Análisis del Desempeño de una Red Óptica Flexible Definida por Software Empleando un Método Cognitivo de Asignación de Espectro Basado en Machine Learning

Jhonatan Eduardo Achinte Sanjuan

Wilson Bernardo Benavides

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Telecomunicaciones

Grupo I+D Nuevas Tecnologías en Telecomunicaciones - GNTT

Popayán, 2022

Análisis del Desempeño de una Red Óptica Flexible Definida por Software Empleando un Método Cognitivo de Asignación de Espectro Basado en Machine Learning

Jhonatan Eduardo Achinte Sanjuan
Wilson Bernardo Benavides

Trabajo de Grado presentado como requisito para obtener el título de Ingeniero en Electrónica y Telecomunicaciones

Director: MsC. Catalina Muñoz Collazos

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Telecomunicaciones

Grupo I+D Nuevas Tecnologías en Telecomunicaciones - GNTT

Popayán, 2022

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

ACRONIMOS

API	*Application Programming Interface,* Interfaz de Programación de Aplicación
BVT	*Bandwidth Variable Transponder,* Transpondedor de Ancho de Banda Variable
BV-WXC	*Bandwidth variable wavelength cross connect switches,* Conmutador Optico de Ancho de Banda Variable
CF	*Central Frequency,* Frecuencia Central
EON	*Elastic Optical Networking,* Redes Ópticas Elásticas
FAN	*Flexi Access Network Node,* Nodo de red de Acceso Flexible
FDM	*Frequency Division Multiplexing,* Multiplexación por División de Frecuencia
GPON	*Gigabit-capable Passive Optical Network,* Conmutador virtual de Red Óptica Pasiva con Capacidad de Gigabit
ILP	*Integer Lineal Programming,* Programación Lineal Entera
Lx	*Logical xBar,* Conmutador Lógico xBar
ML	*Machine Learning,* Aprendizaje Automático
NFV	*Network Function Virtualization,* Virtualización de Funciones de Red
NOS	*Network Operating System,* Sistema Operativo de Red
NP-hard	*Nondeterministic Polynomial Time Hard,* Problema de Tiempo Difícil Polinomial No Determinístico
OA	*Optical Amplifier,* Amplificador Óptico
OADM	*Optical Add/Drop Multiplexers,* Multiplexores de Inserción/extracción Ópticos

OFDM	*Orthogonal Frequency Division Multiple Access,* Acceso Múltiple por División de Frecuencia Ortogonal
OLT	*Optical Line Terminal,* Terminales de Línea Ópticos
OTDM	*Optical Time Division Multiplexing,* Multiplexación Óptica por División de Tiempo
OTS	*Open Transport Switch,* Conmutador de Transporte Abierto
OWB	*Optical White Box,* Caja blanca Óptica
OXC	*Optical Cross-Connectors,* Cross-conectores Ópticos
QoS	*Quality of Service,* Calidad del Servicio
ROADM	*Reconfigurable Optical Add/Drop Multiplexor,* Multiplexores Ópticos Reconfigurables de Adición Eliminación
RSA	*Routing and Spectrum Allocation,* Asignación de Ruta y Espectro
RWA	*Routing and Wavelength Allocation,* Enrutamiento y Asignación de Longitud de Onda
SA	*Spectrum Allocation,* Asignación de Espectro
SW	*Slot Width,* Ancho de Slot
SDN	*Software Defined Networking,* Redes Definidas por Software
SDON	*Software Defined Optical Networking,* Red Óptica Definida por Software
TDM	*Time Division Multiplexing*, Multiplexación por División de Tiempo
WDM	*Wavelength Division Multiplexing,* Multiplexacion por Division de Longitud de Onda
WSS	*Wavelength Selectable Switches,* Conmutadores con Longitud de Onda Seleccionable

INTRODUCCIÓN

Debido al aumento en el tráfico mundial en el campo de las redes de telecomunicaciones y a la demanda de sus recursos, actualmente las investigaciones se encaminan hacia la búsqueda de soluciones que permitan optimizar los recursos de las redes o hacer una mejor gestión de estos. Los principales problemas a los que se hacen frente actualmente son la poca capacidad de las redes, la poca flexibilidad y la falta de redes programables.

Las Redes Ópticas son ampliamente utilizadas gracias a su alta velocidad de transmisión, alta capacidad y en el caso de las Redes Ópticas Elásticas (EON, *Elastic Optical Netwwork)* gracias a su flexibilidad en entornos de red. Además, la integración de la informática con las redes de telecomunicaciones ha llevado a buscar soluciones basadas en Redes Definidas por Software (SDN, *Software Defined Networking)* y Virtualizacion de Funciones de Red (NFV, *Network Function Virtualization*) lo que permite centralizar la intengencia de la red para mejorar su desempeño. Analogamente esta integración también se extiende a la capa óptica obteniendo un paradigma llamado Redes Opticas Definidas por Software (SDON, *Software Defined Optical Networkind*), de esta manera las SDON deben brindar un transporte de datos rápido y a su vez deben adaptarse a cambios dinámicos de tráfico y topología en tiempo real, lo que genera la necesidad de implementar tecnicas de enrutamiento y asignación de espectro óptico que mejoren la Calidad del Servicio (QoS, *Quality of Service*).

Los métodos cognitivos y algoritmos basados en Aprendizaje Automático (ML, *Machine Learning*) permiten monitorear el estado de la red y brindar alternativas para brindar una gestión más eficiente de los recursos dentro de las redes de telecomunicaciones, y principalmente dentro de las redes SDON. Por estos motivos se plantea la idea de diseñar e implementar mediante simulación un método cognitivo basado en ML que permita realizar una asignación de espectro óptico dentro de una SDON Flexigrid y analizar el impacto que este mecanismo tiene sobre su desempeño.

El documento se presenta en cinco capítulos organizados de la siguiente forma:

El **capítulo 1** abarca los aspectos generales más importantes referentes a le teoría sobre las redes definidas por software, las redes ópticas y las redes ópticas definidas por software, lo cual brinda un soporte que permite afianzar los conceptos que conciernen al trabajo de grado.

El **capítulo 2** detalla y describe dos aspectos muy importantes para este trabajo de investigación como lo son el espectro óptico y la cognición en las redes de telecomunicaciones. Aquí se describe el proceso de RSA dentro de las redes ópticas flexibles, la importancia del manejo del espectro óptico y las diferentes formas y algoritmos que existen para realizar una mejor gestión de este, lo cual es la base del desarrollo de esta investigación; además, se hace un recuento de la importancia de la cognitividad y de los procesos que se pueden implementar en las redes de telecomunicaciones.

El **capítulo 3** trata acerca de la metodología utilizada para llevar a cabo tanto el desarrollo del trabajo como la simulación, aquí se describen algunas de las herramientas investigadas y se hace una elección de la que se usara para el desarrollo de la simulación y también se elige la metodología para ello.

El **capítulo 4** describe la implementación de la red de prueba y del método cognitivo propuesto para asignar espectro óptico, se detallan los módulos implementados y diseñados para obtener un modelo correcto de red, se explica el diseño del método propuesto y se ejecutan las simulaciones para cumplir con los objetivos del trabajo de grado.

El **capítulo 5** estudia los resultados obtenidos en la ejecución de las simulaciones, se plantean las conclusiones al analizar los resultados y se dejan algunos posibles trabajos futuros para continuar con la línea de investigación.

CAPÍTULO 1

REDES DEFINIDAS POR SOFTWARE, REDES ÓPTICAS Y REDES ÓPTICAS DEFINIDAS POR SOFTWARE

1.1 Redes Definidas Por Software

Hasta hace unos años, los recursos de almacenamiento, informática y red se mantenían física y operativamente separados, al igual que los sistemas utilizados para administrar esos recursos, lo cual involucraba significativamente las políticas de acceso, los sistemas y los procedimientos de acceso. Los primeros defensores de las Redes Definidas por Software *(*SDN*, Software Defined Networking)* vieron que los proveedores de dispositivos de red no cumplían sus necesidades, especialmente en los espacios de innovación, desarrollo de funciones y el equipo de enrutamiento y conmutación; también se consideró que los componentes del plano de control y sus dispositivos tenían un costo elevado [1] [10] .

Todos estos aspectos son importantes ya que crearon la motivación que llevó a evolucionar el concepto hasta convertirse en lo que hoy se llama SDN como un enfoque de arquitectura de red que optimiza y simplifica las operaciones de red vinculando más estrechamente la interacción, aprovisionamiento, mensajería y alarmas entre aplicaciones, servicios y dispositivos de red, ya sean reales o virtualizados. Siendo la separación del plano de datos y control uno de los aspectos fundamentales para SDN. En la figura 1.1 se puede observar su arquitectura donde se aprecia las relaciones que tienen las diferentes capas y sus principales componentes o funciones [2] [10]

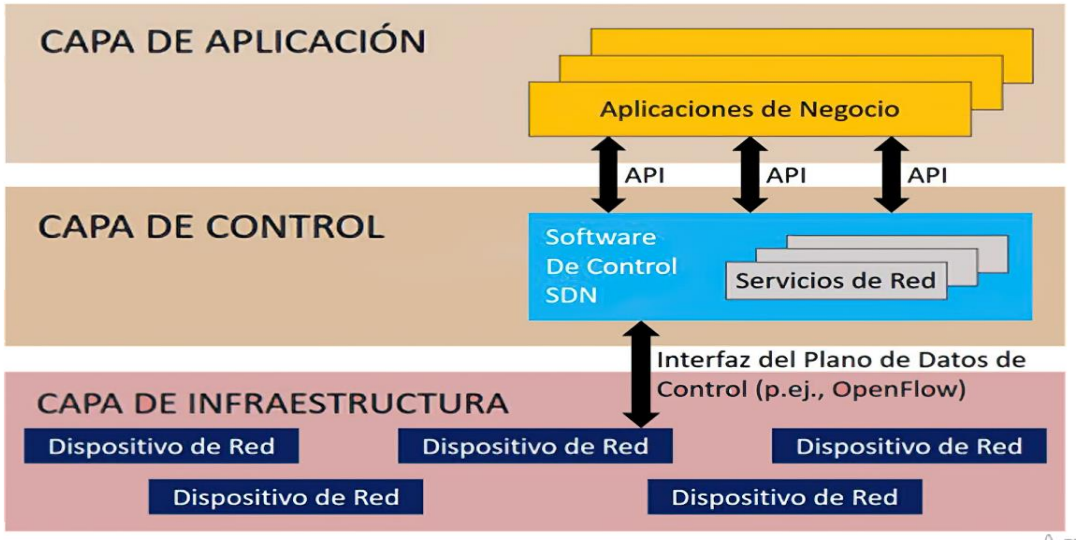

Figura 1.1 Arquitectura SDN. Tomada de [10].

1.1.2 Arquitectura SDN

El objetivo de la arquitectura de SDN es desacoplar los elementos de red tradicionales (*switches, routers*) del plano de datos y del plano de control, centralizando la inteligencia con el estado de la red, esto permite que adquiera una característica programable. Las capas de abstracción SDN constan de tres capas inspiradas en los sistemas informáticos, la capa de aplicación, la capa de control y la capa de infraestructura. La interfaz entre la capa de aplicación y la capa de control se conoce como la interfaz dirección norte (NBI, *NorthBound Interface)*, mientras que la interfaz entre la capa de control y la capa de infraestructura se conoce como la interfaz dirección sur (SBI, *SouthBound Interface),* así mismo, y con la finalidad de establecer comunicación con otros controladores en la red, se cuenta con la interfaz hacia el este (EBI, *EastBound Interface)* e interfaz hacia el oeste (WBI, *WestBound Interface*), en la figura 1.2 se puede observar la interacción entre la interfaces SBI y NBI dentro del dominio de red de SDN y como se pueden interconectar diferentes controladores o dominios SDN gracias a la interfaz WBI, y mediante la interfaz EBI se logran interconectar dominios no SDN [3] [11] [12] .

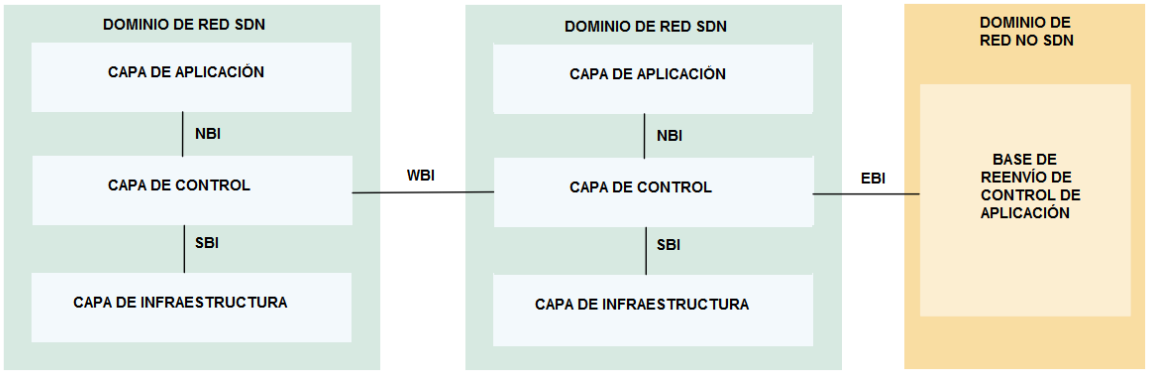

Figura 1.2 Distribución de la arquitectura SDN. Tomada de [12].

Capa de aplicación

La capa de aplicación comprende aplicaciones y servicios de red que utilizan el plano de control para realizar funciones de red sobre la infraestructura física o virtual. Permite crear aplicaciones para automatizar tareas de configuración, provisión y despliegue de nuevos servicios en la red. Esta capa se comunica con la capa de control por medio de una interfaz de programación de aplicación (API, *Application Programming Interface)*, para tener una visión global de las condiciones actuales de la red, con el fin de mejorar la transmisión de datos, mejorar la toma de decisiones locales por nodos de redes individuales acerca de cómo tratar los flujos de tráfico de una aplicación en particular y cómo se deben compartir con las plataformas de control de aplicaciones [11] [12]

Capa de control

La capa de control es responsable de programar y configurar los elementos de red (conmutadores) a través de las SBI. El controlador SDN es una entidad lógica que identifica las instrucciones para configurar la infraestructura de la red basada en los requisitos de la capa de aplicación. Para administrar la red de manera eficiente, los controladores SDN pueden solicitar información de las infraestructuras SDN, como estadísticas de flujo, información de topología, relaciones con los vecinos y estado del enlace de los elementos de la red (nodos). Por medio de esta capa, se crea una vista centralizada de la topología de la red de datos, la cual permite gestionar el flujo de datos en la capa de infraestructura por medio del protocolo *OpenFlow*, también brinda un API para que desde la capa de aplicación se puedan programar flujos y retroalimentar a la aplicación con información de la topología de red o el tráfico de paquetes. La entidad de software que implementa el controlador SDN a menudo se denomina Sistema Operativo de Red (NOS, *Network Operating System*), generalmente, un NOS se puede implementar de forma independiente [11] [12].

Capa de Infraestructura

Está conformada por elementos de red, los cuales son gestionados por medio del protocolo *OpenFlow*, permitiendo simplificar la configuración al administrador de red. La capa de infraestructura incluye un entorno para el reenvío de tráfico de datos ya sea en hardware virtual o real. El plano de datos comprende una red de elementos de red, que exponen sus capacidades a través del SBI hasta el plano de control. SDN desacopla las funciones de control, como algoritmos de reenvío y mueve estas funciones de control fuera de la infraestructura a un nodo lógico central controlado, al hacerlo, los elementos de red actúan sólo como interruptores que actúan sobre las instrucciones del controlador [11] [12].

1.2 Redes Ópticas

1.2.1 Redes Ópticas de Transporte

Aquellas redes de datos que permiten transportar, conmutar, direccionar y verificar información mediante el uso de enlaces de fibra óptica se conocen como Redes Ópticas de Transporte (OTN, *Optical Transport Network*) [4] [13].

La transmisión de múltiples canales ópticos mediante el uso de una misma fibra ha proporcionado una forma simple de extender la capacidad de las OTN. La Multiplexación de canales se puede realizar en el dominio del tiempo o de la frecuencia gracias a la Multiplexación por división de tiempo (TDM, *Time Division Multiplexing*) y Multiplexación por división de frecuencia (FDM, *Frequency Division Multiplexing*), estas técnicas pueden utilizarse también en el dominio eléctrico, por lo cual para realizar una distinción en el dominio óptico es común referirse a estas técnicas como Multiplexación Óptica por División de Tiempo (OTDM, *Optical Time Division Multiplexing*) y Multiplexacion por Division de Longitud de Onda (WDM, *Wavelength Division Multiplexing*) respectivamente [14].

1.2.2 Terminales de línea ópticos

Su función principal es finalizar una conexión punto a punto para multiplexar o demultiplexar las longitudes de onda que conforman los caminos ópticos. Los multiplexores o combinadores combinan las señales ópticas con distintas longitudes de onda, de manera tal que les permite a todas ellas pasar a través de una sola fibra óptica sin interferirse entre sí. Los demultiplexores o divisores separan las señales de distintas longitudes de onda, de forma similar a como los filtros separan las señales eléctricas de distintas frecuencias [15] [16].

1.2.3 Multiplexores de inserción/extracción ópticos

Conocidos como (OADM, *Optical Add/Drop Multiplexers*) se ubican en puntos intermedios del sistema. Son dispositivos que separan una longitud de onda de un cable de fibra, y la pasan a otra fibra que va en dirección distinta. Una vez quitada una longitud de onda, se puede reemplazar por una nueva señal con la misma. Básicamente, los multiplexores de agregar y quitar se usan para reconfigurar cables de fibra óptica [15] [17].

1.2.4 Conectores ópticos

Los Conectores ópticos o cross-conectores (OXC, *Optical Cross-Connectors*) conmutan una longitud de onda desde cualquiera de sus puertos de entrada a cualquiera de sus puertos de salida, esta conmutación puede incluir o no conversión de longitud de onda [16] [18].

1.2.5 Transpondedor de Ancho de Banda Variable

Gracias al uso de distintos formatos de modulación según la arquitectura de red, este dispositivo tiene la capacidad de adaptarse a diferentes velocidades de datos; de esta manera logra hacer un buen uso del espectro, el Transpondedor de Ancho de Banda Variable (BVT, *Bandwidth Variable Transponder)* genera una señal óptica conocida como super-canal solamente haciendo uso de los recursos espectrales necesarios, acorde a la velocidad de datos del usuario. Algunas características importantes de este dispositivo son [5] [19] [20]:

1. Ajuste de numero de subportadoras: el control del número de subportadoras se puede realizar tanto en dominios ópticos como digitales, dependiendo del método de síntesis de señales para los sistemas OFDM, el BVT consta de una fuente óptica variable de múltiples portadoras y un modulador óptico de múltiples portadoras. Al ajustar la luz con diferentes frecuencias de oscilación, es posible controlar el número de subportadoras. En la figura 1.3 se observan las tasas de cada subportadora, correspondientes a un ajuste de granularidad.

2. Modulación adaptativa: Con el ajuste del formato de modulación se obtiene diferentes tazas de datos.

3. Generación de señal: los canales OFDM pueden ser asignados juntos en un supercanal, consiguiendo más capacidad que un único canal OFDM, el cual está limitado por la capacidad máxima de subportadoras del BVT. El flujo de datos se divide en varios canales que posteriormente son modulados en caminos ópticos OFDM sin banda de guarda entre sí, de esta forma ocupa menos recursos espectrales que utilizando una multiplexación WDM.

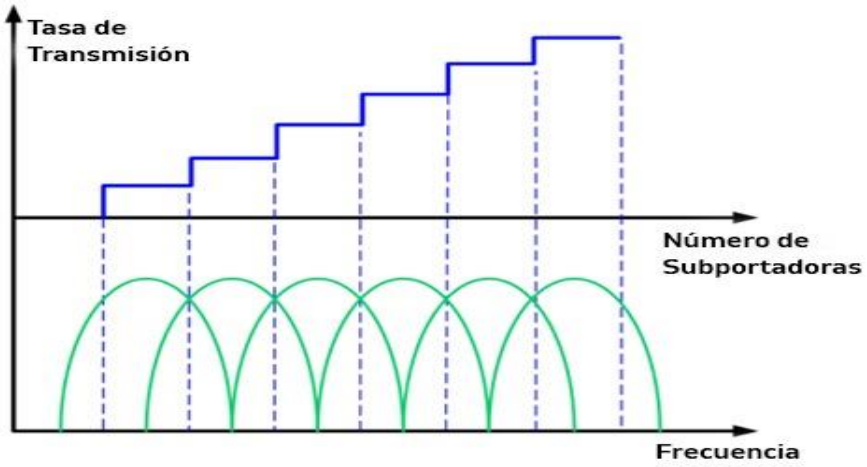

Figura 1.3 variación de la taza de transmisión a través de un ajuste del número de subportadoras. Tomada de [20]

1.2.6 Conmutador Óptico Cruzado de Ancho de Banda Variable

El Conmutador Óptico Cruzado de Ancho de Banda Variable (BV-WXC, *Bandwidth variable-wavelength cross connect switches*) se encarga de asignar una conexión cruzada que tenga un ancho de banda que permita crear una ruta óptica de un nodo a otro, esto lo logra haciendo una configuración flexible de la ventana de comunicación en función del ancho de banda de la señal óptica que ingresa, así logra aumentar o disminuir el ancho de banda de la conexión óptica elástica. Como se observa en la figura 1.4 este conmutador está diseñado para agrupar granularidades de conmutación cercanas que se adaptan de manera flexible al canal [19][20].

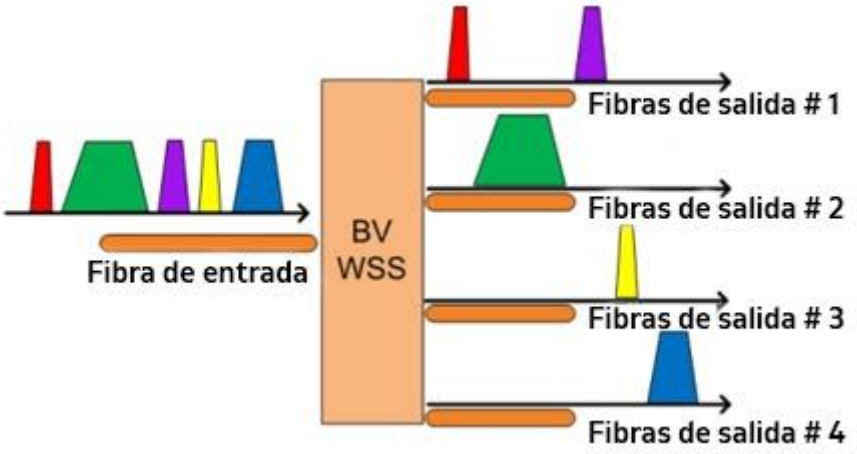

Figura 1.4 Concepto de BV-WXC. Tomada de [20].

Para soportar un camino óptico elástico de inicio a fin, todo BV-WXC a lo largo de dicho camino debe asignar una conexión cruzada de tamaño apropiado que corresponde al ancho de banda del espectro. Para ello debe configurar la ventana de conmutación de forma flexible de acuerdo con el ancho espectral de la señal o solicitud recibida. Los BV-WXC con ancho de banda variable basados en cristal líquido (MEMS, *Micro-Electro Mechanical System*) pueden ser implementados como elementos de conmutación para realizar conexiones cruzadas con un ancho de banda variable y frecuencia central. Sus principales tipos y características son [19] [21]:

1. LCoS-Based BV-WXC: Es una tecnología de visualización que combina cristal líquido y tecnología de semiconductores para crear un mecanismo de estado sólido y alta resolución, estos componentes son utilizados para controlar la fase del haz de luz en cada píxel y así producir una rejilla programable.

2. MEMS-Based BV-WXC: Esta basado en un grado de difracción libre de espacio combinado con un vector lineal de pendiente simple, así múltiples rejillas de 13,2 GHz son combinadas para formar un conmutador que permita una variación de ancho espectral y de ubicación de los canales seleccionados.

3. Banda de guarda y características: En teoría una señal con espectro continúo basado en OFDM no tiene necesidad de una banda de guarda en el dominio de la frecuencia entre canales OFDM. Por lo tanto, cuando una señal OFDM es transmitida por diferentes BV-WX, las subportadoras de los bordes del espectro sufren una perdida mayor debido a imperfecciones en estos dispositivos. Con la inclusión de una banda de guarda entre caminos ópticos adyacentes este problema puede ser reducido, pero a costo de disminuir la eficiencia espectral.

4. Arquitectura nodal de BV-WXC: A través de los BV-WXC mencionados anteriormente, este puede ser construido utilizando una arquitectura de difusión y selección. En la figura 1.5 se puede observar esta arquitectura, donde la estructura de un BV-WXC debe soportar las siguientes características:
   -*Colorless*: se refiere a retirar y agregar servicios para todas las longitudes de onda.
   -*Directionless*: hace referencia a la inclusión y exclusión de servicios para cualquier dirección.
   -*Contentionless*: Se pueden retirar o adherir servicios de una misma longitud de onda en diferentes direcciones.

Como se observa en la figura 1.5 la arquitectura de difusión y selección las señales de entrada son transmitidas para todas las salidas de los canales adecuados seleccionados utilizando el BV-WXC. Las señales de entrada son demultiplexadas por un BV-WXC y después enrutadas para diferentes puertos de salida utilizando conexiones cruzadas ópticas.

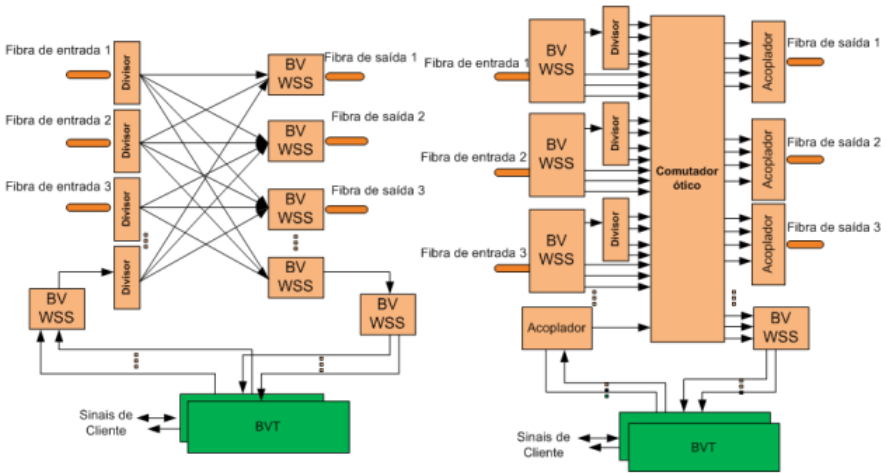

Figura 1.5 Arquitectura de un BV-WXC. Tomada de [20].

1.3 Redes Ópticas Elásticas o Flexgrid

Las Redes Ópticas Elásticas (EON, *Elastic Optical Networks*) o *Flexgrid* son redes adaptativas, flexibles y eficientes en el espectro basadas en OFDM, lo cual proporciona una alternativa a la técnica de modulación de portadora única ya que el flujo de datos se divide y es multiplexado en múltiples subportadoras consecutivas de baja velocidad y, por lo tanto, aumenta la duración del símbolo y proporciona una velocidad de datos más alta**;** las EON pueden proporcionar una solución a largo plazo para manejar el tráfico de datos que aumenta exponencialmente de manera eficiente y económica. En la figura 1.6 se puede observar la arquitectura de una Red Óptica *Flexgrid,* en donde el cliente se conecta a la red a través de un transpondedor de ancho de banda variable que realiza la conversión del dominio eléctrico al dominio óptico para poder llevar la conexión por fibra gracias a el conmutador de ancho de banda variable que encamina las solicitudes a través de la red [7] [22]

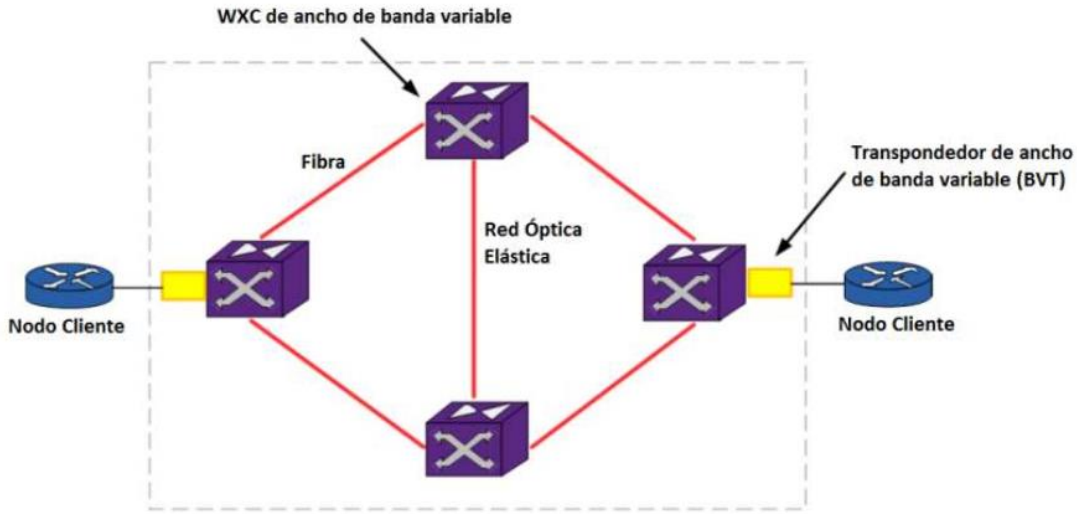

Figura 1.6 Arquitectura de una EON. Tomada de [21].

Las principales razones para el desarrollo del paradigma de las redes ópticas elásticas son lograr superar la barrera de los 400 Gbps, 1 Tbps y la demanda de tasas más altas; la separación entre canales reducida, lograr una mejor eficiencia espectral y obtener una red dinámica. La red óptica elástica se compone de conmutadores ópticos (WXCs) en la red central, y transpondedores de ancho de banda variable (BVT) en el borde de la red basados en Multiplexación por División de Frecuencias Ortogonales (OFDM, *Orthogonal Frequency Division Multiplexing*) [19] [22].

La implementación de redes ópticas elásticas basadas en OFDM puede generar muchos beneficios, tales como:

- Alta eficiencia espectral mediante la asignación de espectro flexible que varía de acuerdo con la velocidad de datos.

- Soporte y agregación de servicios con espectro flexible, lo que permite adaptarse a diferentes velocidades de datos.
- Con la posibilidad de ajustar el formato de modulación y el número de subportadoras, es posible tener un alcance diferente, de acuerdo con las necesidades de cada demanda (tasa y distancia) y modularlas de acuerdo con la velocidad de transmisión y la cantidad de recursos que requieran ser asignados.
- Consumo de energía eficiente, debido a la posibilidad de "apagar" subportadoras que no son necesarias [19] [22].

Para obtener una red EON la rejilla de espectro debe ser una rejilla “flexible” establecida por la ITU-T en el estándar G.694.1, en la cual el espectro óptico de la ventana C (1530-1565 nm) es dividido en pequeñas ranuras de frecuencia de 6.25 GHz cada una, dos segmentos conforman un slot de 12.5 GHz al que se le asigna una Frecuencia Central (CF, *Central Frequency*), que debe coincidir con el principio o el final de estos segmentos [27]. Por lo que la CF tomando como referencia la frecuencia de193.1 THz está dada por la ecuación 1.1 [19] [20].

$$CF = 193.1\ [THz] + n * 0.00625\ [THz] \qquad (1.1)$$

Donde n es entero positivo o negativo (incluido el 0), el cual tiene un valor mínimo de -246 (índice de la frecuencia correspondiente a 1565 nm) y máximo de 455 (índice de la frecuencia correspondiente a 1530 nm), que representa el índice de la frecuencia central de cada segmento [19] [20].

Aquí debe tenerse en cuenta el ancho de ranura (SW, *Slot Width*), que es igual al número de segmentos o intervalos de frecuencia (FSs, *Frequency Slots*) asignados a la conexión, multiplicado por 12.5 (tamaño de un slot en GHz). Por lo anterior, el ancho de ranura se puede expresar mediante la ecuación 1.2, donde m es un número entero positivo, y representa el número de slots de frecuencia necesarios para cumplir con el ancho de banda requerido [19] [20].

$$SW = 12.5\ [GHz] * m \qquad (1.2)$$

Se puede observar de la figura 1.7, que el ancho de banda disponible se divide en slots de 12.5 GHz. Por ejemplo, para lograr conseguir el ancho de banda de 0.0375 THz (conexión 1) y 0.05 THz (Conexión 2) se requieren 3 y 4 slots de frecuencia respectivamente. Por lo anterior, con los conceptos de (frecuencia central y ancho de ranura), es posible definir un canal en función de m y n.

De esta forma un canal puede definirse por su frecuencia central y su respectivo ancho de banda (diferencia entre su frecuencia final e inicial), como lo muestra la ecuación 1.3, en fi es la frecuencia inicial del canal en THz.

$$FC = fi + 0.00625 * m \quad (1.3)$$

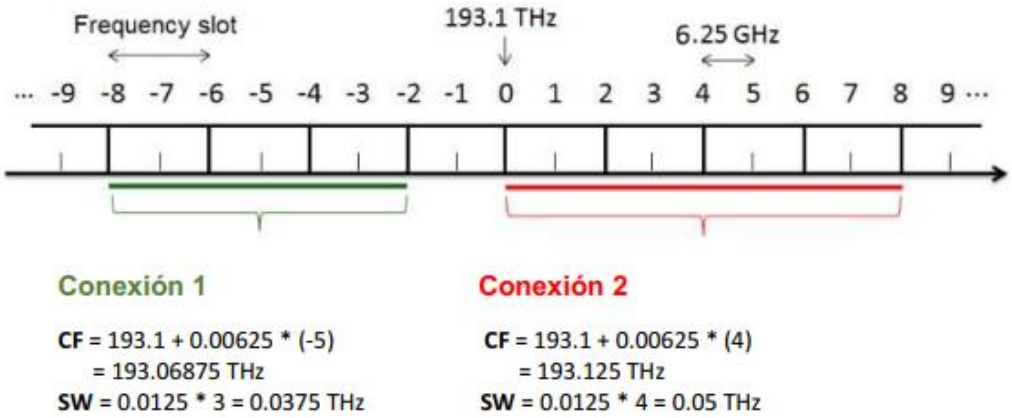

Figura 1.7 Concepto de ancho de ranura. Tomada de [19}.

Las EON generalmente hacen uso de diferentes técnicas de multiplexado, una de las más comunes es el Acceso Múltiple por División de Frecuencia Ortogonal (OFDM, *Orthogonal Frequency-Division Multiple Access)*

OFDM

El acceso múltiple por división de frecuencias ortogonales permite convertir un flujo de datos de alta velocidad en varios flujos de datos de menor tasa de bits, para ello divide una portadora de gran capacidad en varias subportadoras de menor capacidad, logrando así asignar a una conexión la cantidad de subportadoras necesarias para satisfacer la demanda de ancho de banda de la conexión.

En la figura 1.8 se puede observar la característica clave de OFDM, la ortogonalidad de sus portadoras. Esta propiedad permite que las señales se solapen unas con otras, sin causar interferencia entre sí, pues la máxima potencia de cada señal subportadora se encuentra en los mínimos de potencia de las demás señales, permitiendo así acomodar muchas más señales en una menor cantidad del espectro al eliminar la necesidad de bandas de guarda entre señales. [19] [20]

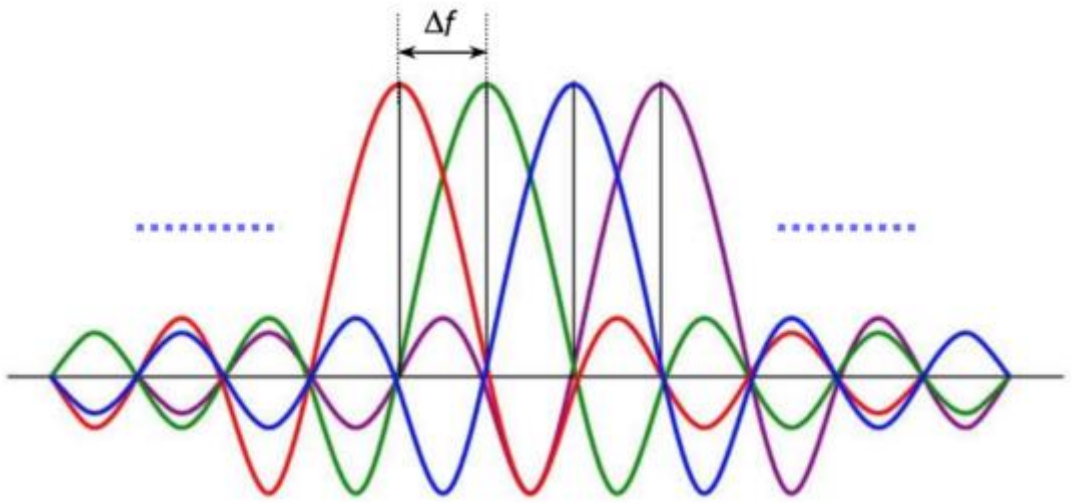

Figura 1.8 Señal OFDM en el dominio de la frecuencia. Tomada de [19].

1.4 Redes Ópticas Definidas por Software

En los últimos años se ha avanzado mucho en la codificación óptica y en os formatos de modulación. A pesar de esos avances, hay un gran desequilibrio entre el crecimiento en la capacidad de transmisión y los ingresos en las redes, para enfrentar este desafío se ha propuesto una extensión de SDN y *OpenFlow* a redes ópticas, lo cual hace que se considere como una tecnología prometedora para el plano de control en las redes IP de conmutación de paquetes, así como la conmutación de longitud de onda en redes ópticas. Los beneficios de aplicar SDN y el estándar de este protocolo en particular a las redes de transporte ópticas incluyen: flexibilidad de gestión y control de la red de transporte, despliegue de sistemas de gestión y control de terceros, e implementar nuevos servicios aprovechando la virtualización y SDN [22] [23] .

Las nuevas tendencias de aplicaciones están impulsando las redes tanto de paquetes como de circuitos en redes ópticas para volverse más flexibles y dinámicas. La red óptica dinámica habilitada por SDN se conoce como red óptica definida por software (SDON, *Software Defined Optical Networking*). Una SDON emplea multiplexores ópticos reconfigurables de adición eliminación (ROADM, *Reconfigurable Optical Add/Drop Multiplexor*) para cambiar dinámicamente de un extremo a otro, circuitos de longitud de onda que utilizan un plano de control óptico. Puede ser entendido como la extensión de SDN a la capa óptica [23] [24] .

Para aprovechar al máximo la agilidad del hardware que ofrecen a las redes los ROADM, los operadores de red deben tener la capacidad de reconfigurar de manera eficiente, confiable y dinámica tanto la longitud de onda como la ruta de los canales ópticos a través de la red, mediante el uso del software de gestión sin ninguna

intervención manual. La implementación de esta capacidad necesita multiplexación y demultiplexación sin dirección, que permiten que la longitud de onda y la ruta que utiliza el canal para entrar / salir de un nodo terminal sean aprovisionados de forma flexible dentro del plano de control de gestión y sin ninguna intervención física. Para ello, se necesita una nueva cartera de componentes ópticos, como conmutadores con longitud de onda seleccionable (WSS, *Wavelength Selectable Switches*), monitores de canal óptico, interruptores multidifusión, arreglos de amplificador óptico (OA, *Optical Amplifier*), ancho de banda variable (BV) y BV-ROADM. La figura 1.9 presenta un modelo aproximado de la arquitectura de una SDON, donde se puede observar cómo los diferentes clientes de tecnologías ópticas se conectan a través de la interfaz SDN para así hacer la gestión de sus recursos y solicitudes de una forma centralizada, esto gracias a la extensión de SDN a redes ópticas [24] [25].

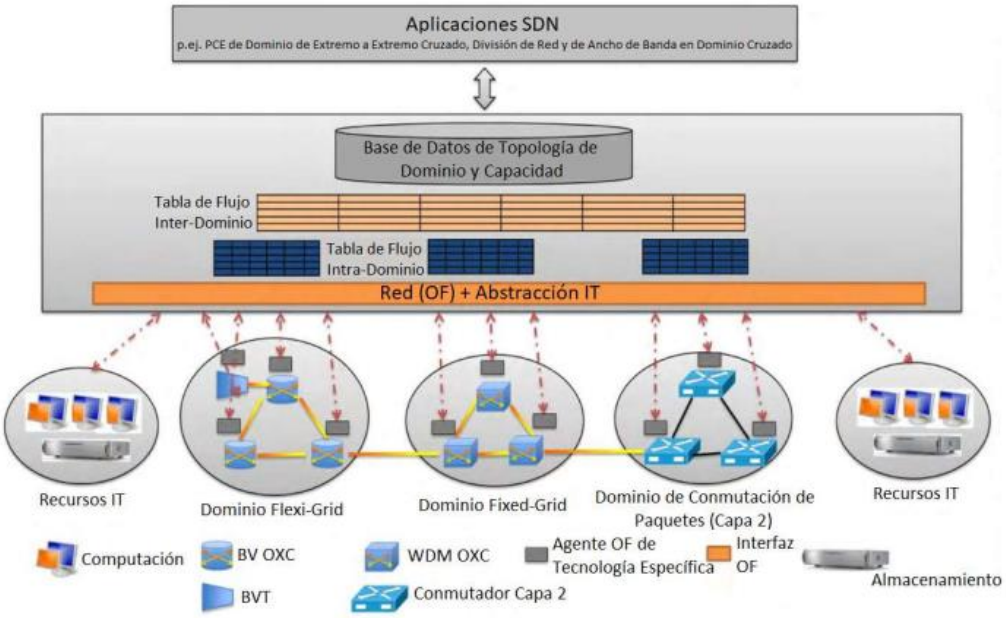

Figura 1.9 Arquitectura SDON. Tomada de [25].

SDON hace uso de los siguientes componentes de red:

- Multiplexor Óptico Reconfigurable de Extracción e Inserción (ROADM, *Reconfigurable Optical Add-Drop Multiplexer*): Mediante el uso de conmutadores ópticos selectivos de longitud de onda, un ROADM puede extraer o insertar, ya sea uno o múltiples canales de longitud de onda que transportan señales de datos ópticos sin requerir la conversión de la señal óptica en señales eléctricas [26].

- Conmutador de Transporte abierto (OTS, *Open Transport Switch*): es un conmutador virtual óptico habilitado para OpenFlow. Su función principal es abstraer la capa de conmutación física subyacente a un conmutador virtual. Se compone de tres módulos: detección, control y plano de datos, los cuales son controlados por un controlador SDN a través de mensajes para interactuar con el hardware de conmutación física [27]].

- Conmutador lógico xBar (Lx, *Logical xBar*): conmutador programable que se pueden combinar para formar un único xBar grande con una única tabla de reenvío [27].
- Caja blanca óptica (OWB, *Optical White Box*): Conmutador que combina un plano programable con elementos programables de un nodo [28].

- Conmutador virtual de Red Óptica Pasiva con Capacidad de Gigabit (GPON, *Gigabit-capable Passive Optical Network*) (GPON *Virtual Switch*): conmutador en el cual la tecnología GPON es totalmente programable el cual abstrae todo el GPON en un conmutador OpenFlow virtual [29].

- Nodo de red de acceso flexible (FAN, *Flexi Access Network Node*): Este nodo tiene capacidades de transmisión eléctrica y óptica que son controladas a través de SDN, de forma tal que el tráfico que está destinado a otros nodos en el área de cobertura de una red de acceso puede ser enviado directamente a la red de acceso [29].

La SDON a implementar contará con una arquitectura similar a la presentada en la figura 1.6, para ello se trabaja con una topología de la fundación nacional de redes para la ciencia (NSFneT, *National Science Foundation's Network)* de 14 nodos, cada uno de los cuales tendrá en su interior dos dispositivos, un BVT y un BV-WXC, los cuales se encargan de generar las peticiones de la red y de recibir las indicaciones del nodo controlador para establecer la ruta y ancho de espectro. Se diseñará un nodo controlador, con el fin de llevar a cabo el proceso de centralización, el cual debe tener la capacidad de gestionar los recursos, tales como establecimiento de ruta y cantidad de espectro para todas las peticiones que se generen dentro de la red.

En la tabla 1.1 se presentan las características de la red que se pretende implementar:

Tabla 1.1 Características de la red.

RED	SDON FLEXIGRID
TOPOLOGIA	NSFNet
BORDE EN LA RED	BVT
RED CENTRAL	BV-WXC

CAPÍTULO 2

ESPECTRO ÓPTICO Y COGNICIÓN EN LAS REDES

2.1 Enrutamiento y Asignación de Espectro

El enrutamiento y asignación de espectro (RSA, *Routing and Spectrum Allocation*) se utiliza para encontrar la ruta adecuada para una conexión desde su origen hasta su destino, y asignar las ranuras de espectro adecuadas a la solicitud. RSA trata el enrutamiento y asignación de espectro para ahorrar recursos espectrales, logrando una mejor operación de la red óptica, en donde el requisito de capacidad de cada solicitud de conexión es caracterizado por una serie de ranuras de subportadoras [8] [30].

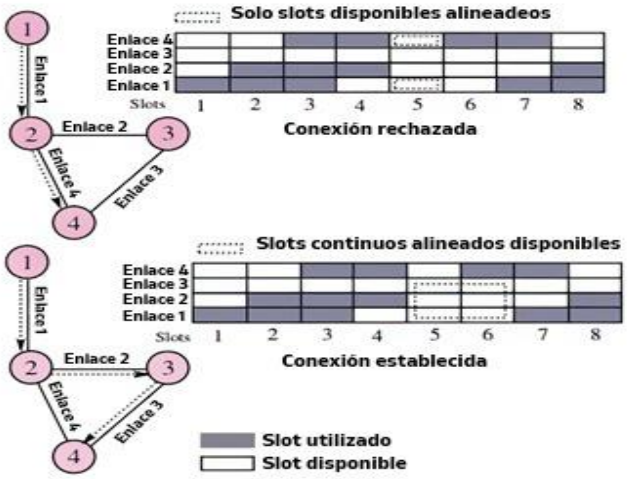

Figura 2.1 Continuidad y contiguidad del espectro.Tomada de [30].

El problema de establecer trayectos de luz para cada conexión o solicitud, seleccionando una ruta adecuada y asignando a ella la longitud de onda requerida se conoce como enrutamiento y asignación de longitud de onda (RWA, *Routing and Wavelength Allocation*). El problema de RSA en redes ópticas elásticas es equivalente al problema de RWA en redes ópticas basadas en WDM. Como se observa en la figura 2.1 en RSA, se asigna un conjunto de ranuras de espectro contiguas a una conexión en lugar de la longitud de onda establecida por RWA en redes basadas en WDM de rejilla fija, estas ranuras de espectro asignadas deben colocarse cerca una de la otra para satisfacer la restricción de contigüidad del espectro, en el caso de que no haya suficientes ranuras contiguas disponibles a lo largo de la ruta, la conexión se

puede dividir en múltiples pequeñas demandas, cada una de estas demandas menores requerirá un número menor de ranuras de subportadoras contiguas [30].

El objetivo de RSA es encontrar una ruta con suficiente espectro libre para brindar el ancho de banda requerido por las demandas de tráfico. La asignación de espectro (SA, *Spectrum Allocation*) de una conexión óptica consiste en encontrar un determinado canal que debe cumplir con las limitaciones de contigüidad y continuidad del espectro; para ello, todas las ranuras de un canal deben estar cerca de la otra (restricción de contigüidad) mientras que el canal asignado debe usar la misma frecuencia central (CF, *Central Frequency*) para todos los enlaces que conforman esa conexión óptica (restricción de continuidad) [31].

RSA puede tener un enfoque de un solo paso o un enfoque de dos pasos. En el enfoque de un solo paso, los algoritmos definen la ruta y el espectro contiguo disponible simultáneamente, por otro lado, se hace un enfoque de dos pasos, el enrutamiento y la asignación de espectro se resuelven secuencialmente. Como presenta la figura 2.1 RSA debe considerar las siguientes condiciones para operar [30]:

- **Contigüidad del espectro:** todos los slots de frecuencia del canal en una conexión óptica, deben ser adyacentes en el espectro.
- **Continuidad del espectro:** en una conexión óptica para un enlace óptico desde el nodo origen al destino se debe asignar el mismo canal, es decir usar la misma frecuencia central.
- **No superposición del Espectro:** un slot de frecuencia en un enlace sólo se puede utilizar para una conexión a la vez.
- **Espectro limitado:** hay un número finito de slots en cada enlace.
- **Bandas de guarda:** es necesario separar las conexiones espectralmente adyacentes en cada fibra, por al menos un slot no asignado para evitar interferencias [31].

En la figura 2.2 se visualiza el modelo de una rejilla de espectro óptico en RSA, la cual está formada por su frecuencia central y un numero de slots.

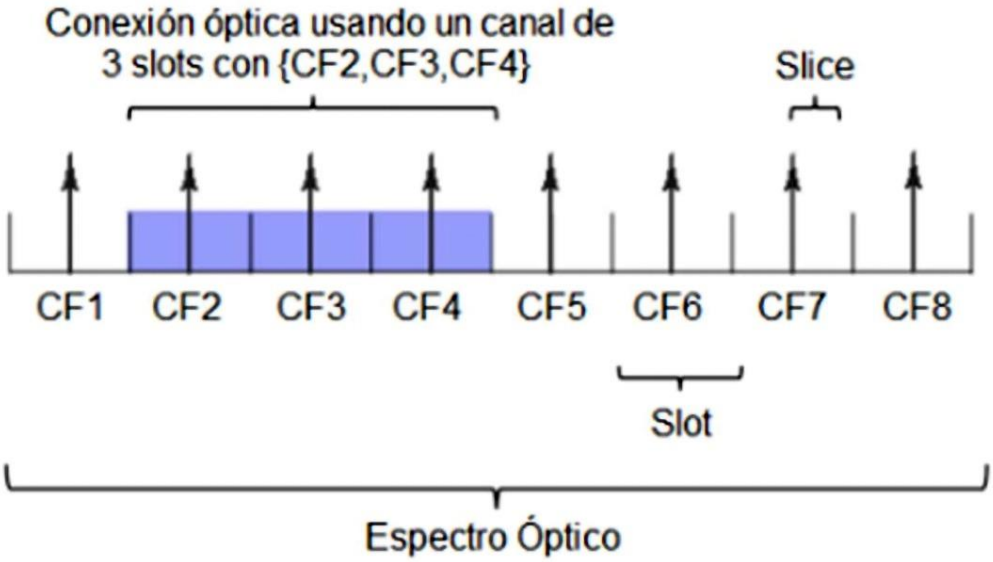

Figura 2.2 Rejilla de espectro óptico en RSA. Tomada de [31].

De acuerdo con los cambios permitidos al espectro asignado en los enlaces, se definen 3 esquemas para RSA [31].

2.1.1 Esquema de asignación de espectro fijo en el tiempo

Como presenta la figura 2.3 este es un esquema de asignación en donde no hay elasticidad, por lo cual bajo este esquema tanto la frecuencia central como el ancho de espectro asignado permanecen estáticos todo el tiempo, esto genera que la asignación de espectro sea independiente de las variaciones de los requisitos de ancho de banda. Se pueden encontrar dos condiciones:

a) Cuando el ancho de banda requerido es menor que la capacidad del espectro asignado, el espectro utilizado para transportar tráfico es, por tanto, más bajo que el asignado, lo que lleva a un uso deficiente de la capacidad de la red.

b) Cuando el ancho de banda requerido es mayor que la capacidad asignada de espectro, parte del ancho de banda es desperdiciado [31].

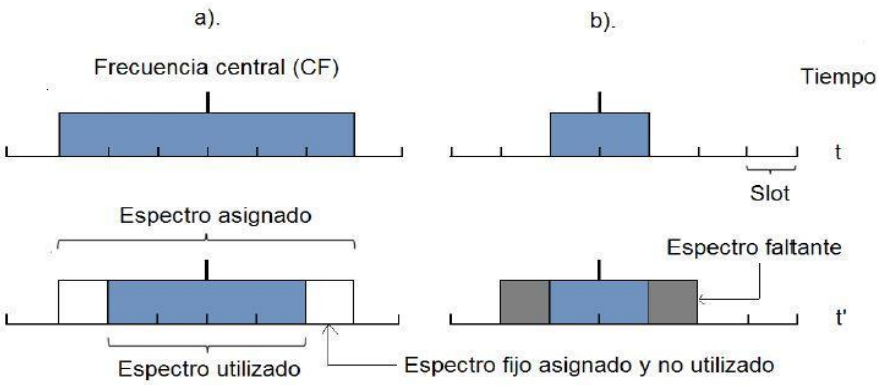

Figura 2.3 Asignación de espectro fija en el tiempo. Tomada de [31].

2.1.2 Esquema de asignación de espectro semi-elástico

La frecuencia central permanece fija, pero el ancho del espectro asignado puede variar. Los cortes de frecuencia asignados a un camino de luz intentan ajustarse al ancho de banda requerido en cualquier momento. Como consecuencia de la elasticidad de este esquema, los cortes de frecuencia pueden ser compartidos por trayectos de luz vecinos, pero cada segmento solo se puede asignar para máximo un camino de luz. Se puede hacer distinción de 2 escenarios:

a) Si hay una reducción del ancho de banda requerido, la capacidad del espectro asignado se reducirá y los cortes no necesarios en cada extremo del espectro asignado se liberarán y estarán disponibles para otros caminos si son necesarios.
b) Si hay un aumento del ancho de banda requerido, nuevos segmentos contiguos pueden ser asignados al final de ambos lados de la frecuencia central, la capacidad del espectro asignado intentará aumentar para servir al máximo del ancho de banda requerido. En la figura 2.4 se aprecia el funcionamiento de este esquema [31].

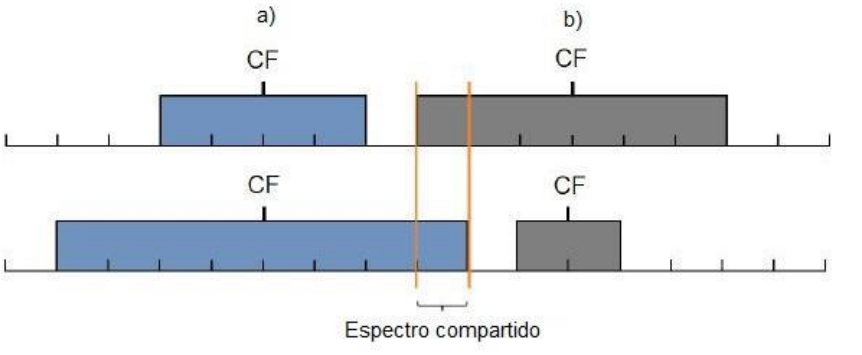

Figura 2.4 Asignación semi-elastica en el tiempo. Tomada de [31].

2.1.3 Esquema de asignación de espectro elástico en el tiempo

Este esquema agrega al esquema anterior un nuevo grado de libertad: no solo se permite variar la cantidad de ranuras por camino de luz en cualquier momento, sino también cambiar la frecuencia central [31]. Para este esquema se pueden analizar dos situaciones:

a) Elástico con expansión/contracción de espectro: En este caso, como se observa en la figura 2.5 los movimientos de la frecuencia central están limitados a un cierto rango; por lo tanto, la reasignación de espectro está restringida a las frecuencias centrales vecinas.

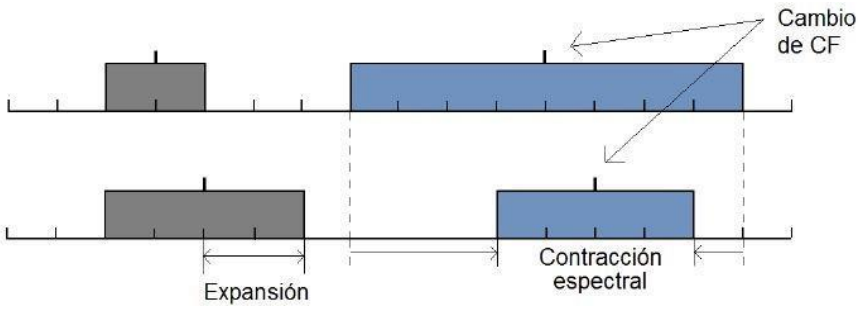

Figura 2.5 Asignación elástica de espectro con expansión/contracción. Tomada de [31].

b) Elástico con reasignación de espectro: Como se presenta en la figura 2.6 el espectro se puede reasignar completamente y no hay limitación de movimiento de la frecuencia central.

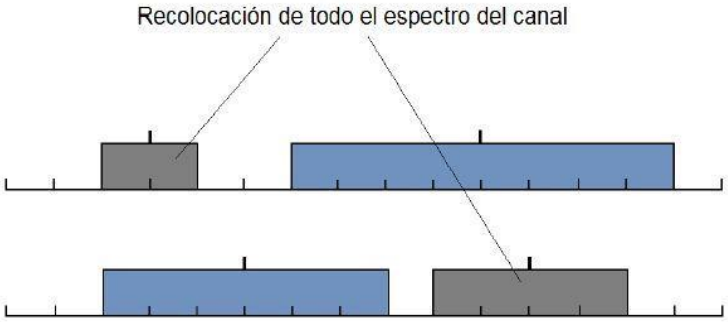

Figura 2.6 Asignación elástica con reasignación de espectro. Tomada de [31].

2.2 Algoritmos RSA

Generalmente RSA se divide en dos partes, por un lado, la operación de enrutamiento (routing), cálculo de la ruta entre el nodo origen y el destino a través de la topología de la red; y por otra, la selección de los recursos espectrales que se asignan a la petición, Asignación de Espectro (SA, *Spectrum Allocation*), definidos por una frecuencia central y un ancho banda o de slot (SW, *Slot Width*). Algunos algoritmos RSA están diseñados para ejecutar ambas operaciones simultáneamente [32] .

Se pueden clasificar los algoritmos de planificación RSA de acuerdo con la manera como llegan las peticiones en dos grandes grupos: algoritmos estáticos (offline) y algoritmos dinámicos (online).

2.2.1 Algoritmos RSA Estáticos (*Offline*)

Se recibe una matriz de demandas de canales con capacidades variables. El procesado de las peticiones se realiza como un conjunto. El cálculo de la solución óptima a este problema se presenta como un problema *NP-hard* y se computa mediante una programación lineal entera (ILP, *Integer Lineal Programming*). El ILP es un método de programación matemático para la resolución de problemas complejos y la obtención de soluciones enteras óptimas [32] [33].

Un ILP se describe como un conjunto de variables, una o varias funciones objetivo, que suelen ser la maximización o minimización de alguna variable, en el caso de las redes elásticas puede ser maximizar el número de slots consecutivos dejados libres tras una petición, minimizar el número de saltos de una ruta, o minimizar la posición en el espectro del último slot asignado a una petición. Y, por último, un número de restricciones o condiciones que definen el problema, el problema de RSA se transforma en uno de enrutamiento y asignación de canal, en la que la asignación de canal implica la asignación de espectro contiguo, y hay que tener en cuenta directamente las limitaciones de continuidad del espectro [32] [33].

2.2.2 Algoritmos RSA Dinámicos (*Online*)

El caso dinámico u online, consiste en servir las peticiones dinámicamente según van llegando. Las peticiones de caminos se van ejecutando según llegan, es decir, la red se va ocupando conforme las peticiones se van atendiendo. Es un problema más complejo debido a la llegada / salida de tráfico aleatorio y la fluctuación de las demandas de tráfico a lo largo del tiempo. Dependiendo del estado de la red, los recursos espectrales disponibles pueden o no ser suficientes para establecer una conexión. El estado de la red consta de los enlaces físicos y la asignación de espectro para todas las conexiones activas, por lo cual a medida que la red evoluciona, es posible que un algoritmo de enrutamiento óptimo actual ya no proporcione una utilización espectral óptima a lo largo del tiempo. Por lo tanto, cada vez que una nueva solicitud de conexión llega, se debe ejecutar un algoritmo en tiempo real para determinar si es factible acomodar la nueva solicitud de conexión y, de ser así, realizar el RSA. Si no se puede atender una nueva solicitud de conexión, se bloquea; por tanto, la probabilidad de bloqueo de las solicitudes de conexión surge como el objetivo clave en un algoritmo RSA [32] [33].

Para solucionar el problema de RSA dinámico los estudios proponen algoritmos heurísticos que requieren menos tiempo de computación. Dichos algoritmos pueden clasificarse en algoritmos de una y dos etapas.

- **Algoritmos de una etapa**

Resuelven el problema RSA simultáneamente, pero de manera poco eficiente, las operaciones de *routing* y asignación de recursos espectrales se realizan a la vez [32] [33].

Algunos ejemplos de este tipo de algoritmos son:

a) *Modified Dijkstra 's Shortest Path* (MSP): Con el fin de comprobar si hay ancho de banda disponible para una solicitud entrante, se realiza una intersección del espectro en los enlaces de la red [34] [33].

b) *Spectrum-Constraint Path Vector Searching* (SCPVS): genera un árbol de búsqueda de ruta vectorial (PVST, *Vector Path Searching Tree*), en el que las rutas son almacenadas con el espectro agregado y se realiza una comprobación en cada nivel del árbol [34] [33].

- **Algoritmos de dos etapas**

El problema RSA divide en dos partes: enrutado y asignación del espectro, primero se calculan k "*shortest paths*" utilizando el algoritmo de Dijkstra u otro algoritmo equivalente, y a continuación se utiliza uno de los siguientes algoritmos para la asignación del espectro:

a) *Lowest Starting Slot:* Asigna la primera secuencia de slots disponibles que cumplan los requisitos de nuestro canal.

b) *Void Filling*: Selecciona la ruta con los menos huecos consecutivos posibles. Esto sirve para que la fragmentación en el espectro sea la menor posible.

c) *First Fit:* Asigna la primera secuencia de slots disponibles que encuentra al canal. [32] [33].

2.2.3 Teoría de grafos

Un grafo es un conjunto de vértices y un conjunto de aristas que conectan pares de vértices distintos con máximo un borde conectando cualquier par de vértices. La teoría de grafos es una rama importante de las matemáticas combinatorias que se ha estudiado intensamente durante cientos de años, un grafo con v vértices tiene como máximo v(v-1) /2 aristas [35].

Una ruta en un grafo es una secuencia de vértices en la que cada vértice sucesivo (después del primero) es adyacente a su predecesor en el camino. En un camino simple, los vértices y los bordes son distintos. Un ciclo es un camino que es simple excepto que el primer y último vértice son los mismos [35].

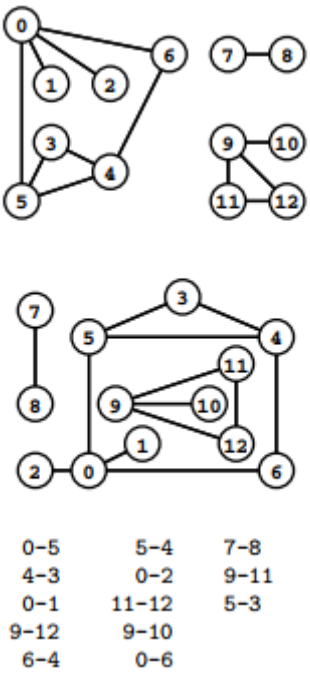

Figura 2.7 Representación de un Grafo. Tomada de [35].

Como se observa en la figura 2.7 un grafo se define por sus vértices y sus bordes, no por la forma en que se elige dibujarlo. Estos dibujos representan el mismo grafo, al igual que la lista de bordes (abajo), da la información adicional que el grafo tiene 13 vértices etiquetados del 0 al 12 [35].

- **Representación de listas de adyacencia**

En la figura 2.8 se muestra una representación del grafo de la figura 2.7 como una serie de listas enlazadas. El espacio utilizado es proporcional al número de nodos más el número de aristas. Para encontrar los índices de los vértices conectados a un vértice dado v, se ubica la posición v en una matriz, que contiene un puntero enlazado a una lista que contiene un nodo para cada vértice conectado a v. El orden en que aparecen los nodos en las listas depende del método que usamos para construir las listas [35].

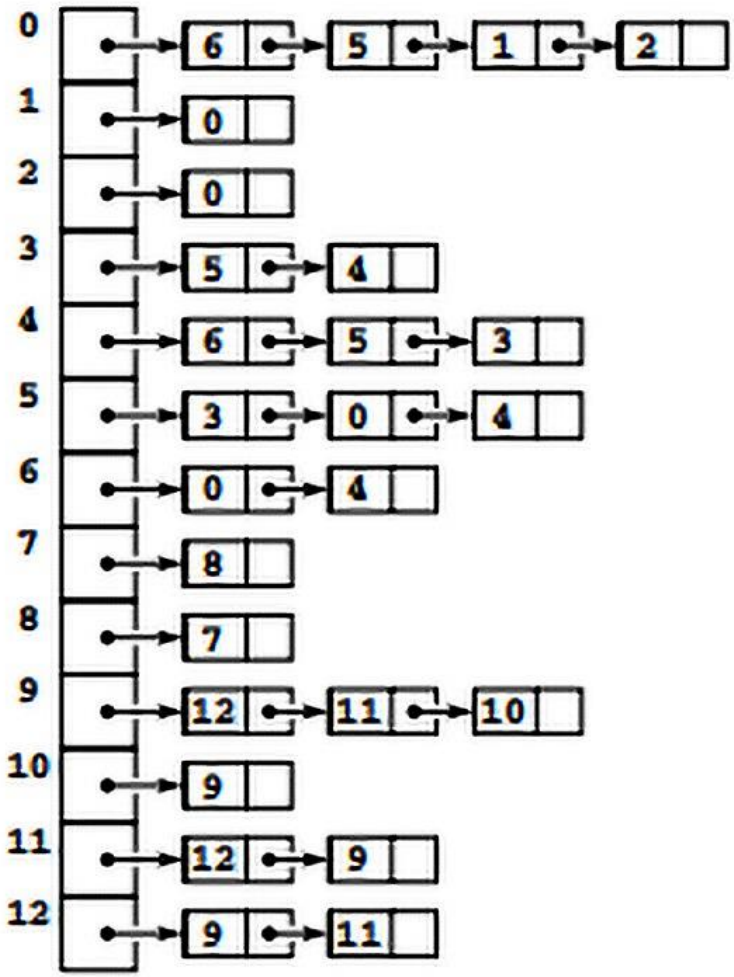

Figura 2.8 Lista de adyacencia de un grafo. Tomada de [35].

2.3 Aprendizaje Automático

En la actualidad con la era del *big data*, solo por mencionar, hay aproximadamente 1 billón de páginas web, se sube una hora de video a YouTube cada segundo, lo que equivale a 10 años de contenido cada día. Este diluvio de datos requiere métodos automatizados de análisis de datos, que es lo que el Aprendizaje Automático (ML, *Machine Learning)* proporciona.

Siendo un subcampo de la inteligencia artificial, en particular, el ML se define como un conjunto de métodos que pueden detectar automáticamente patrones en los datos y luego usa los patrones descubiertos para predecir los datos futuros, o para realizar otro tipo de toma de decisiones en condiciones de incertidumbre, como planificar y recopilar datos. También es aplicable al campo de las telecomunicaciones, gracias a la capacidad de aprender acerca del estado de las redes que, junto con la

cognitividad, hacen posible la “autogestión”. Es un subcampo de la informática, que se ha definido como ‘campo de estudio que da a los ordenadores la capacidad de aprender sin ser programados explícitamente’. Mediante el estudio de la construcción de algoritmos se pueden detectar patrones en grandes cantidades de datos que difícilmente serían captadas por los humanos. Con estos algoritmos, puede hacer predicciones e inferencias sobre el comportamiento de conjuntos de datos ya dados. [6] [36] [37].

El proceso de aprendizaje que utiliza ML podría describirse como un lazo. Los pasos son los siguientes:

•Observación: se identifican patrones en los datos

•Planificación: encuentra todas las posibles soluciones

•Optimización: encuentra las soluciones más convenientes de la lista de posibles soluciones.

•Actuación: se ejecuta la solución anteriormente encontrada.

• Aprendizaje y adaptación: si el resultado es similar al esperado, finaliza el proceso. En caso contrario, se adapta de nuevo, volviendo al principio de los pasos.

2.3.1 Tipos de Machine Learning

- **Aprendizaje supervisado**

  Usado en problemas de clasificación y regresión, requiere entrenamiento previo. Este tipo de ML Permite la búsqueda de patrones en campos de datos históricos para relacionarlos con un campo objetivo. Estos sistemas utilizan datos cuya salida es conocida, es decir, al algoritmo de aprendizaje automático se le proporciona un conjunto de datos de entrada cuya variable de salida asociada es conocida. El objetivo del algoritmo es aprender patrones en los datos y construir un conjunto general de reglas [38] [39].

- **Aprendizaje no supervisado**

  Se encarga de buscar la estructura que corresponde a un grupo de campos de datos con el objetivo de ordenarlos. Requiere entrenamiento previo. No hay etiquetas para los ejemplos u observaciones, el problema fundamental consiste en encontrar la estructura subyacente de un conjunto de datos presentes. Puede requerir entrenamiento previo antes de usar el modelo [38] [39].

El aprendizaje no supervisado se utiliza cuando se desconoce la clase de salida deseada para los datos. Este tipo de algoritmo carece de un conocimiento previo. El objetivo es estudiar los patrones del conjunto de comportamiento de datos de entrada para identificar patrones y comportamientos similares que se puedan agrupar [38] [39].

- **Aprendizaje activo (con refuerzo)**

  Se premia a un agente con un refuerzo positivo cuando realiza la acción adecuada o se le castiga sin dar refuerzo cuando se equivoca. Generalmente se da en un contexto de un agente actuando en un ambiente. No se le dice al agente explícitamente si se equivoca o no. El agente es castigado o premiado (no necesariamente de manera inmediata), el problema fundamental es definir una política que permita maximizar el estímulo positivo (premio) [38] [39].

  Su objetivo es que el algoritmo aprenda por su propia experiencia. Se determinan qué acciones se deben escoger para maximizar una 'recompensa' final. Para la toma de decisiones no sólo se debe considerar la recompensa inmediata, sino la posterior y todas las siguientes [38] [39].

- **Aprendizaje en línea**

  Su objetivo principal es la extracción de la mayor cantidad de información útil con el menor número de interacciones al actuar sobre un gran volumen de datos en tiempo real. Puede ser supervisado o no [38] [39].

De acuerdo con la forma en cómo se enfrenta el problema el ML se puede clasificar en:

- **Aprendizaje por clasificación y regresión**

  Su objetivo es establecer un método para la relación entre un cierto número de valores y una variable objetivo o de salida continua. El ejemplo de regresión más común es la regresión lineal, en el cual se define una recta para proporcionar la tendencia de un conjunto de datos [38] [39].
  Asigna un caso de prueba para uno de un conjunto finito de clases, mientras que la regresión permite predecir el caso empleando variables continuas o atributos. En cuanto al diagnóstico de redes, un problema de clasificación se resume en decidir si la falla en una conexión se debe a que el sitio tomado como objetivo se ha caído. Por analogía, el problema de regresión puede radicar en predecir el tiempo que toma la conexión en reintegrarse a su estado óptimo [36] [40].

- **Aprendizaje para actuar**

  Un agente se encarga de ejecutar planes de actuación que han sido previamente seleccionados, se puede llevar un registro de acciones previas para que en una nueva ejecución futura al agente le tome un tiempo menor realizarla [36] [40].

- **Aprendizaje para interpretación y entendimiento**

  Se identifican instancias dentro de una clase gracias al entendimiento y utilizando modelos profundos se pueden interpretar datos en el ambiente. De esta forma se logra interpretar situaciones y eventos [36][39].

- **Regresión lineal**

  La Regresión Lineal es una técnica ampliamente utilizada en el campo del Aprendizaje Supervisado debido a su sencillez y a su gran utilidad. Consiste en predecir una variable dependiente y en función de una o varias variables independientes x, mediante el trazado de la línea recta que mejor se ajusta al conjunto de datos proporcionados. La Ecuación 2.1 representa la hipótesis de este modelo. [40]

$$h(x) = b0 + b1x1 + b2x2 + b3x3 \quad (2.1)$$

2.4 Redes Cognitivas

Una red cognitiva es una red con un proceso que puede percibir sus condiciones actuales, y luego planificar, decidir y actuar sobre esas condiciones. La red puede aprender de estas adaptaciones y utilizarlas para tomar decisiones futuras. La red debe comprender lo que la aplicación está tratando de lograr, y una aplicación debe poder comprender lo que la red es capaz de hacer en cualquier momento, esto permitiría a una red hacer uso de nuevas capacidades aprendiendo los requisitos de la aplicación y eligiendo dinámicamente el protocolo de red que cumplirá con estos requisitos. Una red cognitiva tiene la opción de implementar un proceso cognitivo centralizado, un proceso cognitivo distribuido, o un proceso parcialmente cognitivo distribuido [41] [42].

Las redes cognitivas deben ser capaces de aprender y planificar y para ello se necesita un mayor sentido de autoconciencia. Se puede decir que el funcionamiento de la red cognitiva es el punto final de evolución natural de una red. Para ello es muy importante entender lo que es el ciclo cognitivo y cómo aplicarlo a la red y lograr que esta sea cognitiva. En la figura 2.7 se puede visibilizar un esquema del ciclo cognitivo en redes, donde la información pasa por las diferentes etapas de este, para después llevar a cabo el procesamiento [42] [43].

- **Proceso cognitivo (método cognitivo)**

Es considerado el corazón de una red cognitiva y consiste en un algoritmo responsable de realizar funciones de razonar, aprender y planear para lo cual se soporta en una diversidad de mecanismos tomados de la inteligencia artificial. Entonces, un método o proceso cognitivos puede ser "implantado" en el nodo de una red y a través de los elementos cognitivos que posea y la posibilidad de realizar cambios tanto estructurales como de funcionamiento en dicho nodo, pueda contribuir a lograr algún tipo de mejora en el desempeño no solo del nodo sino de toda la red.

La figura 2.9 muestra mediante un diagrama de transición de estados las fases del ciclo cognitivo.

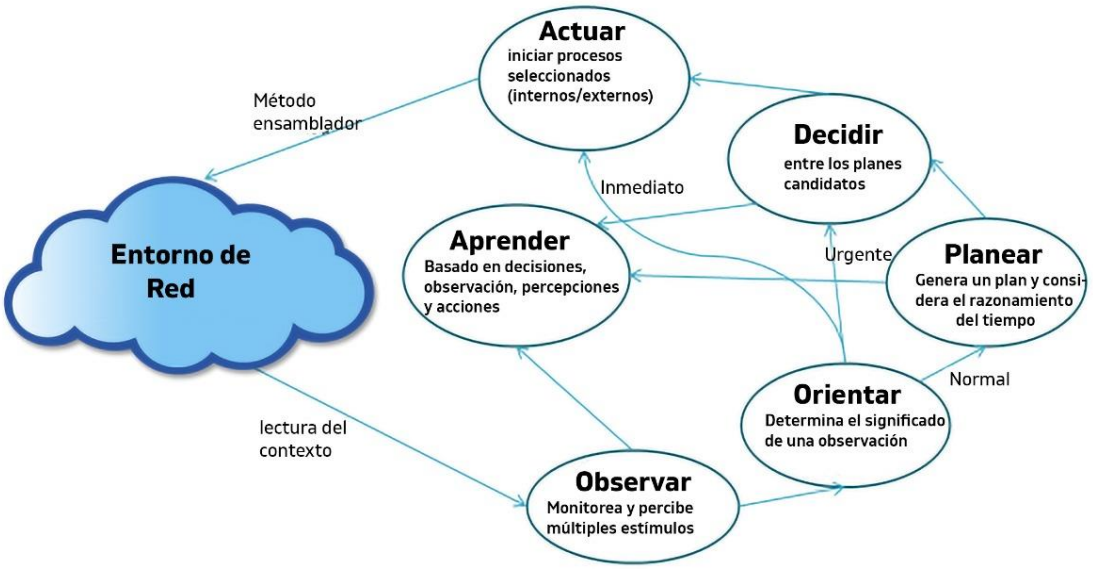

Figura 2.9 Ciclo Cognitivo. Tomada de [43].

Como se observa en la figura 2.9, el ciclo cognitivo consta de unas partes esenciales dentro de las redes de comunicaciones que se describen a continuación:

Observación: Recolectar información acerca del estado de la red.

Orientación o análisis: Evaluar la importancia de la información recolectada en el proceso anterior**.**

Decisión: Escoger la mejor alternativa para actuar teniendo en cuenta la información recopilada.

Planeación: Plantear una línea de acción a largo plazo de acuerdo con la información recopilada.

Acción: Después del proceso de orientación, decisión o planeación, el algoritmo puede pasar a esta fase, donde realiza los ajustes necesarios a la red, de acuerdo con lo aprendido durante el ciclo.

Aprendizaje: Se evalúa el desempeño de la red y se evalúa su cambio respecto a la fase anterior del ciclo, tomando las mejores decisiones en base a la experiencia [49].

Para poder llamar “cognitiva” a una red, debe cumplir con los procesos de observación, análisis (orientar), decisión y acción. La figura 2.10 modela estos procesos cognitivos [43].

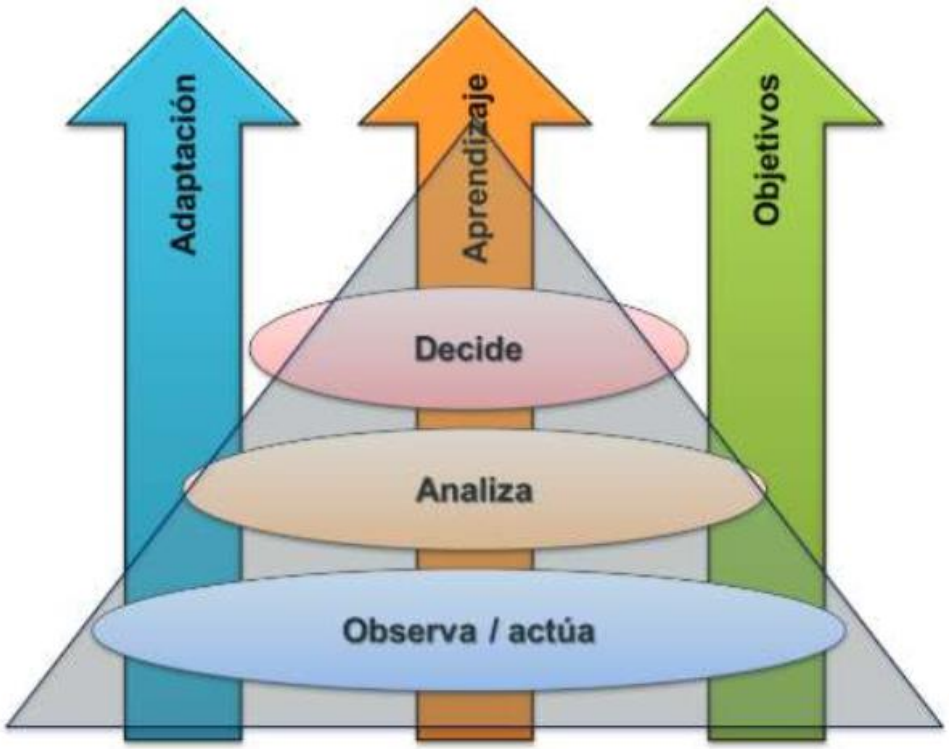

Figura 2.10 Proceso Cognitivo. Tomada de [43].

La figura 2.10 representa las funcionalidades para activar las propiedades cognitivas en una red mostrando estas funcionalidades como elementos de una pirámide en cuya base se encuentran las funciones distribuidas a través de la red (elementos de observación que pueden incluir desde medida de características físicas hasta variables del nivel de aplicación), de otro lado los elementos funcionales que tienen a su cargo la ejecución de acciones pueden llevar a cabo operaciones simples como el ajuste de parámetros para las diferentes interfaces de red. Estas funcionalidades son descritas a continuación:

Adaptación: Debe responder a los cambios en el entorno de la red.

Objetivos: Conjunto de metas a alcanzar mediante la aplicación del aprendizaje y la optimización cognitiva en la red.

Aprendizaje: Es el núcleo del sistema cognitivo y está encargado de habilitar a los demás elementos de la red, para llevar a cabo acciones basadas en la experiencia previa o acciones almacenadas previamente en el sistema que pueden ayudar a tomar decisiones futuras [43]

Un ciclo cognitivo descrito en la literatura tomada como referencia para este trabajo de grado es el ciclo ODAA, el cual se detalla a continuación:

- **Ciclo OODA**

Uno de los primeros modelos de ciclos cognitivos, es el conocido como ciclo OODA (*Observe, Orient, Decide and Act*) mostrado en la figura 2.11, aunque este modelo omite el proceso de aprendizaje en sí, ha sido adaptado y aceptado como punto de partida para una serie de estudios e investigación de donde surgen diferentes estructuras que involucran el concepto de lo cognitivo en distintos entornos [42] [43].

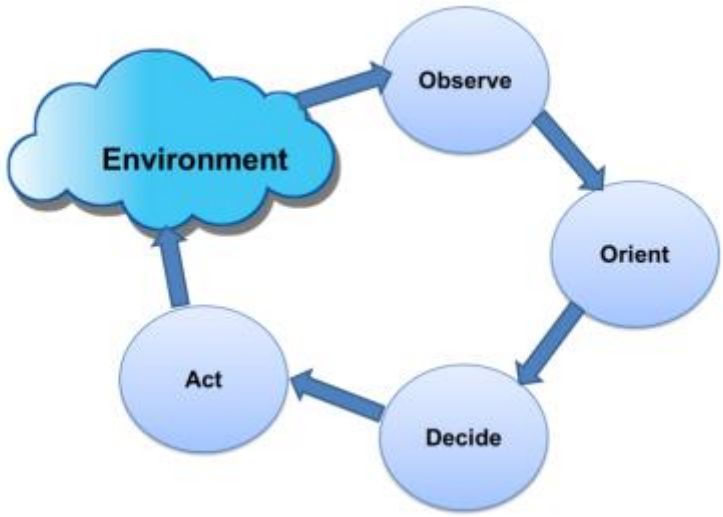

Figura 2.11 Modelo Cognitivo OODA. Tomada de [43].

El modelo OODA fue definido para entrenamiento militar en la fuerza aérea de los Estados Unidos, propuesto por el coronel John Boyd, la importancia de este modelo es que coloca elementos esenciales en estados bien definidos. Estos estados son: **Observación**, corresponde a la recolección de datos, **Orientación**, como resultado del análisis y síntesis de los datos se construye una perspectiva mental del entorno real, **Decisión**, corresponde a la determinación del curso de una acción basado en una perspectiva mental de la realidad y **Acción**, que corresponde a la ejecución física de las decisiones.

Tanto el Ciclo Cognitivo como el ciclo OODA resultan esenciales en el desarrollo del concepto de cognición aplicable a redes de comunicaciones, de ellos se puede extraer aspectos de comportamiento y entrenamiento propios de los seres vivos y emplearlos para construir motores de cognición que impulsen el desarrollo de modelos desempeño en diferentes sistemas de redes de comunicaciones [43].

Para la asignación de espectro en la EON SDON se utilizará un algoritmo ML de tipo regresión lineal, el cual será dinámico y tratará el problema RSA en dos etapas, debido a que se considera necesario mostrar más de una sola ruta entre los nodos, se utiliza la teoría de grafos para implementar el algoritmo de enrutamiento, además de utilizar vectores y colas para conseguirlo. En el método cognitivo se implementa el ciclo OODA ya que contiene las partes necesarias del ciclo cognitivo que se buscan implementar en la propuesta del trabajo de grado, de esta forma se logra cumplir con los requerimientos del proceso cognitivo y aplicarlo a redes, además de utilizar la técnica de regresión lineal de ML y obtener así una asignación de espectro basada en ML.

CAPÍTULO 3

METODOLOGÍA

En este capítulo se presentan las metodologías empleadas para el desarrollo del trabajo de grado, primero se expone la metodología utilizada en el trabajo de grado, definida en el anteproyecto y en segunda instancia la metodología empleada en la simulación.

3.1 Metodología del trabajo

Para desarrollar este trabajo de investigación se ha escogido el uso de la metodología de trabajo conocida como lineal secuencial o denominada también como modelo en cascada presentado en la figura 3.1, la cual es ampliamente utilizada en la academia y en el desarrollo de soluciones software.

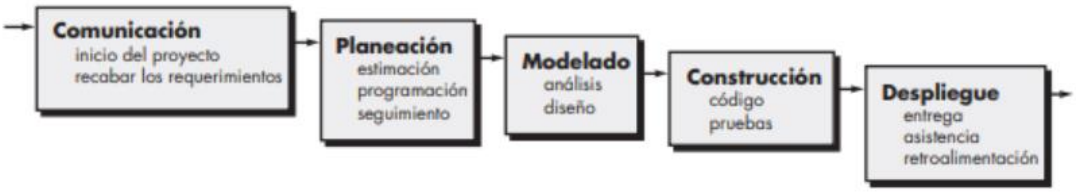

Figura 3.1 Modelo en cascada. Tomada de [44].

Este modelo está conformado por cinco fases, las cuales se van implementando a medida que se van terminando los requerimientos para cada una de ellas, lo cual brinda un avance claro y ordenado. A continuación, se detallan cada una de sus fases, las cuales brindan la posibilidad de desarrollar el trabajo de grado de manera adecuada: [45]

- **Fase de recopilación de información:** El objetivo principal es discernir los requisitos finales del proyecto para poder proponer de forma concisa todos los requerimientos funcionales del mismo.

- **Fase de planeación:** El proyecto es fraccionado en partes más pequeñas, las cuales puedan ser desarrolladas de forma independiente.

- **Fase de diseño o modelado:** Aquí es diseñado el entorno en donde se va a implementar la simulación y todos los algoritmos que nos brindan la posibilidad de dar una solución al problema planteado en el trabajo de grado.

- **Fase de construcción:** Aquí son ejecutados los algoritmos en el entorno para posteriormente realizar pruebas de desempeño y recolectar los datos necesarios.

- **Fase de despliegue:** En esta fase se realiza un análisis detallado de los resultados obtenidos, para hacer una retroalimentación y poder corregir los posibles errores y así dar por terminado el proyecto de grado y efectuar su entrega.

3.2 Metodología de simulación

Para realizar el estudio de la evaluación del desempeño de los modelos de red se acude a la información incluida en [45] donde se sigue una metodología de enfoque practico, se muestra como una evaluación de desempeño de un modelo de red soportado en simulación contiene una serie de pasos como se muestra en la figura 3.1, aunque esta serie de pasos puede considerarse como una metodología de desarrollo de la simulación a seguir, no es de estricto cumplimiento y es susceptible de ajustes.

Como siguiente medida se desarrollan las fases propuestas dentro de la metodología que se muestra en la figura 3.1, para la obtención de los modelos de desempeño y simulación. A continuación, se detallan las fases:

1. Definición de los escenarios de prueba: Con base a los propósitos de estudio de evaluación y acorde con los objetivos planteados en el trabajo de grado, teniendo en cuenta aspectos teóricos que se investigan en la literatura, se asumen límites y restricciones, además se selecciona las métricas que servirán para realizar la evaluación, se busca realizar un análisis del desempeño de una red SDON flexigrid implementando un método cognitivo para asignar espectro basado en ML.

2. Obtención del modelo de red de prueba: Este proceso se plantea para cada escenario propuesto, con elementos teóricos y conceptuales que ya están definidos en capítulos anteriores y contando con una arquitectura base propuesta, se plantea el modelo de la red de prueba, es una parte muy importante ya que un buen diseño conduce a crear un ambiente que se adecue a los requerimientos hechos y a la pregunta de investigación que se desea resolver.

3. Diseño del método cognitivo: Con base a la literatura investigada y teniendo ya un modelo de red propuesto se puede concebir la siguiente fase que sería diseñar el método cognitivo que realizara la asignación de espectro dentro del modelo de red de prueba y analizar su desempeño, para lo que se debe realizar una comparación con otro algoritmo o método convencional que emplean las EON.

4. Prueba de la red sin método cognitivo: Se ejecuta la simulación para observar el comportamiento de la red bajo características propias de la EON y posteriormente con estos elementos, realizar una comparación con el método propuesto en el trabajo de grado.

5. Prueba de la red con método cognitivo: Una vez se tiene el modelo de desempeño convencional de la red y el método diseñado, se realiza el análisis bajo las características que tiene este último y realizar un contraste y así obtener resultados que permitan analizar el comportamiento de la red bajo estas condiciones.

6. Comparación de los resultados: Después de ejecutar las diferentes simulaciones se realiza una validación y análisis de los resultados obtenidos en los dos escenarios propuestos, se puede realizar una comparativa de las salidas obtenidas escenarios y haciendo un contraste de estas.

7. Conclusiones: Con base a los resultados obtenidos y al contraste entre los dos escenarios se pueden definir las conclusiones que derivan de las pruebas bajo las condiciones ya establecidas.

En la figura 3.2 se puede observar el esquema de la metodología propuesta con sus diferentes fases.

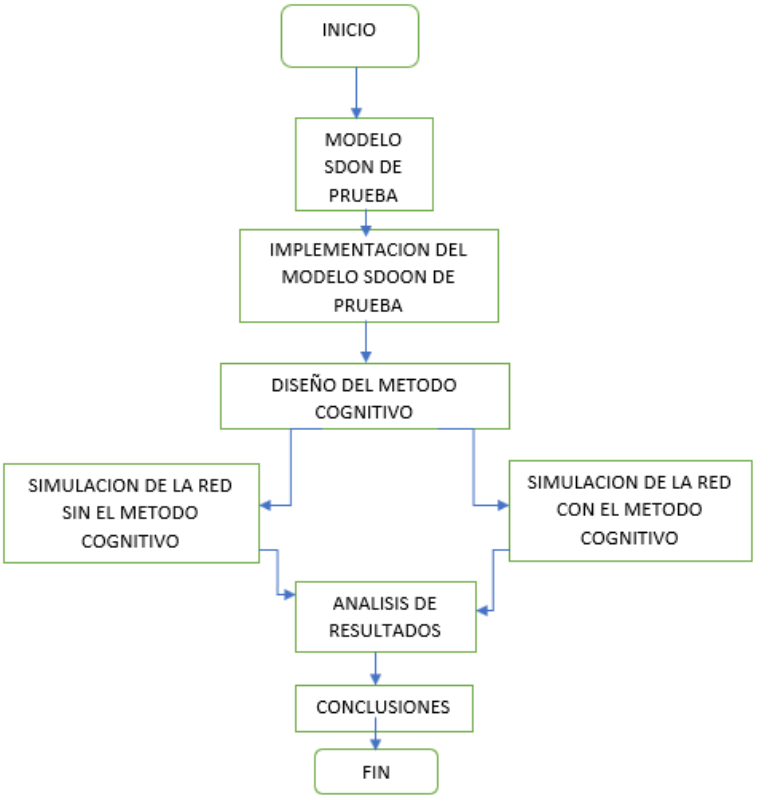

Figura 3.2 Diagrama de la metodología de simulación.

3.3 Herramientas de simulación

Para la implementación del modelo de simulación se exploraron diferentes herramientas consecuentes con la propuesta y los objetivos del trabajo de grado, a continuación, se describen algunas de las más relevantes y la que finalmente se seleccionó para llevar a cabo el proyecto.

- **Simulador ns (Network Simulator)**

Es una herramienta disponible en múltiples plataformas que ofrece soporte para la simulación de todo tipo de redes tanto cableadas como inalámbricas. Se trata de uno de los simuladores de redes más ampliamente utilizado entre la comunidad docente e investigadora del área de redes de computadores [46] .

Se suministra con el código fuente completo. Consta de un núcleo principal escrito en C + + al que se invoca simplemente tecleando ns en la línea de comandos (una vez este ha sido correctamente instalado). A partir de este punto el usuario puede interactuar directamente con el simulador, a través de un lenguaje de interfaz. El lenguaje de interfaz es OTcl, una versión de Tcl 2 orientada a objetos [46]. En la figura 3.3 se observa el logo del simulador.

Figura 3.3 Logo NS-2. Tomada de [46].

- **OMNET++**

El entorno de desarrollo integrado OMNET ++ se basa en la plataforma Eclipse y lo amplía con nuevos editores, vistas, asistentes y funciones adicionales. OMNeT ++ agrega funcionalidad para crear y configurar modelos (archivos NED e ini), realizar ejecuciones por lotes y analizar los resultados de la simulación, mientras que Eclipse proporciona Edición de C ++, modelado UML, integración de seguimiento de errores, acceso a bases de datos, entre otras, a través de varios códigos abiertos y comerciales [47] [48].

OMNET ++ es un marco de simulación de red de eventos discretos modular orientado a objetos. Tiene una arquitectura genérica, por lo que puede (y ha sido) utilizada en varios dominios de problemas:

• modelado de redes de comunicación inalámbricas y por cable
• modelado de protocolos
• modelado de redes de colas
• modelado de multiprocesadores y otros sistemas de hardware distribuidos
• validación de arquitecturas de hardware
• evaluar aspectos de desempeño de sistemas de software complejos
•en general, modelado y simulación de cualquier sistema en el que se acerque a un evento discreto es adecuado y se puede mapear convenientemente en entidades que se comunican intercambiando mensajes [49] [50]. Se observa el logo de la herramienta en la figura 3.4.

Figura 3.4 LOGO OMNET. Tomada de [48].

- **OPNET**

Lenguaje de simulación orientado a las comunicaciones, proporciona acceso directo al código fuente, es utilizado por empresas de telecomunicaciones, gubernamentales y del ejército, posee una interfaz gráfica de alto nivel y librerías de modelos que facilitan su uso. OPNET permite la conexión de diferentes nodos de una red que a su vez se componen de módulos y distintas conexiones programadas en lenguaje C + + [51] [52].

Posee una interfaz que incluye varias librerías de modelos, el código fuente de estas librerías es accesible si se dispone de la versión OPNET Modeler que permite que el programador entienda la jerarquía interna del programa [51] [52]. La figura 3.5 muestra el logo de esta herramienta.

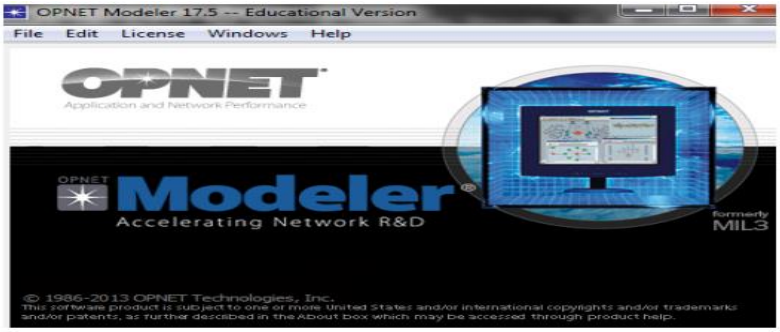

Figura 3.5 Logo OPNET. Tomada de [52].

- **MATLAB**

Herramienta informática y lenguaje de programación de alto nivel, destinado a la realización de cálculos matemáticos, especialmente operaciones de matrices. Permite la realización de cálculos complejos, creación y manejo de gráficas, creación de interfaces, interacción con otros entornos computacionales, simulación de eventos, procesamiento de señales, etc., siendo ampliamente usado en muchos campos, desde la física, química, hasta prácticamente en todas las ramas de la ingeniería [53] [54]. En la figura 3.6 se encuentra el logo de esta herramienta.

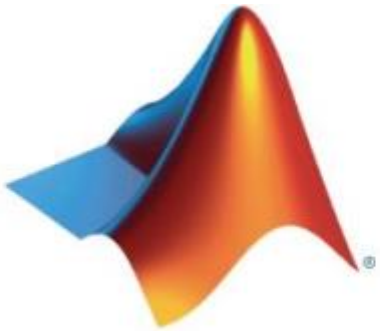

Figura 3.6 Logo MATLAB. Tomada de [53].

La programación es realizada mediante un lenguaje similar a los de alto nivel como BASIC o C, lo cual brinda al usuario la posibilidad de agrupar sentencias que utiliza con regularidad dentro de un programa invocado posteriormente, esto permite que haya una reducción de tiempo en sesiones secuenciales ya que no es necesario que se redacten todas las sentencias de nuevo [54] [55] .

- **Optsim**

Soporta el diseño y la evaluación de desempeño del nivel de transmisión de sistemas de comunicaciones ópticas. Esta herramienta que requiere el uso de una licencia para su operación se basa en la interconexión de componentes modulares, puede ser usado en conjunto con MATLAB, en lo que se conoce como entorno co-simulado. Se utiliza para la simulación de sistemas de fibra óptica. Permite la simulación a nivel de transmisión o capa uno del modelo OSI, siendo ampliamente usado por la industria y el campo educativo. OptSim (de RSOFT) es ideal para evaluar a nivel físico el desempeño de sistemas WDM, dispositivos ópticos, sistemas totalmente ópticos, etc [56] [57]. La figura 3.7 muestra la interfaz de usuario de optsim.

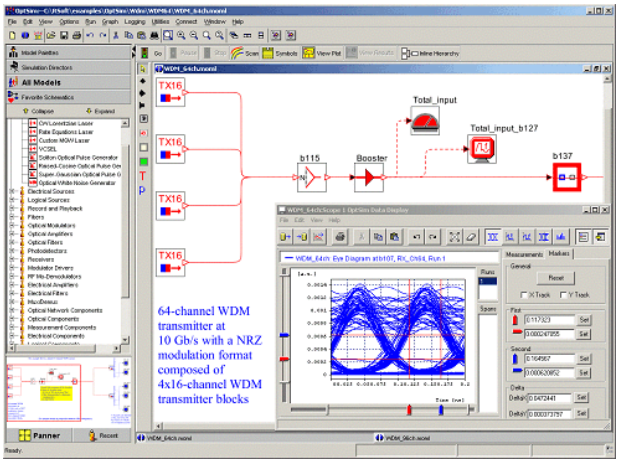

Figura 3.7 GUI OPTSIM. Tomada de [57]

- **ONOS**

ONOS es el acrónimo de *Open Networking Operating System*, un proyecto de la comunidad open source hospedado por la Linux Foundation. El objetivo es ofrecer una solución a nivel de operador por eso está diseñado para ofrecer escalabilidad, altas prestaciones y alta disponibilidad. El software está creado por *The Open Networking Lab* que junto a otros colaboradores como AT & T y NTT *Communications* lanzaron, en diciembre de 2014, la primera versión [58] [59].

El software ONOS partió de la necesidad de ofrecer un sistema robusto, tolerante a fallos que sirviese como base para futuras plataformas que puedan ser utilizadas por empresas operadores. En el momento de desarrollar ONOS las plataformas SDN de entonces estaban enfocadas a la investigación con lo que no se preocupaban de que el sistema fuera escalable y de alta disponibilidad como tampoco ofrecían una facilidad de su programación. Plataformas como NOX, Beacon, SNAC y POX fueron desarrolladas más para explorar el potencial de las redes SDN que para su uso en un entorno empresarial [58] [59]. En la figura 3.8 se muestra el logo de ONOS.

Figura 3.8 Logo ONOS. Tomada de [59].

- **Mininet**

Un emulador es un software que permite ejecutar programas sobre una plataforma diferente a la que fue originalmente diseñada, las herramientas de simulación difieren de un simulador en que este último solo reproduce el comportamiento del programa mientras que un emulador modela de forma precisa un dispositivo pudiendo ser comparado con el hardware original [60] [61].

Mininet es uno de los primeros emuladores desarrollados explícitamente para apoyar SDN, permitiendo la ejecución eficaz de redes de pequeña escala con tráfico artificial en computadoras no necesariamente potentes, su licencia es libre y permisiva (BSD – Berkeley Software Distribution). La filosofía de Mininet es crear redes virtuales, hosts corriendo kernels reales y dispositivos de red virtualizados de forma simple y rápida a través de un host anfitrión de características simples, con un ambiente abierto y libre como lo es Linux. [61] [62]. En la figura 3.9 se muestra la interfaz de usuario de mininet.

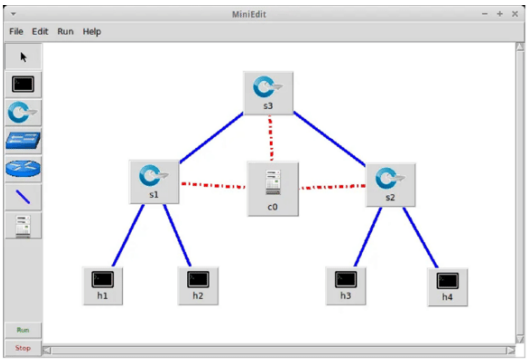

Figura 3.9 GUI MININET. Tomada de [62].

En la tabla 3.1 se realiza una comparación de las distintas herramientas de simulación exploradas. De las cuales, después de realizar una evaluación de las más relevantes, teniendo en cuenta las referencias encontradas y los trabajos realizados en ellas, relacionados con las temáticas del trabajo de grado, finalmente, después de analizarlas y compararlas, para llevar a cabo la implementación de este proyecto se selecciona la herramienta OMNET ++, debido a sus características y modularidad, que hacen de ella la más conveniente para llevar a cabo los objetivos planteados, esto también gracias a la facilidad que posee de poder realizar modificaciones en características, protocolos y algoritmos desarrollados, ya que cada componente se puede diseñar, codificar en C++ y realizar las pruebas necesarias, empezando prácticamente de cero debido a que la herramienta no cuenta con nodos predefinidos.

Tabla 3.1 Comparación Herramientas de Simulación

Característica	**Simulador Ns**	**Omnet**	**Opnet**	**Matlab**	**Mininet**	**ONOS**	**Opstim**
Licencia	gratuita	gratuita	comercial	comercial	gratuita	gratuita	comercial
Interfaz	media	media	media	alta	alta	alta	alta
Plataforma	windows, mac	windows,unix	windows	windows	Linux, ubuntu	Linux, Ubuntu	windows, unix
Heurísticas	no	Si	no	si	Si	Si	si
Graficación	regular	buena	buena	muy buena	buena	muy buena	muy buena
Soporte nivel 2 y 3 (enlace y red)	alto	Alto	alto	bajo	si	si	bajo
Uso en la academia	alto	Alto	alto	alto	alto	alto	alto

CAPITULO 4

IMPLEMENTACIÓN y SIMULACIÓN DE LA SDON FLEXGRID

En este capítulo se describe la configuración realizada para implementar la arquitectura del modelo de red propuesto, la configuración de los nodos y la topología NSFnet, de acuerdo con la metodología de simulación, la cual será evaluada bajo los parámetros establecidos sin el método y posteriormente con el método.

De acuerdo con la metodología de simulación se presentan las etapas de la implementación de la red SDON *flexigrid* y del método cognitivo de asignación de espectro óptico.

4.1 Modelo SDON de prueba

En la figura 4.1 se observa el diagrama que muestra las relaciones de uso entre módulos simples y compuestos, interfaces de módulo, redes, canales e interfaces de canal, que componen la red de prueba.

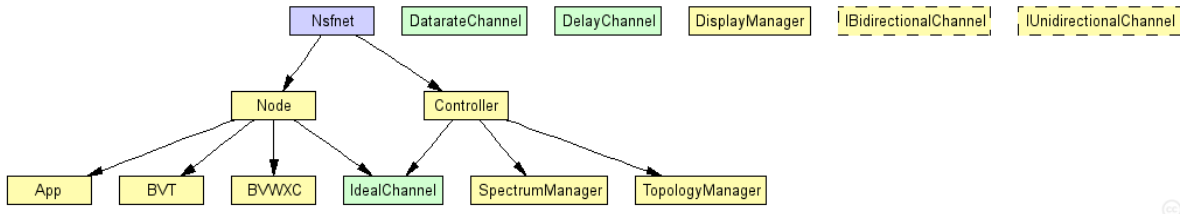

Figura 4.1 Diagrama de uso de la red de prueba. Obtenido de OMNET++

El diagrama presentado en la figura 4.2 muestra la herencia jerárquica entre módulos simples y compuestos, interfaces de módulo, redes, canales e interfaces de canal.

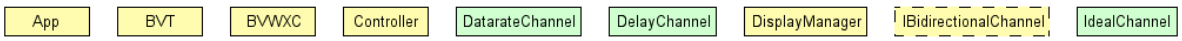

Figura 4.2 Herencia jerárquica de los módulos. Obtenido de OMNET ++

4.2 Implementación del Modelo SDON de prueba

En las figuras 4.3 y 4.4 se observa la configuración que tiene el módulo compuesto llamado *Node* o Nodo, para transmisión y recepción los cuales están formados por 3 componentes principales o módulos simples en su interior.

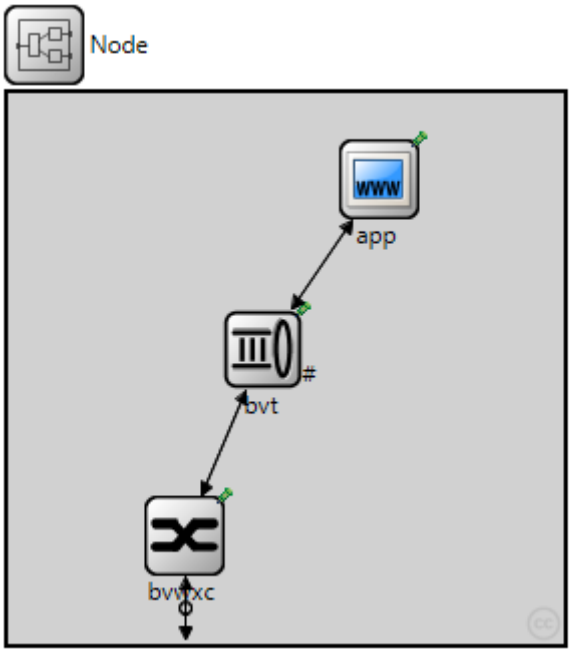

Figura 4.3 Nodo de la red de prueba SDON. Obtenido de OMNET ++

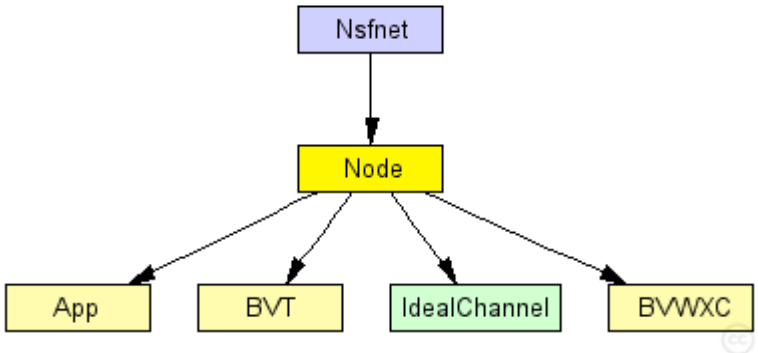

Figura 4.4 Diagrama del módulo compuesto NODE. Obtenido de OMNET ++

La red está formada por 14 de estos módulos compuestos, que desempeñan el papel de nodos en la red, internamente están formados por módulos simples que funcionan de la siguiente manera:

4.2.1 Módulo Simple APP

Se encarga de realizar las peticiones a la red, estas solicitudes llegan con un requerimiento de ancho de banda, cumple la función de usuario final por lo cual podría representar un host o muchos conectados a la red para generar solicitudes de conexión. Las peticiones se realizan siguiendo una distribución uniforme, cada una tomándola entre 100 y 200 ms. Este módulo es el encargado de generar todo el tráfico en la red.

Como muestran las figuras 4.5 y 4.6 cada petición que realiza la fuente está formada por unos parámetros y el mensaje o paquete que lleva la petición, los lleva implícitos en su interior como se evidencia a continuación.

Figura 4.5 Parámetros de petición de la app. Obtenido de OMNET ++

Se observa que la petición que genera la fuente está formada por una dirección de destino o *address*, un número de slots que se generan para la solicitud, son asignados por el módulo app y pueden ser aleatorios o fijos.

El *sendIaTime* es el tiempo que tarda en producirse nuevas peticiones, cuanto más corto sea este tiempo, las peticiones llegan con mayor frecuencia. Este es el parámetro básico de la red para controlar el tráfico.

PacketLenght hace referencia a la cantidad de información en bytes que cada paquete debe transportar.

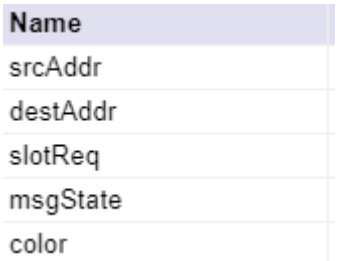

Figura 4.6 Mensaje Óptico. Obtenido de OMNET ++

Cada petición lleva en su interior un mensaje óptico que está formado por una dirección de origen, una dirección de destino, un requerimiento de slots para dicha petición y un estado del mensaje para confirmar la solicitud al controlador, puede ser la solicitud o la confirmación de que ya fue resuelta. El parámetro color sirve para graficar en la rejilla de slots y diferenciar visualmente las solicitudes.

4.2.2 Módulo Simple BVT

Este módulo simple cumple la función de pasarela o de interfaz óptico-eléctrica o eléctrica-óptica, según el caso, en transmisión o recepción. Las solicitudes son generadas en la app, en donde previamente se encuentra una interfaz eléctrica, por lo cual este componente debe transformar dicha interfaz a una tecnología óptica para permitir la conexión con el BWXC. Este módulo representa la interfaz de red para una conexión punto a punto; como muestra la figura 4.7 sus parámetros se refieren a entradas y salidas que llegan desde la fuente de peticiones o del módulo simple BWXC según sea el caso, transmisión o recepción. Las conexiones *in, out* son ópticas mientras que las conexiones *localin* y *localout* son conexiones eléctricas.

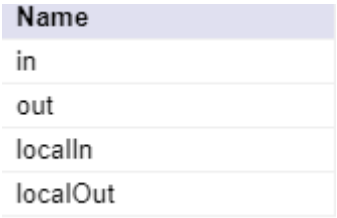

Figura 4.7 parámetros del BVT. Obtenido de OMNET ++

4.2.3 Módulo Simple BWXC

Es el conmutador o enrutador óptico que se encarga de direccionar los mensajes que ingresan a la red, está diseñado para realizar un enrutamiento dinámico dentro de la red. Funciona de acuerdo con los parámetros de la figura 4.8

La entrada *directin* es la comunicación que tiene el BVWXC con el controlador; en OMnet existen unas entradas especiales llamadas entradas directas que no requieren de una conexión física y tampoco consumen tiempo de ejecución dentro de la red, a través de esta entrada se envía el mensaje óptico, si el estado del mensaje es *LIGHTPATH_REQUEST*, es enviado al controlador por el contrario si tiene un estado *LIGHTPATH_ASSIGNMENT*, el BVWXC envía el paquete por la compuerta y al nodo que este descrito en la tabla de enrutamiento.

Las entradas in, out, son las conexiones entre enrutadores y estas pueden variar dependiendo de la capacidad de la red.

Las entradas *localin, localout* son las conexiones eléctricas con el BVT.

Name
directIn
in [ ]
out [ ]
localIn
localOut

Figura 4.8 parámetros del BV-WXC. Obtenido de OMNET ++

4.2.4 Modulo compuesto CONTROLADOR

Para obtener una red centralizada y con las características de una *SDON flexgrid* se diseñó un módulo compuesto que funciona como un controlador que se encarga de la gestión de la red y del manejo de sus recursos, en este caso ruta y espectro. Para ello se diseñaron dos componentes o módulos simples, como se aprecia en las figuras 4.9 y 4.10, que permiten el correcto funcionamiento de estas propiedades de las SDON. A continuación, se describe la función de cada componente perteneciente al controlador de la red:

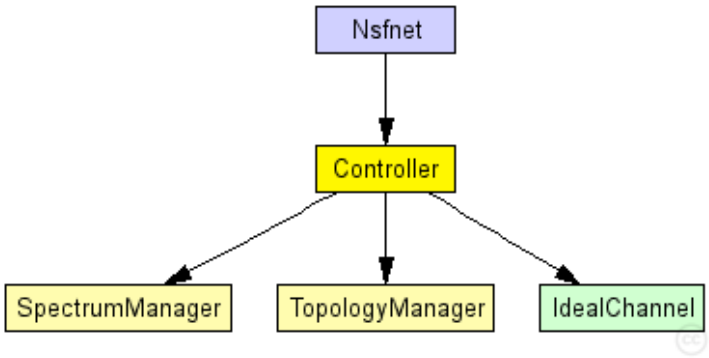

Figura 4.9 Diagrama del controlador. Obtenido de OMNET ++

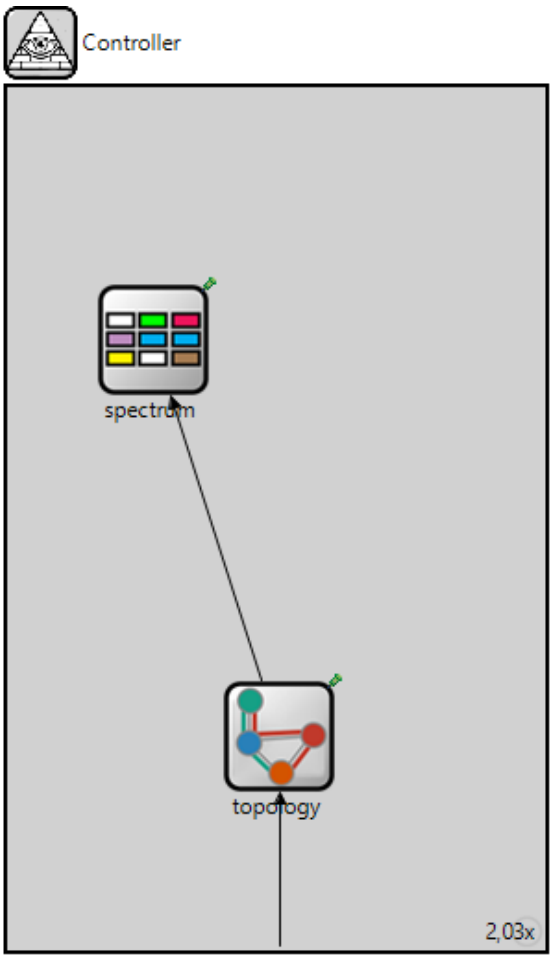

Figura 4.10 Modulo compuesto controlador. Obtenido de OMNET ++

El controlador actúa de acuerdo con la cantidad de slots que requiere la petición y la cantidad de slots disponibles en el canal óptico, como se ve en la figura 4.11

OMNET ++ tiene integradas diversas clases especiales para tratar con redes como lo es *cTopology*, que trabaja con nodos y compuertas para representar las diferentes conexiones de red. Esta clase tiene incorporadas varias funciones que permiten calcular la ruta más cercana a un nodo destino. Sin embargo, esta función solo provee de una ruta, por lo cual fue necesario implementar un método que calcule más rutas entre nodos. Lo anterior se logra apoyándose en la teoría de grafos [35] en donde se muestra como implementar esa estructura de datos en C++ y así se consigue calcular y almacenar las rutas necesarias para una solicitud de conexión.

SlotBanwidth es el ancho de banda que tienen los slots 12.5 Ghz según la recomendación [63] y [64] *channelBandwidth* la capacidad del canal o cantidad de slots disponibles.

Name
slotBandwidth
channelBandwidth

Figura 4.11 parámetros del controlador. Obtenido de OMNET ++

Las rutas entre nodo fuente y destino se almacenan en un archivo de texto llamado routes.csv. Este archivo guarda información relevante como cada nodo y cada compuerta del enrutador por las cuales atraviesa cada ruta encontrada un numero identificador de paquete, un color, el número de saltos de ruta, un numero identificador para cada ruta, entre otros. Esta información se calcula cuando el paquete llega al módulo controlador, se almacena en el archivo y se usa posteriormente para nuevas peticiones, si una ruta entre dos nodos ya fue encontrada primero el controlador busca esta fuente y destino y asigna el enrutamiento dinámico de acuerdo con el archivo y no hace el cálculo de la ruta nuevamente, es decir aprende la mejor alternativa de ruta.

Routes.csv

```
13,0,106,1681,#5C83D0,3,0
8,0,106,785,#5C83D0,3,0
3,0,106,225,#5C83D0,3,0
13,56,106,1905,#5C83D0,4,1
9,0,106,1233,#5C83D0,4,1
6,0,106,337,#5C83D0,4,1
1,0,106,1,#5C83D0,4,1
13,112,106,2129,#5C83D0,4,2
12,0,106,1345,#5C83D0,4,2
7,0,106,561,#5C83D0,4,2
2,0,106,113,#5C83D0,4,2
10,56,138,1793,#53E8AF,3,0
9,0,138,1233,#53E8AF,3,0
6,56,138,1121,#53E8AF,3,0
10,0,138,1569,#53E8AF,4,1
8,0,138,785,#53E8AF,4,1
3,56,138,672,#53E8AF,4,1
4,112,138,1008,#53E8AF,4,1
10,112,138,2241,#53E8AF,4,2
12,0,138,1345,#53E8AF,4,2
7,56,138,897,#53E8AF,4,2
4,112,138,1008,#53E8AF,4,2
8,0,211,785,#0DBB0A,3,0
3,56,211,672,#0DBB0A,3,0
4,56,211,896,#0DBB0A,3,0
8,56,211,1568,#0DBB0A,3,1
10,112,211,2241,#0DBB0A,3,1
12,0,211,1345,#0DBB0A,3,1
8,112,211,1680,#0DBB0A,3,2
13,112,211,2129,#0DBB0A,3,2
12,0,211,1345,#0DBB0A,3,2
5,0,240,1009,#B631A0,3,0
4,56,240,896,#B631A0,3,0
```

Figura 4.12 Archivo route.csv. Obtenido de OMNET ++

Dentro del controlador hay dos módulos simples, muy importantes y de complejo diseño que se encargan de gestionar los recursos de la red ya sea ruta o espectro y asignarles dichos recursos a las peticiones de conexión que son generadas dentro de los nodos ya descritos. Estos módulos simples son:

- **Módulo *Simple Topology***

Se encarga de recibir las peticiones que realizan los nodos de la red y asignarles una ruta de conexión, para ello en su interior aplica el algoritmo de las *K-rutas más cercanas* que permite que el controlador pueda revisar la topología de la red y establecer una ruta de conexión para cada solicitud que se genera. Además, se encarga de escribir la tabla de enrutamiento dinámico para que los BVWXC sepan hacia donde direccionar el paquete cuando le llega.

- **Módulo *Simple Spectrum***

Este componente es el encargado de realizar la asignación de espectro a las conexiones dentro de la red, cuando ya se ha establecido la ruta para las solicitudes que provienen de los nodos, se encarga de asignar el espectro que permita satisfacer dicha petición. En este componente se implementan dos algoritmos que permiten realizar dicha asignación que son el *First Fit* [32], el cual es un algoritmo propio de las EON y el método cognitivo a ser diseñado. Este módulo se diseña de tal manera que cumpla con las condiciones de continuidad y contigüidad de espectro, también lleva la cuenta de paquetes perdidos en la red, para obtener la probabilidad de bloqueo, que servirá como métrica de evaluación del desempeño de la red.

En la figura 4.12 se puede observar el establecimiento de una conexión en el modelo de red propuesto.

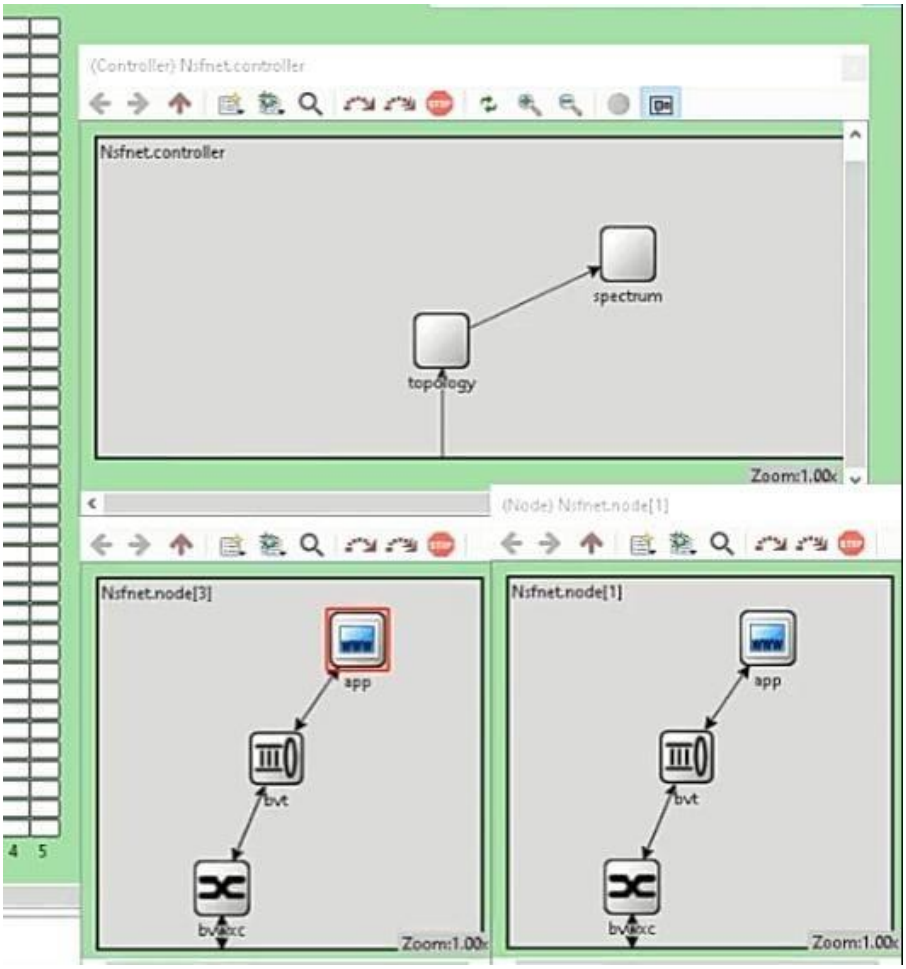

Figura 4.13 Nodo estableciendo conexión con el controlador de la red SDON. Obtenido de OMNET++

En el modelo SDON de prueba se utilizan nodos formados por un BWXC, UN BVT, en transmisión y recepción, esto con la finalidad de obtener un modelo de arquitectura que corresponda a una red EON o flexgrid tomando como referencia el modelo presentado en [21]. Ya diseñados e implementados los módulos anteriormente descritos, lo siguiente es interconectarlos a través de la topología de la red NSFNet (NSFnet, *National Science Foundation Network*).

La red NSFNet está compuesta por catorce nodos distribuidos a lo largo y ancho de los Estados Unidos de América, conectando ciudades principales. Esta red ha sido diseñada para la investigación y estudio de las telecomunicaciones, está compuesta por 14 nodos interconectados a través de 42 enlaces bidireccionales. La red es gestionada por un controlador, de esta forma se obtiene la red de prueba SDON flexgrid, la cual debe cumplir con las características propias de este tipo de redes.

En la figura 4.14 se puede observar la topología de red seleccionada para implementar el modelo de red de prueba que se analizara en este trabajo de grado, y en la figura 4.15 la topología de red SDON propuesta.

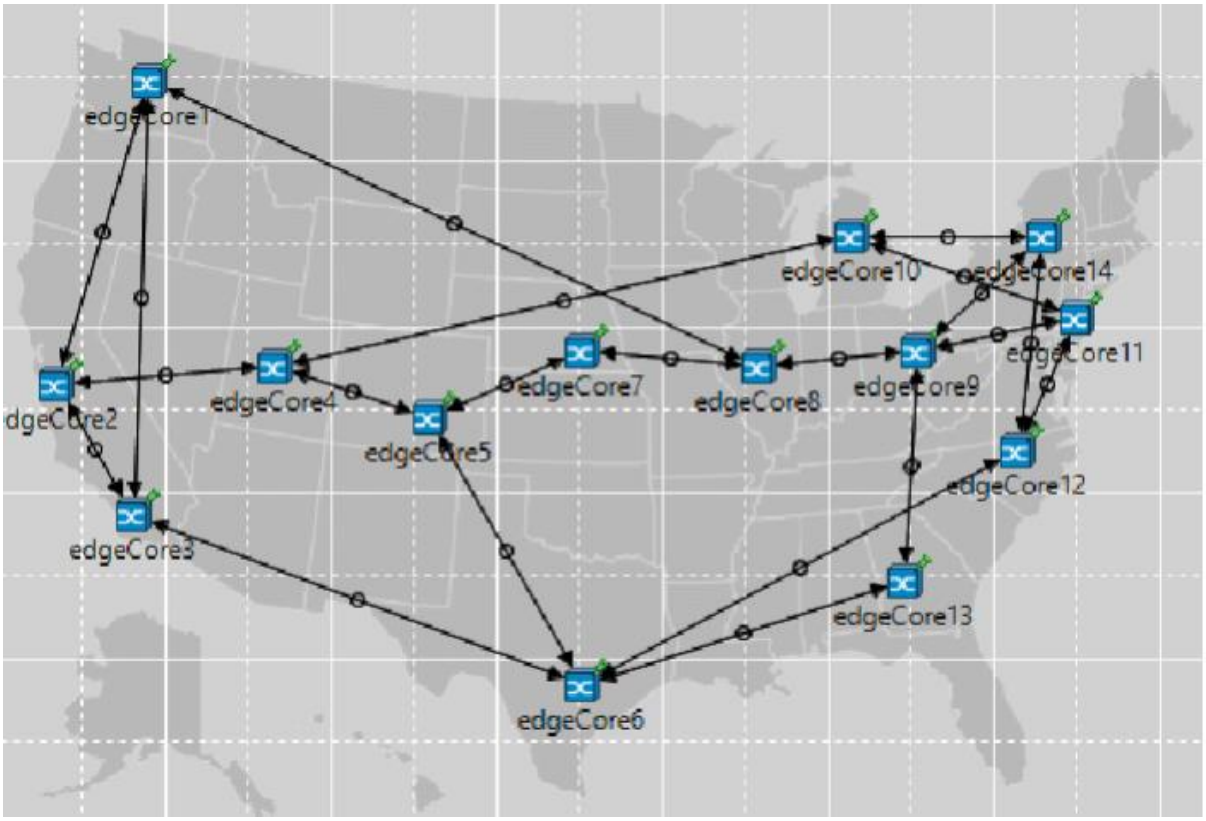

Figura 4.14 Topología NSFNet. Tomada de [43].

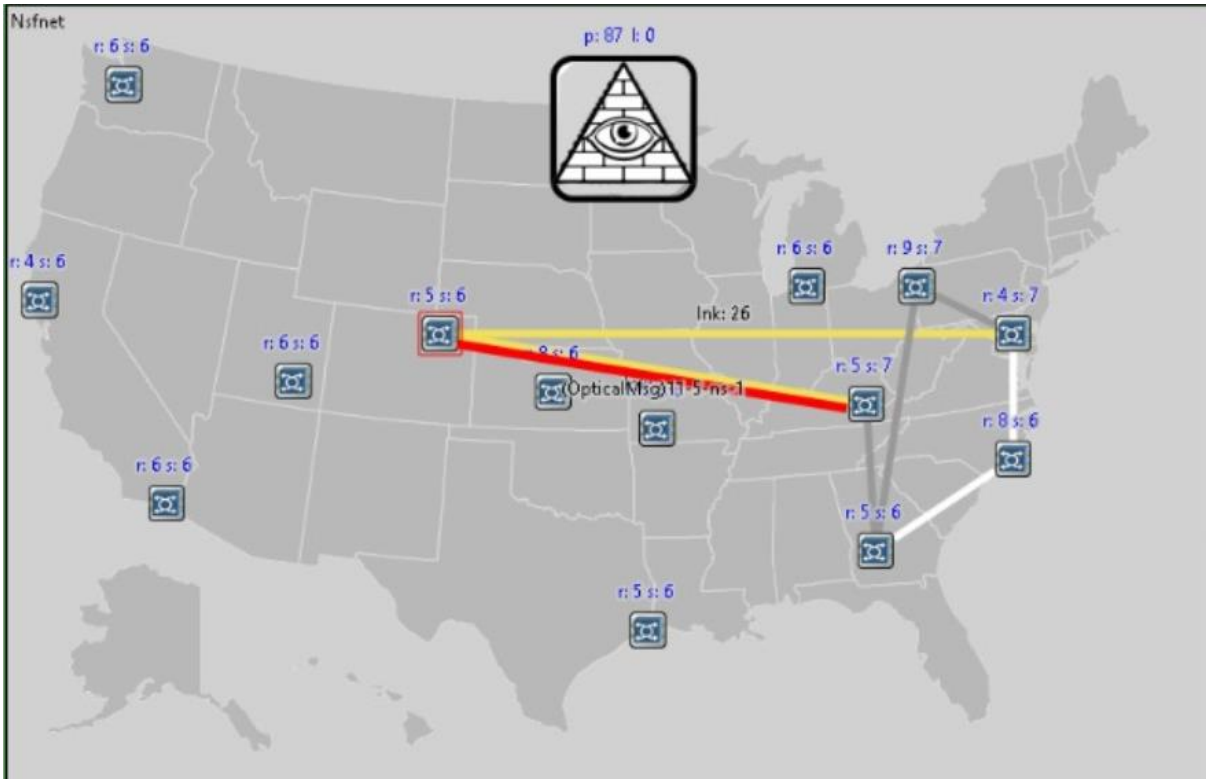

Figura 4.15 Topología SDON Flexigrid. Obtenido de OMNET ++

En la figura 4.16 se observa la red centralizada con el controlador y la configuración de los nodos.

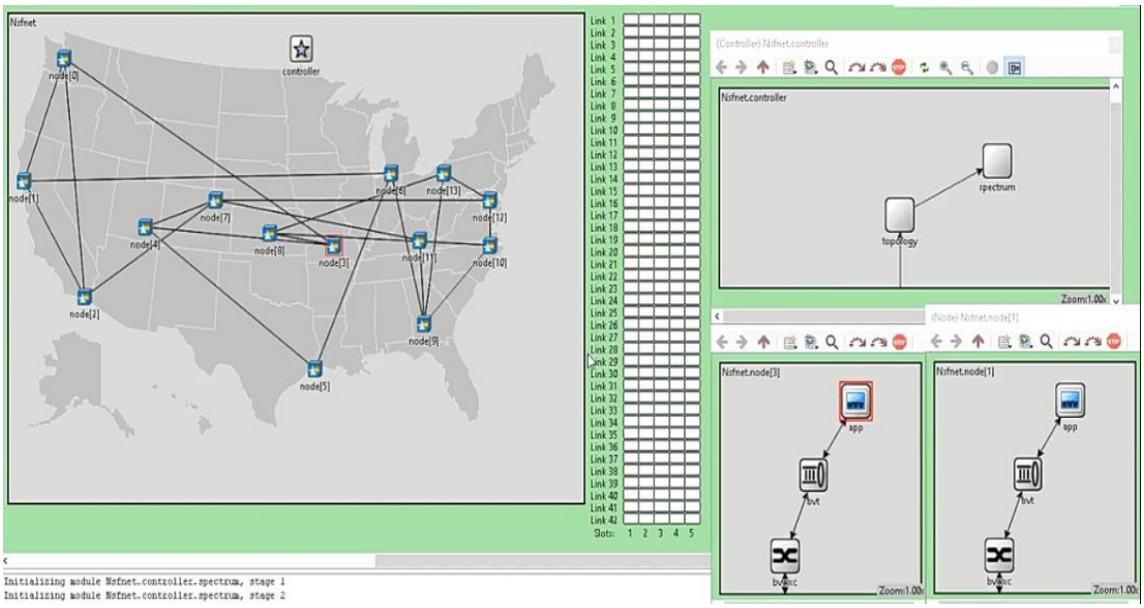

Figura 4.16 Conexión en la red SDON Flexigrid.

- **Centralización de la red**

La tarea de manejar los recursos de red, llenar las tablas de enrutamiento, y las funciones que permiten tener un control sobre estos recursos de red, le corresponde al módulo “controlador”. El primer miembro de este módulo es el módulo *TopologyManager*, quién se encarga de buscar diferentes rutas entre nodos y almacenar está información en un archivo de texto llamado *Routes.txt* que contiene la información sobre los nodos y compuertas correspondientes a cada nodo de una ruta determinada. En la figura 4.17 se observa una parte del código implementado para este propósito.

```
cTopology::Node *srcNode = topo->getNodeFor(getParentModule()->getParentModule()->getSubmodule("node", src));
cTopology::Node *targetNode = topo->getNodeFor(getParentModule()->getParentModule()->getSubmodule("node", dst));
std::vector<cTopology::LinkIn*> route;
std::vector<cTopology::LinkIn*> path;
std::vector<std::vector<cTopology::LinkIn*>> pathList;
std::vector<std::vector<cTopology::LinkIn*>> routeList;
std::vector<int> nodeDis;
for (int i = 0; i < topo->getNumNodes(); i++) {
    nodeDis.push_back(MAXIMUM);
    path.push_back(nullptr);
}
nodeDis.at(targetNode->getModule()->getIndex()) = 0;
for (int idx = 0; idx < nodeDis.size(); idx++) {
}
std::deque<cTopology::Node*> q;
q.push_back(targetNode);
while (!q.empty()) {
    cTopology::Node *v = q.front();
    q.pop_front();
    for (int i = 0; i < v->getNumInLinks(); i++) {
        cTopology::Node *w;
        w = v->getLinkIn(i)->getRemoteNode();

        if (w == srcNode) {
            pathFound++;
            nodeDis.at(w->getModule()->getIndex()) = 1 + nodeDis.at(v->getModule()->getIndex());
            path.at(w->getModule()->getIndex()) = v->getLinkIn(i);
            pathList.push_back(path);
            for (int idx = 0; idx < nodeDis.size(); idx++) {
            }
            break;
        }
        int dist = nodeDis.at(w->getModule()->getIndex());
        if (dist == MAXIMUM) {
            nodeDis.at(w->getModule()->getIndex()) = 1 + nodeDis.at(v->getModule()->getIndex());
            path.at(w->getModule()->getIndex()) = v->getLinkIn(i);
            q.push_back(w);
        }

        if (pathFound == numPaths)
            break;
    }
    if (pathFound == numPaths)
        break;
}
```

Figura 4.17 porción de Código del módulo controlador

Para calcular las rutas, se aprovechan funciones incorporadas en la clase cTopology para utilizar la teoría de grafos y así determinar más de una ruta entre nodos; OMNET ++ ya trae implementada una función para encontrar la ruta entre dos nodos, sin embargo, esta función sólo entrega una ruta entre dos nodos. El algoritmo de asignación de espectro tiene como requisito un conjunto de rutas preestablecido entre nodos, por esta razón es necesario implementar un método que permita encontrar más rutas.

El método que implementa esta funcionalidad se muestra en la figura 4.17, las variables “route” y “path” son vectores de enlaces que pertenecen a la clase cTopology de omnet, las variables “pathList” y “routeList” son vectores formados por otros vectores de enlaces de la clase cTopology, estos serán los encargados de almacenar las diferentes rutas entre nodos.

El vector “nodeDis” es quién guarda las distancias entre nodos, en un principio este vector se llena con un valor “MAXIMUM” para representar que están separados una distancia infinita como lo requiere la teoría de grafos.

En una cola, en el código denominada “q” se van a ir almacenando los diferentes nodos que se visitan cada vez, tomando como nodo fuente constante *srcNode del tipo cTopology:Node y como nodo destino *targetNode del mismo tipo, se puede ir visitando cada nodo y preguntando si está conectado al anterior, si es así escribir este número en la cola.

El proceso de buscar rutas entre nodos depende del valor de rutas que se hayan especificado en el archivo de configuración omnet.ini llamado “numPaths”. Las rutas encontradas se almacenan en “pathList” y posteriormente se escriben en el archivo de texto Routes.txt. La figura 4.18 muestra parte del código implementado para poder lograr este proceso.

```
public:
    SpectrumManager() = default;
    virtual ~SpectrumManager();

protected:

    virtual void initialize(int stage) override;
    virtual int numInitStages() const override; // NOTE THE const MODIFIER!!!
    virtual void finish() override;
    virtual void handleMessage(cMessage *msg) override;
    virtual void refreshDisplay() const override;
    void drawSlotGrid();
    void drawSlotsOnGrid(int lnkPos, int sltPos, int numSlots, cFigure::Color color);
    void drawSingleSlotOnGrid(int id, cFigure::Color color);
    void updateSlotGrid();
    void cleanSlotGrid();
    std::vector<int> contiguousSpectrum(std::vector<Link*> lnk);
    std::vector<std::vector<int>> continuousSpectrum(std::vector<int> slots);
    Link* searchLink(int id);
    std::vector<int> algorithmFirstFit(std::vector<std::vector<int>> continuous, std::vector<Link*> lnk, int slrequest, cFigure::Color color, int msgid);
    std::vector<int> algorithmLastFit(std::vector<std::vector<int>> continuous, std::vector<Link*> lnk, int slrequest, cFigure::Color color, int msgid);
    std::vector<int> algorithmRegFit(std::vector<std::vector<int>> continuous, std::vector<Link*> lnk, int slrequest, cFigure::Color color, int msgid);
    void printCurrentLinkMatrix();
    void printTotalLinkMatrix();
    void clearLinkMatrix();
    void saveResults();
};
```

Figura 4.18 poción de código del módulo topology.

El módulo responsable de tomar las decisiones y guardar la información de red es la clase “SpectrumManager”, su funcionamiento es el responsable tanto de la asignación de espectro como también de actualizar la representación gráfica del estado de red en el simulador.

Las funciones refresSisplay (), drawSlotsOnGrid(), drawSingleSlotOnGrid(), updateSlotGrid() y cleanSlotGrid() son las encargadas de llevar le mantenimiento de la rejilla de espectro que se muestra al lado derecho de las simulaciones, está rejilla necesita estar dibujando nuevos cuadros cuando una solicitud de conexión se hace según los slots y enlaces que vaya a ocupar esa comunicación, este recurso debe liberarse una vez esa conexión se termine.

Los “slots” que llevan el mismo color significa que pertenecen a la misma conexión, mientras que si un slot está en color blanco significa que el slot está libre para transmitir.

Las siguientes funciones más específicas como: contiguousSpectrum (), se encargan de verificar la condición de contigüidad del espectro, que ubica en el espectro disponible la nueva conexión de forma que los slots estén alineados.

La función continuosSpectrum () es quién se encarga de asignar slots continuos en el espectro disponible, el número de slots continuos que se requieren depende del número de slots que se hayan pedido en la solicitud, si existen slots disponibles que cumplan está condición se asignan de caso contrario se elimina el paquete.

Las funciones que siguen en el simulador ya son las encargadas de asignar los distintos tipos de algoritmos que se pueden utilizar para asignación de espectro, para efectos de este trabajo de grado fue necesario implementar tres algoritmos: FirstFit, LastFit y RegressionFit.

Estos algoritmos solicitan como parámetros un conjunto de vectores continuos, que se calcularon anteriormente donde se aplicará el algoritmo y se asignará el espectro.

4.3 Diseño del método cognitivo de asignación de espectro óptico y algoritmos implementados

Estos algoritmos fueron implementados y puestos en funcionamiento en el módulo compuesto del controlador, debido a que contiene toda la información de la red, lo que permite analizar los recursos de esta y su disponibilidad en las diferentes rutas que se puedan seleccionar conforme van llegando solicitudes de conexión. A continuación, se presenta detalladamente el algoritmo de enrutamiento y los algoritmos de asignación de espectro elegidos para este proyecto de investigación.

4.3.1 Algoritmo de la K-rutas más cercanas

Este algoritmo es implementado dentro del módulo compuesto del controlador y en su interior dentro del módulo simple topology. Establece un conjunto de rutas más cortas desde un nodo de origen hasta un nodo destino elegido aleatoriamente. Para este propósito se implementa una función en C++ que utiliza la clase cTopology y además variables y vectores para almacenar esta información, en [35] se encuentra maneras de atravesar un grafo con el método BFS, por cual se realiza una adaptación para la lógica de la red. A continuación, se observa el pseudocódigo de este algoritmo:

Entradas:
k= número de rutas por encontrar,
n= rutas encontradas hasta ahora,
s= nodo fuente,
t= nodo destino,
G [i, j]= Matriz de enrutamiento de red,
C [i, j]= Matriz de capacidad de red,
H [u]= Costo por nodo,
Inf= un valor constante más grande que la mayor longitud para un camino

Salida:
K-rutas entre nodo fuente y destino

1. Inicializar G [i, j] y C [i, j] con valores de red
2. G [s, t]= Inf, C [s, t]=0
3. Llamar k-Dijkstra (k, n, s, t, G, C)
4. K-Dijkstra (k, n, s, t, G, C)
5. **While** n ¡k **do**
6. Dijkstra (s, t, G, C)
7. Registrar la ruta encontrada como un vector predecesor
8. Quitar cada entrada en G [i, j] que aparece en la ruta
9. n++
10. **end while**

La figura 4.19 presenta el diagrama de flujo para el algoritmo de las k-rutas más cercanas.

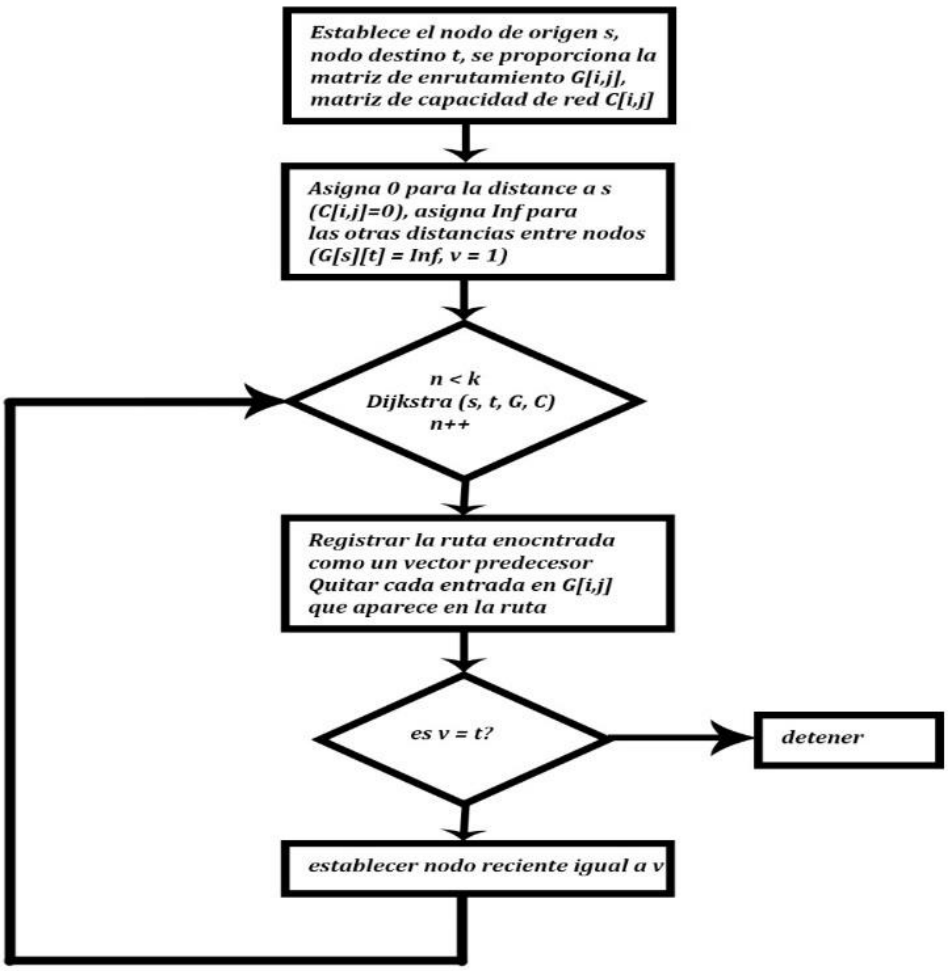

Figura 4.19 Algoritmo de las k-rutas más cercanas

4.3.2 Algoritmo de asignación First Fit

La asignación del espectro en el primer caso se implementa utilizando el algoritmo *First Fit* [32][33*]*, el cual funciona realizando una búsqueda de los slots requeridos, recorriendo la rejilla de espectro desde los slots con índice más bajo hasta los más altos.

Su funcionamiento toma en cuenta el número de slots que son requeridos para resolver la solicitud de conexión, después realiza una búsqueda del número de slots haciendo un recorrido en la rejilla de izquierda a derecha desde el primer slot, garantizando el principio de contigüidad, en caso de no lograr la conexión, el algoritmo busca desde el siguiente slot y repite el proceso hasta que se establece la conexión o recorre la rejilla por completo y si los slots requeridos no se encuentren disponibles se rechaza la conexión, de acuerdo con los siguientes pasos:

Entradas:

V [i, j]: Conjunto de rutas preestablecidas
C [i, j]: Matriz de capacidad de red
G [i, j]: Matriz de enrutamiento de red
n: Solicitud de numero de slots
f: Clase enlace, abstracción de funciones y propiedades de un enlace

Salida:

Conjunto de slots para una ruta seleccionada

1. **If** f. assigned **then**
2. **return** antigua ruta asignada para f
3. **foreach** p E P do
4. **foreach** i E p do
5. l.avail_bw←capacity
6. encontrar el mínimo l.avail_bw
7. **return** l.minimum
8.p. avail_bw←l.minimum
9. P= {para todos p.avail_bw}
10. p.sorted=sort p
11. **if** p used + f.rate < p. capacity **then**
12 p. used ←p. used +f.rate
13. **return** p
14 h= HASH(f)
15. **return** p= psrc ←→dst(h)

La figura 4.20 muestra el diagrama de flujo del algoritmo de asignación de espectro Firt Fit.

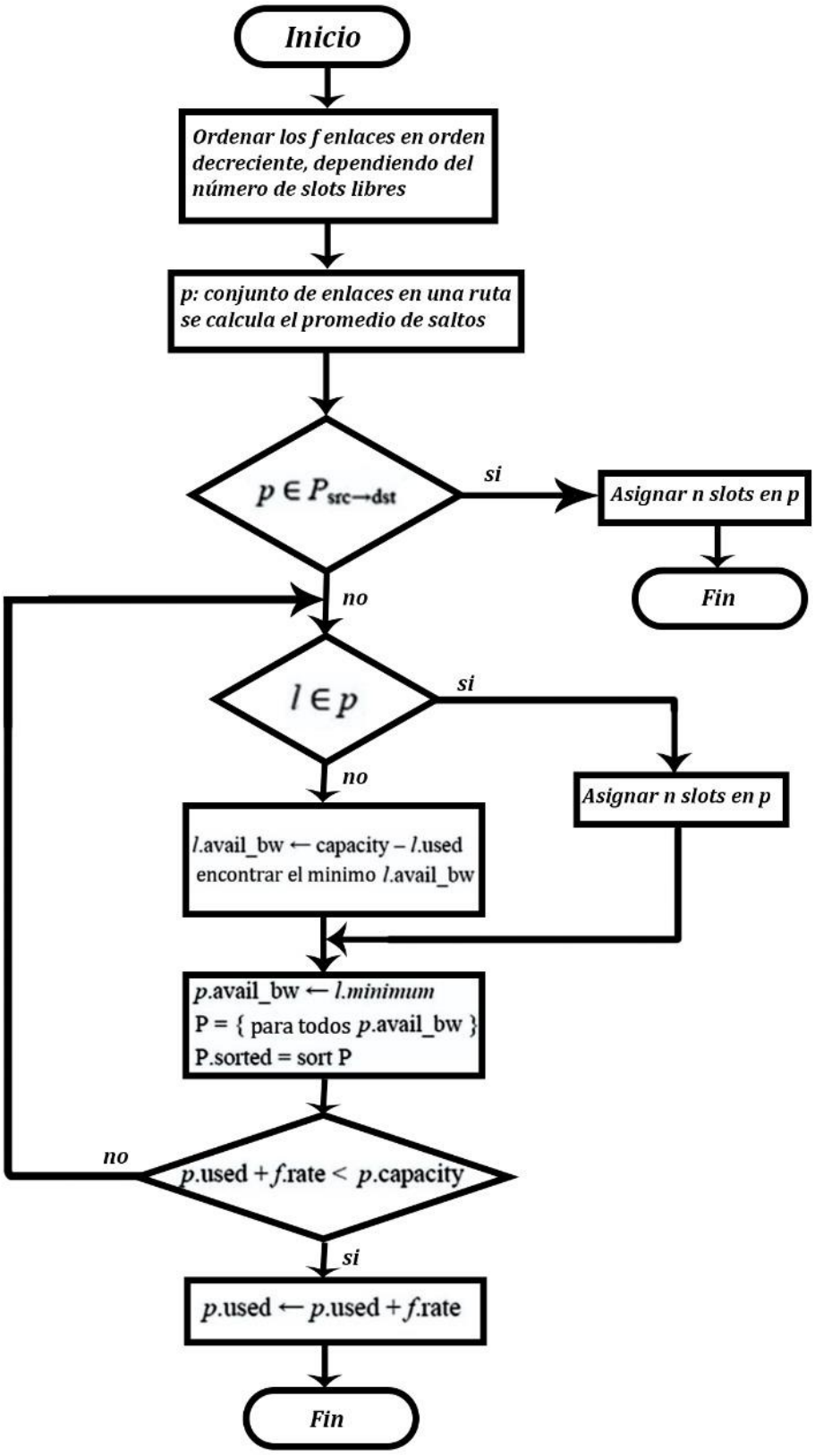

Figura 4.20 Diagrama de flujo del algoritmo first fit.

4.3.3 Algoritmo Last Fit

Esta política siempre intenta elegir el espacio indexado más alto de la lista de ranuras disponibles y lo asigna a la ruta para atender la solicitud de conexión. Cuando se completa, la ranura vuelve a la lista de disponibles ranuras. Intenta invariablemente seleccionar la ranura disponible con el índice más alto y utilizarla para el aprovisionamiento de la ruta. Cuando la ruta es liberada, el espacio se agrega a la lista de espacios libres [32][33]. Este algoritmo trabaja de la siguiente forma:

Entradas:
V [i, j]= Conjunto de rutas preestablecidas
C [i, j]= Matriz de capacidad de red
G [i, j]= Matriz de enrutamiento de red
n= solicitud de numero de slots
f= clase enlace, abstracción de funciones y propiedades de un enlace

Salida:
Conjunto de slots para una ruta seleccionada

1. **If** assigned **then**
2. **return** antigua ruta asignada para f
3. **foreach** p E p do
4. **foreach** l E p do
5. l. avail_bw←capacity- l. used
6. encontrar el máximo l. avail_bw
7. **return** l. máximum
8. p. avail_bw←l.maximum
9. p= {para todos p.avail_bw}
10. p.sorted = sort p
11. **if** p.used +f.rate<p.capacity **then**
12. p.used ←p.used + f.rate
13. **return** p
14. h=HASH(f)
15. **return** p=psrc←→dst (h)

En la figura 4.21 se muestra el diagrama de flujo del algoritmo *last fit.*

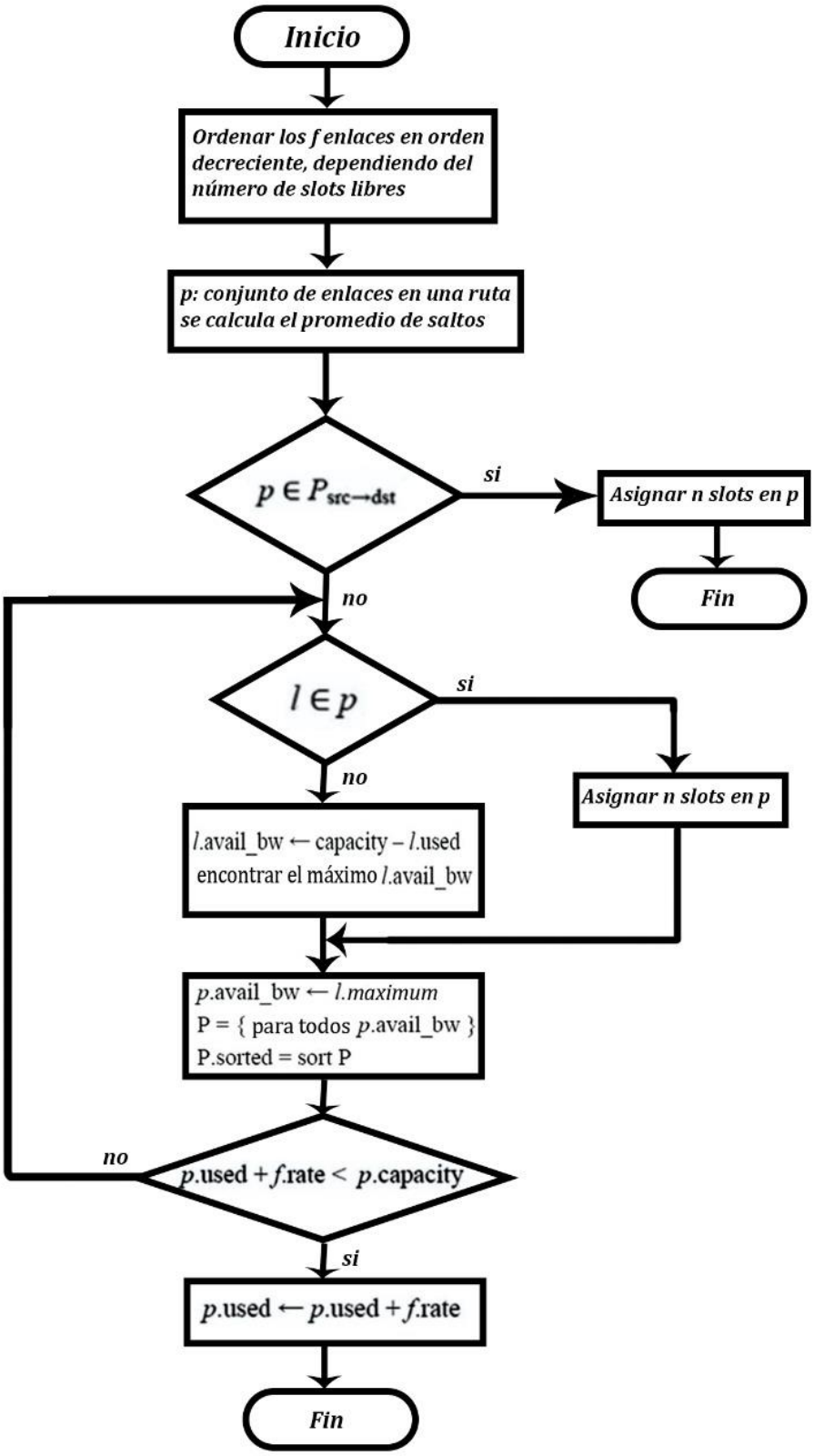

Figura 4.21 Diagrama de flujo del algoritmo last fit.

4.3.4 Método cognitivo de asignación de espectro óptico basado en machine learning

En la figura 4.22 se observa el diagrama de flujo del método cognitivo implementado sobre la red.

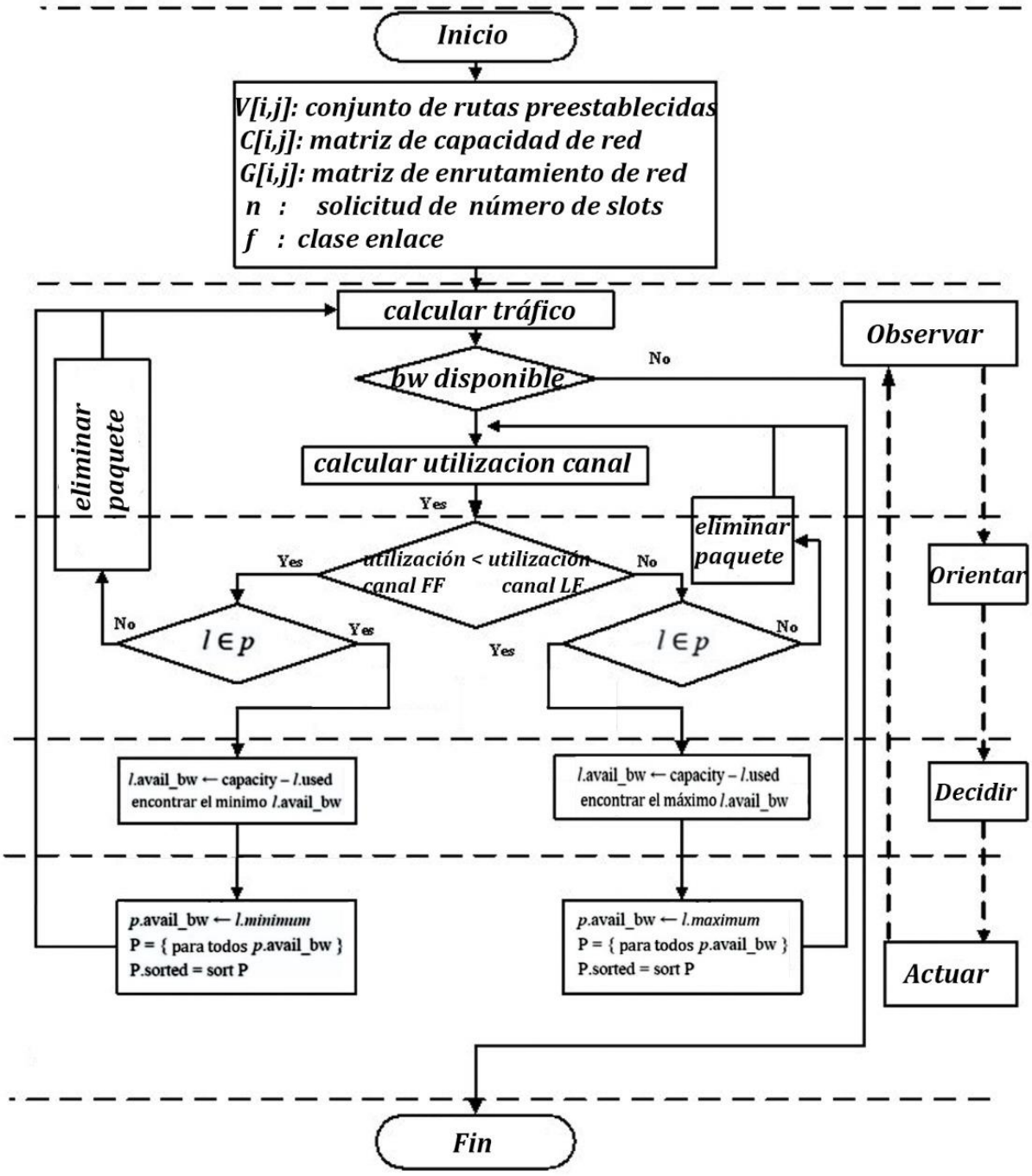

Figura 4.22 Diagrama de flujo Regression Fit

El seudocódigo que describe el funcionamiento del método cognitivo es el siguiente:

Entrada:
V [i, j]= Conjunto de rutas preestablecidas
C [i, j]= Matriz de capacidad de red
G [i, j]= Matriz de enrutamiento de red
n= solicitud de numero de slots
f= Clase enlace, abstracción de funciones y propiedades de un enlace

Salida:
Conjunto de slots para una ruta seleccionada

1.Calcular tráfico de red
2. Calcular promedio de utilización del canal
3. **if** f assigned **then**
4. **return** antigua ruta asignada para f
5. **foreach** p E psrc do
6. **foreach** l E p do
7. l. avail_bw←capacity-l.used
8. encontrar el máximo l. avail_bw
9. **return** l.maximum
10. p.avail_bw←l.maximum
11. p= {para todos p.avail_bw}
12. p.sorted=sort o
13. **if** p.used+f rate < p.capacity then
14. p.used ←p.used +f.rate
15. **return** p
16. **foreach** p E Psrc do
17. **foreach** l E p do
18.l. avail_bw←capacity-l.used
19. encontrar el mínimo l.avail_bw
20. **return** l.minimum
21. p.avail_bw←l.minimum
22. p= {para todos p.avail_bw}
23. p.sorted=sort p
24.**if** p.used + f.rate < p.capacity **then**
25. p.used ←p.used +f.rate
26. **return** p
27. h= HASH(f)
28. **return** p=psrc→dst(h)

Para realizar la evaluación y análisis del desempeño de la red bajo el método cognitivo de asignación de espectro basado en ML se debe comparar con un algoritmo de asignación de espectro en redes EON, en este caso el algoritmo *FirstFit*, encontrado en [32][33]. Para aplicar el método que se propone en este trabajo de grado es necesario que la red esté calculando continuamente diferentes métricas que esta tiene durante el tiempo de transmisión; para lograr este objetivo se acude a un concepto conocido como cognitividad o proceso cognitivo en redes, particularmente el modelo OODA descrito en [43].

Se define llamar al método diseñado, *Regression Fit* y las métricas que este método debe monitorear constantemente son dos: el tráfico en un instante de tiempo y la utilización promedio del canal. Los pasos que el método debe seguir para cumplir con el ciclo cognitivo son:

- **Observación**

Es la primera etapa del modelo OODA, se encarga de colectar diferentes datos que son necesarios para el cálculo del tráfico de red. Es necesario almacenar datos como las solicitudes de slots, retardo extremo a extremo, tiempo entre generación de paquetes y saltos de rutas en diferentes vectores como atributos de una clase llamada *Regression*, quien es la encargada de la cognitividad en este simulador de redes ópticas.

```
class Regression
{
public:
    std::string m_name;
    std::vector<double> endToEndDelay;
    std::vector<double> slotAverage;
    std::vector<double> holdingTime;
    std::vector<double> routeHops;

    Regression()    {    }

    void addSlotAverage(int slotReq)    {

    void addHoldingTime(double holding_time)    {

    void addRouteHops(double hops)    {

    double calculateSlotAverage()    {

    double calculateRouteHopsAverage()

    double calculateHoldingTimeAverage()

    double calculateEndToEndDelayAverage()
```

Figura 4.23 Porción de código del método cognitivo.

La etapa de observación se compone de dos partes, la primera es recolectar en vectores los datos antes mencionados y la segunda es calcular el promedio de estos datos. Esta información es indispensable para el cálculo del tráfico de la red, cabe resaltar que únicamente son necesarios estos procedimientos cuando el algoritmo seleccionado para la asignación de espectro es *"RegressionFit"* en el archivo de configuración de omnet, si el caso de simulación es *"FirstFit"* todos estos cálculos y promedios no se realizan, esto ya muestra que la cognitivdad en redes ópticas implica un costo computacional adicional en comparación con una red que no posea esta cualidad. La figura 4.23 muestra una porción del código implementado para realizar cálculos de las métricas necesarias en el método

- **Orientación**

El tráfico es una medida que permite tener un conocimiento del desempeño de la red en un determinado momento, sin embargo, no brinda un conocimiento específico sobre algún recurso en particular, como por ejemplo los slots disponibles en los enlaces. Para suplir esta necesidad, en las redes ópticas elásticas se proporciona una métrica llamada promedio de utilización del canal, lo que permite estimar la utilización de los slots disponibles para cada canal. Como lo que se busca es que el método constantemente este monitoreando la red, requiere conocer esta métrica para definir su funcionamiento, para ello se recurre a una función encontrada en [65] que define el promedio de utilización del canal según la ecuación 4.1, así:

$$U(v) = v.\ hops\ avg/|F|X|E| \quad (4.1)$$

dónde $|F|y|E|$ corresponden al número de slots de espectro disponibles por enlace y el número de enlaces disponibles en la red, respectivamente. hops_avg corresponde al promedio del número de saltos de cada ruta y v corresponde al tráfico de red en un instante específico.

Este dato proporciona una medida que se usa para comparar cuál de los dos algoritmos, en este caso, *FirstFit* y *LastFit*, tiene una mejor utilización del canal en un momento dado y con un tráfico determinado. Para llevar esta tarea acabo se sigue el siguiente procedimiento: primero se calcula el tráfico de la red en un instante dado, luego se asigna el espectro para los algoritmos de asignación de espectro, tanto *FirstFit* como *LastFit* y posteriormente a esto se calcula el promedio de utilización del canal para cada algoritmo. Este procedimiento orienta a la red sobre que algoritmo tiene un mejor desempeño en cuanto a la asignación de espectro respecto del otro.

Este método calcula constantemente esta métrica para los dos algoritmos en busca de asignar uno u otro dependiendo de su desempeño, como muestra la figura 4.24.

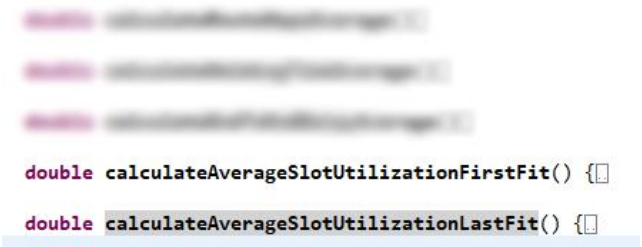

Figura 4.24 Porción de código de selección del algoritmo en el método cognitivo.

Con base en lo anterior la clase Regression se encarga de proporcionar la cognitividad de red al simulador.

- **Decisión**

Esta etapa se implementa en el proceso cognitivo del método propuesto haciendo el cálculo de un valor siguiente al estado actual de la red, para esto y debido a que el promedio de utilización del canal es una función lineal del tráfico, se utiliza una técnica de ML conocida como regresión lineal para predecir cuál de los dos algoritmos tendrá un mejor comportamiento, si el algoritmo *FirstFit* o el algoritmo *LastFit*, y este dato sirve para tomar la decisión de cual algoritmo se utiliza en determinado momento.

- **Actuación**

La última etapa del modelo cognitivo OODA, para este caso la acción se da cuando se utiliza el algoritmo que en un momento dado tenga un menor valor para el promedio de utilización del canal y se realice la asignación de espectro.

En la figura 4.25 se observa una porción del espectro cuando se han enviado un cierto número de paquetes, los slots que están ocupados enviando información se pintan de color y e organizan de acuerdo al algoritmo de asignación, los slots que no se han ocupado en ningún momento, se muestran con el borde fino y el fondo de color blanco, mientras que los slots que ya han sido ocupados y liberados posteriormente, se muestran con el borde grueso y el color blanco que indica que se encuentran disponibles para una nueva conexión, de esta forma se demuestra como el nodo controlador libera el recurso una vez la conexión ha sido finalizada.

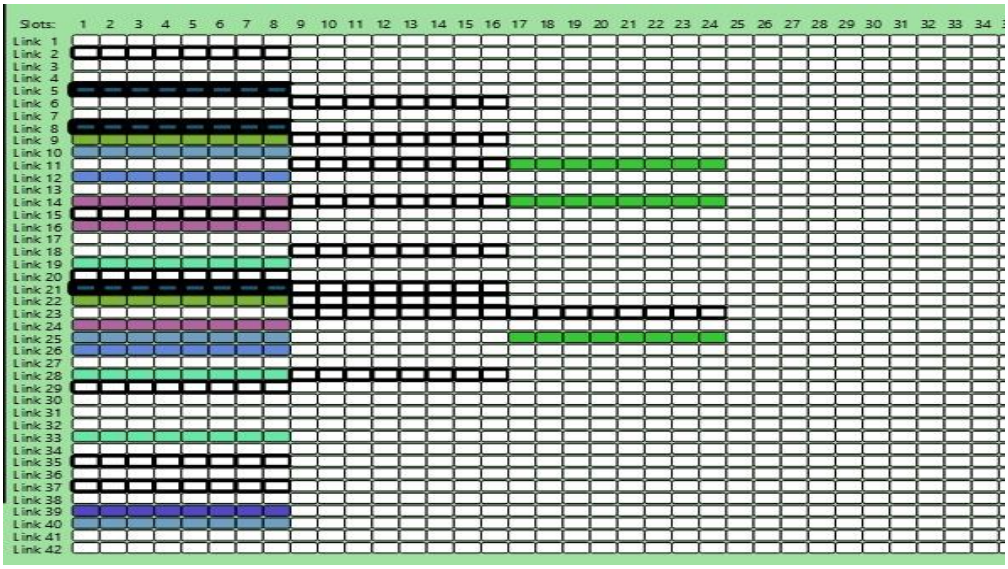

Figura 4.25 Porción de Espectro.

En la figura 4.26 se observa la porción de código que realiza la regresión lineal.

```
std::vector<int> SpectrumManager::algorithmRegFit(std::vector<std::vector<int>> continuous, std::vector<Link*> lnk, int slrequest, cFigure::Color color, int msgid)
{
    double UFF = regresion.calculateLinearRegressionFirstFit();
    double ULF = regresion.calculateLinearRegressionLastFit();

    if (UFF < ULF) {
        return algorithmFirstFit(continuous, lnk, slrequest, color, msgid);
    }
    else {
        return algorithmLastFit(continuous, lnk, slrequest, color, msgid);
    }
}
```

Figura 4.26 Porción de código método cognitivo. Regresión lineal.

La función *algorithmRegFit* en el simulador es quien se encarga de asignar un algoritmo *FirstFit* o *LastFit*, dependiendo del valor que se haya predicho en la clase Regresión.

4.4 Escenarios de simulación

Para realizar el análisis del desempeño de la red implementada y de los algoritmos RSA desarrollados, además del método cognitivo diseñado, es necesario llevar a cabo la propuesta de unos escenarios de simulación que permitan simular y evaluar el desempeño de la red, sometida bajo ciertos parámetros y condiciones de simulación.

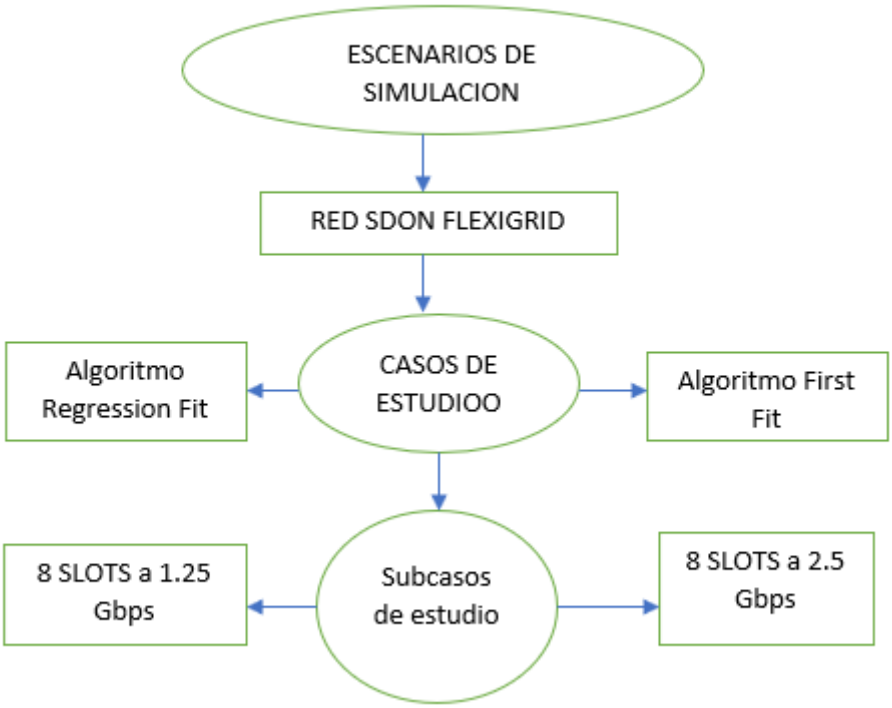

Figura 4.27 Escenarios de simulación.

4.4.1 Caso y subcasos de simulación

Una vez obtenido el escenario o escenarios de simulación, se plantean como plan de prueba distintos casos de estudio, que permitan evaluar el desempeño de la red y del método diseñado, esto ceñido a los objetivos ya propuestos y de esta manera obtener una respuesta a la pregunta de investigación formulada. Los casos de estudio son:

Tabla 4.1 Casos de Simulación

Escenarios de simulación			
topología de red NSFnet- red SDON Flexgrid			
Casos de simulación			
Caso 1		**Caso 2**	
Algoritmo *First Fit*		Algoritmo *Regression Fit*	
Subcasos de simulación			
8 slots disponible		8 slots disponible	
1.25 Gbps	2.5 Gbps	1.25 Gbps	2.5 Gbps

Para cada caso de simulación se especifican unas cargas de tráfico y se analiza el desempeño de la red de acuerdo con los parámetros establecidos en los objetivos del trabajo de grado. Después de someter la red a diferentes cargas de tráfico, y observar su comportamiento, con la finalidad de que el desempeño de la red pueda verse afectado o presentar saturación, se definen 3 niveles de carga así:

Tráfico bajo: carga con 5Mb
Tráfico medio: carga con 15Mb
Tráfico alto: carga con 30Mb

Configuración de los parámetros de simulación:

Teniendo en cuenta lo descrito en [63] y [64] en la tabla 4.2 se muestran los parámetros utilizados para ejecutar los casos de simulación con 8 slots, valor escogido ya que variando la velocidad del enlace entre 1.25 y 2.5 Gbps. Se utiliza NSFNET como topología de red. Las solicitudes de conexión entre origen y destino se generan aleatoriamente siguiendo una distribución uniforme con valor medio 100ms (ʎ), el número de slots promedio por petición (β), el promedio del retardo extremo a extremo de la red (h), el número de paquetes procesados por la red (n). Se define la carga de tráfico de la red, en Erlangs, como se describe en la ecuación 4.2 tomada como referencia de [65]

$$v = n * \lambda * \beta * h \qquad (4.2)$$

Siguiendo las recomendaciones encontradas en [63], [64] y [65] se definen los siguientes parámetros para las simulaciones

Tabla 4.2 Parámetros de simulación

Parámetro	Variable	Valores
velocidad de transmisión	datarate_channel	1.25 Gbps – 2.5 Gbps
Número de slots	slotRandomSize	8
Longitud del paquete	packetLength	5 MB, 15MB, 30MB
Algoritmo de Enrutamiento	K-Shorthest Path	numPaths
Algoritmo de Asignación de Espectro	spectrumAlgorithmAssignment	FirstFit – RegressionFit

El desempeño de esta red se analizará de acuerdo con las siguientes métricas:

Perdida de paquetes

Teniendo en cuenta la probabilidad de bloqueo evaluando el bloqueo entre conexiones físicas y despreciando la probabilidad de bloqueo de la red, es decir que se toma la relación entre los paquetes que se envían y los que se reciben en un enlace determinado. Cuando se genera el tráfico, se tiene en cuenta la cantidad de paquetes que se han recibido o enviado tanto en transmisión como en la recepción y con la ayuda de la ecuación 4.3 se calcula la pérdida de paquetes que posteriormente sirve para evaluar el desempeño de la red [66] .

$$pb=paquetes\ perdidos/paquetes\ totales \quad (4.3)$$

Retardo extremo a extremo

El retardo extremo a extremo se toma como una medida entre las conexiones físicas, es decir entre una fuente y el destino [66]. El retardo extremo a extremo se calcula por la diferencia entre el tiempo de llegada de cada conexión y el tiempo de generación de la solicitud, en donde ya se tiene en cuenta el tiempo de procesamiento y el tiempo de duración del paquete en la red. En la ecuación 4.4 se expresa el cálculo del retardo extremo a extremo:

$$Rext = Tllegada - Tgeneracion \quad (4.4)$$

CAPITULO 5

ANALISIS DE RESULTADOS, CONCLUSIONES Y TRABAJOS FUTUROS

Este capítulo aborda el análisis de los resultados obtenidos al ejecutar las simulaciones propuestas en el capítulo 4, para posteriormente realizar una comparación entre los dos modelos de red propuestos, con y sin el método cognitivo de asignación de espectro y así realizar algunas conclusiones con base a lo encontrado, además se proponen algunos trabajos futuros para seguir ampliando la línea de investigación.

5.1 Análisis de resultados

5.1.1 Algoritmo *Firts Fit* con 8 slots, velocidad de enlace de 1.25Gbps y 2.5Gbps

En la figura 5.1 se observar una comparación de la probabilidad de bloqueo del algoritmo *first fit* al ser sometido a las 3 diferentes cargas de tráfico en la red, con 5MB,15MB Y 30MB, con 8 slots disponibles y una velocidad de 1.25Gbps. De acuerdo a lo que se presenta en la figura 5.1 la probabilidad de bloqueo para las cargas de 5MB y 15MB presenta estabilidad y constancia en sus valores lo que muestra que el algoritmo tiene eficiencia al distribuir el espectro y contribuye a que el desempeño de la red se mantenga en un nivel que permita una menor perdida de paquetes, como muestra la figura para 5MB el algoritmo se mantiene constante y no pierde paquetes por lo cual para cargas bajas tiene un buen desempeño, en el caso de 15MB antes de llegar a los 0.8 erlangs la probabilidad de bloqueo tiene un crecimiento hasta llegar a un valor poco menor al 10% de saturación en la red lo que permite inferir que la red empieza a experimentar un crecimiento en el número de paquetes que pierde. Para el caso de la carga más alta de tráfico que es 30MB y con la cual se busca que la red se sature para ver su comportamiento, en la figura se nota que la probabilidad de bloqueo supera la barrera del 30% bastante rápido, mucho antes de llegar a 0.1 erlangs de tráfico, lo cual indica que el algoritmo tiene inconvenientes para distribuir el espectro, causando que la red pierda muchos paquetes y la probabilidad de bloqueo sea alta, después de ello se empieza a estabilizar y tiene algunos decrementos hasta alcanzar un máximo de probabilidad de bloqueo del 50% a los 0.7

erlangs, lo cual muestra que el algoritmo intenta estabilizar el desempeño de la red pero las cargas son muy altas lo cual produce que se sature.

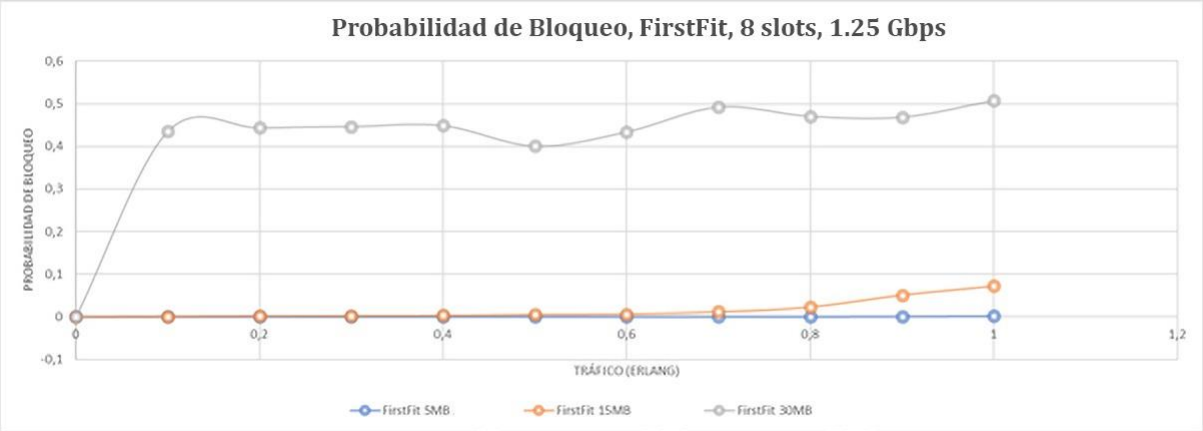

Figura 5.1 Comparativa probabilidad de bloqueo algoritmo first fit con 8 slots y velocidad de 1.25Gbps para cargas de 5, 15 y 30MB

En la figura 5.2 se aumenta la velocidad a 2.5Gbps para analizar el efecto que tiene en la probabilidad de bloqueo del algoritmo, se observa que para cargas bajas de 5MB el algoritmo mantiene su probabilidad de bloqueo en 0 mostrando un comportamiento estable y generando que la red tenga una eficiente gestión del espectro contribuyendo así a su buen desempeño. Para el caso de las cargas con 15MB el algoritmo empieza a tener un crecimiento antes de los 0.8 erlangs y alcanza un máximo de probabilidad de bloqueo de 5% por lo cual mejora el desempeño de la red respecto a la velocidad anterior. En el caso de las cargas con 30MB el algoritmo crece exponencialmente entre 0 y 0.1 erlangs hasta un valor cercano al 25% experimentando así un aumento en la perdida de paquetes dentro de la red, posterior a ello se estabiliza y alcanza máximos de 25% en la probabilidad de bloqueo a los 0.7 y 0.9 erlangs, de esta forma el algoritmo logra mantener la red sin saturarse y permite que haya una menor perdida de paquetes respecto a la velocidad anterior.

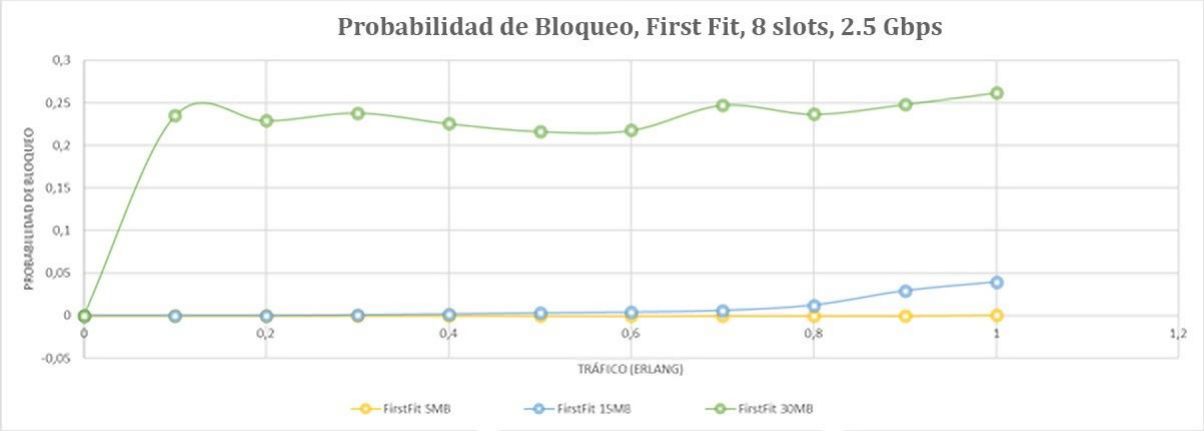

Figura 5.2 Comparativa probabilidad de bloqueo algoritmo first fit con 8 slots y velocidad de 2.5Gbps para cargas de 5, 15 y 30MB

En la figura 5.3 se presenta una comparación en el retardo extremo a extremo para el algoritmo *first fit* y una velocidad de 1.25 Gbps, mostrando las 3 diferentes cargas a la que fue sometida la red. Como se observa en la figura en las cargas bajas con 5MB el retardo extremo a extremo es 0 lo que muestra que el desempeño de la red es ideal y que el algoritmo contribuye a ello, al aumentar la carga a 15MB el retardo aumenta sus valor pero se mantiene constante hasta los 0.8 erlangs en donde disminuye sus valores con 30MB muestra un comportamiento similar, pero presentando una crecimiento en los valores del retardo extremo a extremo, lo cual indica que el algoritmo empieza a tener problemas para realizar las conexiones dentro de la red con estos niveles de tráfico, aun así después de los 0.2 erlangs estos valores empiezan a decrecer ya que el algoritmo se estabiliza y empieza a contribuir a que la red tenga un mejor desempeño y logre establecer de forma correcta las conexiones.

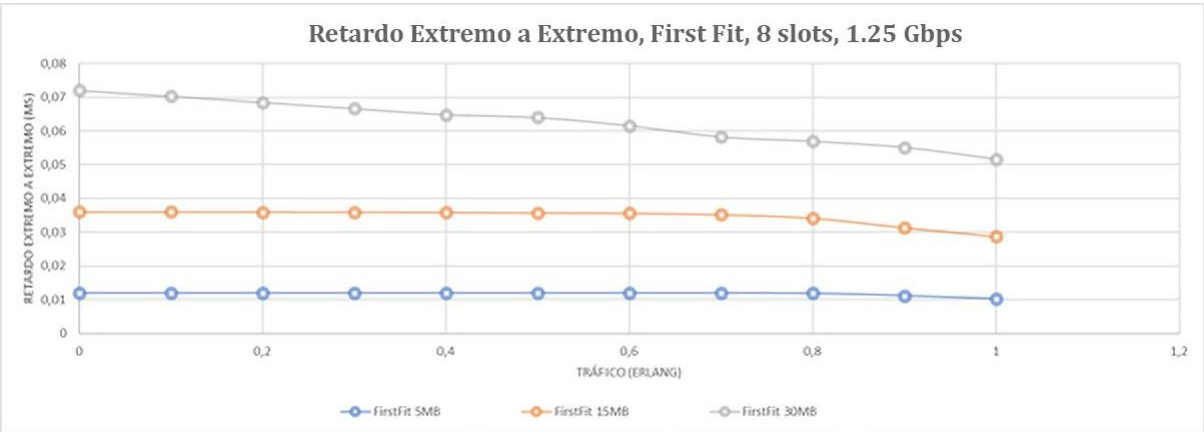

Figura 5.3 Comparativa retardo extremo a extremo algoritmo First Fit a 1.25 Gbps y 8 slots

En la figura 5.4 se muestra la comparación del retardo extremo a extremo al aumentar la velocidad del enlace a 2.5 Gbps, se observa el algoritmo se comporta de forma ideal para cargas bajas con 5MB sin presentar problemas al establecer las conexiones dentro de la red. Con las cargas de 15MB la red ya empieza a mostrar inconvenientes y hay un aumento en el retardo en los enlaces tendiendo a un valor constante hasta los 0.6 erlangs, punto en el cual el algoritmo se estabiliza y contribuye a que la red mejore su desempeño produciendo que haya un decremento en el retardo entre las conexiones. Cuando se somete a valores altos de carga como lo son las de 30MB la red experimenta un crecimiento súbito en el retardo entre sus conexiones alcanzando valores de 0.035ms, aun así, el algoritmo logra mantenerse en un valor constante de retardo y desde los 0.1 erlangs empieza a presentar mayor estabilidad que mejora el desempeño de la red en un valor cercano al 40% y los valores de retardo empiezan a bajar hasta un mínimo de 0.025ms por lo cual el algoritmo contribuye a que la red tenga un mejor desempeño.

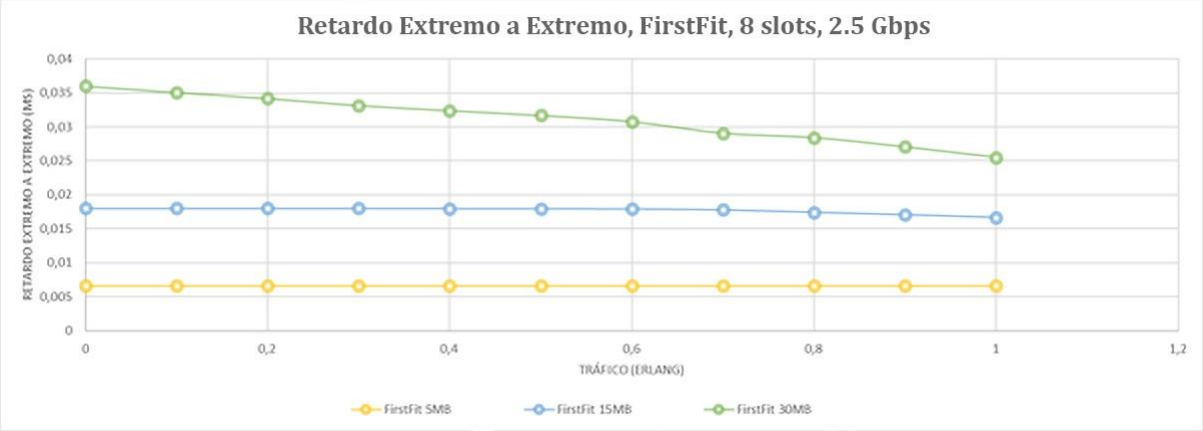

Figura 5.4 Comparativa retardo extremo a extremo algoritmo First Fit a 2.5 Gbps y 8 slots

5.1.3 Algoritmo *regression fit* con 8 slots, velocidad de enlace de 1.25Gbps y 2.5Gbps

En la figura 5.5 se presenta la comparación entre las 3 diferentes cargas de tráfico para el algoritmo regression fit de asignación de espectro a una velocidad de 1.25Gbps y con 8 slots disponibles. Como muestra la figura para la carga de 5MB el algoritmo tiene un comportamiento ideal manteniendo una probabilidad de bloqueo nula por lo que la red tiene un desempeño deseado ya que no pierde paquetes, al aumentar la carga a 15MB el método mantiene un comportamiento estable hasta los 0.7 erlangs en donde la probabilidad de bloqueo experimenta un crecimiento hasta alcanzar un valor máximo del 5% lo que denota que el método tiene eficiencia en la distribución y asignación de espectro contribuyendo a que la red mantenga un desempeño en donde la perdida de paquetes no genere saturación. Cuando la red es sometida a cargas más altas, en este caso de 30MB, la figura muestra que al llegar

a 0.1 erlangs la red tiene un crecimiento exponencial en su probabilidad de bloqueo y llega a un valor cercano al 30% y lo supera por poco hasta que en los 0.2 erlangs se estabiliza y vuelve a bajar su valor y a partir de los 0.3 erlangs se mantiene estable y constante en un valor del 30% hasta los 0.7 erlangs mostrando que el método cognitivo tiende a estabilizarse en ese intervalo y mantener una probabilidad de bloqueo constante pero con un valor alto que puede generar una saturación dentro de la red produciendo problemas en el desempeño y se presente mayor pérdida de paquetes, a partir de los 0.7 erlangs la probabilidad de bloqueo tiene un crecimiento súbito y supera el valor de 35% lo que muestra que el método cognitivo tiene problemas para asignar el espectro de forma eficiente al ser sometida a estas cargas de tráfico altas y ello contribuye a que ocurran problemas con el desempeño de la red.

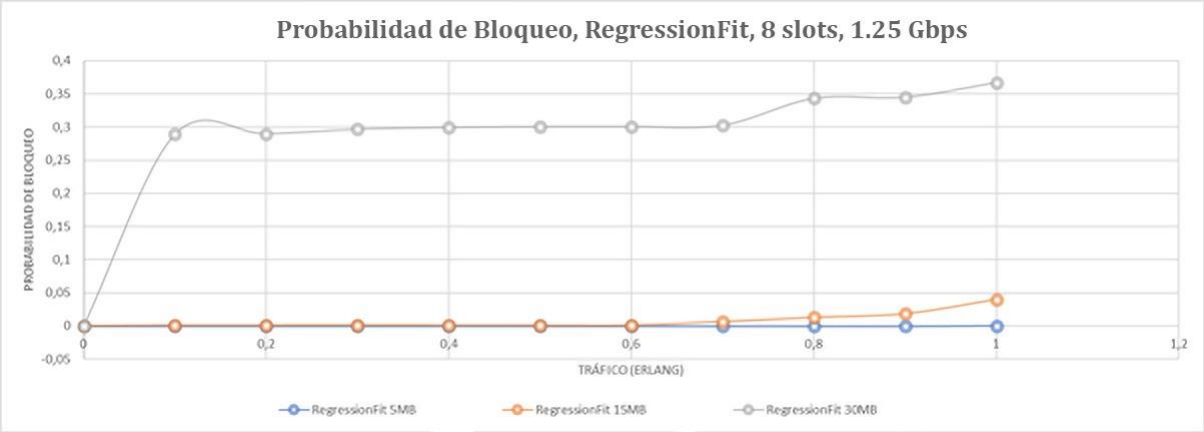

Figura 5.5 Comparativa probabilidad de bloqueo método cognitivo a 1.25Gbps y 8 slots

Como se observa en la figura 5.6 la velocidad de enlace ha sido aumentada a 2.5 Gbps y los valores de la probabilidad de bloqueo han cambiado, para el caso de 5MB se observa que el método cognitivo mantiene su comportamiento ideal al no perder ningún paquete ya que su probabilidad de bloqueo sigue siendo 0, cuando se aumenta la carga de tráfico a 15MB al llegar a los 0.7 erlangs ocurre un aumento en el valor de la probabilidad de bloqueo lo que indica que se empiezan a perder más paquetes, aun así alcanza un valor máximo que no llega ni siquiera al 5% por lo que el método mantiene un comportamiento muy estable y contribuye a que el desempeño de la red se mantenga en un nivel en donde la perdida de paquetes no genere una saturacion, lo que indica que hay una eficiente gestión del espectro óptico. Ya con el valor máximo de carga de trafico de 30MB ocurre un crecimiento súbito en la probabilidad de bloqueo, como se observa en la figura al llegar a los 0.1 erlangs la probabilidad de bloqueo ha aumentado hasta llegar a un 20% superando este umbral y estabilizándose nuevamente en los 0.2 erlangs y manteniendo este valor hasta los 0.5 erlangs, en este lapso el método se mantiene estable, ya a partir de los 0.5 erlangs hay un nuevo incremento en los valores de probabilidad de bloqueo, el método empieza a tener problemas para gestionar lo recurso y alcanza máximos de 25% y

26% lo que puede traer problemas al desempeño de la red ya que son valores cercanos a producir saturación dentro de esta.

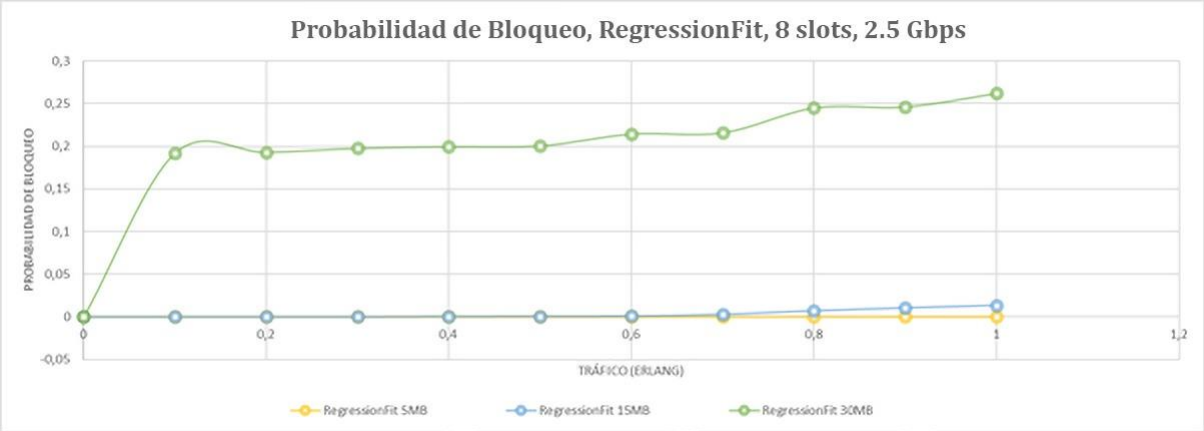

Figura 5.6 Comparativa probabilidad de bloqueo método cognitivo a 2.5 Gbps y 8 Slots.

En la figura 5.7 se compara el retardo extremo a extremo para el método cognitivo a una velocidad de 1.25Gbps y 8 slots, para la carga de 5MB hay un comportamiento ideal ya que las conexiones son establecidas sin mayor inconveniente, al aumentar la carga a 15MB aparecen retrasos en el establecimiento de las solicitudes, la figura muestra como el retardo es mayor respecto al algoritmo *First Fit* y se mantiene un retardo constante y finalmente decrece al llegar a los 0.9 erlangs, para la carga de 30MB el retardo aumenta cerca de 2.2 veces y se mantiene en valores de 0.07ms y cercanos a este valor por debajo, hasta los 0.4 erlangs en donde empieza a decrecer y a experimentar una mejora en el retardo de las conexiones para finalmente alcanzar un valor mínimo de 0.06ms el cual es mayor al conseguido con el algoritmo *first fit.*

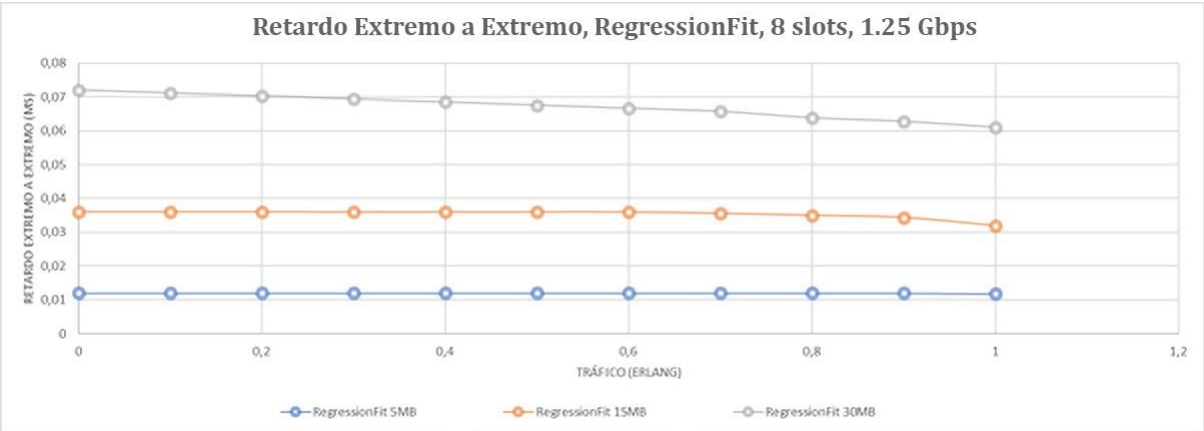

Figura 5.7 Comparativa retardo extremo a extremo método cognitivo a 1.25 Gbps y 8 slots.

En la figura 5.8 se muestra el retardo extremo a extremo al aumentar la velocidad del enlace a 2.5Gbps, como se observa en cargas bajas a 5MB hay un comportamiento esperado y el método se comporta de forma estable experimentando retardos con valores máximos de 0.006ms lo que contribuye a que las conexiones se establezcan

dentro de la red mejorando así su desempeño, con cargas de trafico de 15MB el retardo sufre un aumento considerable llegando a valores que superan los 0.015ms y cercanos a 0.02ms a partir de los 0.6 erlangs comienza a tener una estabilidad y este valor decrece hasta un mínimo de 0.016 mostrando así que el método tienda a estabilizar los retardos en las conexiones. Al aumentar la carga a 30MB este retardo aumenta considerablemente, como muestra la figura 5.6 supera el umbral de los 0.035 hasta los 0.3 erlangs a partir de allí comienza a decrecer alcanzando un mínimo de 0.0226ms, estos valores son más altos que los obtenidos con el algoritmo *First Fit,* como muestra la figura el algoritmo *regression fit* tiene inconvenientes al establecer las solicitudes en un mayor tiempo.

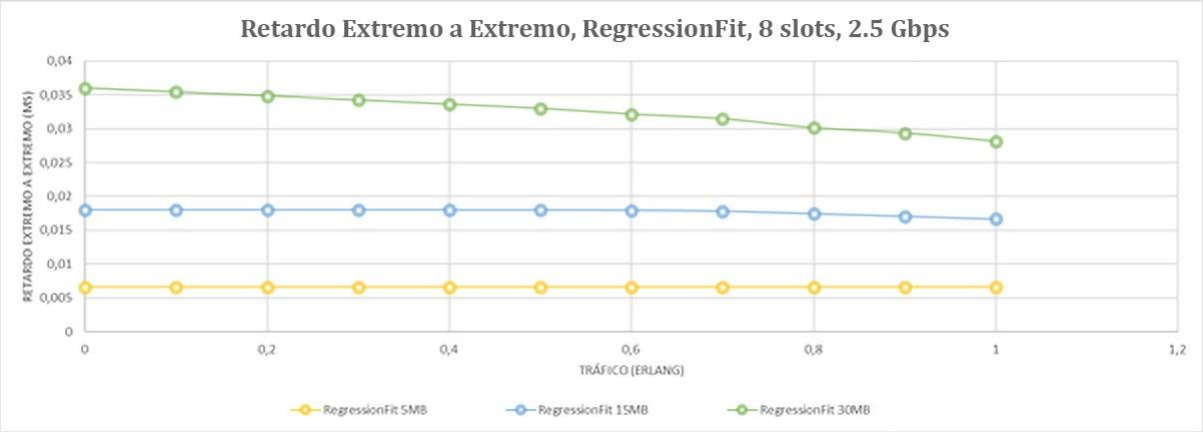

Figura 5.8 Comparativa retardo extremo a extremo método cognitivo a 2.5 Gbps y 8 slots.

En la figura 5.9 se busca hacer una comparación de los promedios obtenidos en los casos de simulación respecto a la probabilidad de bloqueo para ambos métodos, el cognitivo y el algoritmo *First Fit* y con las velocidades de 1.25Gbps y 2.5Gbps. Como se muestra en la figura para la velocidad de 1.25Gbps el método cognitivo *Regression Fit* tiene un mejor comportamiento alcanzando valores promedio de probabilidad de bloqueo por debajo del 15% mientras que el algoritmo *First Fit* llega a valores cercanos al 20% lo que permite observar que se pierden menos paquetes al utilizar el *Regression Fit*. Con la velocidad a 2.5Gbps se mantiene esta tendencia ya que el método *Regression Fit* tiene un promedio de probabilidad de bloqueo inferior al 10% mientras el algoritmo *First Fit* alcanza el 10% e incluso lo supera por poco, esto permite observar que el método cognitivo tiene un comportamiento más estable y contribuye a que la red tenga un mejor desempeño.

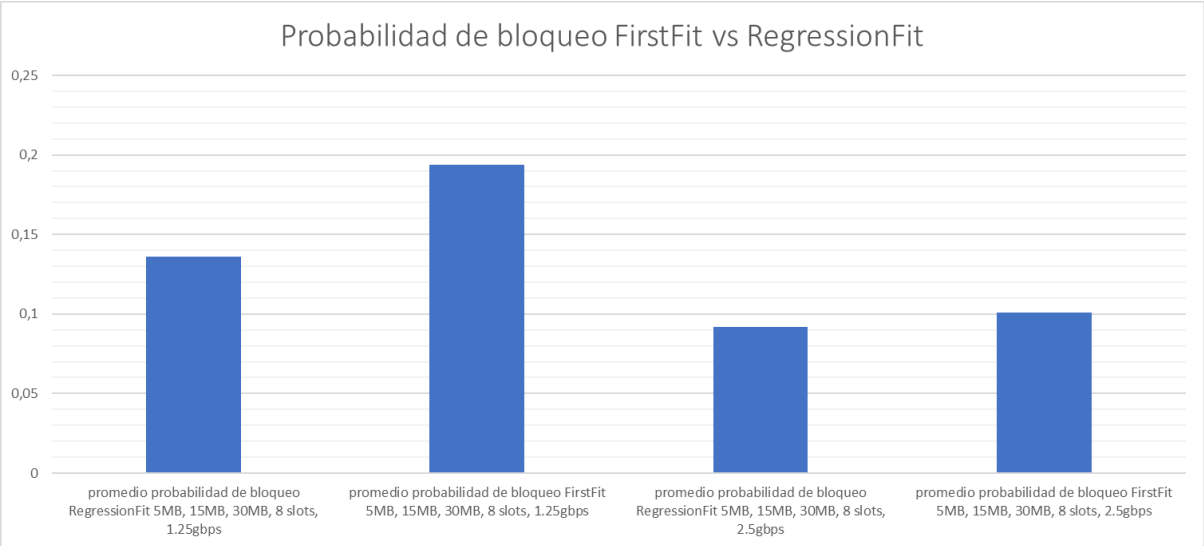

Figura 5.9 Comparativa promedio probabilidad de bloqueo entre algoritmo First Fit y método cognitivo de asignación de espectro Regression Fit

En la figura 5.10 se compara los promedios del retardo extremo a extremo para ambos métodos de asignación de espectro y las velocidades de 1.25Gbps y 2.5 Gbps. Para el caso de 1.25 Gbps hay una diferencia notable ya que el *First Fit* tiene un mejor comportamiento y con un promedio de 0.03ms frente al método cognitivo que alcanza un valor 0.038ms, muy cercano a los 0.04ms, lo que muestra que el método cognitivo tiene problemas con el retardo en las conexiones debido a que tiene procesos adicionales antes de establecerlas. Con las velocidades de 2.5Gbps se mantiene esta tendencia, aunque los valores de retardo disminuyen, el *First Fit* sigue teniendo un mejor desempeño ya que el método cognitivo alcanza un valor de 0.02 ms y el *First Fit* apenas supera los 0-015ms, por lo cual sigue manteniendo un mejor comportamiento respecto al promedio en el retardo entre conexiones.

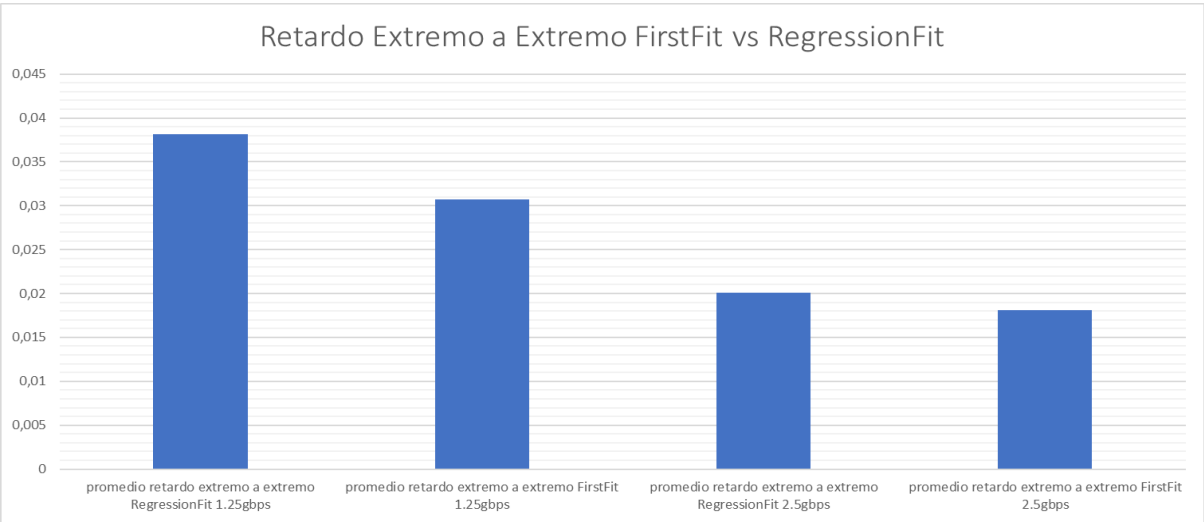

Figura 5.10 Comparativa promedio retardo extremo a extremo entre algoritmo First Fit y método cognitivo Regression Fit.

OMNET++ es un simulador de redes programables que permite realizar escalamiento de tiempo mientras se realizan las pruebas de funcionamiento, por tanto, las simulaciones tardan milisegundos, de esta forma se evidencia que el tiempo de ejecución de c++ es comparable al tiempo de simulación que se asigna en OMNET++, por tanto, este tiene un peso medible que afecta los resultados en los casos de simulación, generando que el Regression Fit tenga un mayor retardo extremo a extremo. En el simulador no se implementaron métricas sobre el grado de fragmentación del espectro en un determinado momento, por lo cual no es posible tener un registro de medidas para comparar los algoritmos en cuanto a esta métrica, sin embargo, los resultados obtenidos en las simulaciones muestran que en todos los casos de estudio el algoritmo Regression Fit tiene una menor probabilidad de bloqueo, lo que en principio indica que tiene un menor grado de fragmentación del espectro.

Debido a que el método cognitivo toma mejores decisiones al momento de escoger la ruta y el espectro para las peticiones, logra tener un valor menor de probabilidad de bloqueo, pero para que pueda tomar estas decisiones, debe realizar una serie de procesos y cálculos descritos anteriormente, que generan que el método tienda a tardar más tiempo en establecer las conexiones, por tanto, su retardo extremo a extremo aumenta. El efecto que causa el algoritmo en la red a largo plazo es que distribuye los slots en 2 conjuntos disjuntos entre sí, esto tiene un efecto de incrementar el número de slots libres alineados en la parte central del espectro, más slots continuos y alineados entre si logran una menor probabilidad de bloqueo.

Como se observa en la figura 5.11, se realizaron algunas pruebas con cargas superiores a los 500MB, en este caso una carga de 2GB, evidenciando una saturación inmediata de la red, lo que demuestra que tanto la red como el método tienen problemas al manejar estas magnitudes en el tráfico, lo que denota que existen limitaciones en la red implementada y en la herramienta de simulación empleada.

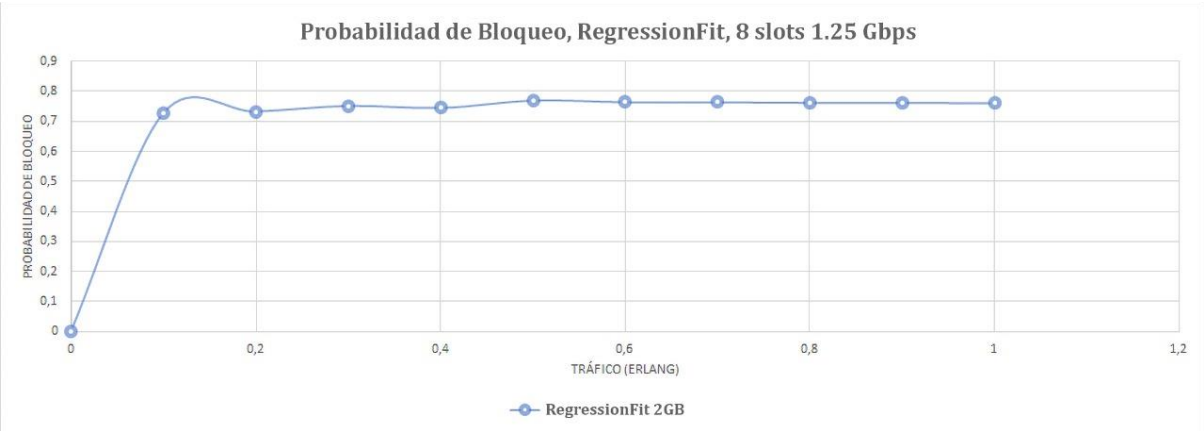

Figura 5.11 Carga de saturación

5.2 CONCLUSIONES

- El diseño de un método cognitivo de asignación de espectro en la red SDON *flexgrid* basado en ML se logró implementando un tipo de ML que fue la regresión lineal y el ciclo ODAA, lo cual permitió que el método diseñado cumpliera con el lazo del ciclo cognitivo y que se soportara en un mecanismo basado en aprendizaje automático.
- La red SDON implementada se evaluó con y sin el método cognitivo, en relación con la perdida de paquetes teniendo en cuenta la probabilidad de bloqueo y al retardo extremo a extremo entre las conexiones, evidenciando que el método propuesto tiene un mejor desempeño respecto a la perdida de paquetes, pero tiene un desempeño menor respecto al retardo extremo a extremo, debido a que el método cognitivo toma mejores decisiones al momento de escoger las rutas y asignar el espectro, pero debido a que realiza procesos extras para tomar dichas decisiones consume mas recursos computacionales y le toma mayor tiempo establecer la alternativa de ruta y espectro.
- El desempeño de la red SDON flexgrid está relacionado con la velocidad de transmisión, ya que en los resultados obtenidos se evidencio que al aumentar la velocidad de los enlaces tanto la probabilidad de bloqueo como el retardo extremo a extremo disminuyen.
- La red diseñada y el método cognitivo propuesto tienen problemas al trabajar con cargas altas de tráfico entre los100 MB y el orden de los GB, lo que denota que existen limitaciones en la herramienta de simulación, al implementar este tipo de redes ópticas y aplicaciones basadas en ML, ya que c++ se ha aplicado poco en este ámbito.
- La aplicación de técnicas de ML a los procesos de gestión del espectro en las redes ópticas puede brindar grandes beneficios en su desempeño ya que hay una mayor eficiencia al gestionar este recurso, porque se puede conocer el estado de la red en tiempo real y tomar decisiones que busquen asignar los recursos como lo es el espectro óptico de una forma más eficiente de acuerdo con las solicitudes que llegan a la red, con su estado actual, lo que permite tomar mejores decisiones.
- Los procesos y ciclos cognitivos pueden contribuir a que las redes ópticas tengan una mejor gestión en sus recursos, tales como el espectro óptico, y por tanto un mejor desempeño ya que las redes pueden tener un conocimiento constante en la disponibilidad de sus recursos y en la forma en cómo se gestiona su uso.
- El proceso de centralización de la red incremento el gasto de recursos computacionales, debido a que se requirió la implementación de módulos que realizaran procesos que requerían de diferentes cálculos que llevaron a consumir más recursos, al igual que la implementación del método cognitivo dentro de la red.

5.3 TRABAJOS FUTUROS

Con el desarrollo de este trabajo de grado surgen nuevas propuestas de investigación.

- Actualizar el módulo BV-WXC para que sea deslizable en el espectro, es decir que pueda deslizar los slots a diferentes posiciones.
- Implementar métricas que midan el grado de fragmentación del espectro en la red.
- Implementar algoritmos de desfragmentación del espectro dentro de esta misma red.
- Analizar el desempeño de la red empleando el método cognitivo basándose en otro modelo de inteligencia artificial tales como algoritmos genéticos.
- Analizar el desempeño del método cognitivo de asignación de espectro basado en ML sobre una red OBS SDON u OPS SDON.

REFERENCIAS

[1] Danda B. Rawat, «Software Defined Networking Architecture,Security and Energy Efficiency: A Survey,» *IEEE,* vol. 19, nº 1, 2017.

[2] Hamid Farhady, «Software-Defined Networking: A survey,» *Computer Networks,* 2015.

[3] Selvaraj, «Need of Algorithm Selection in Next Generation Optical Networks,» *International Journal of Business Data Communications and Networking,* vol. 14, nº 1, 2018.

[4] Yeison Andres Quijano, Daniel Fernando Benavides Quirá, «Analisis Comparativo Del Desepeño entre una Red OPS Distribuida y una Red OPS SDON,» *Universidad del Cauca,* 2019.

[5] Shih-Chun Lin∗, «QoS-aware Adaptive Routing in Multi-layer Hierarchical Software Defined Networks: A Reinforcement Learning Approach,» *IEEE International Conference on Services Computing,* 2016.

[6] Raouf Boutaba, «A Comprehensive Survey on Machine Learning for Networking: Evolution, Applications and Research Opportunities,» *Journal of Internet Services and Applications,* 2018.

[7] Bijoy Chand Chatterjee IEEE, «Routing and Spectrum Allocation in Elastic Optical Networks: A Tutorial,» *IEEE Communications Surveys & Tutorials,* 2015.

[8] Shuiyan Zhang, «Routing and Spectrum Assignment Algorithm with Traffic Prediction and Periodic Rerouting in Elastic Optical Networks,» *IEEE 11th International Conference on Communication Software and Networks,* nº 11, 2019.

[9] Ching-Fang Hsu, «Graph-Model-Based Dynamic Routing and Spectrum Assignment in Elastic Optical Networks», *J. OPT. COMMUN. NETW,* vol. 8, 2016.

[10] Akhilesh Thyagaturu, «Software Defined Optical Networks (SDONs): A Comprehensive Survey,» *IEEE Communications Surveys & Tutorials,* 2016.

[11] Hidalgo, «Diseño e implementación de una aplicación de red bajo la arquitectura SDN, » Bogota: Pontificia Universidad Javeriana, 2014.

[12] D. P. VILLARROEL, Aplicación de SDN en redes opticas: analisis preliminar, valladolid: universidad de Valladolid, 2015.

[13] International Telecomunication Union, «Recommendation ITU-T G.872», 2017.

[14] S. Govind P. Agrawal, «Fiber-Optic Communications Systems», New York: TheInstitute of Optics university of rochester, 2002.

[15] W. Tomasi, «Sistemas de Comunicaciones Electronicas», Phoenyx Arizona: DeVry Institute of Technology, 2003.

[16] M. Á. T. Martín, «Evaluación de Arquitecturas de Red Hibridas OCS/OBS, Barcelona: Escuela Técnica Superior de Ingeniería de Telecomunicación de Barcelona», 2009.

[17] Zhe Ding, «Hybrid routing and spectrum assignment algorithms based on distance-adaptation combined coevolution and heuristics in elastic optical networks,» *Optical Engineering,* 2014.

[18] J. G. L. Perafan, «Diseño De Metodos Cross Layer Cognitivos para Redes de Comunicación Optica de Rafagas (OBS), » Popayan: Universidad del Cauca, 2014.

[19] P. M. Pereira, «Redes Opticas Elasticas, » sao paulo: Universidad de Sao Paulo, Escuela de Ingenieria, 2013.

[20] André C. S. Donza, «Spectrum Allocation Policies in Fragmentation Aware and Balanced Load Routing for Elastic Optical Networks,» *The Ninth International Conference on Digital Society,* 2015.

[21] R. Costa y A. C. Drummond, «New Distance-Adaptive Modulation Scheme for Elastic Optical Networks,» *in IEEE Communications Letters,* vol. 21, nº 2, 2017.

[22] K. Deepak Sharma, «An Overview of Elastic Optical Networks and its Enabling Technologies,» *International Journal of Engineering and Technology,* 2017.

[23] J.P. Elbers, «From Static to Software-Defined Optical Networks,» *16th International Conference on Optical Network Design and Modelling,* 2012.

[24] S. Mayur Channegowda, «Software-Defined Optical Networks Technology and Infrastructure: Enabling Software-Defined Optical Network Operations,» *Journal of Optical Communications and Networking,* vol. 5, nº 10, 2013.

[25] JunHea, «A survey on recent advances in optical communications,» *Computers & Electrical Engineering,* vol. 40, pp. 216-240, 2014.

[26] Abhinava Sadasivarao, «Open Transport Switch: A Software Defined Networking Architecture for Transport Networks,» *Conference: ACM SIGCOMM workshop on Hot topics in Software Defined Networking (HotSDN)At: Hong Kong,* 2013.

[27] R. Nejabati, «Toward a completely softwareized optical network [Invited],» *in Journal of Optical Communications and Networking,* vol. 7, 2015.

[28] .W. Lee, «Design and Implementation of a GPON-Based Virtual OpenFlow-Enabled SDN Switch,» *in Journal of Lightwave Technology,* vol. 34, nº 10, 2016.

[29] A. F. Sevilla Majin y E. C. Zúñiga Quisoboní, «Análisis comparativo del desempeño de algoritmos RSA y RWA sobre una red óptica basada en la topología NSFNET,» Popayan: Universidad del Cauca, 2017.

[30] A. Asensio Garcia, «Elastic spectrum allocation in flexgrid optical networks, » Barcelona: Universidad Politecnica de Cataluña, 2012.

[31] S. Shakya, «Management of Spectral Resources in Elastic Optical Networks, » Dissertation, Georgia: State University, 2015.

[32] A. Mayoral López de Lerma, «Algoritmos de planificación para redes elásticas, » Madrid: Universidad Autónoma de Madrid. Departamento de Tecnología Electrónica y de las Comunicaciones, 2013.

[33] I. Olszewski, «Algorithms of Routing and Spectrum Assignment in Spectrum Flexible Transparent Optical Networks,» *Przeglad Elektrotechniczny ,* vol. 89, 2013.

[34] G. Nadeau, SDN: Software Defined Networks: an authoritative review of network programmability technologies., O'Reilly Media, Inc, 2013.

[35] R. Sedgewick, Algorithms in C++ Part 5: Graph Algorithms 3rd Edición, Princeton: Addison-Wesley Professional, 2001.

[36] K. Jafar Alzubi1, «Machine Learning from Theory to Algorithms: An Overview,» *Journal of Physics: Conference Series, Second National Conference on Computational Intelligence,* vol. 1142, 2018.

[37] C. Artacho Gómez, «Desarrollo y aplicación de técnicas de Machine Learning para la predicción de contagios por Covid-19,» Sevilla: Universidad de Sevilla, 2021.

[38] M. Cuadrado, «Utilizacion del Machine Learning en la Indrustria 4.0, » Valladolid: Universidad de Valladolid, 2019.

[39] W. Thomas, «Cognitive networks: adaptation and learning to achieve end-to-end performance objectives,» *IEEE Communications Magazine,* vol. 44, nº 12, 2006.

[40] F. D'Angiolo, «Algoritmos de regresión lineal aplicados al mantenimiento de un datacenter,» *XXV Congreso Argentino de Ciencias de la Computación (CACIC),* pp. 12-21, 2019.

[41] Q. e. Mahmoud, «Cognitive networks: towards self-aware networks», John Wiley & Sons, 2007.

[42] F. Akyildiz, «A survey on spectrum management in cognitive radio networks,» *IEEE Communications Magazine,* vol. 46, nº 4, pp. 40-48, 2004.

[43] J. L. Rivera Hurtado, «Método de control cognitivo aplicado al enrutamiento de una red bajo arquitectura SDN/NFV, » Popayan: Universidad del Cauca, 2018.

[44] D. Moncada Zarzosa, «Implementación de un simulador de redes ópticas metropolitanas flexibles (gridless), Valladolid: Universidad de Valladolid, 2016.

[45] R. S. Pressman, «Ingenieria de Softwre un enfoque practico,» University of Connecticut, 2010.

[46] J. V. Capella Hernández, «Introducción al simulador de redes NS-2, » Valencia, España: Universidad Politecnica de Valencia, 2011.

[47] H. Varga, «An overview of the OMNeT++ simulation environment,» *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops,* pp. 1-10, 2008.

[48] O. Ltd, «OMnet ++ user Manual, » 2014.

[49] J. Morales, «Simulando con OMNET: Selección de la herramienta y su utilización, » Universidad Icesi, 2013.

[50] G. López Lecumberri, «Simulación de redes de computadores con OMNeT++4, » Pamplona: Universidad Publica de Navarra, 2011.

[51] OPNET, «Manual de Usuario, » Barcelona: Universidad Politecnica de Cataluña, 2005.

[52] d. I. Telematica, «Aquitectura de Redes Sistemas y Servicios: Conceptos de simulacion y OPNET, » Universidad Politecnica de Navarra, 2011.

[53] C. Fernández, «Manual Basico de Matlab, » Madrid: Servicios Informáticos U.C.M , 2012.

[54] P. Julio Benitez Lopez, «Introduccion a MATLAB, » Valencia, España: Universidad Politecnica de Valencia, 2011.

[55] S. Valentín Gregori Gregori, «Matlab, matrices y transformaciones geometricas en el plano y en el espacio, » Valencia, España: Universitat Politècnica de València, 2021.

[56] B. Stromatas, «Lidar signal simulation for the evaluation of aerosols in chemistry transport models,» *Geoscientific Model Development,* vol. 5, nº 6, 2012.

[57] L. d. M. Dynamique, «OPTSIM 1.1 User´s Guide, » Institut Pierre-Simon Laplace, 2014.

[58] F. P. Neyra, «Depliegue de un controlador SDN basado en ONOS, Barcelona: Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona, Departament d'Arquitectura de Computadors, 2016.

[59] V. Marian, «An SDN Architecture for IoT Networks Using ONOS Controller.,» *Conference: 2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet),* pp. 1-6, 2020.

[60] K. Bob Lantz, «Demonstration of Software-Defined Packet-Optical Network Emulation with Mininet-Optical and ONOS,» *Optical Fiber Communication Conference (OFC) 2020,* 2020.

[61] H. RAMIREZ, «Guia de Implementacion y Uso del Emulador de Redes Mininet, » Pereira: Universidad Tecnologica de Pereira, 2015.

[62] S. C. López, «Estudio de redes SDN mediante Mininet y Miniedit, » Valencia: Universidad Politecnica de Valencia, 2019.

[63] I. T. Union, «ITU-T Recommendation G.694.1, » ITU-T SERIES G, 2002.

[64] U.I. Telecomunicaciones, «Recomendación UIT-T G.694.2, » ITU-T SERIES G, 2003.

[65] E. O. ByBijoy Chand Chatterjee, «Elastic Optical Networks: Fundamentals, Design, Control, and Management, » CRC Press., 2020.

[66] D. I. G. Horna, «Evaluación de algoritmos de asignación de recursos que ofrezcan protección en redes ópticas elásticas, » Valladolid, España: Universidad de Valladolid, 2017.

ANEXOS

1. Archivo de configuración de red Omnet.ini

```
[General]
network = networks.Nsfnet
include ./networks/NsfnetParams.ini
record-eventlog = false
seed-0-mt=4  # or any other 32-bit value

**.topology.numPaths = 3 #1,2,max3
**.topology.assignedRoute = 0 #0,1,2
**.app.slotRandomSize = intuniform(1, 16)  #num random slots
**.app.sendIaTime = uniform(100ms, 101ms)

**.app.packetLength = 10MB

**.spectrum.assignmentAlgorithm = "RegressionFit"  #"FirstFit", "LastFit",
"RegressionFit"

**.channelBandwidth = 1THz
**.slotBandwidth = 12.5GHz #based on ITU G694.1 recommendation
```

---

2. Archivo de configuración de topología de red Nsfnet.ned

```
network Nsfnet {
    parameters:
        @display("bgb=786,562;bgi=maps/usa;bgu=km");
        double slotBandwidth @unit(Hz);
        double channelBandwidth @unit(Hz);
    types:
        channel OpticalChannel extends ned.DatarateChannel        {
            delay = default(uniform(10us, 20us));
//            datarate = default(uniform(11Gbps, 12Gbps));
            datarate = 1.25Gbps;
        }
    submodules:
        node[14]: Node {
            parameters:
                @display("i=block/optical_s;p=$xpos,$ypos");
                address = index;
        }
```

```
    controller: Controller {
        @display("p=402.992,80.372");
        slotBandwidth = slotBandwidth;
        channelBandwidth = channelBandwidth;
    }
connections allowunconnected:
    for i=0..int((channelBandwidth / slotBandwidth) - 1) {
        node[0].portOut++ --> OpticalChannel --> node[1].portIn++; //1
        node[0].portIn++ <-- OpticalChannel <-- node[1].portOut++;
    }
    for i=0..int((channelBandwidth / slotBandwidth) - 1) {
        node[0].portOut++ --> OpticalChannel --> node[2].portIn++; //2
        node[0].portIn++ <-- OpticalChannel <-- node[2].portOut++;
    }
    for i=0..int((channelBandwidth / slotBandwidth) - 1) {
        node[0].portOut++ --> OpticalChannel --> node[3].portIn++; //3
        node[0].portIn++ <-- OpticalChannel <-- node[3].portOut++;
    }

    for i=0..int((channelBandwidth / slotBandwidth) - 1) {
        node[1].portOut++ --> OpticalChannel --> node[6].portIn++; //4
        node[1].portIn++ <-- OpticalChannel <-- node[6].portOut++;
    }
    for i=0..int((channelBandwidth / slotBandwidth) - 1) {
        node[1].portOut++ --> OpticalChannel --> node[2].portIn++; //5
        node[1].portIn++ <-- OpticalChannel <-- node[2].portOut++;
    }
    for i=0..int((channelBandwidth / slotBandwidth) - 1) {
        node[2].portOut++ --> OpticalChannel --> node[7].portIn++; //6
        node[2].portIn++ <-- OpticalChannel <-- node[7].portOut++;
    }
    for i=0..int((channelBandwidth / slotBandwidth) - 1) {
        node[3].portOut++ --> OpticalChannel --> node[4].portIn++; //7
        node[3].portIn++ <-- OpticalChannel <-- node[4].portOut++;
    }
    for i=0..int((channelBandwidth / slotBandwidth) - 1) {
        node[3].portOut++ --> OpticalChannel --> node[8].portIn++; //8
        node[3].portIn++ <-- OpticalChannel <-- node[8].portOut++;
    }
    for i=0..int((channelBandwidth / slotBandwidth) - 1) {
        node[4].portOut++ --> OpticalChannel --> node[7].portIn++; //9
        node[4].portIn++ <-- OpticalChannel <-- node[7].portOut++;
    }
    for i=0..int((channelBandwidth / slotBandwidth) - 1) {
```

```
    node[4].portOut++ --> OpticalChannel --> node[5].portIn++; //10
    node[4].portIn++ <-- OpticalChannel <-- node[5].portOut++;
}
for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[5].portOut++ --> OpticalChannel --> node[6].portIn++; //11
    node[5].portIn++ <-- OpticalChannel <-- node[6].portOut++;
}
for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[6].portOut++ --> OpticalChannel --> node[9].portIn++; //12
    node[6].portIn++ <-- OpticalChannel <-- node[9].portOut++;
}
for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[7].portOut++ --> OpticalChannel --> node[12].portIn++; //13
    node[7].portIn++ <-- OpticalChannel <-- node[12].portOut++;
}
for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[7].portOut++ --> OpticalChannel --> node[11].portIn++; //14
    node[7].portIn++ <-- OpticalChannel <-- node[11].portOut++;
}
for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[8].portOut++ --> OpticalChannel --> node[10].portIn++; //15
    node[8].portIn++ <-- OpticalChannel <-- node[10].portOut++;
}
for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[8].portOut++ --> OpticalChannel --> node[13].portIn++; //16
    node[8].portIn++ <-- OpticalChannel <-- node[13].portOut++;
}
for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[9].portOut++ --> OpticalChannel --> node[10].portIn++; //17
    node[9].portIn++ <-- OpticalChannel <-- node[10].portOut++;
}
for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[9].portOut++ --> OpticalChannel --> node[13].portIn++; //18
    node[9].portIn++ <-- OpticalChannel <-- node[13].portOut++;
}
for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[9].portOut++ --> OpticalChannel --> node[11].portIn++; //19
    node[9].portIn++ <-- OpticalChannel <-- node[11].portOut++;
}
for i=0..int((channelBandwidth / slotBandwidth) - 1) {
    node[12].portOut++ --> OpticalChannel --> node[13].portIn++; //20
    node[12].portIn++ <-- OpticalChannel <-- node[13].portOut++;
}
for i=0..int((channelBandwidth / slotBandwidth) - 1) {
```

```
        node[12].portOut++ --> OpticalChannel --> node[10].portIn++; //21
        node[12].portIn++ <-- OpticalChannel <-- node[10].portOut++;
    }
}
```

3. Algoritmo para la generación y envío de paquetes

```
if (msg == generatePacket) {

    char msgname[40];
    int src = getParentModule()->getIndex(); // our module index
    int size = getParentModule()->getVectorSize() - 2;
    int dst = intuniform(0, size);
    if (dst >= src)
        dst++;
    int ns = slotRandomSize;

    sprintf(msgname, "%i-%i-ns%i", src, dst, ns);
    OpticalMsg *opmsg = new OpticalMsg(msgname);
    opmsg->setSrcAddr(src);
    opmsg->setDestAddr(dst);
    opmsg->setSlotReq(ns);
    opmsg->setMsgState(LIGHTPATH_REQUEST);
    opmsg->setColor(intrand(16777213));
    opmsg->setByteLength(packetLength);
    opmsg->setKind(intuniform(0, 7));
    send(opmsg, "out");
    numSent++;

    scheduleAt(simTime() + sendIATime->doubleValue(), generatePacket);
  }
```

4. Algoritmo de comportamiento del dispositivo BVWXC dependiendo del estado del paquete.

```
OpticalMsg *rcvMsg = check_and_cast<OpticalMsg*>(msg);
int src = rcvMsg->getSrcAddr();
int dst = rcvMsg->getDestAddr();
int id = rcvMsg->getId();
int state = rcvMsg->getMsgState();
std::string cTime = std::to_string(rcvMsg->getCreationTime().dbl());
   if (state == LIGHTPATH_REQUEST) {
      cModule *control = getParentModule()->getParentModule()-
>getSubmodule("controller");
      sendDirect(msg, control, "in");
```

```
    }
    if (state == LIGHTPATH_ASSIGNMENT) {
        tmpCount = 0;
        rcvMsg->setHopCount(rcvMsg->getHopCount() + 1);
        std::ifstream ifs("./node/data/RoutingTable.csv");

        if (ifs.is_open()) {
            std::string node;
            std::string gate;
            std::string msgid;
            std::string sltid;
            std::string available;
            while (ifs.good()) {
                std::getline(ifs, node, ',');
                std::getline(ifs, gate, ',');
                std::getline(ifs, msgid, ',');
                std::getline(ifs, sltid, ',');
                std::getline(ifs, available, '\n');
                if (node.length() != 0 && gate.length() != 0 && msgid.length() != 0 &&
sltid.length() != 0 && available.length() != 0) {
                    if (getParentModule()->getIndex() == std::stoi(node) && std::stoi(msgid)
== id && tmpCount == 0) {

                        if (!getParentModule()->gate("portOut", std::stoi(gate))-
>getTransmissionChannel()->isBusy()) {
                            send(msg, "out", std::stoi(gate));
                        }
                        tmpCount++;
                    }
                    else if (getParentModule()->getIndex() == std::stoi(node) &&
std::stoi(msgid) == id && tmpCount != 0) {

                        if (!getParentModule()->gate("portOut", std::stoi(gate))-
>getTransmissionChannel()->isBusy()) {
                            send(msg->dup(), "out", std::stoi(gate));
                        }
                        tmpCount++;
                    }
                }
            }
            ifs.close();
        }
    }
```

5. Despliegue topología en pantalla

```
   int pathFound = 0;
   assignedRoute = 0;
   OpticalMsg *rcvMsg = check_and_cast<OpticalMsg*>(msg);
   int src = rcvMsg->getSrcAddr();
   int dst = rcvMsg->getDestAddr();
   int state = rcvMsg->getMsgState();
   int numSlots = rcvMsg->getSlotReq();
   int blue = rcvMsg->getColor() / 65536;
   int green = (rcvMsg->getColor() - blue * 65536) / 256;
   int red = rcvMsg->getColor() - blue * 65536 - green * 256;
   int pack_id = rcvMsg->getId();
   int slot = 0;
   rcvMsg->setMsgState(LIGHTPATH_ROUTING);
   cTopology *topo = new cTopology("topo");
   std::vector<std::string> nedTypes;
   nedTypes.push_back(getParentModule()->getParentModule()-
>getSubmodule("node", 0)->getNedTypeName());
   topo->extractByNedTypeName(nedTypes);
   for (int i = 0; i < topo->getNumNodes(); i++) {
      cTopology::Node *srcN = topo->getNodeFor(getParentModule()-
>getParentModule()->getSubmodule("node", i));
      int numOutLinks = srcN->getNumOutLinks();
      for (int j = 0; j < numOutLinks; j++) {
         if (true) {
            cDisplayString &dStr = srcN->getLinkOut(j)->getLocalGate()-
>getDisplayString();
            dStr.parse("ls=#939393,0");
         }
      }
   }
```

6. Algoritmo para el descubrimiento de nuevas rutas utilizando la teoría de grafos y almacenamiento de estas en el archivo Routes.csv

```
cTopology::Node *srcNode = topo->getNodeFor(getParentModule()-
>getParentModule()->getSubmodule("node", src));
   cTopology::Node *targetNode = topo->getNodeFor(getParentModule()-
>getParentModule()->getSubmodule("node", dst));
   std::vector<cTopology::LinkIn*> route;
   std::vector<cTopology::LinkIn*> path;
   std::vector<std::vector<cTopology::LinkIn*>> pathList;
   std::vector<std::vector<cTopology::LinkIn*>> routeList;
   std::vector<int> nodeDis;
```

```
    for (int i = 0; i < topo->getNumNodes(); i++) {
        nodeDis.push_back(MAXIMUM);
        path.push_back(nullptr);
    }
    nodeDis.at(targetNode->getModule()->getIndex()) = 0;
    for (int idx = 0; idx < nodeDis.size(); idx++) {
    }
    std::deque<cTopology::Node*> q;
    q.push_back(targetNode);
    while (!q.empty()) {
        cTopology::Node *v = q.front();
        q.pop_front();
        for (int i = 0; i < v->getNumInLinks(); i++) {
            cTopology::Node *w;
            w = v->getLinkIn(i)->getRemoteNode();

            if (w == srcNode) {
                pathFound++;
                nodeDis.at(w->getModule()->getIndex()) = 1 + nodeDis.at(v->getModule()-
>getIndex());
                path.at(w->getModule()->getIndex()) = v->getLinkIn(i);
                pathList.push_back(path);
                for (int idx = 0; idx < nodeDis.size(); idx++) {
                }
                break;
            }
            int dist = nodeDis.at(w->getModule()->getIndex());
            if (dist == MAXIMUM) {
                nodeDis.at(w->getModule()->getIndex()) = 1 + nodeDis.at(v->getModule()-
>getIndex());
                path.at(w->getModule()->getIndex()) = v->getLinkIn(i);
                q.push_back(w);
            }
            if (pathFound == numPaths)
                break;
        }
        if (pathFound == numPaths)
            break;
    }
    while (srcNode != targetNode) {
        for (int i = 0; i < topo->getNumNodes(); i++) {
            if (path[i] != nullptr) {
                if (srcNode == path[i]->getRemoteNode()) {
                    route.push_back(path[i]);
```

```
                srcNode = path[i]->getLocalNode();
            }
        }
    }
    for (int i = 0; i < pathList.size(); i++) {
        srcNode = topo->getNodeFor(getParentModule()->getParentModule()-
>getSubmodule("node", src));
        routeList.push_back(std::vector<cTopology::LinkIn*>());
        while (srcNode != targetNode) {
            for (int j = 0; j < pathList[i].size(); j++) {
                if (pathList[i][j] != nullptr) {
                    if (srcNode == pathList[i][j]->getRemoteNode()) {
                        routeList[i].push_back(pathList[i][j]);
                        srcNode = pathList[i][j]->getLocalNode();
                    }
                }
            }
        }
    }
    std::vector<const char*> cols = { "ls=#ffffff,5", "ls=#dddddd,5", "ls=#999999,5",
"ls=#666666,5", "ls=#333333,5", "ls=#111111,5" };
    for (int i = 0; i < routeList.size(); i++) {
        for (int j = 0; j < routeList[i].size(); j++) {
            cDisplayString &dispStr = routeList[i][j]->getRemoteGate()-
>getDisplayString();
            dispStr.parse(cols[i]);
        }
    }
    int routeid = 0;
    std::string fileName("./node/data/Routes.csv");
    std::ofstream routingTable;
    for (std::vector<cTopology::LinkIn*> routes : routeList) {
        for (cTopology::LinkIn *tmp : routes) {
            cDisplayString &dispStr = tmp->getRemoteGate()->getDisplayString();
            char conn_id[25];
            int lnk_id = tmp->getRemoteGate()->getConnectionId() + 1;
            sprintf(conn_id, "lnk: %d", lnk_id);
            char tcol[30];
            sprintf(tcol, "ls=#%X%X%X,5", red, green, blue);
            int gate = tmp->getRemoteGate()->getIndex();
            int node = tmp->getRemoteNode()->getModule()->getIndex();
            int id = tmp->getRemoteGate()->getConnectionId();
            char color[30];
```

```
            sprintf(color, "#%.2X%.2X%.2X", red, green, blue);
            routingTable.open(fileName, std::ios_base::app);
            routingTable << node << "," << gate << "," << pack_id << "," << id << "," <<
color << "," << routes.size() << "," << routeid << endl;
            routingTable.close();
        }
        routeid++;
    }
    delete topo;
    cModule *spec = getParentModule()->getSubmodule("topology");
    sendDirect(rcvMsg, spec, "directTopoOut");
```

---

7. Inicialización de la matriz de enlaces de la red y dibujado en pantalla de la rejilla de slots.

```
slotBandwidth = par("slotBandwidth");
    channelBandwidth = par("channelBandwidth");
    assignmentAlgorithm = par("assignmentAlgorithm");
    slotSize = (channelBandwidth / slotBandwidth);
    canvas = getParentModule()->getParentModule()->getCanvas(); // toplevel canvas

    numProcessed = 0;
    numLost = 0;
    WATCH(numProcessed);
    WATCH(numLost);

    if (stage == 1) {

        cTopology *topo = new cTopology("topo");
        topo->extractByModulePath(cStringTokenizer("**.node[*]").asVector());
        numLinks = 0;
        int slotIndex = 0;
        for (int i = 0; i < topo->getNumNodes(); i++) {
            cTopology::Node *srcNode = topo->getNodeFor(getParentModule()-
>getParentModule()->getSubmodule("node", i));
            int numOutLinks = srcNode->getNumOutLinks();
            for (int j = 0; j < numOutLinks; j++) {
                int node = srcNode->getModule()->par("address");
                int dst = topo->getNode(i)->getLinkOut(j)->getRemoteNode()-
>getModule()->par("address");
                int gate = srcNode->getLinkOut(j)->getLocalGate()->getIndex();
                int slotId = srcNode->getLinkOut(j)->getLocalGate()->getConnectionId();

                if (j % slotSize == 0) {
                    Link *l = new Link(slotId, true, numLinks);
```

```
                l->addSlot(slotId, 0, slotIndex, node, gate);
                linkMatrix.push_back(l);
                numLinks++;

            }
            else {
                linkMatrix.back()->addSlot(slotId, 0, slotIndex, node, gate);
            }
            slotIndex = (slotIndex + 1) % slotSize;
        }
    }
    drawSlotGrid();
    delete topo;
  }
}
```

8. Algoritmo de selección y asignación de espectro

```
  updateSlotGrid();
   cleanSlotGrid();
   numProcessed++;
   routeLinks.clear();
   saveResults();
   char msgname[20];
   OpticalMsg *msgPath = check_and_cast<OpticalMsg*>(msg);
   int src = msgPath->getSrcAddr();
   int dst = msgPath->getDestAddr();
   int slreq = msgPath->getSlotReq();
   int id = msgPath->getId();
   int blue = msgPath->getColor() / 65536;
   int green = (msgPath->getColor() - blue * 65536) / 256;
   int red = msgPath->getColor() - blue * 65536 - green * 256;

   msgPath->setMsgState(LIGHTPATH_ASSIGNMENT);
   cFigure::Color col = cFigure::Color(red, green, blue);
   cTopology *topo = new cTopology("topo");
   std::vector<std::string> nedTypes;
   nedTypes.push_back(getParentModule()->getParentModule()-
>getSubmodule("node", 0)->getNedTypeName());
   topo->extractByNedTypeName(nedTypes);
   std::ifstream ifs("./node/data/Routes.csv");
   if (ifs.is_open()) {
      std::string node;
      std::string gate;
      std::string msgid;
```

```
        std::string lnkid;
        std::string color;
        std::string hops;
        std::string routeid;
        while (ifs.good()) {
            std::getline(ifs, node, ',');
            std::getline(ifs, gate, ',');
            std::getline(ifs, msgid, ',');
            std::getline(ifs, lnkid, ',');
            std::getline(ifs, color, ',');
            std::getline(ifs, hops, ',');
            std::getline(ifs, routeid, '\n');
            if (node.length() != 0 && gate.length() != 0 && msgid.length() != 0 &&
lnkid.length() != 0 && color.length() != 0 && hops.length() != 0 && routeid.length() !=
0) {

                if (id == std::stoi(msgid) && 0 == std::stoi(routeid)) {
                    cTopology::Node *tmpNode = topo->getNodeFor(getParentModule()-
>getParentModule()->getSubmodule("node", std::stoi(node)));
                    routeLinks.push_back(searchLink(std::stoi(lnkid)));
                }
            }
        }
        ifs.close();
    }
    std::vector<int> temporal = contiguousSpectrum(routeLinks);
    std::vector<std::vector<int>> slts = continuousSpectrum(temporal);

    std::vector<int> slotFF;
    std::string firstFit = "FirstFit";
    std::string lastFit = "LastFit";
    std::string regFit = "RegressionFit";
    if (std::strcmp(assignmentAlgorithm, firstFit.c_str()) == 0) {
        slotFF = algorithmFirstFit(slts, routeLinks, slreq, col, id);
    }
    else if (std::strcmp(assignmentAlgorithm, lastFit.c_str()) == 0) {
        slotFF = algorithmLastFit(slts, routeLinks, slreq, col, id);
    }
    else if (std::strcmp(assignmentAlgorithm, regFit.c_str()) == 0) {
        slotFF = algorithmRegFit(slts, routeLinks, slreq, col, id);
    }
    if (slotFF.size() > 0) {
        std::string fileName("./node/data/RoutingTable.csv");
        std::ofstream routingTable;
```

```
        for (Link *lnk : linkMatrix) {
            for (Slot *sl : lnk->slots) {
                if (sl->m_available == 1 && sl->m_msgid == id) {
                    int sl_id = sl->m_id;
                    int node = sl->m_node;
                    int gate = sl->m_gate;
                    int available = sl->m_available;
                    routingTable.open(fileName, std::ios_base::app);
                    routingTable << node << "," << gate << "," << id << "," << sl_id << "," <<
available << endl;
                    routingTable.close();
                }
            }
        }

        cModule *srcNode = getParentModule()->getParentModule()-
>getSubmodule("node", src)->getSubmodule("bvwxc");
        sendDirect(msgPath, srcNode, "directIn");
    }
    else {
        numLost++;
        getParentModule()->bubble("lost packet");
        std::ifstream ifs("./node/data/RoutingTable.csv");
        std::ofstream tmp;
        if (ifs.is_open()) {
            tmp.open("./node/data/tmp.csv");
            std::string node;
            std::string gate;
            std::string msgid;
            std::string slid;
            std::string available;
            while (ifs.good()) {
                std::getline(ifs, node, ',');
                std::getline(ifs, gate, ',');
                std::getline(ifs, msgid, ',');
                std::getline(ifs, slid, ',');
                std::getline(ifs, available, '\n');

                if (node.length() != 0 && gate.length() != 0 && msgid.length() != 0 &&
slid.length() != 0 && available.length() != 0) {
                    if (id != std::stoi(msgid)) {
                        tmp << node << "," << gate << "," << msgid << "," << slid << "," <<
available << endl;
                    }
```

```
                else if (id == std::stoi(msgid)) {
                    std::string zero("0");
                    tmp << node << "," << gate << "," << msgid << "," << slid << "," <<
zero << endl;
                }
            }
        }
        tmp.close();
        ifs.close();
    }
    std::remove("./node/data/RoutingTable.csv");
    std::rename("./node/data/tmp.csv", "./node/data/RoutingTable.csv");
    delete msgPath;
  }
}
```