerone
/domains/\(DEFAULT\)/logical_ports.json
```

Figura 3.47. Comando puertos lógicos desde VTN.

Tabla 3.22 Puertos lógicos topologias de red desde VTN.

Puertos lógicos red Prueba TEST

{"logical_ports":[{"logical_port_id":"PP-OF:openflow:1-s1-eth1"},{"logical_port_id":"PP-OF:openflow:1-s1-eth2"},{"logical_port_id":"PP-OF:openflow:1-s1-eth3"},{"logical_port_id":"PP-OF:openflow:1-s1-eth4"},{"logical_port_id":"PP-OF:openflow:2-s2-eth1"},{"logical_port_id":"PP-OF:openflow:2-s2-eth2"},{"logical_port_id":"PP-OF:openflow:2-s2-eth3"},{"logical_port_id":"PP-OF:openflow:2-s2-eth4"},{"logical_port_id":"PP-OF:openflow:3-s3-eth1"},{"logical_port_id":"PP-OF:openflow:3-s3-eth2"},{"logical_port_id":"PP-OF:openflow:3-s3-eth3"},{"logical_port_id":"PP-OF:openflow:3-s3-eth4"},{"logical_port_id":"PP-OF:openflow:4-s4-eth1"},{"logical_port_id":"PP-OF:openflow:4-s4-eth2"},{"logical_port_id":"PP-OF:openflow:4-s4-eth3"},{"logical_port_id":"PP-OF:openflow:4-s4-eth4"}]}[root@localhost bin]#

Puertos lógicos red NSFNET

//{"logical_ports":[{"logical_port_id":"PP-OF:openflow:1-s1-eth1"},{"logical_port_id":"PP-OF:openflow:1-s1-eth2"},{"logical_port_id":"PP-OF:openflow:1-s1-eth3"},{"logical_port_id":"PP-OF:openflow:1-s1-eth4"},{"logical_port_id":"PP-OF:openflow:10-s10-eth1"},{"logical_port_id":"PP-OF:openflow:10-s10-eth2"},{"logical_port_id":"PP-OF:openflow:10-s10-eth3"},{"logical_port_id":"PP-OF:openflow:11-s11-eth1"},{"logical_port_id":"PP-OF:openflow:11-s11-eth2"},{"logical_port_id":"PP-OF:openflow:11-s11-eth3"},{"logical_port_id":"PP-OF:openflow:11-s11-eth4"},{"logical_port_id":"PP-OF:openflow:11-s11-eth5"},{"logical_port_id":"PP-OF:openflow:12-s12-eth1"},{"logical_port_id":"PP-OF:openflow:12-s12-eth2"},{"logical_port_id":"PP-OF:openflow:12-s12-eth3"},{"logical_port_id":"PP-OF:openflow:12-s12-eth4"},{"logical_port_id":"PP-OF:openflow:13-s13-eth1"},{"logical_port_id":"PP-OF:openflow:13-s13-eth2"},{"logical_port_id":"PP-OF:openflow:13-s13-eth3"},{"logical_port_id":"PP-OF:openflow:13-s13-eth4"},{"logical_port_id":"PP-OF:openflow:14-s14-eth1"},{"logical_port_id":"PP-OF:openflow:14-s14-eth2"},{"logical_port_id":"PP-OF:openflow:14-s14-eth3"},{"logical_port_id":"PP-OF:openflow:14-s14-eth4"},{"logical_port_id":"PP-OF:openflow:2-s2-eth1"},{"logical_port_id":"PP-OF:openflow:2-s2-eth2"},{"logical_port_id":"PP-OF:openflow:2-s2-eth3"},{"logical_port_id":"PP-OF:openflow:2-s2-eth4"},{"logical_port_id":"PP-OF:openflow:3-s3-eth1"},{"logical_port_id":"PP-OF:openflow:3-s3-eth2"},{"logical_port_id":"PP-OF:openflow:3-s3-eth3"},{"logical_port_id":"PP-OF:openflow:3-s3-eth4"},{"logical_port_id":"PP-OF:openflow:4-s4-eth1"},{"logical_port_id":"PP-OF:openflow:4-s4-eth2"},{"logical_port_id":"PP-OF:openflow:4-s4-eth3"},{"logical_port_id":"PP-OF:openflow:4-s4-eth4"},{"logical_port_id":"PP-OF:openflow:5-s5-eth1"},{"logical_port_id":"PP-OF:openflow:5-s5-eth2"},{"logical_port_id":"PP-OF:openflow:5-s5-eth3"},{"logical_port_id":"PP-OF:openflow:5-s5-eth4"},{"logical_port_id":"PP-OF:openflow:6-s6-eth1"},{"logical_port_id":"PP-OF:openflow:6-s6-eth2"},{"logical_port_i* Connection #0 to Host 192.168.1.33 left intact
d":"PP-OF:openflow:6-s6-eth3"},{"logical_port_id":"PP-OF:openflow:6-s6-eth4"},{"logical_port_id":"PP-OF:openflow:6-s6-eth5"},{"logical_port_id":"PP-OF:openflow:7-s7-eth1"},{"logical_port_id":"PP-OF:openflow:7-s7-eth2"},{"logical_port_id":"PP-OF:openflow:7-s7-eth3"},{"logical_port_id":"PP-OF:openflow:8-s8-eth1"},{"logical_port_id":"PP-OF:openflow:8-s8-eth2"},{"logical_port_id":"PP-OF:openflow:8-s8-eth3"},{"logical_port_id":"PP-OF:openflow:8-s8-eth4"},{"logical_port_id":"PP-OF:openflow:9-s9-eth1"},{"logical_port_id":"PP-OF:openflow:9-s9-eth2"},{"logical_port_id":"PP-OF:openflow:9-s9-eth3"},{"logical_port_id":"PP-OF:openflow:9-s9-eth4"}]}[root@localHost bin]#

Puertos lógicos red IRIS10

{"logical_ports":[{"logical_port_id":"PP-OF:openflow:1-s1-eth1"},{"logical_port_id":"PP-OF:openflow:1-s1-eth2"},{"logical_port_id":"PP-OF:openflow:1-s1-eth3"},{"logical_port_id":"PP-OF:openflow:1-s1-eth4"},{"logical_port_id":"PP-OF:openflow:1-s1-eth5"},{"logical_port_id":"PP-OF:openflow:10-s10-eth1"},{"logical_port_id":"PP-OF:openflow:10-s10-eth2"},{"logical_port_id":"PP-OF:openflow:10-s10-eth3"},{"logical_port_id":"PP-OF:openflow:11-s11-eth1"},{"logical_port_id":"PP-OF:openflow:11-s11-eth2"},{"logical_port_id":"PP-OF:openflow:11-s11-eth3"},{"logical_port_id":"PP-OF:openflow:11-s11-eth4"},{"logical_port_id":"PP-OF:openflow:11-s11-eth5"},{"logical_port_id":"PP-OF:openflow:11-s11-eth6"},{"logical_port_id":"PP-OF:openflow:12-s12-eth1"},{"logical_port_id":"PP-OF:openflow:12-s12-eth10"},{"logical_port_id":"PP-OF:openflow:12-s12-eth2"},{"logical_port_id":"PP-OF:openflow:12-s12-eth3"},{"logical_port_id":"PP-OF:openflow:12-s12-eth4"},{"logical_port_id":"PP-OF:openflow:12-s12-eth5"},{"logical_port_id":"PP-OF:openflow:12-s12-eth6"},{"logical_port_id":"PP-OF:openflow:12-s12-eth7"},{"logical_port_id":"PP-OF:openflow:12-s12-eth8"},{"logical_port_id":"PP-OF:openflow:12-s12-eth9"},{"logical_port_id":"PP-OF:openflow:13-s13-eth1"},{"logical_port_id":"PP-OF:openflow:13-s13-eth2"},{"logical_port_id":"PP-OF:openflow:13-s13-eth3"},{"logical_port_id":"PP-OF:openflow:14-s14-eth1"},{"logical_port_id":"PP-OF:openflow:14-s14-eth2"},{"logical_port_id":"PP-OF:openflow:14-s14-eth3"},{"logical_port_id":"PP-OF:openflow:15-s15-eth1"},{"logical_port_id":"PP-OF:openflow:15-s15-eth2"},{"logical_port_id":"PP-OF:openflow:15-s15-eth3"},{"logical_port_id":"PP-OF:openflow:16-s16-eth1"},{"logical_port_id":"PP-OF:openflow:16-s16-eth2"},{"logical_port_id":"PP-OF:openflow:16-s16-eth3"},{"logical_port_id":"PP-OF:openflow:16-s16-eth4"},{"logical_port_id":"PP-OF:openflow:16-s16-eth5"},{"logical_port_id":"PP-OF:openflow:16-s16-eth6"},{"logical_port_id":"PP-OF:openflow:16-s16-eth7"},{"logical_port_id":"PP-OF:openflow:17-s17-eth1"},{"logical_port_id":"PP-OF:openflow:17-s17-eth2"},{"logical_port_id":"PP-OF:openflow:17-s17-eth3"},{"logical_port_id":"PP-OF:openflow:18-s18-eth1"},{"logical_port_id":"PP-OF:openflow:18-s18-eth2"},{"logical_port_id":"PP-OF:openflow:18-s18-eth3"},{"logical_port_id":"PP-OF:openflow:2-s2-eth1"},{"logical_port_id":"PP-OF:openflow:2-s2-eth2"},{"logical_port_id":"PP-OF:openflow:2-s2-eth3"},{"logical_port_id":"PP-OF:openflow:3-s3-eth1"},{"logical_port_id":"PP-OF:openflow:3-s3-eth2"},{"logical_port_id":"PP-OF:openflow:3-s3-eth3"},{"logical_port_id":"PP-OF:openflow:4-s4-eth1"},{"logical_port_id":"PP-OF:openflow:4-s4-eth2"},{"logical_port_id":"PP-OF:openflow:4-s4-eth3"},{"logical_port_id":"PP-OF:openflow:4-s4-eth4"},{"logical_port_id":"PP-OF:openflow:4-s4-eth5"},{"logical_port_id":"PP-OF:openflow:5-s5-eth1"},{"logical_port_id":"PP-OF:openflow:5-s5-eth2"},{"logical_port_id":"PP-OF:openflow:5-s5-eth3"},{"logical_port_id":"PP-OF:openflow:6-s6-eth1"},{"logical_port_id":"PP-OF:openflow:6-s6-eth2"},{"logical_port_id":"PP-OF:openflow:6-s6-eth3"},{"logical_* Connection #0 to host 192.168.1.33 left intact
port_id":"PP-OF:openflow:6-s6-eth4"},{"logical_port_id":"PP-OF:openflow:6-s6-eth5"},{"logical_port_id":"PP-OF:openflow:7-s7-eth1"},{"logical_port_id":"PP-OF:openflow:7-s7-eth2"},{"logical_port_id":"PP-OF:openflow:7-s7-eth3"},{"logical_port_id":"PP-OF:openflow:8-s8-eth1"},{"logical_port_id":"PP-OF:openflow:8-s8-eth2"},{"logical_port_id":"PP-OF:openflow:8-s8-eth3"},{"logical_port_id":"PP-OF:openflow:9-s9-eth1"},{"logical_port_id":"PP-OF:openflow:9-s9-eth2"},{"logical_port_id":"PP-OF:openflow:9-s9-eth3"},{"logical_port_id":"PP-OF:openflow:9-s9-eth4"},{"logical_port_id":"PP-OF:openflow:9-s9-eth5"}]}[root@localhost bin]#

Después de haber obtenido los puertos lógicos de las diferentes topologías de red, se da inicio a la implantación del algoritmo de enrutamiento bajo la metaheuristica ACO.

ALGORIMO DE ENRUTAMIENTO ACO

La aplicación del algoritmo de enrutamiento ACO, sobre los puertos lógicos, da como resultado la obtención de las mejores rutas para él envio de flujo de paquetes, es así que dentro del CDS se hace necesario el almacenamiento de los resultados obtenidos anteriormente y así mejorar el parámetro de desempeño.

Para tal caso, se realiza la instalación del motor de base de datos MYSQL en el servidor ServVTN y se crea la base de datos "*datos_VTN*", sobre el cual se almacena información importada desde *OpenDayLight* además del almacenamiento de los resultados obtenidos por el algoritmo de enrutamiento bajo la metaheuristica ACO.

Con el fin de dar inicio con la implantación del algoritmo de enrutamiento ACO, expresamos la topología de red mediante un grafico $G = (v, e)$, donde v, es el conjunto de nodos y e es el conjunto de enlaces.

Las decisiones de enrutamiento se basan en el algoritmo de optimización por colonia de hormigas cuyo código se muestra en el algoritmo 1. Cada hormiga utiliza la Pij para seleccionar el siguiente nodo de acuerdo con la ecuación 2.

$$Pij = \frac{\tau ij^{\alpha} * \mu ij^{\beta}}{\sum \tau ij^{\alpha} * \mu ij^{\beta}} \qquad (2)$$

Donde τij representa la concentración de feromonas del enlace (ij), μij es la conveniencia y $\alpha\ y\ \beta$ son la influencia relativa de todos los factores.

Las hormigas a regreso actualizan la concentración de feromona, para mejorar la tasa de convergencia, se aplica la actualización global de feromonas de la ruta óptima por medio de la ecuación 3.

$$\tau ij = (1 - p) * \tau ij + \Delta \tau ij \qquad (3)$$

Donde p es el parámetro de evaporación de la feromona, que es el valor crítico para el rendimiento del algoritmo ACO, $\Delta \tau ij$ es la cantidad de feromona que queda en el camino óptimo de las hormigas y está dado por la ecuación 4.

$$\Delta \tau ij = \frac{Q}{L} \qquad (4)$$

Donde Q es la cantidad total de feromonas y $L = B$ donde $B = Probabilidad\ de\ bloqueo$.

$$1 - \left(\frac{Pr}{Pe}\right) \qquad (5)$$

$$donde\ Pr = Paquetes\ recibidos; \qquad Pe = Paquetes\ enviados$$

En la tabla 3.23. se toma un grupo de valores razonables de acuerdo a la experimentación realizada en [84], donde los autores toman muestras de flujos ratón y elefante para el valor de los parametros.

Tabla 3.23. Parámetros de inicio ACO.

ηij	Δτij	α	β	ρ
bij	Q/L	1	3	0.1

En ese mismo sentido, para dar inicio a la compilación del algoritmo de enrutamiento basado en la optimización de colonia de hormigas, se deben establecer las

condiciones iniciales en relación a la ruta fuente y la ruta destino las cuales están dadas en el archivo temporal "*/usr/local/vtn/sbin/demand"* en las líneas 2 y 3 respectivamente, además para determinar el valor máximo de interaciones del algoritmo nos basamos en [89] donde se logra demostrar que el desempeño de la red es dependiente al esfuerzo computacional, por lo cual se determina *(iteracion<=250)* con recurso computacional de memoria Ram de 2Gb y un procesador de 2 core a 1.8 Ghz.

Algoritmo 1: enrutamiento por colonización de hormigas ACO

```
Entradas: G(v,e), src, dst                    ; entradas del algoritmo
Salidas: mejor_ruta                           ; salida del algoritmo
begin
    while iteracion<=M                        ; número de iteraciones del algoritmo
        for                                   ; inicio ciclo for para cada hormiga
        set nodo=src
          while nodo=dst                      ; cuando nodo=dst cambia al siguiente
                                                nodo con la mejor Pij
            ecuación (2)
            nodo=lista_rutas                  ; nodo = lista de rutas aplicadas del
                                                algoritmo
          end while
          mejor_ruta = selecciont_ruta (lista_rutas)   ; la ruta de la lista_rutas con mayor
                                                         probabilidad es la mejor_ruta
          end for
          ecuación (3)                        ; actualización de la feromona.
          Iteracion ++
end while
      Imprimir mejor_ruta                     ; exporta a la tabla datos_cong campo
                                                connection_route la mejor_ruta.
end
```

Lo anterior implica que el algoritmo de enrutamiento ACO, logra encontrar la ruta con menor probabilidad de bloqueo de una lista de rutas (connection_route), las cuales serán almacenadas una a una en la tabla "*datos_cong*" de la base de datos "*datos_VTN*" y son importados a la tabla 3.22, además importa los datos (port_connection y bp) contenidos en la base de datos de *OpenDayLight*data de *OpenDayLight.*

Tabla 3.24. Importación base de datos "*datos_VTN*".

port_connection	Ip_address	connection_route	Bp

De acuerdo a la información almacenada en la base de datos "*datos_VTN*" del CDS, durante cada iteración el método cognitivo tiene la capacidad de comparar las nuevas peticiones de rutas contra la experiencia obtenida a través del tiempo. Y así tomar la decisión de seguir enviando el algoritmo de enrutamiento bajo la metaheuristica ACO

o enviar una ruta anteriormente obtenida gracias a la experiencia.

Técnicamente el método cognitivo aprende debido a la experiencia brindada en el almacenamiento de las rutas escogidas y de otros parámetros importados de la base de datos de *OpenDayLight,* es así que la implantación del método cognitivo está dada por el siguiente *Script.*

Algoritmo 2: mejor ruta.

```
$src = 'tail -n +2 /usr/local/vtn/sbin/demand'                    ;cargar ip de entrada
$dst = 'tail -n +3 /usr/local/vtn/sbin/demand'                    ;cargar ip de salida
$mejor_route = 'tail -n +22 /home/vtn/routing_aco.sh'             ;mejor ruta escogida por el
                                                                  algoritmo ACO
LOAD DATA INFILE 'ftp 92.168.1.30/home/jlrivera/consulta_bp_TEST.csv'
INTO TABLE datos_cong FIELDS TERMINATED BY ';'                    ; traer por medio de ftp las celdas
(port_connetion, ip_address, bp);                                 ;port_connetion, ip_address, bp
                                                                  ;archivo consulta_bp_TEST.csv

$servername = "192.168.1.33";                                     ; conexion base de datos MYSQL
del servidor VTN.
$username = "root";
$database = "datos_VTN";
$password = "mona1951nina";
select port_connetion, ip_address, connection_route, bp from datos_cong
        where port_connetion=port_connetion and ip_address=ip_address;
        case
        do{
                i+=1
           }
        while (i<10) max(bp)<bp then @sh /home/vtn/routing_aco.sh     ; seleccionar los datos
                                                                      necesarios, si el ultimo
                                                                      valor de bp es mayor; a los
                                                                      datos anteriores de bp
        else $mejor_ruta=>bp.connection_route                         ;entonces enviar el
                                                                      algoritmo ;ACO, de lo
                                                                      contrario enviar la ruta
                                                                      ;con la menor probabilidad
                                                                      de bloqueo
        from datos_cong
end
```

3.2.6.1. Resultados obtenidos desde la topología de red Prueba TEST

La preparación de la prueba en la topología de red Prueba Test para realizar el análisis de resultados desde el parámetro de desempeño se determina de acuerdo a la siguiente lista de chequeo.

Tabla 3.25. Lista de chequeo preparación de la red prueba TEST cognitiva.

	Prueba TEST	**NSFNET**	**IRIS10**
Carga de Topología			
MININET	X		
OpenDayLight	X		
Virtual Tenant Network			
Método Cognitivo			
Sin			
Con	X		
Carga de tráfico, espacio de los escenarios			
TráficoPruebaTEST.sh	X		
TráficoNSFNET.sh			
TráficoIRIS10.sh			
Variaciones de tráfico			
HostelefanteTEST.sh	X		
HostratonTEST.sh	X		
HostelefanteyratonTEST.sh	X		
HostelefanteNSFNET.sh			
HostratonNSFNET.sh			
HostelefanteyratonNSFNET.sh			
HostelefanteIRIS10.sh			
HostratonIRIS10.sh			
HostelefanteyratonIRIS10.sh			

Dentro de la lista de chequeo se carga la topología de red Prueba TEST tanto en la herramienta MININET como el aplicativo *OpenDayLight*. Seguidamente se hace la verificación del funcionamiento del algoritmo de enrutamiento bajo la metaheuristica ACO, para así, cargar de tráfico el espacio del escenario por medio del *Script*. TráficoTEST.sh uniformemente. Finalmente se cargan uno a uno las tres variaciones de tráfico HostratonTEST.sh, HostelefanteyratonTEST.sh y HostelefanteTEST.sh sobre la red Prueba TEST.

3.2.6.1.1. Variación 1: Tráfico ratón.

La variación 1, consiste en inyectar tráfico (HostratonTEST.sh) ratón entre los diferentes Host, con intervalo de 2 segundos, durante 5 días tomando muestras cada 12 horas de acuerdo a la configuración realizada en el CRONTAB del servidor que aloja OpenDayLight, de la probabilidad de bloqueo de los enlaces y la red por medio del PHP, (consulta_bp_TEST.php), como se muestra en la tabla 3.26.

Tabla 3.26. Resultado probabilidad de bloqueo de la red Prueba TEST cognitiva con Tráfico raton.

# Iteraciones	horas	Bp
21600	12	0.001466667
43200	24	0.002266667
64800	36	0.003016667
86400	48	0.003441667
108000	60	0.003816667
129600	72	0.004166667
151200	84	0.004408333
172800	96	0.004575
194400	108	0.00465
216000	120	0.004716667

En la Figura 3.48 se da a conocer el consolidado de las pruebas realizadas durante 5 días (120 Horas) de la probabilidad de bloqueo de la red Prueba TEST con Tráfico Ratón en relación al tiempo en horas.

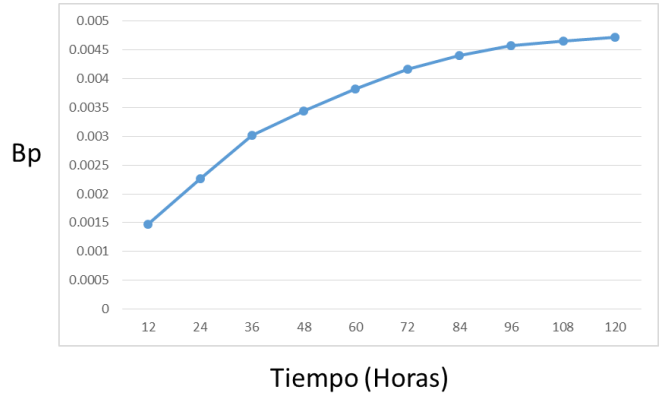


Figura 3.48. BP en la red de prueba TEST con algoritmo cognitive y tráfico ratón.

3.2.6.1.2. Variación 2: Tráfico elefante y ratón.

La variación 2, consiste en inyectar tráfico (*HostelefanteyratonTEST.sh*) elefante y ratón entre los diferentes Host, con intervalo de 2 segundos, durante 5 días tomando muestras cada 12 horas del tráico ratón y con intervalo de 60 segundos, durante 5 días tomando muestras cada 12 horas de tráfico elefante, de acuerdo a la configuración realizada en el *CRONTAB* del servidor que aloja *OpenDayLight*, de la probabilidad de bloqueo de los enlaces y la red por medio del PHP, *(consulta_bp_TEST.php),* como se muestra en la tabla 3.27.

Tabla 3.27: Resultado probabilidad de bloqueo de la red Prueba TEST cognitiva con tráfico elefante y ratón.

horas	Bp
12	0.005692
24	0.007025
36	0.008133
48	0.009183
60	0.010125
72	0.010867
84	0.011492
96	0.011908
108	0.012250
120	0.012508

En la Figura 3.49 se da a conocer el consolidado de las pruebas realizadas durante 5 días (120 Horas) de la probabilidad de bloqueo de la red Prueba TEST con tráfico elefante y ratón en relación al tiempo en horas.

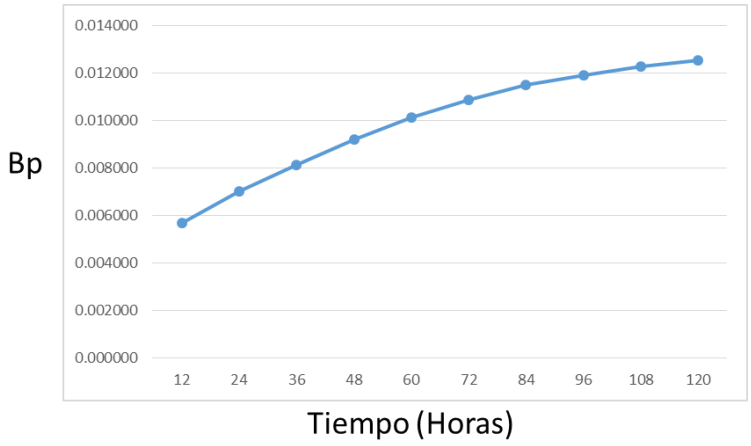


Figura 3.49. BP en la red de prueba TEST con algoritmo cognitive y tráfico elefante y ratón.

3.2.6.1.3. Variación 3: Tráfico elefante.

La variación 3, consiste en inyectar tráfico (*HostelefanteTEST.sh*) ratón entre los diferentes Host, con intervalo de 60 segundos, durante 5 días tomando muestras cada 12 horas de acuerdo a la configuración realizada en el *CRONTAB* del servidor que aloja OpenDayLight, de la probabilidad de bloqueo de los enlaces y la red por medio del PHP, *(consulta_bp_TEST.php),* como se muestra en la tabla 3.28.

Tabla 3.28. Resultado probabilidad de bloqueo de la red Prueba TEST cognitiva con Tráfico elefante.

# Iteraciones	horas	Bp
720	12	0.01085
1440	24	0.015233333
2160	36	0.018891667
2880	48	0.022375
3600	60	0.024658333
4320	72	0.026441667
5040	84	0.027891667
5760	96	0.029266667
6480	108	0.029891667
7200	120	0.030291667

En la figura 3.50 se da a conocer el consolidado de las pruebas realizadas durante 5 días (120 Horas) de la probabilidad de bloqueo de la red Prueba TEST con Tráfico Elefante en relación al tiempo en horas.

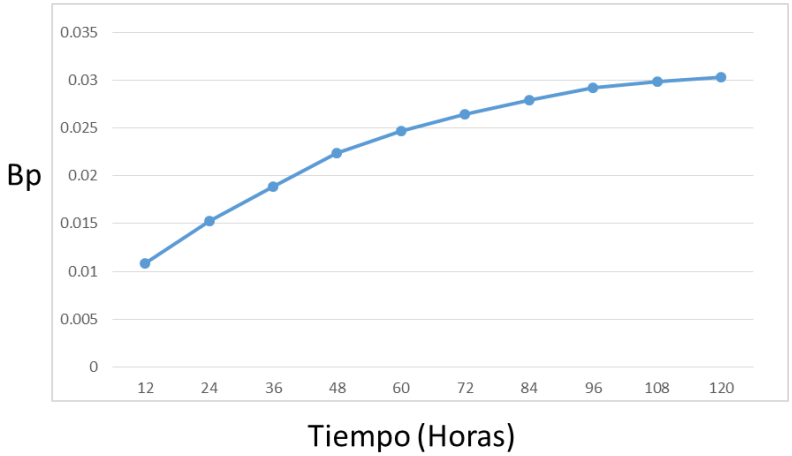


Figura 3.50. BP en la red de prueba TEST con algoritmo cognitive y tráfico elefante.

3.2.6.2. Resultados obtenidos desde la topología de red NSFNET.

La preparación de la prueba en la topología de red NSFNET para realizar el análisis de resultados desde el parámetro de desempeño se determina de acuerdo a la siguiente lista de chequeo.

Tabla 3.29: Lista de chequeo preparación de la red NSFNET cognitiva.

	Prueba TEST	NSFNET	IRIS10
Carga de Topología			
MININET		X	
OpenDayLight		X	
Virtual Tenant Network			
Método Cognitivo			
Sin			
Con		X	
Carga de tráfico, espacio de los escenarios			
TráficoPruebaTEST.sh			
TráficoNSFNET.sh		X	
TráficoIRIS10.sh			
Variaciones de tráfico			
HostelefanteTEST.sh			
HostratonTEST.sh			
HostelefanteyratonTEST.sh			
HostelefanteNSFNET.sh		X	
HostratonNSFNET.sh		X	
HostelefanteyratonNSFNET.sh		X	
HostelefanteIRIS10.sh			
HostratonIRIS10.sh			
HostelefanteyratonIRIS10.sh			

Dentro de la lista de chequeo se carga la topología de red NSFNET tanto en la herramienta MININET como el aplicativo *OpenDayLight*. Seguidamente se hace la verificación del funcionamiento del algoritmo de enrutamiento bajo la metaheuristica ACO, para así, cargar de tráfico el espacio del escenario por medio del SHELL. *TráficoNSFNET.sh* uniformemente. Finalmente se cargan uno a uno las tres variaciones de tráfico *HostratonNSFNET.sh, HostelefanteyratonNSFNET.sh y HostelefanteNSFNET.sh* sobre la red NSFNET.

3.2.6.2.1. Variación 1: Tráfico ratón.

La variación 1, consiste en inyectar tráfico (*HostratonNSFNET.sh*) ratón entre los diferentes Host, con intervalo de 2 segundos, durante 5 días tomando muestras cada 12 horas de acuerdo a la configuración realizada en el *CRONTAB* del servidor que aloja *OpenDayLight*, de la probabilidad de bloqueo de los enlaces y la red por medio del PHP, *(consulta_bp_NSFNET.php),* como se muestra en la tabla 3.30.

Tabla 3.30. Resultado probabilidad de bloqueo de la red NSFNET cognitiva con Tráfico raton.

# Iteraciones	horas	Bp
21600	12	0.002675824
43200	24	0.003377473
64800	36	0.004004945
86400	48	0.004603846
108000	60	0.00495989
129600	72	0.005267582
151200	84	0.005545055
172800	96	0.005789011
194400	108	0.005984615
216000	120	0.006138462

En la Figura 3.51 se da a conocer el consolidado de las pruebas realizadas durante 5 días (120 Horas) de la probabilidad de bloqueo de la red NSFNET con tráfico Ratón en relación al numero de iteraciones dadas cada 12 horas.

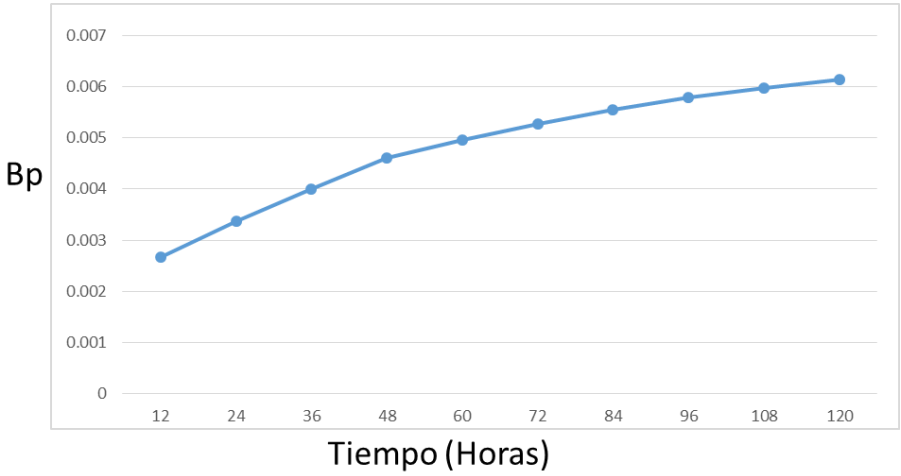


Figura 3.51. BP en la red NSFNET con algoritmo cognitive y tráfico ratón.

3.2.6.2.2. Variación 2: Tráfico elefante y ratón.

La variación 2, consiste en inyectar tráfico (*HostelefanteyratonNSFNET.sh*) elefante y ratón entre los diferentes Host, con intervalo de 2 segundos, durante 5 días tomando muestras cada 12 horas del tráfico ratón y con intervalo de 60 segundos, durante 5 días tomando muestras cada 12 horas de acuerdo a la configuración realizada en el *CRONTAB* del servidor que aloja *OpenDayLight*, de la probabilidad de bloqueo de los enlaces y la red por medio del PHP, *(consulta_bp_NSFNET.php),* como se muestra en la tabla 3.31.

Tabla 3.31. Resultado probabilidad de bloqueo de la red NSFNET cognitiva con tráfico elefante y ratón.

horas	Bp
12	0.007515
24	0.009045
36	0.010279
48	0.011473
60	0.01254
72	0.01316
84	0.013719
96	0.014204
108	0.014631
120	0.014863

En la Figura 3.52 se da a conocer el consolidado de las pruebas realizadas durante 5 días (120 Horas) de la probabilidad de bloqueo de la red NSFNET con tráfico elefante y ratón en relación al numero de iteraciones dadas cada 12 horas.

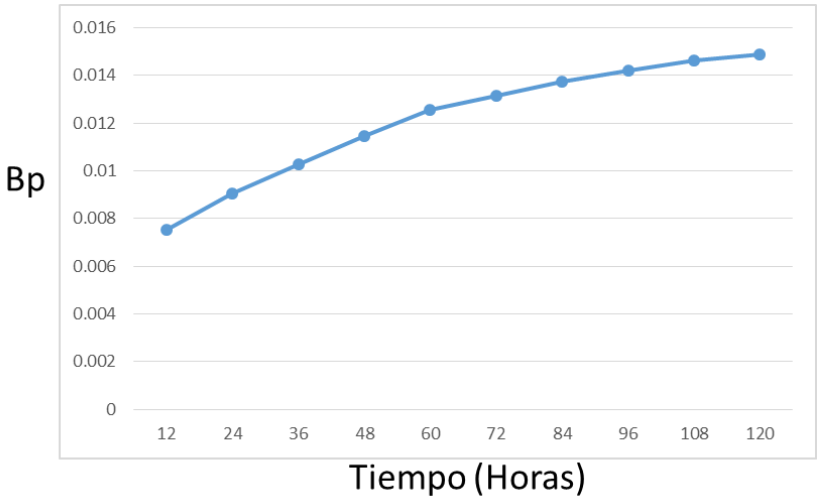


Figura 3.52. BP en la red NSFNET con algoritmo cognitive y tráfico elefante y ratón.

3.2.6.2.3. Variación 3: Tráfico elefante.

La variación 3, consiste en inyectar tráfico (*HostelefanteNSFNET.sh*) elefante entre los diferentes Host, con intervalo de 60 segundos, durante 5 días tomando muestras cada 12 horas de acuerdo a la configuración realizada en el *CRONTAB* del servidor que aloja OpenDayLight, de la probabilidad de bloqueo de los enlaces y la red por medio del PHP, *(consulta_bp_NSFNET.php),* como se muestra en la tabla 3.32.

Tabla 3.32. Resultado probabilidad de bloqueo de la red NSFNET cognitiva con Tráfico elefante.

# Iteraciones	horas	Bp
720	12	0.028821
1440	24	0.03294
2160	36	0.036885
2880	48	0.039916
3600	60	0.042917
4320	72	0.045905
5040	84	0.047893
5760	96	0.049435
6480	108	0.050679
7200	120	0.051488

En la Figura 3.53 se da a conocer el consolidado de las pruebas realizadas durante 5 días (120 Horas) de la probabilidad de bloqueo de la red NSFNET con Tráfico Elefante en relación al numero de iteraciones dadas cada 12 horas.

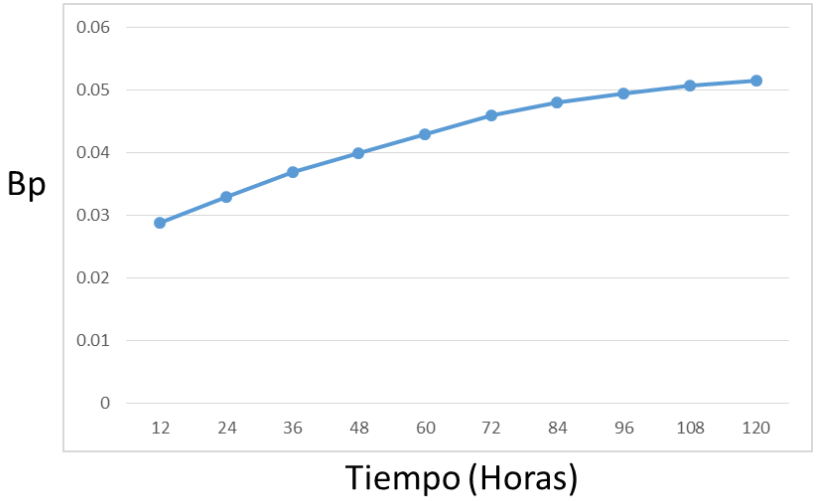


Figura 3.53. BP en la red NSFNET con algoritmo cognitive y tráfico elefante.

3.2.6.3. Resultados obtenidos desde la topología de red IRIS10.

La preparación de la prueba en la topología de red IRIS10 para realizar el análisis de resultados desde el parámetro de desempeño se determina de acuerdo a la siguiente lista de chequeo.

Tabla 3.33. Lista de chequeo preparación de la red IRIS10 cognitiva.

	Prueba TEST	NSFNET	IRIS10
Carga de Topología			
MININET			X
OpenDayLight			X
Virtual Tenant Network			
Método Cognitivo			
Sin			
Con			X
Carga de tráfico, espacio de los escenarios			
TráficoPruebaTEST.sh			
TráficoNSFNET.sh			
TráficoIRIS10.sh			X
Variaciones de Tráfico			
HostelefanteTEST.sh			
HostratonTEST.sh			
HostelefanteyratonTEST.sh			
HostelefanteNSFNET.sh			
HostratonNSFNET.sh			
HostelefanteyratonNSFNET.sh			
HostelefanteIRIS10.sh			X
HostratonIRIS10.sh			X
HostelefanteyratonIRIS10.sh			X

Dentro de la lista de chequeo se carga la topología de red IRIS10 tanto en la herramienta MININET como el aplicativo OpenDayLight. Seguidamente se hace la verificación del funcionamiento del algoritmo de enrutamiento bajo la metaheristica ACO, para así, cargar de tráfico el espacio del escenario por medio del SHELL. TráficoIRIS10.sh uniformemente. Finalmente se cargan uno a uno las tres variaciones de tráfico HostelefanteIRIS10.sh, HostratonIRIS10.sh, HostelefanteyratonIRIS10.sh sobre la red IRIS10.

3.2.6.3.1. Variación 1: Tráfico ratón.

La variación 1, consiste en inyectar tráfico (*HostratonIRIS10.sh*) ratón entre los diferentes Host, con intervalo de 2 segundos, durante 5 días tomando muestras cada 12 horas de acuerdo a la configuración realizada en el *CRONTAB* del servidor que aloja OpenDayLight, de la probabilidad de bloqueo de los enlaces y la red por medio del PHP, *(consulta_bp_ IRIS10.php),* como se muestra en la tabla 3.34.

Tabla 3.34. Resultado probabilidad de bloqueo de la red IRIS10 cognitiva con tráfico ratón.

# Iteraciones	horas	Bp
21600	12	0.003833224
43200	24	0.004412171
64800	36	0.004818092
86400	48	0.005250329
108000	60	0.005628947
129600	72	0.005955263
151200	84	0.006176316
172800	96	0.006351645
194400	108	0.006540789
216000	120	0.006662829

En la figura 3.54 se da a conocer el consolidado de las pruebas realizadas durante 5 días (120 Horas) de la probabilidad de bloqueo de la red IRIS10 con tráfico Ratón en relación al numero de iteraciones dadas cada 12 horas.

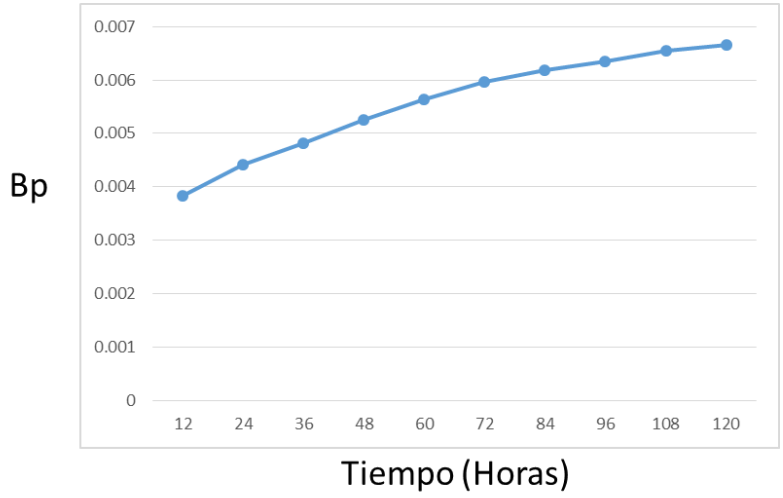


Figura 3.54. BP en la red IRIS10 con algoritmo cognitive y tráfico ratón.

3.2.6.3.2. Variación 2: Tráfico elefante y ratón.

La variación 2, consiste en inyectar tráfico (*HostelefanteyratonIRIS10.sh*) elefante y ratón entre los diferentes Host, con intervalo de 2 segundos, durante 5 días tomando muestras cada 12 horas del Tráfico ratón y con intervalo de 60 segundos, durante 5 días tomando muestras cada 12 horas de acuerdo a la configuración realizada en el *CRONTAB* del servidor que aloja *OpenDayLight*, de la probabilidad de bloqueo de los enlaces y la red por medio del PHP, *(consulta_bp_IRIS10.php),* como se muestra en la tabla 3.35.

Tabla 3.35. Resultado probabilidad de bloqueo de la red IRIS10 cognitiva con Tráfico elefante y ratón.

horas	Bp
12	0.009687171
24	0.011216447
36	0.012415461
48	0.013348684
60	0.014340461
72	0.015225
84	0.015894737
96	0.016430921
108	0.01683125
120	0.017010197

En la Figura. 3.55 se da a conocer el consolidado de las pruebas realizadas durante 5 días (120 Horas) de la probabilidad de bloqueo de la red IRIS10 con Tráfico elefante y ratón en relación al numero de iteraciones dadas cada 12 horas.

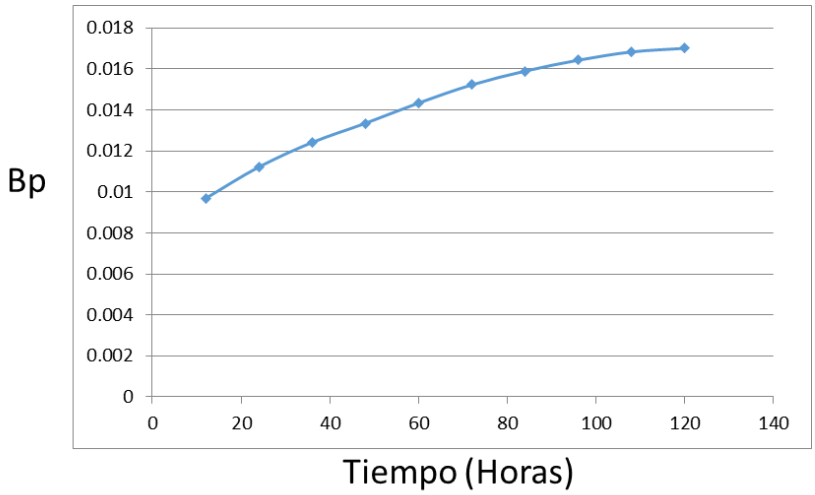


Figura. 3.55. BP en la red IRIS10 con algoritmo cognitive y tráfico elefante y ratón.

3.2.6.3.3. Variación 3: Tráfico elefante.

La variación 3, consiste en inyectar tráfico (*HostelefanteIRIS10.sh*) elefante entre los diferentes Host, con intervalo de 60 segundos, durante 5 días tomando muestras cada 12 horas de acuerdo a la configuración realizada en el *CRONTAB* del servidor que aloja OpenDayLight, de la probabilidad de bloqueo de los enlaces y la red por medio del PHP, *(consulta_bp_IRIS10.php),* como se muestra en la tabla 3.36.

Tabla 3.36. Resultado probabilidad de bloqueo de la red IRIS10 cognitiva con tráfico elefante.

# Iteraciones	horas	Bp
720	12	0.0336
1440	24	0.0389
2160	36	0.0427
2880	48	0.0463
3600	60	0.0499
4320	72	0.0529
5040	84	0.0554
5760	96	0.0576
6480	108	0.0593
7200	120	0.0595

En la Figura. 3.56 se da a conocer el consolidado de las pruebas realizadas durante 5 días (120 Horas) de la probabilidad de bloqueo de la red IRIS10 con tráfico elefante en relación al numero de iteraciones dadas cada 12 horas.

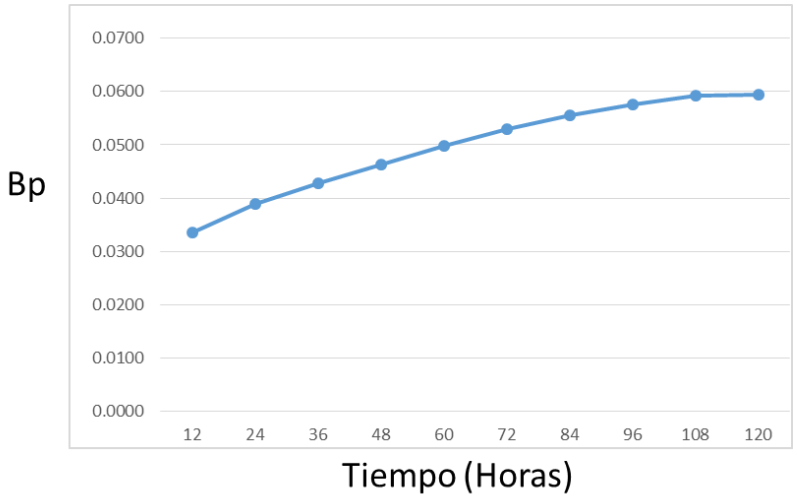


Figura. 3.56. PB para la red IRIS10 cognitiva tráfico elefante.

3.3. Análisis comparativo de las topologías de red sin método cognitivo y con método cognitivo.

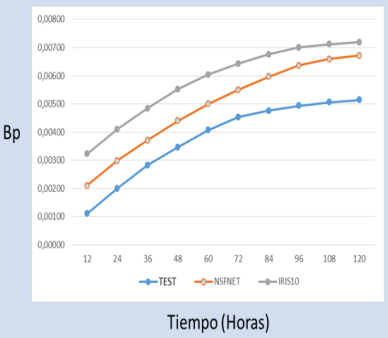
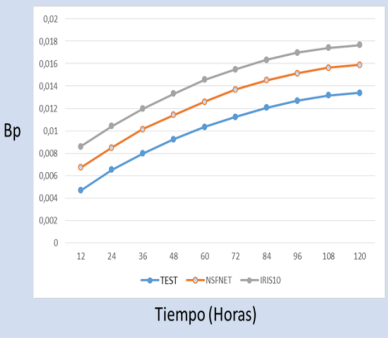
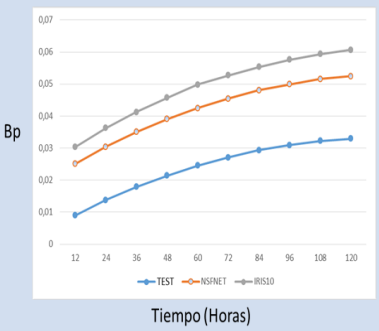
A continuación se presenta el análisis comparativo de las técnicas estudiadas, a partir de los resultados y evaluación de las simulaciones obtenidas en los casos de estudio, con lo cual se pudo determinar algunos factores que inciden directamente en el parámetro de desempeño. El comportamiento de las redes simuladas Prueba TEST, NSFNET e IRIS10 con respecto a las variaciones de flujo de tráfico ratón y elefante, se realiza con el propósito de establecer cual caso de estudio presenta un mejor desempeño en relación a la probabilidad de bloqueo de la red.

3.3.1. Análisis de resultados caso de estudio 1.

Continuando con el análisis y teniendo en cuenta cada uno de los resultados obtenidos en el caso de estudio 1, se realiza una comparación de acuerdo a las variaciones realizadas (Tráfico ratón, tráfico elefante y ratón y Tráfico elefante) en cada una de las topologías de red simuladas buscando identificar el parámetro de desempeño.

La tabla 3.37 da a conocer la comparación de las diferentes topologías en relación a la probabilidad de bloqueo de la red, demostrando así, que los flujos de tráfico y la cantidad de enlaces están directamente relacionados con la probabilidad de bloqueo.

Tabla 3.37 Relacion topologias de red para las variaciones en el caso de estudio 1.

Variación 1, BP respecto a la variación del tráfico ratón	Variación 2, BP respecto a la variación del tráfico ratón	Variación 3, BP respecto a la variación del tráfico ratón
		

El análisis se enfoca en el comportamiento esperado, en la medida que el tráfico de la red incrementa, el parámetro de desempeño Bp muestra convergencia en el tiempo

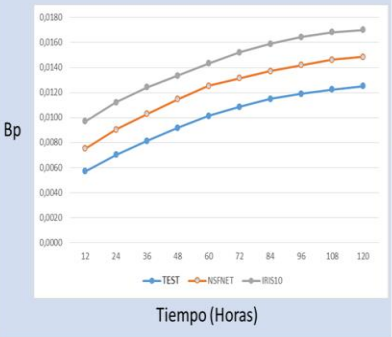
para cada escenario, ademas es de analizar, con el aumento de numero de nodos y numero de enlaces en las topologías empleadas la Bp es mayor en IRIS10 que en NSFNET y Prueba TEST.

3.3.2. Análisis de resultados caso de estudio 2.

De acuerdo a los resultados obtenidos en el caso de estudio 2, se continua con el análisis propuesto en el caso de estudio 1 donde se realiza la comparación de las diferentes variaciones (Trafico ratón, trafico elefante y ratón y trafico elefante), para cada una de las topologías simuladas en relación al parámetro de desempeño.

Las Figuras 3.59, 3.60, 3.61 realizan las comparaciónes de las diferentes topologías de red, en relación a la probabilidad de bloqueo de la red, demostrando así, que los flujos de tráfico y la cantidad de enlaces están directamente relacionados con la probabilidad de bloqueo como se muestra en la tabla 3.38.

Tabla 3.38. Relacion topologias de red para las variaciones en el caso de estudio 2.

Variación 1, BP respecto a la variación del tráfico ratón	Variación 2, BP respecto a la variación del tráfico ratón	Variación 3, BP respecto a la variación del tráfico ratón
		

Por último es muy importante resaltar, que se demuestra una vez más, que en relación al análisis realizado en el caso de estudio 1, en el tiempo la probabilidad de bloqueo para cada escenario de simulacion esta directamente relacionado con el tráfico de la red, los enlaces de conexión y el número de nodos interconectados, además se logra evidenciar que el comportamiento del parámetro de desempeño al interior de la red SDN con método cognitivo tiende a crecer levemente en las tres variaciones de tráfico (tráfico raton, tráfico raton y elefante y tráfico elefante) de acuerdo a las topologías de red configuradas.

3.3.3. Comparación de resultados entre la red con método cognitivo y la red sin método cognitivo

Este trabajo de investigación busca como aporte central mejorar el comportamiento del parámetro de desempeño probabildiad de bloqueo en la red con método cognitivo, por medio de la creación, implantación y puesta en marcha del algoritmo de enrutamiento basado en la metaheuristica ACO con enfoque cognitivo, el cual ha demostrado un mejor conportamiento después de un determinado intervalo de tiempo en relación al caso de estudio 1, además se logra determinar que el algoritmo de enurtamiento por defecto de OpenDayLight sin cognición es mejor al inicio de las pruebas debido a los minimos retardos para encontrar la mejor ruta y el poco trafico implantado al inicio sobre las redes de prueba.

Una vez realizado el análisis individual de cada uno de los casos de estudio, acorde a lo definido en los capítulos 2 y 3, se pudo determinar algunos factores que inciden significativamente en la probabilidad de bloqueo de las redes simuladas, dentro de los cuales se resaltan:

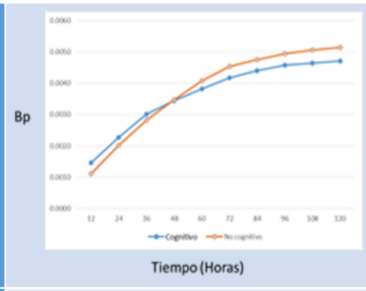
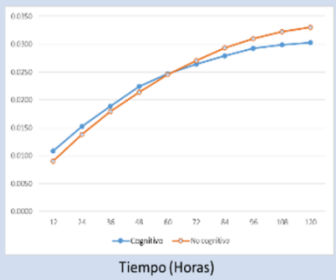
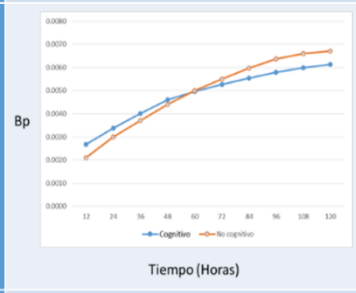
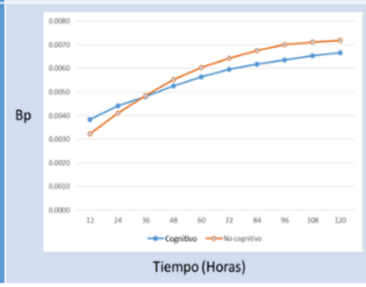
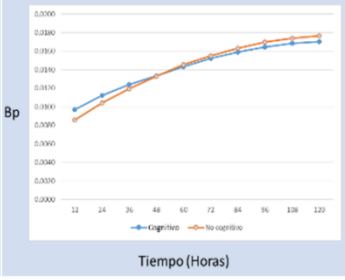
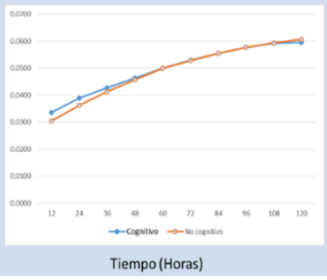
- Los resultados obtenidos para el parámetro de desempeño en los casos de estudio evaluados, con las condiciones realistas dadas en los escenarios de prueba son relativamente similares a los que se obtienen en condiciones del mundo real.
- El algoritmo de enrutamiento por defecto (*Simple Forwarding*) de SDN, obtiene como resultado bajo las condiciones dadas, al inicio de las simulaciones una menor probabildiad de bloqueo en relación al algoritmo de enrutamiento ACO. Con el tiempo la red SDN con el método cognitivo tiende a mejorar en relación al parámetro de desempeño, esto se debe en gran medida al aprendizaje dado por la experiencia del método cognitivo.
- El método cognitivo configurado para la red SDN bajo el algoritmo de enrutamiento ACO inicia con una BP elevada en relación a la red SDN sin método cognitivo debido al tiempo de ejecución que toma el envio del algoritmo ACO y la toma de las primeras decisiones, seguidamente el método cognitivo almacena las rutas escogidas con el propósito de tomar o no tomar la decisión de enviar nuevamente el algoritmo ACO para encontrar rutas de conexión. Logrando asi, almacenar las rutas de conexión dadas por el algoritmo ACO y en el futuro interacion a interacion actuar de acuerdo a su experiencia.
- Seguidamente la importancia del uso del método cognitivo bajo la

metaheurística ACO es bastante relevante dado que la implentacion del método cognitivo reduce significativamenrte el parámetro de desempeño en el tiempo, mientras que el algoritmo de enrutamiento ACO en relación al algoritmo de enrutamiento (*Simple Forwarding*) tiene una mayor probalidad de bloqueo.

- Finalmente la probabilidad de bloqueo depende tanto del tráfico de la red, el número de nodos interconectados y el número de enlaces de conexión en proporisiones similares para los dos casos de estudio, a medida que aumentan los parámetros anteriormente mensionados la probabilidad de bloqueo para los enlaces y para la red aumenta ligeramente para la red sin método cognitivo y levemente para la red con método cognitivo.

Los resultados comparativos se muestran en la tabla 3.35 donde se toma los escenarios de simulación con las variaciones establecidas para el comportamiento de la probabilidad de bloqueo en el tiempo.

Tabla 3.39. Resumen comparativo casos de estudio con y sin cognición.

	Variación 1: Tráfico ratón	Variación 2: Tráfico ratón y elefante	Variación 3: Tráfico elefante
Topología de red Prueba TEST			
Topología de red NSFNET			
Topología de red IRIS10			

Capítulo 4. Conclusiones, recomendaciones y trabajos futuros.

En este capítulo se presentan las conclusiones, recomendaciones y trabajos futuros. Las conclusiones son resultado de los objetivos y el trabajo desarrollado para alcanzarlos, las recomendaciones y trabajos futuros son producto de la metodología y ejecución de la simulación.

4.1. Conclusiones.

- Se comprobó que el método cognitivo enfocado en CHRON en el tiempo tiende a mejorar el parámetro de desempeño en relación a la red bajo arquitectura SDN, además se pudo determinar que la probabilidad de bloqueo de los enlaces y de la red en los casos de estudio con método cognitivo y sin método cognitivo estan directamente relacionados con el tráfico de red, numero de enlaces de conexión y el numero de nodos interconectados.
- la implementación del método cognitivo CHRON bajo el algoritmo de enrutamiento ACO, sobre el algoritmo de enrutamiento (*Simple Forwarding*) de la red SDN, es competitivo para resolver el problema probabilidad de bloqueo.
- Se puede concluir que el algoritmo de enrutamiento ACO sin método cognitivo seria poco competivo en relación al parámetro de desempeño en comparación con el algoritmo de enrutamiento (*Simple Forwarding*).
- Con la implementación de *OpenDayLight* se permite tener una interfaz de configuración y administración más precisa de los recursos de red en relación a nuestros requerimientos, todo esto con el fin de facilitar la administración de los equipos de comunicación en la capa de infraestructura además de monitorear en tiempo real el comportamiento de la Red.
- Al utilizar la red definida por software y la virtualización de funciones de red para la configuración de equipos de comunicación, la controladora (*OpenDayLight*) y la aplicación (*Virtual Tenant Network*) permiten implantar la meta heurística ACO sobre los puertos lógicos en función del enrutador (vRouter).

4.2. Recomendaciones.

- Se recomienda implementar escenarios de simulación con flujos de tráfico incremental, ya que permiten un análisis más claro del comportamiento de la probabilidad de bloqueo en las redes diseñadas.
- En relación a las pruebas realizadas, se recomienda tomar muestras de la probabilidad de bloqueo, después de las 72 horas del inicio de la carga de flujo de tráfico para obtener resultados más confiables.

- De acuerdo a las pruebas realizadas al algoritmo de enrutamiento ACO en el método cognitivo CHRON, se recomienda realizar variaciones a las rutas escogidas por ACO, con el fin de mejorar el desempeño de la red al inicio de las simulaciones.

4.3. Trabajos futuros.

- Analizar el desempeño de latencia y Throughput en topologías de red con gran cantidad de nodos y enlaces de conexion en redes bajo arquitectura SDN mediante *OpenDayLight*.
- Analizar el desempeño de una red SDN mediante la aplicación del método cognitivo CHRON basado en PSO respecto a la probabildiad de bloqueo de la red en tres diferentes topologías.
- Analizar la probabilidad de bloqueo en la función de enrutamiento utilizando vBridge de VTN, por medio del método cognitivo CHRON para SDN y VTN.
- Diseñar e implementar el algoritmo de enrutamiento bajo metaheuristicas basadas en población, en la función de QoS en una red bajo arquitectura SDN/NFV.
- Trabajar con distintos métodos cognitivos con el propósito de mejorar el desempeño de la probabilidad de bloqueo en altos flujos de tráfico.

Bibliografía.

[1] N. McKeown. (2009). Software defined networking. INFOCOM keynote talk.

[2] R. Thomas, L. DaSilva, and A. MacKenzie, “Cognitive networks,” in *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. First IEEE International Symposium on*, Nov 2005, pp. 352–360.

[3] L. F. Akyildiz, A. Lee, P. Wang, M. Luo, W. Chou “A Roadmap for Traffic Engineering in Software Defined Networks” ELSEVIER Jun 2014.

[4] A. F. Ruiz, “Estado del Arte Redes Definidas Por Software,” *Universidad Católica de Pereira*, Pereira, 2014 pp. 20–22.

[5] C. M. Castaño, J. E. Valencia Henao, “Analisis Comparativo Entre el Tiempo de Vida de los Paquetes y el uso del Ancho de Banda, en las Redes Basadas en SDN y las Redes Basadas en el Modelo OSI”, *Universidad Tecnologica de Pereira,* pp. 23–25. 2016

[6] Y. Li, D. Li, W. Cui, R. Zhang “Research Base don OSI Model” *IEEE*, pp. 554–556, 2011.

[7] I. Maathuis, W. A. Smit, “The Battle Between Standards: TCP/IP vs OSI Victory Through Path Dependency or by Quality?” *IEEE*, pp 161-163, 2003.

[8] Q. Ge, Y. Zhu, F. Chen “Internal Structures and Design Models for Implementing TCP/IP Protocol Stack on Embedded Applications,” Conference on Circuits, Communications and System, pp 1-2, 2009.

[9] S. Zaman, F. Karray, “TCP/IP Model and Intrusion Detection Systems,” International Conference on Advanced Information Networking and Applications Workshops, pp 91-93, 2009.

[10] “TCP/IP Overview,” *Cisco Systems, Inc. All rights reserved., Document ID: 13769*, 2005.

[11] “Comparacion Modelo OSI y TCP/IP,” tomado https://es.slideshare.net/wsar85/comparacion-modelo-osi-y-tcp-ip.

[12] “Traditional vs Software Defined Networking,” *IP Knowledge We Accelerate your Business, Document,* pp 2-5.

[13] IEEE Criteria for Class IE Electric Systems (Standards style), “*IEEE Standard*” 308, 1969.

[14] Oscar, R, “Software Defined Networking,” *UNIVERSIDAD POLITÈCNICA DE CATALUNYA*, Catalunya. 2014

[15] M. Pan, P. Li, Y. Song, Y. Fang, P. Lin, and S. Glisic. (2014). when spectrum meets clouds: Optimal session based spectrum trading under spectrum uncertainty. IEEE Journal on Selected Areas in Communications, 32(3). 615–627.

[16] A. L. Valdivieso, A. B. Peral, L. I. Barona, L. J. Garcia “SD: Evolution and Opportunities in the Development IoT Applications,” *Hindawi Publishing Corporation,* 2014.
[17] Universidad técnica de Ambato facultad de ingeniería en sistemas electrónica e industrial “red definida por software (SDN) en base a una infraestructura de software de libre distribución”Pag 16 2015.
[18] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe, “Design and implementation of a routing control platform,” *in Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2, ser. NSDI’05. Berkeley*, pp. 15–28, 2005.
[19] K. Calvert. “Reflections on network architecture: An active networking perspective”. *ACM SIGCOMM Computer Communications Review*, pp. 27–30, 2006.
[20] N. Feamster, J. Rexford, E. Zegura “the road to SDN: An intellectual history of programmable networks” *IEEE.* pp 2.
[21] D. Kreutz, F. M. Ramos, P. Verissimo, Fellow, C. E. Rothenberg, S. Azodolmolky, S. Uhlig “Software-Defined Networking: A Comprehensive Survey,” *IEEE,* pp 1-2 2014.
[22] S. Schenker, “The Future of Networking, and the Past of Protocols,”October 2011. [Online]. Available: http://www.youtube.com/watch?v=YHeyuD89n1Y
[23] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden,“A survey of active network research,” Communications Magazine, IEEE, vol. 35, no. 1, pp. 80–86, 1997.
[24] A. Lazar, K.-S. Lim, F. Marconcini, “Realizing a foundation for programmability of atm networks with the binding architecture,” IEEE *Journal on, vol. 14*, *no. 7*, pp. 1214–1227, 1996.
[25] A. Lazar, “Programming telecommunication networks,” *IEEE*, *vol. 11*, *no. 5*, pp. 8–18, Sep 1997.
[26] D. Sheinbein and R. P. Weber, “800 service using SPC network capability,” *The Bell System Technical Journal, vol. 61, no. 7*, Sep.1982.
[27] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe, “Design and implementation of a routing control platform,” *in Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2, ser. NSDI’05. Berkeley*, pp. 15–28, 2005.
[28] Foundation, O. N. (06 de Sep de 2017). *Open Networking Foundation.* Obtenido de https://www.opennetworking.org/sdn-resources/sdn-definition.
[29] Networking, F. O. (08 de Sep de 2017). *Open Networking Foundation*. Obtenido de

https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf.

[30] *OpenDayLight*. (28 de Sep de 2017). *OpenDayLight*. Obtenido de https://www.*OpenDayLight*.org/what-we-do/current-release.

[31] AT&T et al., "Network functions virtualisation (nfv): an introduction, benefits, enablers, challenges & call for action," Network Functions Virtualisation - Introductory White Paper, Oct. 2012. [Online]. Available: https://portal.etsi.org/nfv/nfv_white_paper.pdf.

[32] San Francisco. Ana. 2015 T.P (SDN-NFV) Arquitecturas SDN-NFV Conceptos, Ambitos de Aplicacion y Caso de Uso en la Red de BT GLOBAL SERVICE ESPAÑA 2015 Pag. 41.

[33] "Network functions virtualisation (nfv): network operator perspectives on industry progress," Network Functions Virtualisation - Update White Paper, Oct. 2013. [Online]. Available: http://portal.etsi.org/NFV/NFV_White_Paper2.pdf

[34] Diego Kreutz, M. F. (2014). Software-Defined Networking: A Comprehensive Survey. *IEEE*.

[35] Ángel Leonardo Valdivieso Caraguay, A. B. (2014). SDN: Evolution and Opportunities in the Development. *International Journal of Distributed Sensor Networks*.

[36] Guohui Wang, T. S. (2012). Programming Your Network at Run-time. *Watson Research Center, Rice University*.

[37] Sahrish Khan Tayyaba, S. T. (Mayo de 2016). Routing Techniques in Software Defined Networks: A Survey. COMSATS Institute of Information Technology.

[38] Botero, J. J. (s.f.). Network Functions Virtualization: A Survey.

[39] Mohammed Basheer Al-Somaidai, E. B. (Diciembre de 2014). Survey of software components to emulate OpenFlow protocol as an SDN implementation. American Journal of Software Engineering and Applications.

[40] E. Gelenbe, R. Lent, and Z. Xu, "Design and performance of cognitive packet networks," *Performance Evaluation, vol. 46, no. 2-3*, pp. 155–176, 2001.

[41] Glover F., Laguna M. y Dowsland K. A, 1993. Modern Heuristic Techniques for Combinatorial Problems. C.R. Reeves (ed.), Blackwell, London, 1993.

[42] F. Luna Valero. "Metaheuristicas Avanzadas Para Problemas Reales en Redes de Telecomunicaciones". Universidad de Málaga. pp 19-21. 2008.

[43] J. F. Chicano, "Metaheuristicas e Ingeniería del Software", Universidad de Malaga, pp 50-52, 2007.

[44] M. Dorigo y G. Di Caro. "The Ant Colony Optimization meta-heuristic". En D. Corne, M. Dorigo, y F. Glover, editores, New Ideas in Optimization, páginas 11-32. McGraw Hill, London, UK, 1999.

[45] J. Liu, J. Yang, H. Liu, X. Tian, and M. Gao, “An improved ant colony algorithm for robot path planning,” Soft. Comput., pp. 1-11, May 2016.
[46] P. Moradi and M. Rostami, “Integration of graph clustering with ant colony optimization for feature selection,” Knowl-based. Syst., vol. 84, pp. 144-161, Aug. 2015
[47] O. Dridi, S. Krichen, and A. Guitouni, “A multiobjective hybrid ant colony optimization approach applied to the assignment and scheduling problem,” Int. T. Oper. Res., vol. 21, no. 6, pp. 935-953, Nov. 2014.
[48] T. Stützle y H. Hoos. “MAX-MIN Ant System”. Future Generation Computer System. 16(8): páginas 889–914, junio 2000.
[49] O. Gómez y B. Barán. “Òmicron ACO”. En Proceedings de la Conferencia Latinoamericana en Informática (CLEI’04). Septiembre 2004.
[50] M. Schaerer y B. Barán. “A Multiobjective Ant Colony System For Vehicle Routing Problem With Time Windows”, IASTED International Conference on Applied Informatics, Innsbruck, Austria, 2003.
[51] J. T. Ball, "A Cognitively Plausible Model of Language Comprehension," in the Thirteenth Conference on Behavior Representation in Modeling and Simulation Orlando, FL: National Training Systems Association , 2004, pp. 1498-1508.
[52] J. Mitola, Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio. PhD thesis, Royal Institute of Technology (KTH), 2000.
[53] R. Saracco, “Forecasting the future of information technology: How to make research investment more cost-effective,” IEEE Communications Magazine, vol. 41, pp. 38–45 December 2003.
[54] Mitola, J 2007, *Cognitive Radio Architecture*, Cognitive Networks: Towards Self-Aware Networks, Qusay H. Mahmoud.
[55] Mahmoud, Q, “Cognitive Networks Towards Self-Aware Networks,” *John Wiley & Sons, Ltd*., UK, 2007
[56] M. H. Rehmani, A. C. Viana, H. Khalife, and S. Fdida. (2013). A distributed channel selection strategy for data dissemination in multi-hop cognitive radio networks. ELSEVIER. 1172–1185
[57] M. Pan, P. Li, Y. Song, Y. Fang, P. Lin, and S. Glisic. (2014). when spectrum meets clouds: Optimal session based spectrum trading under spectrum uncertainty. IEEE Journal on Selected Areas in Communications, 32(3). 615–627.
[58] R. Thomas, L. DaSilva, and A. MacKenzie, “Cognitive networks,” in *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. First IEEE International Symposium on*, Nov 2005, pp. 352–360.
[59] G. Sun, G. Liu, and Y. Wang. (2014). Sdn architecture for cognitive radio networks. Cognitive cellular systems (ccs). International workshop on. 1–5.

[60] R. W. Thomas, L. A. DaSilva, A. B. Mackenzie, “Cognitive networks,” *in Proc. of IEEE DySPAN*, pp. 352–360, 2005.
[61] S. Haykin, “Cognitive radio: Brain-empowered wireless communication,” *IEEE Journal on Selected Areas in Communication*, *vol. 23*, pp. 201–220, 2005.
[62] R. Berezdivin, R. Breinig, R. Topp, “Next-generation wireless communications concepts and technologies,” *IEEE Communications Magazine, vol. 40*, *no. 3*, pp. 108– 116, 2002.
[63] A. Alexiou, M. Haardt, “Smart antenna technologies for future wireless systems: Trends and challenges,” *IEEE Communications Magazine, vol. 42, no. 9*, pp. 90–97, 2004.
[64] R. Saracco, “Forecasting the future of information technology: How to make research investment more cost-effective,” *IEEE Communications Magazine, vol. 41*, pp. 38–45, 2003.
[65] Y. Yoshiba, K. Kitayama, Y. Kai, M. Nishihara, R. Okabe, T. Tanaka, T. Takahara, J. C. Rasmussen “First Demostration of Cognitive SDN Orquestration: A Real-Time Congestion-aware Services Provisioning Over OFDM-based 400G OPS and Flexi-WDM OCS Networks,” *OFC Postdeadline Papers,* 2016.
[66] Mahmoud, Q, “Cognitive Networks Towards Self-Aware Networks,” *John Wiley & Sons, Ltd*., UK, 2007
[67] Thomas, R, DaSilva, L & MacKenzie, A 2006, Cognitive Networks: adaptation and learning to achieve end to end performance objectives, Proceedings of the Communications Magazine, vol.44, Issue 12, pp.51-57.
[68] R.W. Thomas, D.H. Friend, L.A. DaSilva, and A.B. MacKenzie, “Cognitive networks: Adaptation and learning to achieve end-to-end performance objectives,” *IEEE Commun. Mag.,* vol. 44, n. 12, pp. 51-57, Dec. 2006
[69] G. S. Zervas and D. Simeonidou, “Cognitive optical networks: Need, requirements and architecture,” in *Proc. ICTON 2010*, paper We.C1.3.
[70] J. Strassner, N. Agoulmine, E. Lehtihet, “FOCALE – A N ovel Autonomic Networking Architecture”, ITSSA NJournal, Vol. 3, No. 1, May 2007, pages 64-79.
[71] J. Strassner, “Enabling Autonomic Network Management Decisions Using a Novel Semantic Representation and Reasoning Approach”, Ph.D. thesis, 2008.
[72] Suyang Ju and Joseph B. Evans. Modeling and Analysis of CogNet Architecture for Cognitive Radio Networks. University of Kansas.
[73] A. Caballero, R. Borkowski, Cognitive, Heterogeneous and Reconfigurable Optical Networks: the CHRON Project. IEEE 2014.

[74] Y. Yoshiba, K. Kitayama, Y. Kai, M. Nishihara, R. Okabe, T. Tanaka, T. Takahara, J. C. Rasmussen “First Demostration of Cognitive SDN Orquestration: A Real-Time Congestion-aware Services Provisioning Over OFDM-based 400G OPS and Flexi-WDM OCS Networks,” *OFC Postdeadline Papers,* 2016.
[75] H. Cui, Y. Zhang, C. Ma, W. Lai, N. C. Beaulieu, S. Sobolevsky, Y. Liu. “ Desing and Realization of Cognitive Routing Resources Using Big Data Analysis in SDN” IEEE International Congress on Big Data, 2015.
[76] I. Ahmad, S. Namal, M. Ylianttila, A. Gurtov “Software Defined CognitiveNetworking” IEEE.
[77] R. Thomas, L. DaSilva, and A. MacKenzie, “Cognitive networks,” in New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. First IEEE International Symposium on, pp. 352–360, 2015.
[78] *OpenDayLight* Documentation (16 de agosto de 2017). Obtenido de https://docs.*OpenDayLight*.org/en/stable-oxygen/user-guide/virtual-tenant-network-(vtn).html
[79] Historia de redes y seguridad informatica (28 de agosto de 2018) obtenido de https://www.timetoast.com/timelines/historia-de-las-redes-y-la-seguridad-informatica
[80] Software libre RedIRIS (28 de agosto de 2018) obtenido de http://cristilav.blogspot.com/2011/02/rediris.html.
[81] R. Borkowski, C. Kachris. Cognitive optical network testbed: EU project CHRON [Invited].IEEE. 2015.
[82] I. Stiakogiannakis, D. Klonidis. Cognitive Heterogeneous Reconfigurable Optical Network: A technoeconomic evaluation. IEEE. 2013.
[83] Gond, J & Goel, A 2010, Performance Evaluation of Wavelength Routed Optical Network with Wavelength Conversion, *Journal of telecommunications*, vol.2, iss.1, April.
[84] C. Wang, G. Zhang. An ACO-based Elephant and Mice Flow Scheduling System in SDN. IEEE 2nd International Conference on Big Data Analysis. 2017
[85] I. Tafur, D. Zibar, N. Guerrero, R. Borkowski. Cognitive Heterogeneous Reconfigurable Optical Networks (CHRON) : Enabling Technologies and techniques. IEEE. 2011.
[86] I. Akyildiz, A. Lee, P. Wang. A roadmap for traffic engineering in software defined networks. ELSEVIER. 2014.
[87] M. Erel, E. Teoman. Scalability Analysis and Flow Admission Control in Mininet-based SDN Environment. IEEE, 2015.
[88] M. Afaq, S. Rehman. Visualization of Elephant Flows and QoS Provisioning in SDN-Based Networks. IEEE. 2015.

[89] J.A. Fernández-Vargas. A. Bonilla-Petriciolet. Desarrollo de un algoritmo de optimización global en colonias de hormigas con selección de región factible para espacios continuos. ELSEVIERDOYMA. 2014