ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

COMUNICACIONES HABILITADAS POR DRONES: REQUISITOS, TECNOLOGÍAS Y AUTOMATIZACIÓN

IMPLEMENTACIÓN DE NETWORK SLICING EN UN AMBIENTE FLYING AD-HOC NETWORK (FANET)

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TELECOMUNICACIONES

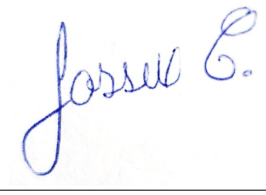
JOSSUE ERNESTO CAMACHO VILLARROEL
jossue.camacho@epn.edu.ec

DIRECTOR: ING. CHRISTIAN JOSE TIPANTUÑA TENELEMA M.Sc.
christian.tipantuna@epn.edu.ec

DMQ, Octubre 2022

CERTIFICACIONES

Yo, JOSSUE ERNESTO CAMACHO VILLARROEL declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento



JOSSUE ERNESTO CAMACHO VILLARROEL

Certifico que el presente trabajo de integración curricular fue desarrollado por JOSSUE ERNESTO CAMACHO VILLARROEL, bajo mi supervisión

M.Sc. CHRISTIAN JOSE TIPANTUÑA TENELEMA
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el producto resultante del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

JOSSUE ERNESTO CAMACHO VILLARROEL

M.Sc. CHRISTIAN JOSE TIPANTUÑA TENELEMA

DEDICATORIA

Dedico este trabajo de titulación a mis padres por su apoyo incondicional y por haber creído en mi.

A mi hermana Jazmín por ser un gran punto de apoyo en mi vida y por ser esa inspiración que me motiva a mejorar.

A mis mascotas que con su fidelidad y compañía endulzaron difíciles momentos y estuvieron conmigo en cada paso de mi desarrollo.

Finalmente, dedico este trabajo a todos mis amigos y familiares que de distintas maneras me hicieron la persona que ahora soy tanto a nivel humano como profesional.

Este logro es para ustedes,

Jossue Camacho

AGRADECIMIENTOS

Agradezco a todas las personas cuya ayuda me permitió culminar este trabajo. De manera especial a mis padres cuyo infaltable apoyo me permitió llegar hasta aquí.

Agradezco a mi tutor, Dr.-Ing. Hernán Barba, por su apoyo durante los períodos más cruciales de mi carrera, por inspirarme a aprender y por fomentar mi desarrollo profesional.

A mi director de tesis, M.sC. Christian Tipantuña, por su guía durante el desarrollo de este trabajo de titulación.

A mi serendipia, Melanny Dávila, por sus consejos, su apoyo y sobre todo por inspirarme a ser un buen profesional.

A mi compañera de alegrías y sufrimientos, Mariela Anaguano, por acompañarme en cada paso de mi desarrollo académico.

Finalmente, agradezco mis amigos, por la ayuda y por los buenos momentos.

ÍNDICE DE CONTENIDO

RESUMEN

Las demandas de comunicación para el mundo actual han evolucionado junto con el desarrollo de la tecnología provocado que las redes celulares tradicionales sufran un cambio en la manera que prestan sus recursos. Actualmente, las infraestructuras móviles deben cambiar y adaptarse al tráfico que pasa por ellas, y como respuesta a esto nace *Network Slicing*, que se caracteriza por dividir a los recursos disponibles de manera que favorezcan la comunicación y se utilicen eficientemente; lo que permite implementar una infraestructura de comunicaciones móviles en dispositivos tales como UAVs y dar un nuevo alcance a las comunicaciones celulares. En el presente proyecto se realizó la implementación de una simulación básica de los principales rasgos de *Network Slicing* en el contexto de las redes aréas Ad-hoc (FANET:Flying Ad-hoc Networks).

En el primer capítulo se describieron los conceptos fundamentales que rodean a *Network Slicing* y su implementación dentro de las redes celulares. Asimismo, se detalló su relación con dispositivos UAVs junto con sus características más importantes en cuanto a comunicaciones celulares.

En el segundo capítulo se explicaron los elementos usados para llevar a cabo la simulación del sistema propuesto en Matlab. Además, en los elementos correspondientes, se detallaron los parámetros que permiten asignar una solicitud a determinado tipo de tráfico.

En el tercer capítulo se presentaron los resultados obtenidos para la simulación propuesta y las conclusiones a los que estos llevaron. Se debe tomar en cuenta que los resultados se analizaron individualmente y de manera conjunta para las mismas condiciones.

PALABRAS CLAVE: 5G, segmentación de red, UAV, FANET, incrustación de red virtual, redes bajo demanda, simulación

ABSTRACT

The communication demands for today's world have evolved along with the development of technology, causing traditional cellular networks to undergo a change in the way they provide their resources. Currently, mobile infrastructures must change and adapt to the traffic that passes through them, and in response to this *Network Slicing* was born. *Network Slicing* is characterized by dividing the available resources in a way they can favor communication and use resources efficiently. This allows the implementation of a mobile communications infrastructure in devices such as UAVs and give a new scope to cellular communications. In this project, the implementation of a basic simulation of the main features of *Network Slicing* in the context of Ad-hoc aerial networks (FANET: Flying Ad-hoc Networks) was carried out.

In the first chapter, the fundamental concepts surrounding *Network Slicing* and its implementation within cellular networks were described. Likewise, its relationship with UAVs devices was detailed along with its most important characteristics in terms of cellular communications.

In the second chapter, the elements used to carry out the simulation of the proposed system in Matlab were explained. In addition, in the corresponding elements, the parameters that allow assigning a request to a certain type of traffic were detailed.

In the third chapter, the results obtained for the proposed simulation and the conclusions to which they led were presented. It should be noted that the results were analyzed individually and jointly for the same conditions.

KEYWORDS: 5G, Network Slicing, UAV, FANET, Virtual Network Embedding, on demand networks, simulation.

1. INTRODUCCIÓN

La innovación de tecnologías celulares y los diferentes tipos de tráfico de datos que se espera fluyan a través de las mismas, han dado como resultado la era de quinta generación de telefonía celular (5G). Se espera que los sistemas 5G brinden a la sociedad una conexión total, que pueda superar las limitaciones de tiempo y espacio para crear conexiones centradas en el usuario, en el servicio o entre las personas y las cosas.

Entre los principales enfoques que promueven el avance de las redes 5G están la segmentación de la red (NS: Network Slicing) y la asignación dinámica de recursos en redes virtuales. Estas características permiten, por una parte, hacer un uso más eficiente de la infraestructura física, y por otra, otorgar recursos especializados a un tipo de servicio específico [1].

La base de NS se encuentra en la incrustación de red virtual (VNE: Virtual Network Embedding). Con esto se pretende distribuir los recursos de hardware a las diferentes redes que se soliciten y que cada uno de estos recursos sea independiente uno del otro. Con la ayuda de VNE, se pueden representar la capacidad de procesamiento y del enlace en grafos ya sea para las redes físicas o virtuales [2]. Así, un nodo tendrá una capacidad de procesamiento y una arista un ancho de banda asignados. Para fijar estos pesos, se consideran requerimientos como el uso de la unidad central de procesamiento (CPU: Central Processing Unit), memoria, ancho de banda y condiciones dadas por la ubicación.

Dado que la implementación de NS implica un uso más eficiente de recursos y con el fin de dar un alto nivel de movilidad a la infraestructura de telefonía móvil, se propone el uso de vehículos aéreos no tripulados (UAV: Unmanned aerial vehicles). Estos, al ser equipados con hardware equivalente al de estaciones base pueden comportarse como tales. Asimismo, estos dispositivos podrían ser usados para proveer una red emergente en casos de que la infraestructura terrestre se encuentre dañada, o para dar cobertura a sitios de difícil acceso [2].

Dado que se implementan varios UAVs para proveer la infraestructura física de la red, se plantea como su mecanismo de comunicación una red Ad-hoc[1], principalmente por las limitaciones en cuanto a recursos energéticos que estos dispositivos tienen y por la capacidad que esta red brinda para reponerse ante la pérdida de nodos. A este tipo de redes se les denomina Redes Aéreas Ad-hoc (FANET: Flying Ad-hoc Networks).

Cada una de estas tecnologías convergen en la virtualización de las redes, el aumento

[1] Una red Ad-hoc es una red inalámbrica que permite un fácil establecimiento de conexión entre dispositivos de clientes inalámbricos en la misma área física sin el uso de un dispositivo de infraestructura [3].

de la eficiencia del uso de los recursos físicos y ajuste de la red móvil de acuerdo a las necesidades de los usuarios.

Este trabajo propone presentar una simulación donde se pueda visualizar la interacción de las diferentes tecnologías mencionadas anteriormente, con la finalidad de conocer cómo se realiza la asignación de recursos de red para cada tipo de tráfico definido para 5G y el avance tecnológico desarrollado para cada una de ellas.

1.1. OBJETIVO GENERAL

Implementar en software de simulación Matlab la tecnología de Network Slicing en un ambiente de una FANET.

1.2. OBJETIVO ESPECÍFICOS

- Describir los fundamentos teóricos de la tecnología de Network Slicing en 5G y de los sistemas de comunicaciones habilitados con UAVs incluyendo las FANETs.
- Describir la incorporación de la tecnología Network Slicing en ambiente habilitado por UAVs.
- Realizar el modelamiento y la caracterización de una red basada en UAVs así como la simulación del sistema de comunicación considerando Network Slicing.
- Realizar la simulación de un sistema de comunicación de UAV que incorpora la tecnología de Network Slicing.

1.3. ALCANCE

En el presente Trabajo de Integración Curricular se ha propuesto implementar dentro de Matlab la tecnología de Network Slicing haciendo énfasis en las comunicaciones entre UAVs. Para esto se abordó la arquitectura de la red de acceso de 5G con el fin de conocer principalmente la división del tráfico que recorre la red móvil. Asimismo, se examinó la arquitectura y la infraestructura de Network Slicing teniendo como resultado la definición del ciclo de vida de los segmentos de red y del propósito de esta tecnología. Finalmente, se establecieron los conceptos fundamentales para poder simular VNE dentro de una red Ad-hoc como lo sería el mapeo de recursos dentro de una infraestructura de base y la comunicación entre estos elementos.

El sistema de comunicaciones simulado comprende la reproducción de la creación de los "segmentos de red" de acuerdo a los requerimientos solicitados dentro de diferentes topologías. Las topologías físicas se conformaron por UAVs y los enlaces que los conectan, mientras que la topologías virtuales se relacionaron a las capacidades de los dispositivos o de los enlaces de acuerdo a la petición realizada.

1.4. MARCO TEÓRICO

En esta sección se presentan los conceptos básicos de la arquitectura de la red de acceso en 5G, de Network Slicing incluyendo el VNE y del funcionamiento de un sistema de comunicaciones con UAVs.

1.4.1. ARQUITECTURA DE LA RED DE ACCESO DE 5G

Con el fin de soportar la quinta generación de tecnología celular (5G), el 3GPP ha desarrollado una nueva red de acceso denominada NG-RAN. Esta se encuentra entre el equipo del usuario (UE: User Equipment) y la red de core de 5G.
La base donde se construye la arquitectura de NG-RAN es el nodo NG-RAN; el nodo NG-RAN es un nodo lógico que se caracteriza entre otras cosas por lo siguiente:

- Son posibles diferentes implementaciones, completamente independientes de la arquitectura lógica. Es decir, la arquitectura lógica no exige ninguna implementación específica (por ejemplo, coubicada, virtualizada, centralizada, etc.), sino que se adapta a muchos de estos tipos de implementación de manera transparente. Por ejemplo, dos nodos lógicos podrían implementarse como una sola entidad física (una sola "caja") sin necesidad de cambiar nada en la arquitectura lógica estandarizada: el único efecto tangible sería que la interfaz lógica entre ellos "desaparecería" dentro la "caja" física y no sería visible en la implementación [4].

- Si las funciones lógicas de los nodos cambiaran, también lo harían las interfaces lógicas. En otras palabras, las interfaces lógicas dependen y se definen en función del conjunto de funciones lógicas del nodo [4].

La Union Internacional de Telecomunicaciones (ITU: International Telecommunications Union) especifica tres casos de uso para 5G que se enfocan en diferentes requisitos, estos son: i) banda ancha móvil mejorada (eMBB:enhanced mobile broadband), ii) comunicaciones ultra confiables de baja latencia (URLLC:ultra reliable low-latency) y iii) comunicaciones máquina

a máquina masivas (mMTC:massive machine type communication) [5].Los requisitos que cada uno de estos casos debe satisfacer se muestran en la figura 1.1.

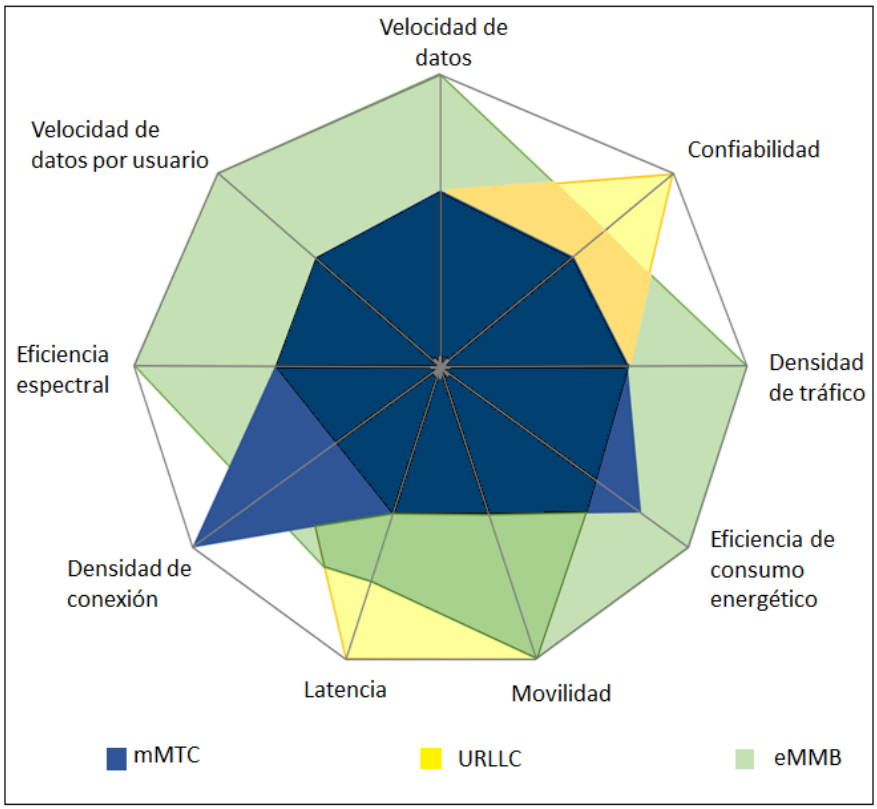


Figura 1.1: Casos de uso básicos para 5G y los requerimientos que deben cumplir basado en [5]. En la figura, mientras más lejos el requerimiento del centro, más importante es para el caso de uso.

La arquitectura de las redes 5G está compuesta por diferentes dominios, entre estos destacan la red de núcleo (CN:Core Network), la red de acceso (RAN: Radio Access Network) y las funciones de la capa de servicio [5]. Esta división en dominios presenta la siguientes características:

- Se puede añadir una nueva tecnología fácilmente debido a que los dominios son independientes [6].
- Se permite la movilidad de los usuarios ya que algunas de las funciones de la CN se mantienen aunque se cambie de nodo [6].
- Se permite el uso de múltiples vendedores para la interoperabilidad de la CN/RAN [6].

Asimismo, si se compara con otras tecnologías de acceso, como la disponible en LTE (E-UTRAN: Evolved Universal Terrestrial Radio Access Network), se tienen las siguientes diferencias:

- **Interfaces aéreas múltiples:** A diferencia de E-UTRAN, NG-RAN admite dos interfaces aéreas. La establecida para LTE (eNB), que va a pasar a llamarse “ng-eNB” y la nueva interfaz para 5G que se conoce como “gNB” [7].

- **Conectividad dual multi radio (MR-DC):** La NG-RAN soporta varias opciones de conectividad múltiple, en las cuales un solo UE podría estar conectado a dos diferentes nodos de la red, uno que se encargue de brindar acceso a la red de 5G y otro que provea acceso sea a E-UTRA o a 5G [7].

- **División:** La división permite a los operadores de red móvil categorizar el tráfico de los clientes en tipos de calidad de servicio, cada uno con diferente requerimiento [7].

1.4.2. NETWORK SLICING (NS)

Se define a NS como un paradigma que permite la compartición de la misma infraestructura para brindar distintos tipos de servicios definidos por 5G [5]. Las principales razones por las que se ideó esta solución son:

- Desplegar múltiples e independientes funciones del core de 5G en la misma red.

- Crear múltiples configuraciones del core y de la red de acceso de 5G.

- Seleccionar el tipo de servicio que requiere el cliente y configurarlo dinámicamente dependiendo de la aplicación y nivel de suscripción que tenga el cliente.

1.4.3. Arquitectura de Network Slicing

Debido a que NS debe ser capaz de proveer diferentes tipos de servicios usando una sola infraestructura física, cada una de las redes que implemente NS se debe diseñar con la capacidad de proveer servicio de todos a todos (E2E: Everything to everything). Al ser difícil de alcanzar esta condición con un solo enfoque y tecnología, se propone dividir la arquitectura en cinco capas, de esta manera cada capa será independiente de otra y se simplificará la solución; las capas son las siguientes: infraestructura, función de red, orquestación, funciones de negocio (business layers) y capas de servicio [8]. En este Trabajo de Integración Curricular se estudiará la capa de infraestructura, responsable de asignación de recurso físicos a redes virtuales.

1.4.4. Capa infraestructura

La capa infraestructura comprende todos los recursos de hardware y software que forman la red del operador, es decir, equipos de computo, almacenamiento y red, incluido el equipo del usuario. Este conjunto de equipos se puede utilizar para implementar nodos de red físicos y/o virtuales. Es por esto que las funciones de esta capa se alinean al paradigma de la infraestructura como servicio (IaaS [2]: Infrastructure-as-a-Service) donde se pueden arrendar diferentes recursos de infraestructura que cubren diferentes requisitos para adaptarse a las necesidades de los diversos servicios [8].

Debido a las diferentes restricciones entre los diversos casos de uso implementados en la red, una infraestructura simple ubicada en el centro podría no ser adecuada. Asimismo, se espera que la RAN que consta de múltiples estaciones base abarque diversas tecnologías de acceso por radio (RAT: Radio Access Technologies), incluidas LTE y Wi-Fi. Es por estas razones que se espera manejar un despliegue de estaciones base definidas por software que utilicen hardware genérico [8].

Es necesario tener en cuenta que la virtualización no solo comprende otorgar los recursos necesarios (procesamiento, almacenamiento, red y radio) para cada caso de uso, sino también la capacidad de admitir diferentes tipos de operaciones de control sobre los recursos de manera virtualizada según los requisitos del servicio y brindar el aislamiento apropiado entre los casos de uso. Esta propiedad de proporcionar un entorno virtualizado de extremo a extremo, que cabe destacar, puede ser potencialmente abierto y controlado completamente por terceros, es una de las características clave que separa el NS de las soluciones de uso compartido de red ya existentes [10].

1.4.5. Ciclo de vida del segmento de red

De acuerdo al estándar establecido por el 3GPPP, el ciclo de vida del segmento de red consiste en los siguientes cuatro fases: puesta en marcha, activación, supervisión del tiempo de ejecución y desmantelamiento. Estas fases pueden ser apreciadas en la figura 1.2.

Durante la fase de puesta en marcha se maneja la preparación y el diseño de un segmento, por ejemplo, se crea una plantilla de segmento en la primera fase. Esta plantilla de segmento contiene información sobre los requisitos del usuario y los recursos que se implementarán.

[2] IaaS es un modelo de provisión en el que una organización subcontrata los equipos utilizados para dar soporte operaciones, incluidos el almacenamiento, el hardware, los servidores y los componentes de red. El proveedor de servicios es propietario del equipo y es responsable de alojarlo, operarlo y mantenerlo. El cliente normalmente paga por uso [9]

Después de eso, la plantilla de segmento diseñada se implementa en la infraestructura subyacente; el recurso especificado debe asignarse a un segmento y activarse. Los recursos de segmento activados se monitorean continuamente a través de las herramientas de monitoreo y, en caso de falla, se notificará al orquestador para resolver el error. En la fase de desmantelamiento, el segmento creado se eliminaría automáticamente en el momento especificado [5].

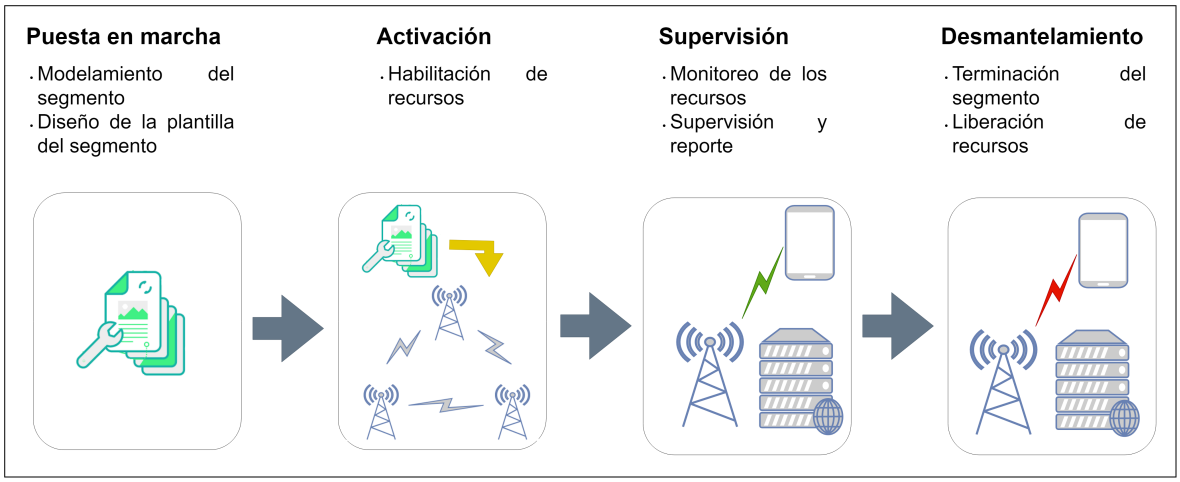


Figura 1.2: Administración del ciclo de vida de Network Slicing basada en [5].

1.4.6. VIRTUAL NETWORK EMBEDDING (VNE)

A través de NS se propone que el proveedor de 5G cree redes virtuales y disponga de una infraestructura física flexible y programable. Como se ha venido revisando, el servicio NS se compone de un conjunto de actividades encadenadas para producir una red virtual, entre estas actividades, una primordial es el mapeo de redes virtuales en recursos físicos, matemáticamente conocido como VNE [11].

VNE se ocupa de la asignación de recursos virtuales tanto en nodos como en enlaces. Por lo tanto, su funcionamiento puede ser divido en dos operaciones: *Mapeo de Nodos Virtuales (VNoM: Virtual Node Mapping)* donde se ubican nodos virtuales en nodos físicos y *Mapeo de Enlaces Virtuales (VLiM: Virtual Link Mapping)* donde se sitúan los enlaces virtuales para conectar los nodos correspondientes en la red física. Para poder realizar las peticiones de los recursos tanto para nodos como para enlaces se utiliza un mecanismo denominado solicitud de red virtual (VNR: Virtual Network request) [12].

Bajo un enfoque centralizado de la asignación de recursos, una sola entidad en la infraes-

tructura del proveedor, como el orquestador [3], recibe las VNRs y realiza la asignación de recursos. Esta entidad requiere de todos los conocimientos y recursos globales necesarios para esa tarea [12].

En la figura 1.3 se presenta un ejemplo de un entorno de virtualización de red donde dos redes virtuales están integradas en la misma infraestructura física. Con el propósito de que exista equilibrio de carga, dos nodos virtuales de la misma red virtual generalmente se integran en diferentes nodos físicos, aunque dos nodos virtuales de diferentes redes virtuales se pueden integrar en el mismo nodo físico.

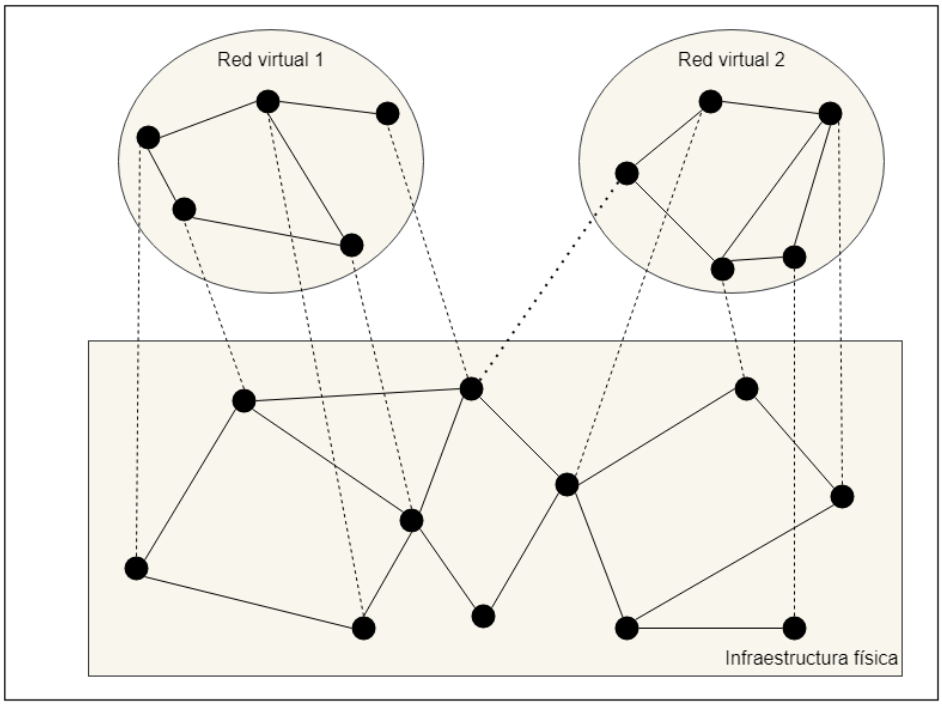


Figura 1.3: Entorno de virtualización de red basada en [12].

El proceso de virtualización que VNE propone puede ser llevado a la teoría de grafos. Esto debido a que tanto la red virtual, que describe cada una las solicitudes de los clientes, como la red física, que describe la infraestructura física, se pueden modelar como grafos [13]. Dentro de estos grafos, se representará fácilmente la carga que cada CPU (nodo) y enlace (arista) tienen.

En este sentido, las expresiones para los grafos que simbolizan las redes están dadas de acuerdo a la ecuación 1.1 [10]:

$$G_S = (N_S, L_S, A_S^N, A_S^L) \tag{1.1}$$

[3] Se define a un orquestador como aquella entidad cuya tarea es identificar y asignar las funciones y tecnologías necesarias para garantizar que los requerimientos de un segmento de red sea cumplido [5].

Donde:

- G_S representa la red física.
- N_S representa la asociación de todos los nodos en la VNR.
- L_S representa la asociación de todos los enlaces en la red física.
- A_S^N representa el atributo para la asociación de todos los nodos en la red física.
- A_S^L representa la asociación de todos los enlaces en la red física.

Se debe tener en cuenta que el término A_S^N también contiene características como la ubicación, la capacidad de la CPU y la capacidad de almacenamiento del nodo. Mientras tanto, el término A_S^L contiene características como el ancho de banda. Al igual que el modelo de red física, en una red virtual, cada solicitud de red virtual o VNR también se puede modelar como un grafo ponderado no dirigido [4] y se simboliza mediante un solo nodo de la red física que puede hospedar varios nodos virtuales. En algunos casos, se pueden combinar varios recursos de la red física para crear nuevos recursos virtuales [10].

1.4.7. UAVS PARA COMUNICACIONES

Las comunicaciones a través de UAVs, conocidos también como drones, han ganado popularidad debido a que dotan a la red de flexibilidad, movilidad y facilidad en el despliegue [15]. Debido a la cobertura amplia que las redes celulares tienen, los UAVs podrían funcionar como una infraestructura complementaria para proveer conectividad amplia y segura, especialmente más allá de la línea de vista [15].

Proveer recursos como ancho de banda en el cielo es una tarea difícil comparado con los servicios que ya se presentaba en generaciones celulares anteriores que ya contemplan ciertos obstáculos como edificios entre estaciones base.

Las comunicaciones con UAVs pueden ser realizadas a través de tres tecnologías distintas, estas son: enlace directo, enlace satelital, red Ad-hoc [16] y se describen a continuación:

- **Enlace directo:** Se caracteriza por el uso de bandas industriales, científicas y médicas (ISM: Industrial, Scientific, and Medical), pero de manera general está limitada a una comunicación por línea de vista. Esto dificulta la implementación de comunicaciones con UAVs a gran escala en el futuro.

[4] Un grafo ponderado es un grafo en el que a cada arista se le asigna un peso numérico. Por lo tanto, un grafo ponderado es un tipo especial de grafo etiquetado en el que las etiquetas son números (que para este caso y de manera general se toman como positivos). Asimismo, un grafo no dirigido es aquel donde las aristas no tienen dirección [14].

- **Enlace satelital:** Esta opción ofrece una gran cobertura para los UAVs, además que permitiría que los satélites reenvíen la información entre UAVs que se encuentran muy lejos de las estaciones base. Incluso, la conexión con satélites permitiría conocer la posición de los UAVs. Sin embargo, se debe recalcar que los enlaces satelitales introducen latencias y muchas pérdidas, lo que es indeseable en comunicaciones sensibles al retraso.

- **Red Ad-hoc:** Se caracteriza por su flexibilidad, poca necesidad de infraestructura, y por su capacidad de tomar la forma de varias topologías, esto, beneficiaría las comunicaciones entre UAVs ya que estos se proponen como una red complementaria de fácil y rápido despliegue. Sin embargo, las redes Ad-hoc son funcionales solamente en tamaños reducidos, esto debido a las dificultades para encontrar un protocolo de enrutamiento confiable sobre toda la red.

1.4.8. FLYING AD-HOC NETWORK (FANET)

Las FANETs son redes Ad Hoc constituidas por nodos móviles aéreos, entre sus características más atractivas están su simplicidad, versatilidad y flexibilidad. Además, presentan las características de proveer comunicaciones inalámbricas dispositivo a dispositivo (D2D: Device to Device) sin requerir una infraestructura [17].
Gracias al avance de los sistemas embebidos y a la tendencia de miniaturización de los sistemas electrónicos, ha sido posible producir UAVs de tamaño y costo reducido. Sin embargo, la capacidad de un único UAV pequeño es limitada. La coordinación y colaboración de múltiples UAV puede crear un sistema que está más allá de la capacidad de un solo UAV de tamaño y mayor capacidad. Los sistemas compuestos por varios UAVs presentan las siguientes características [17]:

- **Escalabilidad:** A diferencia del uso de un solo UAV, con *n* UAVs se alcanzaría a maximizar la cobertura del sistema *n* veces. Esto debido a que en conjunto cubrirían más área.

- **Costo:** Debido a la facilidad que se tiene al elaborar un UAV pequeño, se puede minimizar el costo de la agregación de cada uno de estos dispositivos al sistema que se plantea usar.

- **Robustez:** En el caso de que un UAV falle durante la realización de una misión, esta no fallará ya que existirán otros UAVs que compartirán la carga del UAV que falló.

1.5. Características de diseño

Las FANET presentan varias características únicas producidas por su nivel de movilidad y por los diferentes escenarios de operación. Es por esto que se presentan las siguientes características que son fundamentales al momento de plantear una FANET.

- **Modelo de propagación de radio:** Debido a que los UAVs se van a situar a gran altura, es fácil conseguir línea de vista entre los dispositivos, especialmente en zonas abiertas.
- **Consumo de potencia y tiempo de vida de la red:** Una de las limitaciones que tendrían este tipo de redes es precisamente el consumo de potencia ya que no solo se consumiría potencia realizando operaciones computacionales relacionadas con la comunicación, sino que también se debe mantener al UAV en el aire. Es por esto que se debe considerar el tiempo de vida del UAV (duración de la batería) en el tiempo de actividad de actividad de la red a desplegar.
- **Potencia computacional:** Generalmente, los UAVs como parte de su payload contienen pequeños sistemas computacionales (por ejemplo, una plataforma Raspberry Pi), que, debido a las limitaciones de tamaño y energía, propias de las baterías, tienen un poder computacional limitado.

2. METODOLOGÍA

En el presente capítulo se muestra la implementación de los procesos mediante los cuales se lleva a cabo la simulación de NS en un ambiente FANET. Para esto se considera exclusivamente el proceso de virtualización de solicitudes por medio de VNE, por este motivo no se examinan elementos tales como el modelo de canal, los protocolos de enrutamiento o demás componentes que pueden intervenir durante el proceso de comunicación de los componentes de la NG-RAN. De igual manera, se presenta la relación entre cada una de las funciones que permiten llevar a cabo el proceso de virtualización. Se expone también la interfaz gráfica diseñada para mostrar paso a paso el proceso de virtualización , obtención y análisis de los respectivos resultados.

2.1. ANÁLISIS DE REQUERIMIENTOS

En la simulación a presentar se utilizarán las funciones que se muestran en la figura 2.1.

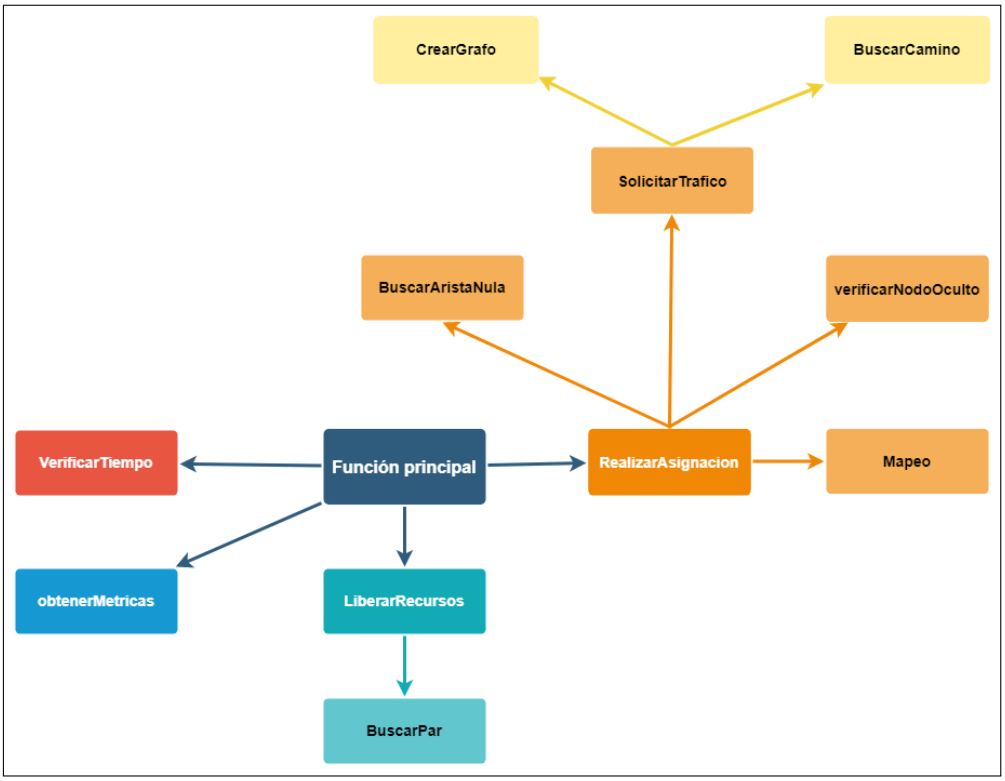


Figura 2.1: Funciones utilizadas para la simulación y su relación.

La simulación a presentar debe cumplir con los siguientes requerimientos:

- Aptitud para mostrar el proceso de virtualización de manera gráfica a la vez que el resto de elementos que interactúan en el mismo.

- Presentar las condiciones para la generación de grafos que puedan describir las topologías tanto solicitadas como la física.
- Crear solicitudes aleatorias con el fin de simular tráfico real.
- Poseer un mecanismo de mapeo de recursos que defina en qué área de la topología física se puede situar una solicitud.
- Disponer de un método de medición de tiempo a fin de retirar solicitudes expiradas de la topología física y llevar el tiempo de simulación.
- Contar con un mecanismo de acceso rápido a grupos de datos con el fin de distinguir fácilmente una solicitud de otra.

2.2. IMPLEMENTACIÓN EN MATLAB

El recurso utilizado para la implementación de la simulación planteada será Matlab debido a que este, en su versión 2022a, presenta objetos en su versión más estable con los cuales se puede generar grafos y manipular los mismos de manera similar a la indicada por VNE. Asimismo, este lenguaje de programación presenta un elevado nivel de versatilidad para el manejo y generación de matrices. Esto satisface el primer requerimiento indicado.

2.2.1. PROGRAMA PRINCIPAL

El programa principal es donde se especifican los parámetros tales como el número de clientes (peticiones) que se van a simular y el tiempo total de la simulación. Dentro de este programa existen varias funciones que se encargan tanto de simular los procesos usados para virtualizar peticiones dentro de la topología física, como la creación y gestión de los mismos.

El escenario bajo el cual se ejecuta la simulación consiste en el mapeo de varias solicitudes aleatorias (VNR) sobre una topología física teniendo en cuenta los recursos de CPU, enlace y posición dentro de la topología física que cada una requiere. Los parámetros relacionados con el uso de recursos de estas solicitudes se generarán de acuerdo al tipo de tráfico al que estas correspondan, y, este, se seleccionará de manera aleatoria. La duración de cada solicitud variará aleatoriamente tratando de simular el tiempo que un usuario podría utilizar un recurso.

Con el fin de generalizar la simulación, se define como unidad de tiempo los *time slots*. Estos podrían representar fracciones de segundo u horas. Esta unidad de tiempo rige tanto el tiempo que dura la simulación como el tiempo de vida de una solicitud.

Dentro de la simulación se puede crear hasta una solicitud por time slot, esto debido a que se trata de mostrar al usuario el proceso de virtualización de la manera más gráfica y sencilla posible. Asimismo, el time slot en el cual se cree una nueva VNR será aleatorio.

El programa principal, como se verá más adelante en este capítulo, se definen varios métodos de control del tiempo y de visualización de la información obtenida con el fin de que el usuario pueda interactuar con la interfaz gráfica y visualizar paso a paso como se realiza la asignación de recursos.

Los procesos que implementa la simulación se describen de manera general en la figura 2.2.

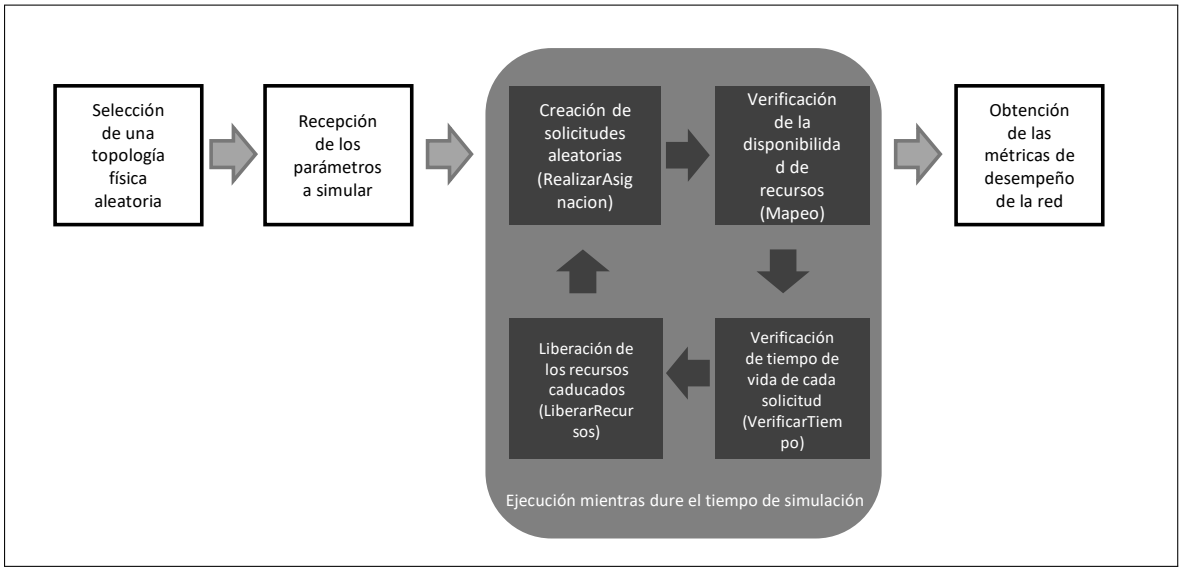


Figura 2.2: Diagrama de bloques del programa principal.

2.2.2. FUNCIÓN GRAPH

La generación de grafos es parte fundamental para la creación y mapeo de solicitudes. Todo esto en virtud de que con la ayuda de grafos se podrá simbolizar la capacidad de CPU utilizada por algún elemento o enlace como lo sugiere la sección 1.4.6. de este trabajo. Para llevar a cabo esta tarea y cumplir con el segundo requerimiento indicado para este trabajo, se utiliza la función `graph` de Matlab [18]. Esta función tiene como objetivo generar grafos con

aristas no dirigidas y, para poder crearlos a partir de su respectiva matriz de adyacencia,[5] se usa la función que está dada por la sección de código 1, de donde para la simulación se usarán todos los elementos de entrada y salida. Cabe indicar que la simulación realiza el proceso de construcción de solicitudes de manera genérica, por lo que al crear la topología de cada solicitud solo se requiere un grafo sin pesos en sus nodos ni en sus aristas.

```
G = graph(A);
```

Segmento de código 1: Creación de un grafo no dirigido de matriz de adyacencia A.

donde el elemento de entrada es [18]:

- **A:** La matriz de adyacencia del grafo que se desea crear.

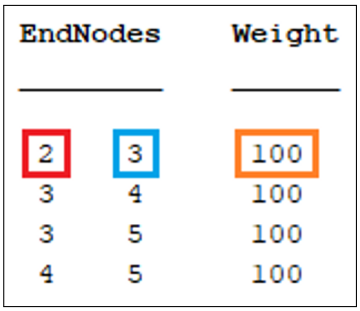
mientras que el elemento de salida es [18]:

- **G:** Objeto grafo generado.

El objeto generado contiene varias propiedades relacionadas con el grafo obtenido, por lo que a este objeto se le denomina *objeto grafo*. Debido a que estas propiedades se usan frecuentemente en el proceso de simulación, son descritas a continuación [18]:

- **Propiedad Edges:** La propiedad “Edges”devuelve una tabla que para esta simulación tiene dimensiones Mx2, donde M es el número de aristas en el grafo y 2 corresponde a dos matrices, una con los nodos de origen y de destino de cada uno de las aristas y otra con el peso que cada arista tiene. El primer elemento de esta tabla (matriz EndNodes) contiene, de izquierda a derecha, los nodos de origen y los nodos de destino correspondientes a cada arista. Asimismo, el segundo elemento de esta tabla muestra el peso que tiene cada arista, tal como se muestra en la figura 2.3. Es importante indicar que se puede acceder a cada elemento de esta tabla con el fin de obtener y/o editar las matrices “EndNodes” y “Weight”.

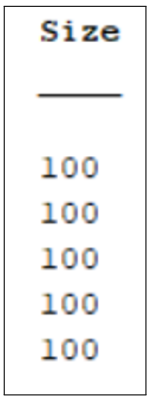
[5] Se conoce como matriz de adyacencia a aquella que contiene filas y columnas que se utilizan para representar un grafo dirigido, con 0 o 1 en la posición de (V_i, V_j) según la condición de que V_i y V_j sean adyacentes (1) o no (0) [14].



EndNodes		Weight
2	3	100
3	4	100
3	5	100
4	5	100

Figura 2.3: Propiedad “Edges” del objeto grafo, dentro del recuadro rojo, se muestra el nodo de origen y dentro del recuadro celeste, se muestra el nodo de destino. Asimismo, dentro del recuadro naranja se muestra el valor del peso de la arista.

- **Propiedad Nodes:** La propiedad “Nodes” devuelve una tabla, que para esta simulación tiene dimensiones Mx1, donde M es el número de nodos en el grafo. El único elemento de esta tabla contiene los pesos de todos los nodos almacenados en la matriz “Size”, estos pesos corresponden desde el nodo 1 hasta el nodo M desde arriba hacia abajo. Como se muestra en la figura 2.4.



Size
100
100
100
100
100

Figura 2.4: Propiedad “Nodes” del objeto grafo, para el caso de esta simulación, solamente se generan los pesos para cada uno de los nodos.

En el segmento de código 2 se muestra la obtención de la matriz *EndNodes* del grafo “G”. El motivo por el que esta matriz es obtenida es analizar los nodos de origen y el de destino de la cada una de las aristas que componen el grafo “G”.

```
EndNodes = G.Edges.EndNodes;
```

Segmento de código 2: Acceso a la matriz *EndNodes* del grafo “G”.

Con el fin de obtener los nodos de origen o de destino de cada una de las aristas que componen el grafo “G”, se debe ejecutar el segmento de código 3. Donde “a” es un número entero que corresponde al número de arista que se desea analizar y “b” es un número entero que varía entre uno (origen) y dos (destino) y corresponde al nodo que se requiere analizar.

```
Origen = G.Edges.EndNodes(a,b);
```

Segmento de código 3: Acceso al origen de la primera arista del grafo “G”.

Luego, para obtener la matriz *Weight*, que contiene los pesos de cada arista del grafo “G”, se coloca el segmento de código 4.

```
Peso = G.Edges.Weight;
```

Segmento de código 4: Acceso al vector de pesos del grafo “G”.

En el que caso de que se requiera analizar el peso de una determinada arista, se debe ejecutar el segmento de código 5, donde “a” es un número entero que corresponde a la arista que se solicita.

```
Peso_1 = G.Edges.Weight(a);
```

Segmento de código 5: Acceso al peso de la primera arista del grafo “G”.

El segmento de código 6 muestra la obtención de la matriz *Size*, que contiene los pesos de cada uno de los nodos del grafo “G”.

```
Nodes = G.Nodes.Size;
```

Segmento de código 6: Acceso a la matriz *Size* del grafo “G”.

Con la finalidad de obtener el valor de alguno de los nodos presentes en el grafo “G”, se requiere ejecutar el código 7. Donde “a” es el número de nodo cuyo peso se requiere analizar.

```
Nodes = G.Nodes.Size(a);
```

Segmento de código 7: Acceso al peso del primer nodo del grafo “G”.

2.2.3. MODELAMIENTO DE SOLICITUDES

Como se mencionó en la subsección 1.4.1, las solicitudes que se realizan a la infraestructura de comunicaciones se crean de acuerdo a las calidades de servicio establecidas para 5G. Estas se diferencian de acuerdo al tipo de procesamiento que se debe asignar, el ancho de banda que van a ocupar y las posiciones sobre la infraestructura que se van a solicitar [12]. Asimismo, dado que los requerimientos de estas solicitudes se reservan por un tiempo de acuerdo con lo que el cliente necesite, se define un tiempo de vida de la solicitud [5]. Todas estas características de las solicitudes se condensan dentro de la simulación de acuerdo a lo mostrado en la sección de código 8.

```
trafico = struct('grafo', {}, ...
    'nombre', '', ...
    'posiciones', {}, ...
    'tiempoVida', {}, ...
    'tiempoInicial', {});
```

Segmento de código 8: Variable donde se definen todas las propiedades que caracterizan una solicitud.

2.2.4. MECANISMOS DE CUMPLIMIENTO DE LOS REQUISITOS DEL SISTEMA

Debido a lo complejo que se volvería el algoritmo de mapeo de recursos, se plantean las siguientes condiciones para la creación de solicitudes:

1. Que no se realicen solicitudes que contengan un nodo de grado cero.
2. Que no existan solicitudes que obliguen al algoritmo a adaptar sus recursos físicos a un nodo oculto[6].

Estas condiciones se revisan durante la ejecución de la función `RealizarAsignacion`. La condición 1 se revisa con el uso de las funciones `isempty` y `BuscarAristaNula`. La función `isempty` se encarga de verificar si la propiedad "Edges" devuelve una tabla vacía de acuerdo a lo que define Matlab [20], lo que significa que no existen enlaces definidos dentro de la solicitud.

Luego, la función `BuscarAristaNula`, revisa si todos los nodos de la solicitud están conectados por al menos un enlace, esto por medio de la verificación de la existencia de $N-1$ enlaces por cada N nodos. Esta operación se puede visualizar en la sección de código 9.

```
function [existe] = BuscarAristaNula(tp)
if (length(tp.Edges.Weight) < length(tp.Nodes.Size)-1)
    existe = 1;
else
    existe = 0;
end
```

Segmento de código 9: Búsqueda de nodos de grado 0.

[6] Se define a un nodo oculto como aquel nodo que se encuentra entre dos nodos solicitados, pero no se encuentra considerado en la solicitud. Un ejemplo de esto se brinda en la figura 2.5 [19].

A diferencia de la condición 1. La condición 2 se revisa con la función `verificarNodoOculto`. Esta verifica que los nodos se conecten de forma secuencial, por ejemplo, en una solicitud de tres nodos completamente conectados, debe existir una arista entre el nodo uno y el nodo dos y también una arista entre el nodo dos y el nodo tres. Esta condición se revisa en los dos sentidos (tanto de nodo menor (primer nodo) a mayor (último nodo), como de nodo mayor a menor). Esto se puede visualizar en la sección de código 10. Donde la variable *matrizR* es la matriz de "EndNodes" y la variable *verXhacer* es el número de filas que tiene la matriz "EndNodes". El objetivo de este procedimiento es descartar a las solicitudes que no cumplan con lo descrito anteriormente debido a que ese generaría un nodo oculto al momento de mapear la solicitud. En la figura 2.5 se muestra un ejemplo de nodo oculto y su respectivo mapeo.

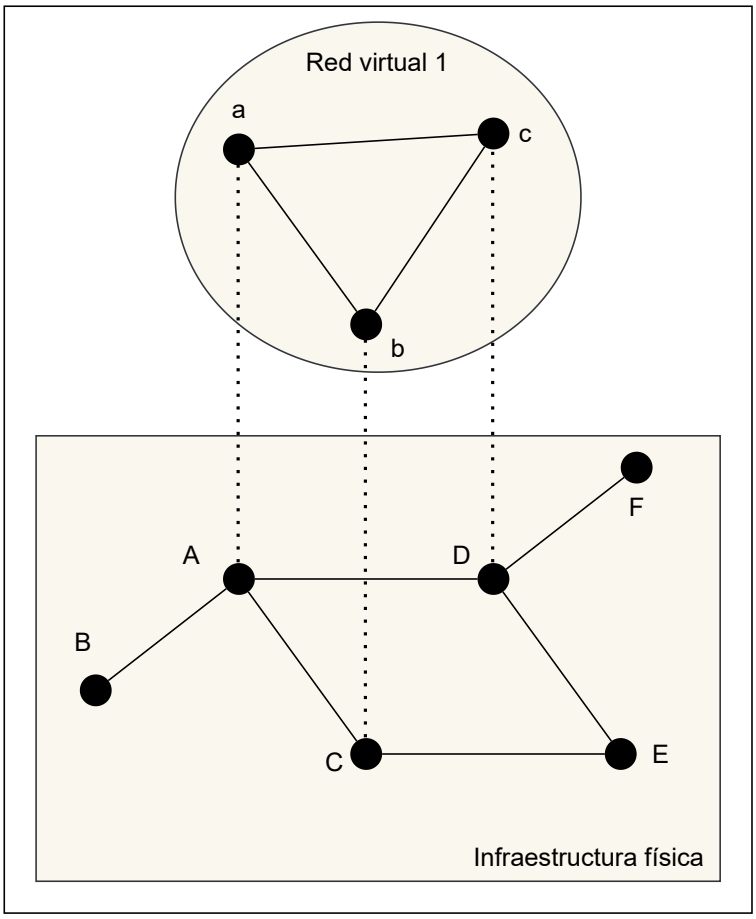


Figura 2.5: Ejemplo de una solicitud con nodo oculto. Los nodos se mapean de la siguiente manera: [a →A, b→C, c→D] y los enlaces de la siguiente manera [(a,b)→(A,C), (a,c)→(A,D),(b,c) → (C,E,D)]. Esto lleva al uso de un nodo más que no se considera dentro de la solicitud original, lo que implica un mayor nivel de complejidad al momento de mapear los recursos dentro de la topología física.

```
for i=1:verXhacer
    if matrizR(i,1) + 1 ~= matrizR(i,2)
        if matrizR(i,1) - 1 ~= matrizR(i,2)
            existe = 1;
        else
            cuentaNo = cuentaNo + 1;
        end
    elseif matrizR(i,1) - 1 ~= matrizR(i,2)
        if matrizR(i,1) + 1 ~= matrizR(i,2)
            existe = 1;
        else
            cuentaNo = cuentaNo + 1;
        end
    end
end
```

Segmento de código 10: Verificación de la unión en un orden secuencial entre los nodos.

2.2.5. FUNCIÓN REALIZAR ASIGNACION

Esta función tiene como propósito realizar dos tareas. La primera es, por medio de la función `SolicitarTrafico`, obtener una solicitud con todos sus campos generados, y la segunda es verificar si esta solicitud puede ser mapeada dentro de la topología física. Esto último por medio de la función `Mapeo`. Las solicitudes realizadas, al momento de someterse a la presente función, se separan en dos categorías: aceptadas y rechazadas. Las solicitudes aceptadas son aquellas que pueden y van a ser virtualizadas dentro de la topología física; mientras que las solicitudes rechazadas son aquellas que debido a sus características (recursos de CPU y/o de enlace) no pueden ser mapeadas. Estas solicitudes son inmediatamente rechazadas para su mapeo dentro de la topología física, por lo que, si se desea obtener esos recursos en el futuro, se debe reintentar hasta que se permita.

En el caso de que no se pueda mapear la solicitud se imprimirá en consola “No se puede asignar la capacidad solicitada RECHAZO” y se aumentará en uno las solicitudes rechazadas, en el caso contrario, se aumentará en uno las solicitudes aceptadas y se concatenará la solicitud al arreglo que contiene todas las solicitudes aceptadas (variable *trafico*). Este proceso se detalla en la figura 2.6.

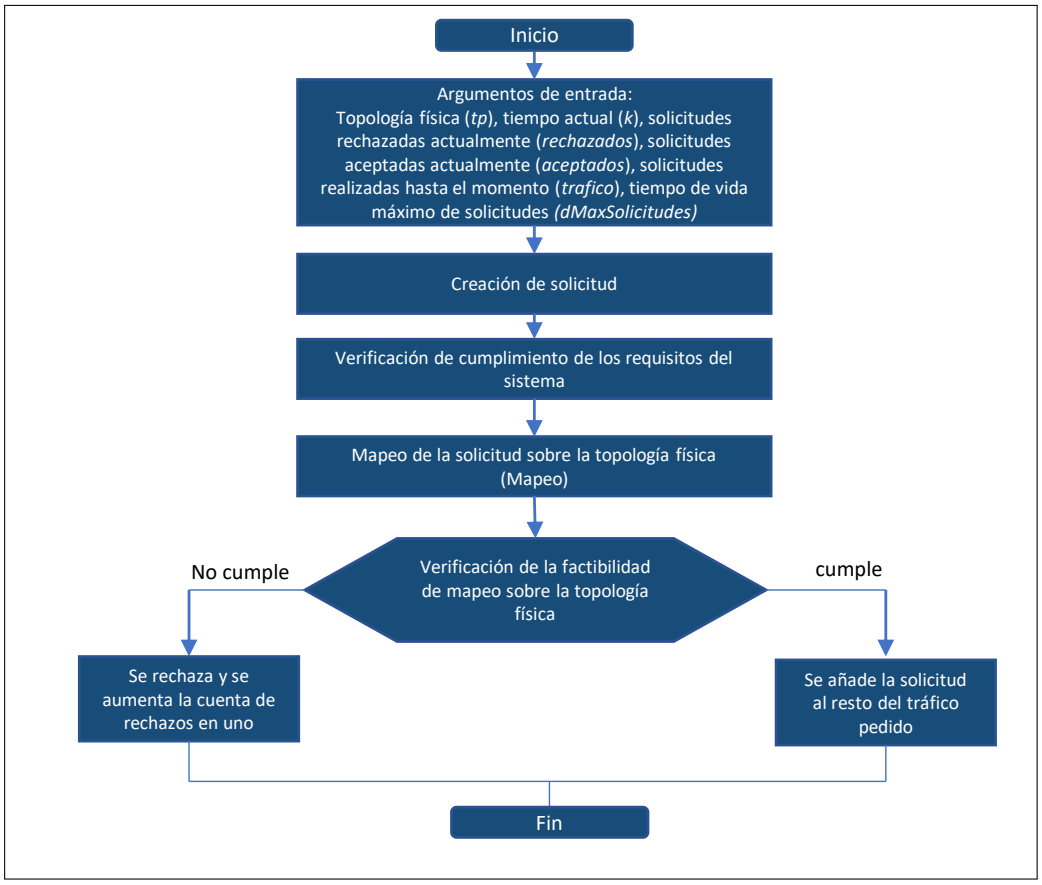


Figura 2.6: Diagrama de flujo el proceso de asignación de recursos dentro de la topología física.

2.2.6. FUNCIÓN SOLICITAR TRAFICO

Con el fin de que se puedan crear solicitudes diferentes de acuerdo al tipo de calidad de servicio y de cumplir con el tercer requisito, se usa la sección de código 11. Dentro de esta sección, se define un número aleatorio de nodos y aristas que conformarán la solicitud (de 2 a 3 nodos y 1 a 2 aristas), este número de nodos y aristas se selecciona con el fin de simplificar la solicitud y por lo tanto su mapeo dentro de la topología física. Los recursos demandados por el grafo que describe la solicitud se crean de manera de manera aleatoria, esto debido a que se busca crear solicitudes de la manera más real posible.

Las calidades de servicio para la simulación se definen de acuerdo a lo mostrado en la tabla 2.1, donde se asignan valores de carga para el CPU (pesos para los nodos) y enlaces (pesos para las aristas) de acuerdo al tipo de calidad de servicio. Los valores que se toman como máximos y mínimos para cada recurso (CPU y enlace) se encuentran normalizados y se basan en una aproximación de lo establecido en la recomendación ITU-R M.2083 (2015) [21]. Asimismo, es en esta función donde se asigna el tiempo de vida a la solicitud, que, de

igual manera, será un valor aleatorio entre 2 y el máximo de tiempo de vida asignado dentro de la función principal. Nuevamente, este tiempo será tratado en *time slots*, que como se indicó anteriormente, es una unidad genérica de tiempo.

Tabla 2.1: Ocupación de CPU y enlace de acuerdo al tipo de tráfico solicitado.

Tráfico	Ocupación de procesamiento mínima [%]	Ocupación de procesamiento máxima [%]	Ocupación de enlace mínima [%]	Ocupación de enlace máxima [%]
eMMB	10	20	30	40
mMTC	1	5	1	2
URLLC	30	40	1	25

```
nodos = randi([2 3], 1,1);
capacidades = zeros(1,nodos);
tipoTrafico = randi([1 3], 1,1);%Eleccion aleatoria
switch tipoTrafico
    case 1 %eMBB
        minimo = 30;
        pico = 40;
        nombre = 'eMMB';
        prominimo = 10;
        promaximo = 20;
    case 2 %mMTC
        minimo = 1;
        pico = 2;
        nombre = 'mMTC';
        prominimo = 1;
        promaximo = 5;
    case 3 %URLLC
        minimo = 1;
        pico = 25;
        nombre = 'URLLC';
        prominimo = 30;
        promaximo = 40;
end
```

Segmento de código 11: Creación de los recursos de una nueva solicitud y selección de la calidad de servicio al que esta corresponderá.

Al tener ya todos los elementos característicos de la solicitud (número de nodos y aristas que la conformarán, a la vez que los recursos de CPU y de enlace requeridos), se ejecuta al sección de código 12, donde se crea el grafo que representará la solicitud (función `CrearGrafo` que se explica a continuación), se verifica que este grafo cumpla con los requisitos establecidos en la subsección 2.2.4, se generará el tiempo de vida en time slots para esta solicitud y se genera la solicitud en su totalidad asociando todos estos elementos en la estructura *nodoT*.

```
grafo = CrearGrafo(nodos, 0, capacidades, minimo, pico);
posiciones = BuscarCamino(tp, nodos);
tiempoVida = randi([2 dMaxSolicitudes], 1,1);
nodoT = struct( ...
    'grafo', grafo, ...
    'nombre', nombre, ... %Para las calidades de servicio
    'posiciones', posiciones, ...
    'tiempoVida', tiempoVida, ...
    'tiempoInicial', tiempoInicial);
```

Segmento de código 12: Generación de una solicitud en su totalidad.

2.2.7. FUNCIÓN CREAR GRAFO

La generación de solicitudes empieza a partir de la creación de grafos, estos, como se indicó en la primera sección de este trabajo, deben ser grafos ponderados no dirigidos. El proceso de creación de la matriz de adyacencia de estos grafos consiste en:

1. **Crear una matriz de adyacencia de tamaño NxN:** Esto se realiza con la ayuda de la función `rand(N)`, que genera una matriz de números pseudo aleatorios entre cero y uno de tamaño NxN [22]. Luego, al ser estos números decimales en el intervalo indicado previamente, se redondea con la función `round` a uno o cero dependiendo de su valor [23]. Se asignan los valores indicados (unos o ceros) ya que de ser otros números, la función que se usará para generar el grafo como tal, asignará el valor correspondiente a cada arista de acuerdo a la matriz de adyacencia, y el objetivo de esta función es crear un grafo en blanco lo más sencillo posible. Este proceso se puede visualizar en la sección de código 13.

```
        G = round(rand(numNodos));
```

Segmento de código 13: Creación de la matriz de adyacencia.

2. **Forzar simetría en la matriz de adyacencia:** Para cumplir con la definición de grafo, la matriz de adyacencia de este debe ser simétrica. Por lo que, para generarla, se capturan todos los elementos que están a la derecha de la diagonal con la ayuda de la función `triu` [24], para luego sumar el resultado obtenido de esta función con la transpuesta del mismo resultado y así obtener una matriz simétrica como se indica la sección de código 14.

```
G = triu(G) + triu(G,1)';
```

Segmento de código 14: Forzado de simetría en la matriz de adyacencia.

3. **Eliminación de bucles:** Al generarse las relaciones entre nodos de manera aleatoria, pueden generarse bucles en uno o más nodos. Para eliminarlos, se asigna un valor de cero a todos los elementos de la diagonal de la matriz de adyacencia como se muestra en la sección de código 15.

```
G = G - diag(diag(G));
```

Segmento de código 15: Eliminación de bucles en la matriz de adyacencia.

Con el fin de crear el grafo, se da como argumento la matriz de adyacencia generada a la función `graph` que construye un grafo ponderado. Hasta este punto aún no se asignan pesos a las aristas del grafo generado.
Finalmente, la función `SolicitarTrafico` es usada tanto para generar la topología física como las solicitudes para la misma, por lo que se verifica qué tipo de solicitud (eMMB, mMTC o URLLC) se está procesando. En el caso de se desee crear una topología física, todos los pesos para los nodos y aristas se crearán con un valor de 100; mientras que, si se trata de una solicitud, se tomarán los valores dados previamente y que son exclusivos para las solicitudes: *prominimo*, *promaximo* y *capacidades*.

2.2.8. FUNCIÓN BUSCAR CAMINO

Como primer acercamiento al mapeo de las solicitudes realizadas, se identifican los nodos dentro de la topología física que van a soportar la carga requerida. El proceso de identificación consiste en fijar un nodo cualquiera dentro de la topología física y calcular la distancia que existe desde este hacia cualquier otro punto dentro de la topología física con la ayuda de la función `shortestpath`, que se describe más adelante dentro del presente capítulo. El camino que cuente con la longitud que se espera tenga la solicitud (número de nodos y

aristas) es el elegido para soportar la nueva solicitud. Este proceso se muestra en la figura 2.7. Donde se toma el ejemplo del mapeo de una solicitud compuesta por tres nodos y dos aritas, en el caso a, no se puede mapear la solicitud pues solamente se encuentra un camino compuesto por dos nodos y una arista, el caso b es similar al a y se rechaza esa propuesta al igual que la propuesta del caso a, por último, para el caso c, se propone un camino que contiene el número de nodos y aristas que se requieren, por lo que se mapea la solicitud. Además, este proceso de la función `CrearGrafo` es descrito en la sección de código 16.

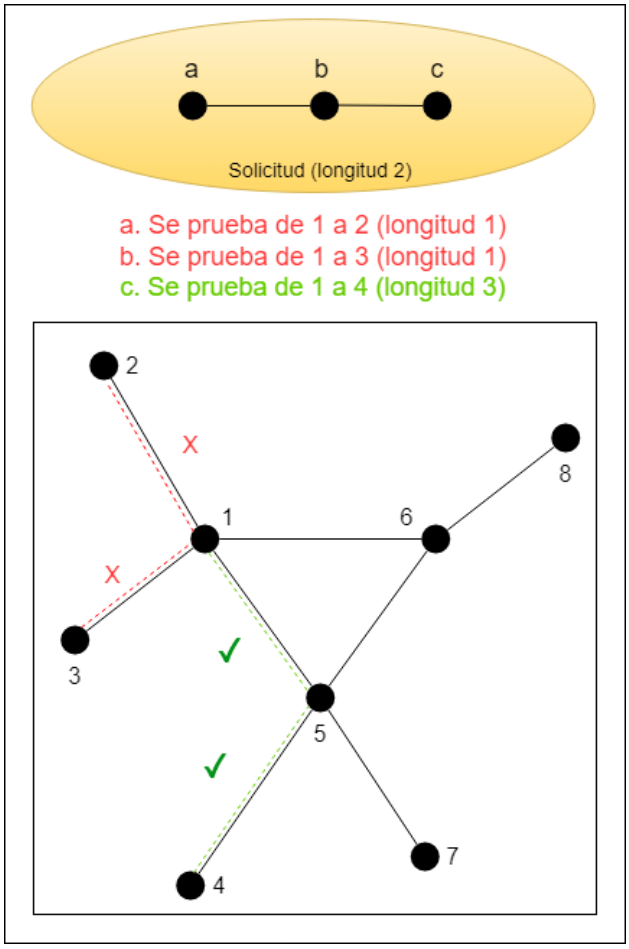


Figura 2.7: Proceso de selección de camino para el mapeo de una solicitud de tres nodos. En rojo se muestran dos posibles caminos que no cumplen con la longitud requerida por la solicitud, uno que va desde el nodo 1 hasta el nodo 2 de la topología física y otro que va desde el nodo 1 hasta el nodo 3 de la topología física. En verde se muestra un posible camino que puede satisfacer los requerimientos de la solicitud, este camino va desde el nodo 1 hasta el nodo 4, pasando por el nodo 5 de la topología física

Se debe tener en cuenta que el proceso de búsqueda de caminos no necesariamente empezará desde el nodo uno de la topología física, sino que se elegirá el punto de inicio de la búsqueda aleatoriamente. Asimismo, si la búsqueda falla para el primer origen seleccionado, se elegirá otro aleatoriamente.

```
function [posiciones] = BuscarCamino(G, nodos)
numnodos = length(G.Nodes.Size);
for inicio=1:3 %doy 3 posibles puntos de partida aleatorios
    var_aleatoria = randi([1 numnodos], 1);
    for i=1:numnodos
        for j=var_aleatoria:numnodos
            if j == i %no se busca un camino que inicie y termine en el
                mismo nodo
                break
            end
            paths = shortestpath(G,j,i, 'Method','positive');
            if length(paths) == nodos
                posiciones = paths;
                return
            end
        end
    end
end
end
```

Segmento de código 16: Búsqueda de caminos para poder mapear una solicitud.

2.2.9. FUNCIÓN SHORTESTPATH

Esta función calcula el camino más corto entre dos nodos [25]. Durante la implementación de esta función se utiliza el algoritmo de Dijkstra ya que este considera pesos en las aristas de los grafos y ningún grafo contiene aristas con pesos negativos. La sintaxis de la función es la que se describe en la sección de código 17.

```
paths = shortestpath(G,inicio,fin, 'Method','positive');
```

Segmento de código 17: Cálculo del camino más corto con el algoritmo de Dijkstra

donde los elementos de entrada son:

- **G:** Es una estructura que contiene los detalles sobre el grafo en el cual se va a calcular el camino más corto.
- **inicio:** Es un entero positivo que específica el nodo desde el que empieza la ruta.

- **fin:** Es un entero positivo que específica el nodo en el que termina la ruta.
- **Method:** Es el parámetro que permite especificar qué tipo de algoritmo se va a usar, en este caso se usa la opción “positive” que corresponde con el algortimo de Dijkstra previamente descrito.

mientras que el elemento de salida es:

- **Paths:** Es un vector que contiene los valores enteros positivos de las posiciones de los distintos nodos que el algoritmo seleccionó como camino más corto.

2.2.10. FUNCIÓN MAPEO

El proceso de mapeo se basa en la técnica de VNE descoordinado, que consiste en, para cada nodo virtual, hacer la búsqueda de un nodo físico sobre el cual mapear la solicitud [26]. Estos nodos físicos son obtenidos previamente con la ayuda de la función `BuscarCamino`. Se verifica si la topología física se encuentra en capacidad de soportar los anchos de banda y capacidades de procesamiento indicados y en el caso de que se puedan asignar, permitir más adelante que se dediquen dichos recursos. En el caso contrario, se emitirá una alerta de solicitud rechazada y se descartará la misma.
La función `Mapeo` se crea con el fin de dar cumplimiento al cuarto requisito indicado y se divide en tres partes, en la primera se realiza una revisión de los nodos a mapear dentro de la topología física. Para esto se usa el vector *posiciones* que contiene las posiciones de los nodos que soportarán la solicitud dentro de la topología física, obtenido de la función `BuscarCamino`. Se extraen las capacidades disponibles de la topología física y se compara con las capacidades solicitadas, si la topología física ofrece los recursos solicitados, se permite la asignación de recursos más adelante, caso contrario la función termina.
Para la segunda parte de la función se revisa las capacidades de los enlaces que componen la ruta solicitada. Para esto se utiliza la función `BuscarPar` que se describe más adelante, con el resultado de esta función se puede obtener la capacidad actual del enlace que soportará la solicitud. Para poder determinar si este enlace soportará la solicitud se realiza la resta de la capacidad actual del mismo con la capacidad solicitada, en caso de que este valor sea negativo, se emite una alerta de que no existen recursos necesarios y se descarta la solicitud, en el caso contrario se permite la asignación de recursos en la tercera parte de esta función. Este proceso se encuentra en la figura 2.8.
En la tercera parte de esta función se realiza la asignación de los recursos, para esto se debe poder mapear los recursos solicitados tanto en nodos como en enlaces, esta condición se

verifica con las variables de control *cumple1* (que verifica que se cumpla la condición de que existan recursos disponibles para los nodos) y *cumple2* (que verifica si se cumplen las condiciones de recursos para los enlaces). Para la asignación de recursos se buscan los nodos y enlaces solicitados dentro de la topología física como se realizó en las dos primeras partes respectivamente, y luego se reservan los recursos indicados por la solicitud. La acción de reservado de recursos consiste en disminuir a la capacidad de la topología física el valor solicitado, tanto para nodos como para enlaces. Finalmente se tiene como salida a la topología física con los cambios realizados de haber sido posible y un aviso de si se pudo realizar o no la asignación solicitada. En el caso de que no se haya podido reservar los recursos específicados por la solicitud, se devuelve a la topología física tal como estaba antes de analizar la factibilidad de mapear una nueva solicitud.

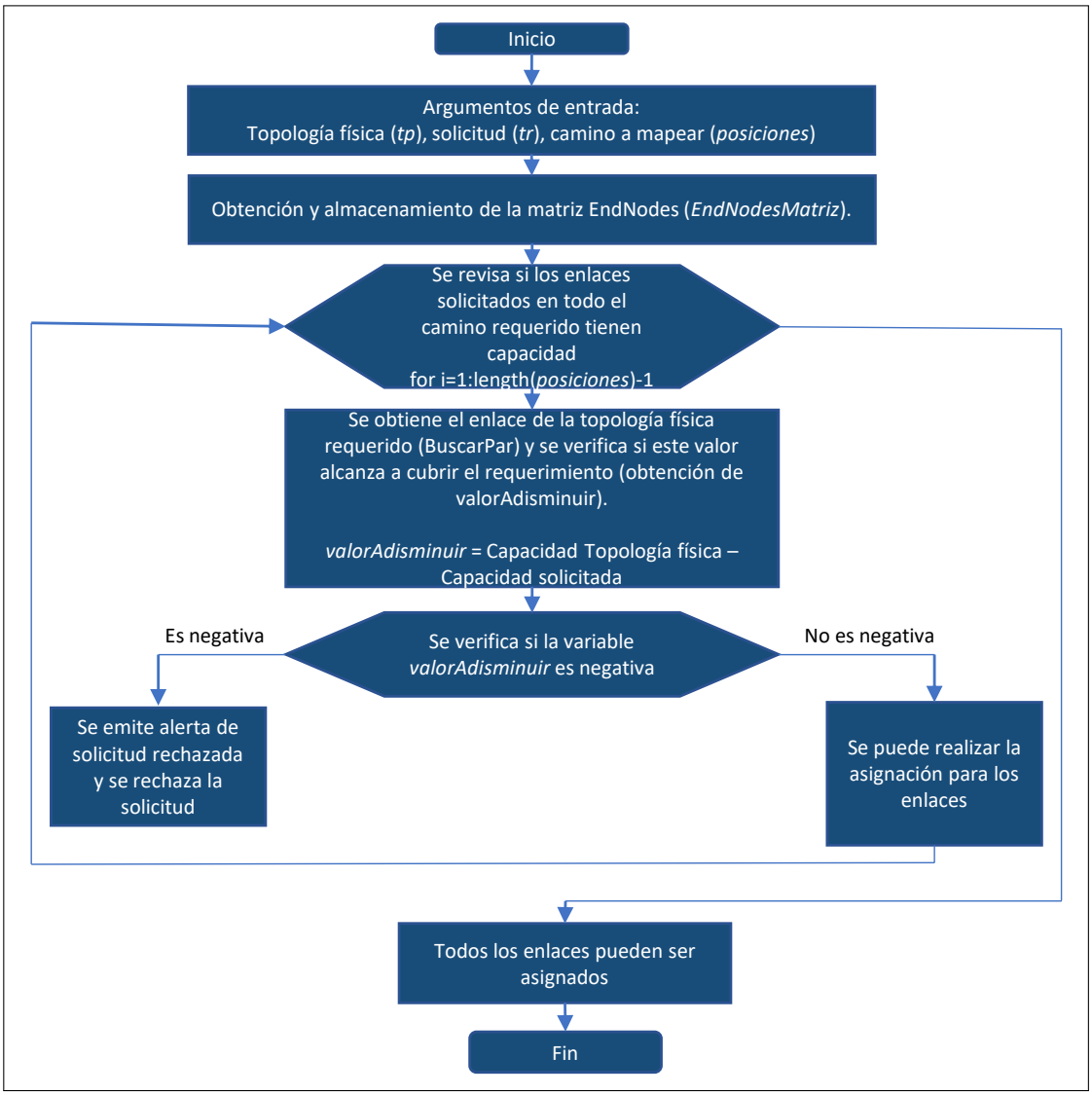


Figura 2.8: Diagrama de flujo de la búsqueda de recursos para los enlaces de la topología física.

2.2.11. FUNCIÓN BUSCAR PAR

Debido a la forma en que Matlab presenta la información relacionada a un grafo, es necesario un método para identificar a los enlaces de una topología. El proceso de identificación consiste en buscar los nodos que se encuentran a los extremos del enlace, estos se pueden encontrar en la matriz “EndNodes”. Para poder encontrarlos, se da los números de los nodos de la solicitud que se realiza, este valor es asignado aleatoriamente al momento de generar la solicitud, así ya se conoce qué nodos deberían encontrarse a los extremos del enlace que se está buscando. Seguido, se establece el rango de búsqueda de los nodos, que estará dado por el número de filas que tiene la matriz “EndNodes”. Con esto se hace la búsqueda por columnas comparando miembro a miembro si los valores de la columna de origen corresponden con los valores de la columna de destino, y de acuerdo a los nodos que se conoce deberían encontrarse a los extremos. Este proceso se puede evidenciar en la sección de código 18.

```
function [encontrado] = BuscarPar(EndNodes, ppar, spar)
dimensionesMatriz = size(EndNodes);
numfilas = dimensionesMatriz(1);
encontrado1 = find(EndNodes==ppar);
encontrado2 = find(EndNodes==spar);
for i=1:length(encontrado1)
    for j=1:length(encontrado2)
        if (encontrado2(j) == encontrado1(i) + numfilas)
            encontrado = encontrado1(i);
            return
        elseif (encontrado2(j) == encontrado1(i) - numfilas)
            encontrado = encontrado2(j);
            return
        end
    end
end
end
```

Segmento de código 18: Algoritmo de búsqueda de enlaces.

2.2.12. FUNCIÓN VERIFICAR TIEMPO

Con el fin de mantener un monitoreo del tiempo de vida de cada solicitud realizada y dar cumplimiento al quinto requisito indicado, se calcula el tiempo transcurrido desde que se crearon todas las solicitudes hasta el momento en el que se está evaluando, que sería el tiempo actual de la simulación. En el caso de que este tiempo transcurrido sea mayor que el tiempo de vida de alguna o algunas de las solicitudes se inicia el proceso de liberación de recursos con la función `LiberarRecursos`, este proceso consiste en asignar nuevamente los recursos que se encontraban ocupados a la topología física. En el caso de que no, se crea un registro de solicitudes activas en ese momento. Además, para mantener un registro de las solicitudes liberadas, se guardan todas estas solicitudes hasta ese momento y se evalúa cada vez que se llama a la función `VerificarTiempo`.

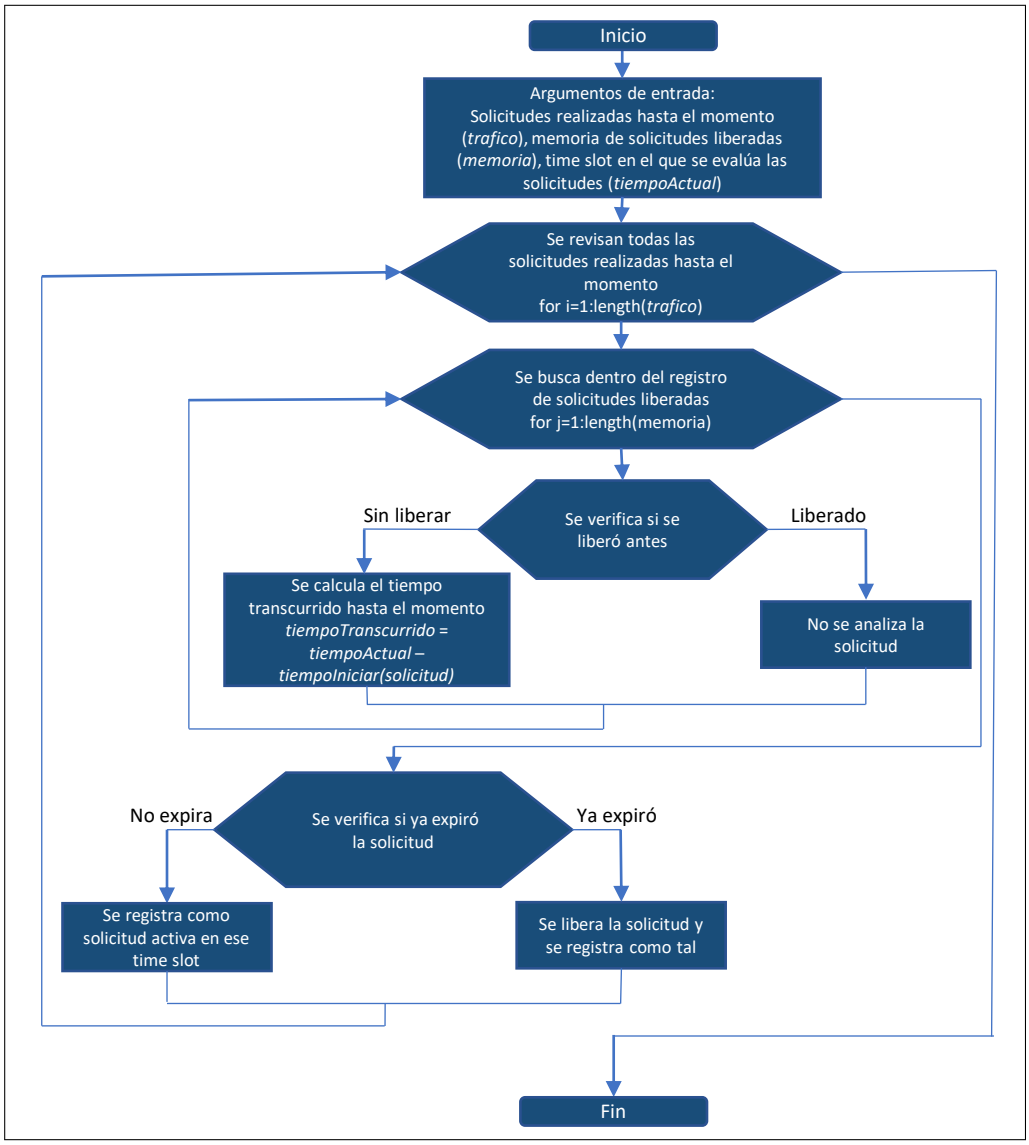


Figura 2.9: Diagrama de flujo del control de tiempo de vida de solicitudes realizadas.

Es necesario indicar que esta función es llamada en cada time slot, por lo que siempre se va a tener un registro de solicitudes liberadas y de solicitudes activas. Este proceso se detalla en la figura 2.9.

2.2.13. FUNCIÓN LIBERAR RECURSOS

Luego de que cada solicitud haya cumplido su tiempo de vida, es necesario liberar los recursos que cada una de estas ocupó. Para esto se vuelve a tomar la solicitud realizada y se buscan los recursos que esta ocupó dentro de la topología física. Primero se compara nodo a nodo y luego enlace por enlace como se realizó con la función `Mapeo`. Cuando se encuentran los recursos ocupados, se liberan sumando a la topología física la capacidad disponible de acuerdo a los requerimientos que se habían demandado. La diferencia de la función `LiberarRecursos` con la función `Mapeo` es que con esta función no se emiten alertas de insuficiencia de recursos y se añaden recursos a la topología física en vez de restarlos.

2.2.14. FUNCIÓN OBTENER METRICAS

Como parte final de toda la simulación se tiene la evaluación del sistema. Para esto, se obtienen las siguientes métricas [6]:

- **Ingresos (Revenue):** Los ingresos se refieren a la suma de los recursos virtuales que realmente solicitaron las entidades virtuales. La obtención de estos se encuentra dada por la siguiente ecuación:

$$Revenue = \sum_{G_r}(A_S^L) + \sum_{G_r}(A_S^N) \tag{2.1}$$

  Donde:

  - G_r representa la red solicitada.

- **Radio de aceptación (Acceptance ratio):** La métrica de radio de aceptación mide la cantidad de solicitudes de red virtual que el algoritmo de mapeo podría integrar por completo, dividida para la cantidad total de solicitudes de red virtual. Esta métrica se encuentra definida por la siguiente ecuación:

$$AR = \frac{A}{T} \tag{2.2}$$

Donde:

- A representa el número de demandas aceptadas.
- T representa el número total de demandas.

- **Manejo de solicitudes:** Esta métrica muestra el número de solicitudes por tipo de tráfico manejadas a la vez por la topología física.

Para el calculo de la métrica ingresos se utiliza principalmente la matriz de solicitudes activas por time slot que fue definida en la función `verificarTiempo`. Para obtener la cantidad de procesamiento y de enlaces utilizados en un tiempo i, se llama a la propiedad "Nodes" y "Edges" de la j - ésima solicitud, esto se puede ver en la sección de código 19.

```
%Para CPU
MatrizcpuUsado = trafico(matrizActivos(i,j)).grafo.Nodes.Size;
cpuPorSolicitud(j) = sum(MatrizcpuUsado); %sumo las capacidades
    solicitadas por cada nodo de cada solicitud
%Para Enlaces
MatrizlinkUsado = trafico(matrizActivos(i,j)).grafo.Edges.Weight;
linkPorSolicitud(j) = sum(MatrizlinkUsado);
```

Segmento de código 19: Obtención de los recursos usados por unidad de tiempo.

Luego, se suman todos los valores de recursos obtenidos, como se muestra en la sección de código 20.

```
sumatoriaCPUPorTiempos(i) = sum(cpuPorSolicitud);%sumo las capacidades de
    todas las solicitudes en un tiempo
sumatoriaLINKPorTiempos(i) = sum(linkPorSolicitud);
```

Segmento de código 20: Obtención de CPU y ancho de banda utilizados por unidad de tiempo.

Posteriormente, se obtiene el ingreso por medio de la suma de los valores de CPU y ancho de banda que se obtuvieron previamente, estos valores se muestran en la tabla 2.1.
Se debe tener en cuenta que como dentro de la matriz de solicitudes activas se almacena el número de solicitud manejada, se debe ignorar el caso de que no exista solicitud a analizar (elemento de la matriz sea igual a cero), de lo contrarío se causaría un error ya que se trata de evaluar la posición 0 del arreglo de solicitudes realizadas.

El radio de aceptación se consigue por medio de la división de las solicitudes aceptadas entre las solicitudes totales como se muestra en la sección de código 21, durante la ejecución de la simulación se calcula este valor, de manera que se obtiene una gráfica acumulada del comportamiento de esta métrica a lo largo del tiempo.

```
rechazados = length(memrechazos);
aceptados = length(trafico);
accratio(tiempo) = aceptados/(aceptados + rechazados);
```

Segmento de código 21: Obtención de la razón de aceptación.

Finalmente, para la obtención de la métrica manejo de solicitudes, se llama al atributo "nombre" de cada solicitud al mismo tiempo y se clasifica de acuerdo a su valor. Al clasificar, se realiza un conteo del tipo de tráfico al que corresponde cada solicitud. El atributo nombre especificará el tipo de tráfico al que corresponde una solicitud de manera que se conseguiría todos los tipos de tráfico solicitados en un time slot. Este proceso se indica en la sección de código 22.

```
QoStrafico = trafico(matrizActivos(i,j)).nombre;
switch QoStrafico
    case 'eMMB'
        contador_eMMB = contador_eMMB + 1;
    case 'mMTC'
        contador_mMTC = contador_mMTC + 1;
    case 'URLLC'
        contador_URLLC = contador_URLLC + 1;
end
```

Segmento de código 22: Clasificación y conteo de solicitudes de acuerdo al tipo de tráfico.

2.2.15. INTERFAZ DE USUARIO

Se desarrolla una interfaz gráfica como se muestra en la figura 2.10 que expone la topología física a la vez que la solicitada, junto a los diferentes elementos que conforman el escenario de simulación, como serían el tiempo de simulación, el tipo de tráfico que se solicita (en el caso de que se solicite) y el número de solicitud que se visualiza (en caso de que se realice una dentro del time slot). Dentro de esta interfaz, se da oportunidad al usuario de elegir el tiempo total de simulación, el número de peticiones (clientes) que se simularán y la topología física sobre la que se mapearán las solicitudes. Asimismo, al momento de que los recursos

son liberados, se notificará al usuario cuando se llegue al time slot donde caduca la solicitud y se permitirá ver la(s) topología(s) liberada(s).

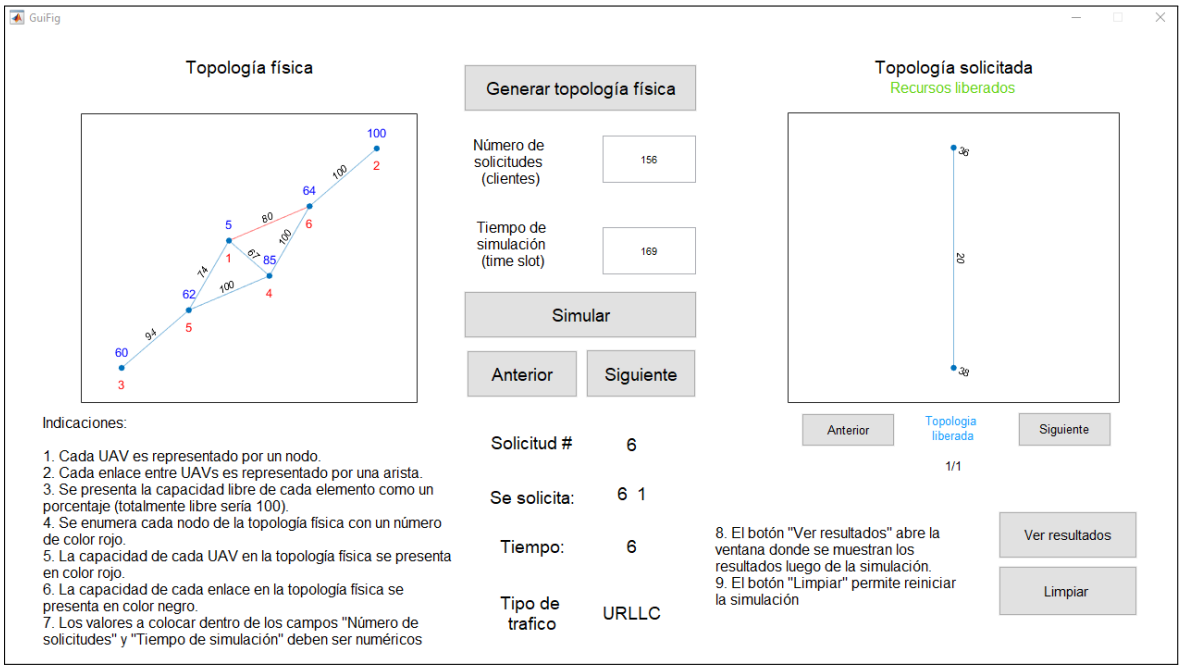


Figura 2.10: Programa de visualización de virtualización para Virtual Network Embedding en un ambiente FANET.

Previo a mostrar la ventana de simulación, se presenta la carátula principal del proyecto que se puede visualizar en la figura 2.11.

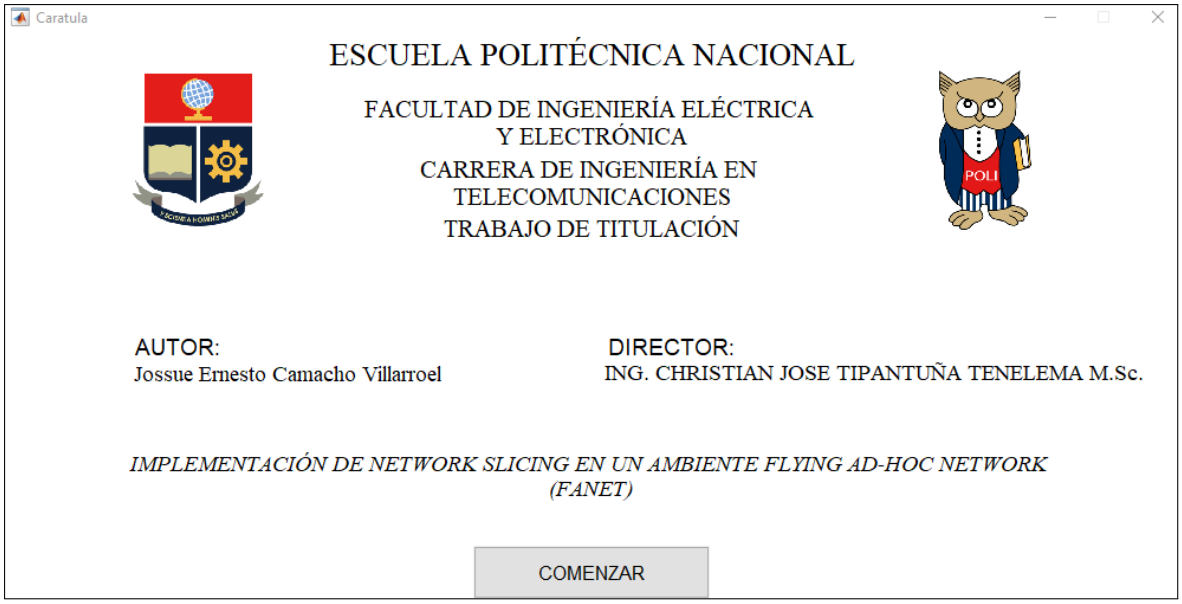


Figura 2.11: Carátula de la interfaz de usuario.

Luego, al realizar click sobre el botón *COMENZAR*, aparecerá la ventana mostrada en la

figura 2.12, aquí el usuario deberá elegir una topología física de acuerdo a su preferencia, estas topologías se irán generando de manera aleatoria manteniendo solamente el número de nodos. Después de esto, deberá colocar el número de usuarios que va a simular, este debe ser un número entero mayor a 0, y el tiempo que va a durar la simulación en time slots, nuevamente, este debe ser un número entero mayor a cero. El orden de selección y llenado de variables recomendado es el indicado por los números en la figura 2.12. Una vez el usuario haya llenado los campos necesarios, deberá presionar el botón *Simular*, que calculará todo el proceso de virtualización de todos los clientes indicados en el intervalo de tiempo designado. Posteriormente, para poder visualizar las solicitudes de una en una, como el espacio que se les asigna dentro de la topología física, se debe usar los botones *Anterior* y *Siguiente* que se encuentran en el centro de la ventana, estos botones mostrarán time slot a time slot la solicitud de virtualización y su respectiva asignación, rechazo y/o liberación.

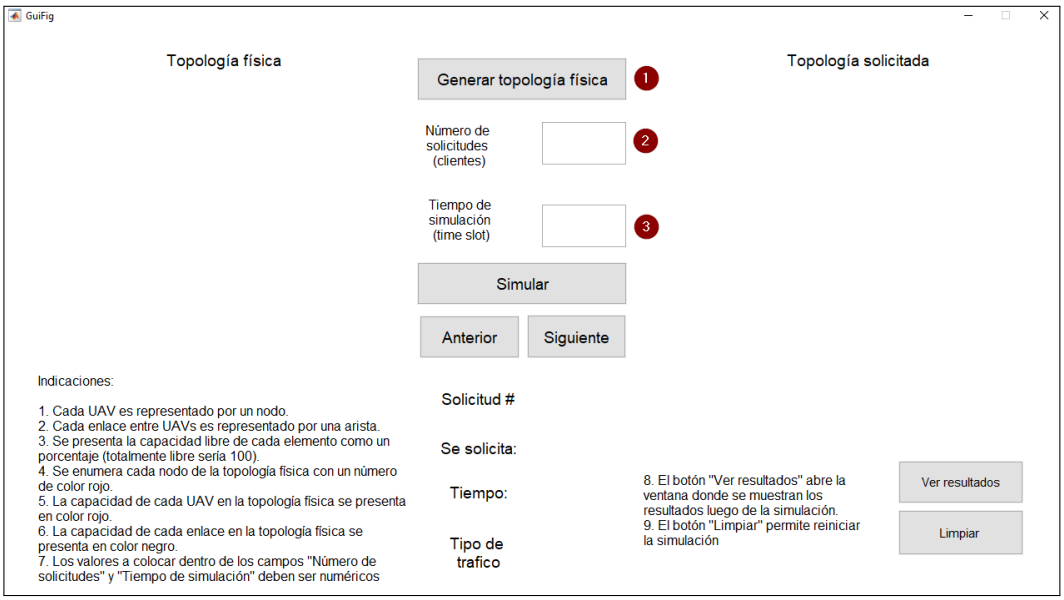


Figura 2.12: Ventana de ingreso y selección de elementos. Se numera el orden recomendado de los campos a seleccionar y/o llenar.

Para el caso de liberación, aparecerán nuevos botones que permitirán navegar entre las solicitudes liberadas como se muestra en la figura 2.13. En el caso de que se haya hecho una solicitud mientras que se liberan recursos, aparecerá primero la solicitud y con el botón *Siguiente* que aparecerá posteriormente, se podrán ver la o las solicitudes liberadas. Cuando se desee evaluar el desempeño del escenario propuesto, se debe realizar click en el botón *Ver resultados*, esto abrirá una venta que, luego de presionar el botón *Mostrar resultados*, mostrará el comportamiento de cada una de las métricas de desempeño en el tiempo, como se muestra en la figura 2.14.

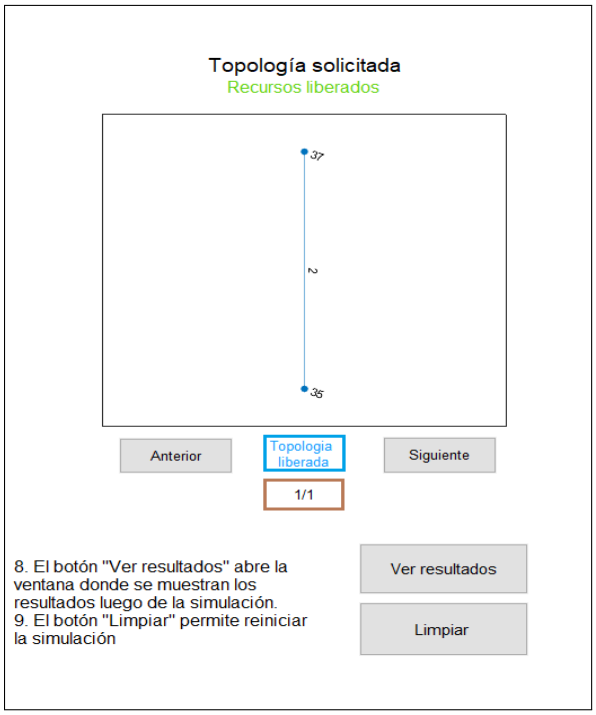


Figura 2.13: Alerta de recursos liberados y muestra de nueva solicitud realizada. Dentro del recuadro azul, el texto será “solicitado” en el caso de que se muestre una solicitud para la topología física y, en el caso de que se muestre una solicitud liberada, el texto será “Topología liberada” en color celeste. Asimismo, el contador que se encuentra dentro del recuadro café mostrará el número de solicitud liberada, en el caso de que este contador sea cero, es debido a que se muestra una solicitud recientemente realizada.

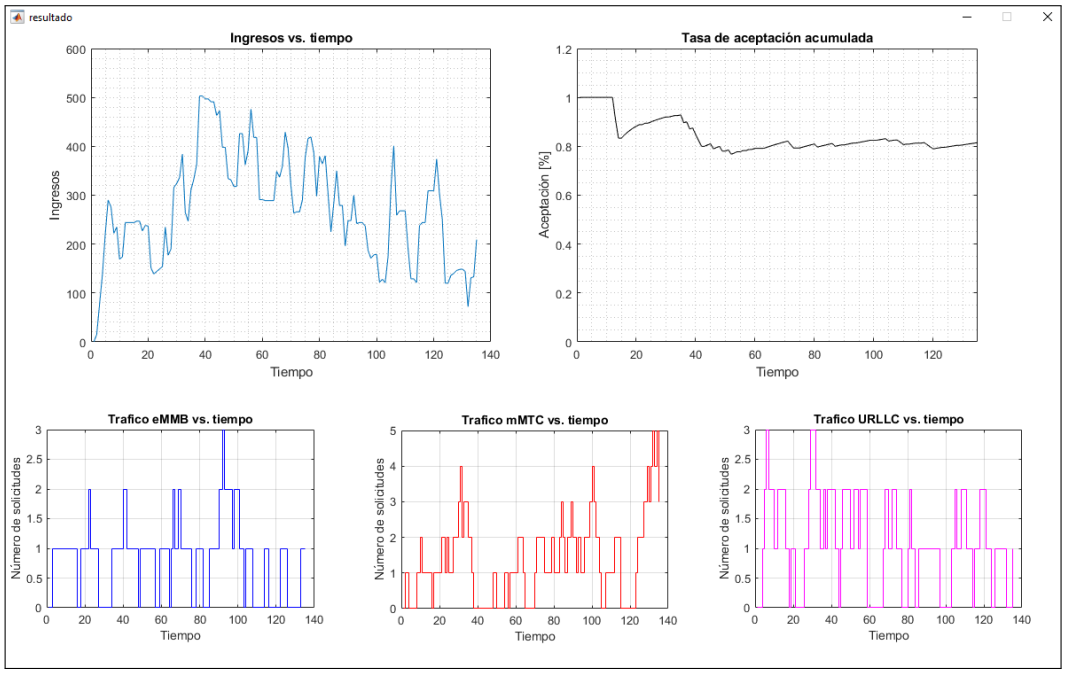


Figura 2.14: Gráficas de las métrcias obtenidas.

3. RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

En el presente capítulo se presentan las gráficas que se obtuvieron como resultado de la simulación de Network Slicing en el contexto de las redes FANET. Los resultados se enfocan en medir el rendimiento de la virtualización de recursos, por lo que se presentan tasas de aceptación para las diferentes solicitudes de acuerdo al tipo de calidad de servicio que se solicita.

3.1. Resultados

Con el fin de evaluar el desempeño de la virtualización propuesta se simularon cuatro escenarios distintos. Los primeros tres escenarios consideran solamente un tipo de servicio (mMMTC, eMMB y URLLC) cada uno, mientras que el último escenario considera a todos los tres tipos de servicio simultáneamente. Asimismo, durante la ejecución de cada escenario, la topología física es sometida a tres niveles distintos de flujos de tráfico con el motivo de visualizar las distintas métricas de desempeño de la virtualización bajo distintas condiciones de demanda de recursos (demandas bajas, moderadas y altas), estos flujos de tráfico son presentados en la tabla 3.1 donde se indica el número de solicitudes propias de cada flujo de tráfico y como se los denomina. Con el fin de obtener las gráficas de manera independiente de la interfaz gráfica, al final de cada simulación se guardaron las variables necesarias para volver a generar las gráficas en archivos de formato *.mat*, de manra que se puedan volver a utilizar dentro de otros scripts.

Tabla 3.1: Denominación de flujo de tráfico de acuerdo al número de solicitudes creadas por unidad de tiempo.

Flujo de tráfico	Número de solicitudes	Período de simulación [time slots]
Bajo	50	150
Moderado	100	150
Alto	150	150

Los resultados presentados a continuación son el producto de realizar la simulación para cada escenario 10 veces por cada tipo de servicio (eMMC, URLLC y mMTC), es decir, por cada tipo de servicio se obtuvieron 30 resultados, 10 para tráfico bajo, 10 para tráfico moderado y 10 para tráfico alto. Con el promedio obtenido por cada flujo de tráfico se obtuvieron las gráficas que se muestran a continuación.

3.1.1. ANÁLISIS PARA TRÁFICO EMMB

En la figura 3.1 se muestran los resultados para los ingresos en el tiempo para los tres niveles planteados en la tabla 3.1 para el servicio eMMB. Se puede visualizar que el servicio eMMB tiene un alto nivel de ingresos sobre la topología física pasado el time slot 10 para todos los flujos de tráfico, lo que significa que un nivel bajo de solicitudes para este servicio coloca a la topología física bajo un alto nivel de estrés con respecto al uso de sus recursos. Esta situación se presenta debido a las características propias del servicio eMMB, cuyas solicitudes individuales, como se ve en la tabla 2.1, requieren un nivel medio de ocupación de CPU y de enlace. Asimismo, que su comportamiento ante una alta carga de tráfico no es el óptimo debido a que la media para un flujo de tráfico alto es cercana a la media para un flujo de tráfico moderado, lo que significa que este tipo de solicitudes saturan rápidamente los recursos disponibles.

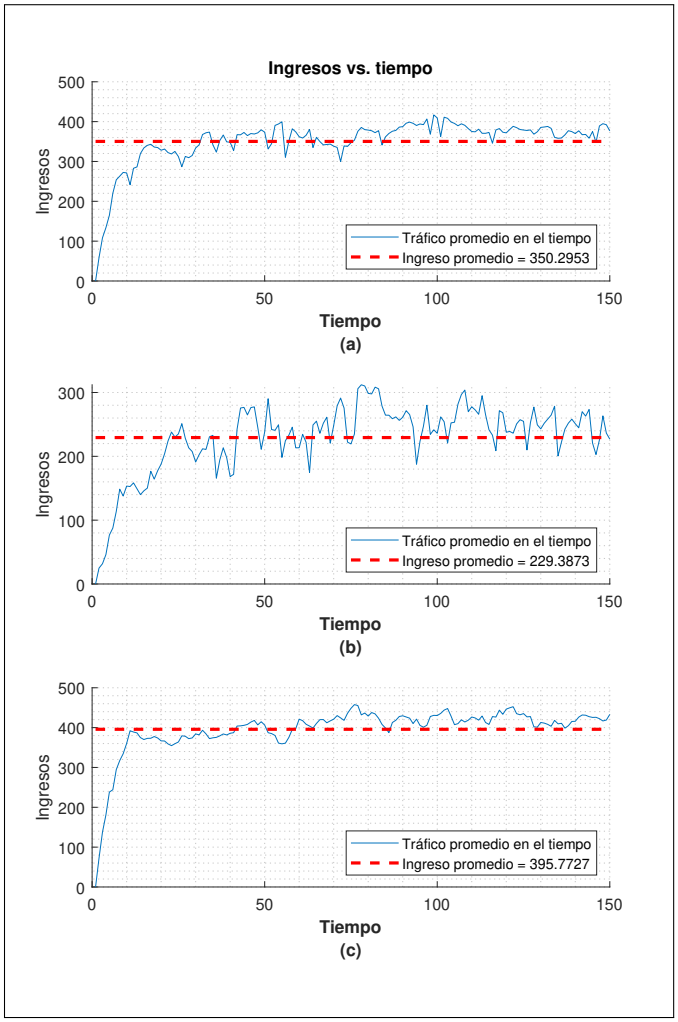


Figura 3.1: Uso de recursos (Revenue) en el tiempo para condiciones de tráfico moderado (a), poco tráfico (b) y tráfico alto (c).

La figura 3.2 corresponde a la tasa de aceptación acumulada para cada flujo de tráfico para el servicio eMMB. En la figura 3.2b se puede visualizar que el servicio eMMB se comporta de mejor manera al tener un flujo de tráfico bajo ya que la tasa de aceptación acumulada promedio es la más alta para este flujo de tráfico (91.05 %). Esta tasa de aceptación se debe a que al haber pocas solicitudes (recursos demandados por los clientes) manejadas por la topología física, se tienen recursos excendentes y por consiguiente una mayor disponibilidad de la red. En cada una de las gráficas se puede apreciar que la tasa de aceptación es mínima (84 %$\pm$4 %) cuando se llega al final del período de simulación, este fenómeno encuentra su justificación en que al final de la simulación, la mayoría de solicitudes se encuentran virtualizadas, provocando así que no existan recursos libres para nuevas solicitudes. Es así como para el caso del manejo de un flujo alto de solicitudes, como se ve en la figura 3.2c, se tendrían los peores valores de aceptación de recursos porque dentro la topología física estaría manejando constantemente nuevas solicitudes de recursos.

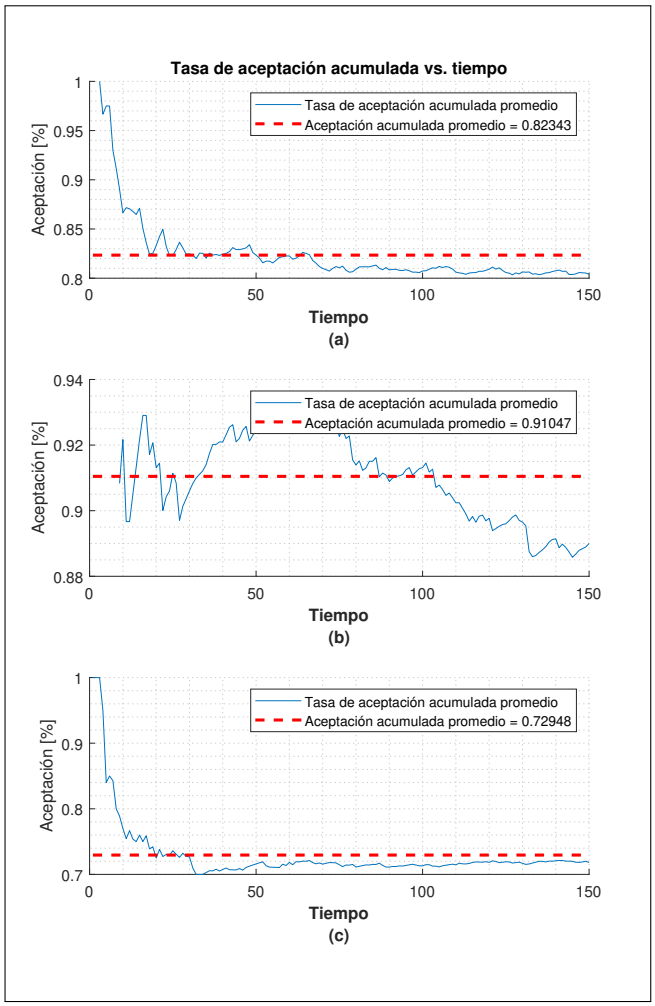


Figura 3.2: Tasa de aceptación acumulada en el tiempo para condiciones de tráfico moderado (a), poco tráfico (b) y tráfico alto (c).

Dentro de cada figura que presenta la tasa de aceptación acumulada, se puede ver un espacio en blanco en los primeros instantes de la simulación, esto se debe a la naturaleza de aparición aleatoria de las solicitudes. Por lo que al inicio no habrían solicitudes realizadas y es imposible medir la tasa de aceptación bajo estas circunstancias.

En la figura 3.3 se presentan las gráficas correspondientes al número de solicitudes por unidad de tiempo que maneja la topología física para el tráfico eMMB, teniendo como máximo un valor de cuatro solicitudes manejadas a la vez por time slot y como promedio tres solicitudes manejadas a la vez.

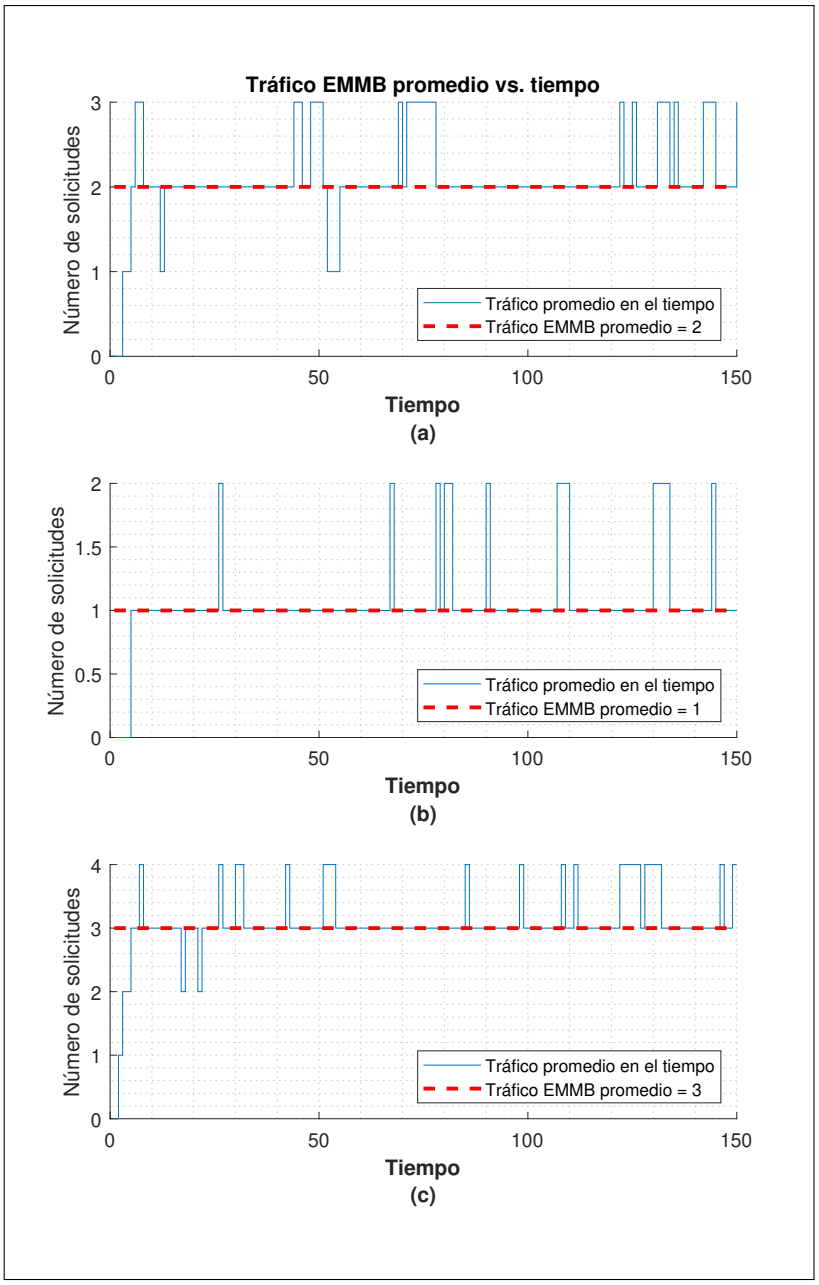


Figura 3.3: Número de solicitudes en el tiempo para condiciones de tráfico moderado (a), poco tráfico (b) y tráfico alto (c).

Estos valores indican que un flujo de tráfico alto (50 solicitudes para un tiempo de simulación igual a 150 time slots) satura con facilidad los recursos ofrecidos por la topología física, aunque, a cambio se obtiene que el número de clientes atendidos a la vez es más alto (para este caso cuatro). Se puede observar en la figura 3.3c un alto nivel de estrés sobre la topología física ya que esta presenta solamente dos momentos donde su manejo de solicitudes por unidad de tiempo disminuye a un valor por debajo al promedio, de igual manera, el número de solicitudes manejadas a la vez solamente aumenta dentro de los primeros siete time slots de la simulación.

El comportamiento en general para el servicio eMMB se puede resumir en que, al tener este servicio solicitudes que exigen una cantidad considerable de recursos (entre 10 % y 20 % para uso de procesador y entre 30 % y 40 % para el uso del enlace) es capaz de saturar-la topología física con relativa facilidad ya que, como se vio en la figura 3.1, los ingresos promedios del flujo de tráfico moderado eran bastante similares a los ingresos promedios del flujo de tráfico alto. Asimismo, se debe tener en cuenta que, por como esta configura la solicitud, los recursos que primero se agotan para este tipo de servicio son los de enlace. Finalmente se debe tener en cuenta que a pesar de la demanda de recursos que este servicio genera, las tasas de aceptación acumuladas tienen un valor por encima del 70 % incluso para el peor de los casos (tráfico alto, figura 3.2).

3.1.2. ANÁLISIS PARA TRÁFICO MMTC

La figura 3.4 presenta los ingresos realizados en la topología física en el tiempo para el servicio mMTC. De los resultados obtenidos se puede visualizar que el servicio mMTC tiene un bajo nivel de ingresos sobre la topología física, esto, debido a que el valor máximo al que se alcanza en el caso de mayor demanda de tráfico (flujo de tráfico alto) es de 63 ingresos. Se debe tener en cuenta que en ningún momento para este tipo de tráfico se saturan los recursos de la topología física, sino que más bien muchos quedan desocupados (debido a que no se supera ni los 100 ingresos por time slot), dotando al topología física la capacidad de aceptar nuevas solicitudes. Dando como resultado que, en el caso de alto flujo de tráfico (figura 3.4c), se tenga una curva estable desde el time slot 15. Asimismo se puede visualizar que para esta caso, los ingresos promedios para los distintos casos de flujo de tráfico son diferentes entre ellos (incremento de entre 15 ingresos por time slot entre aumento de tipo de flujo de tráfico).

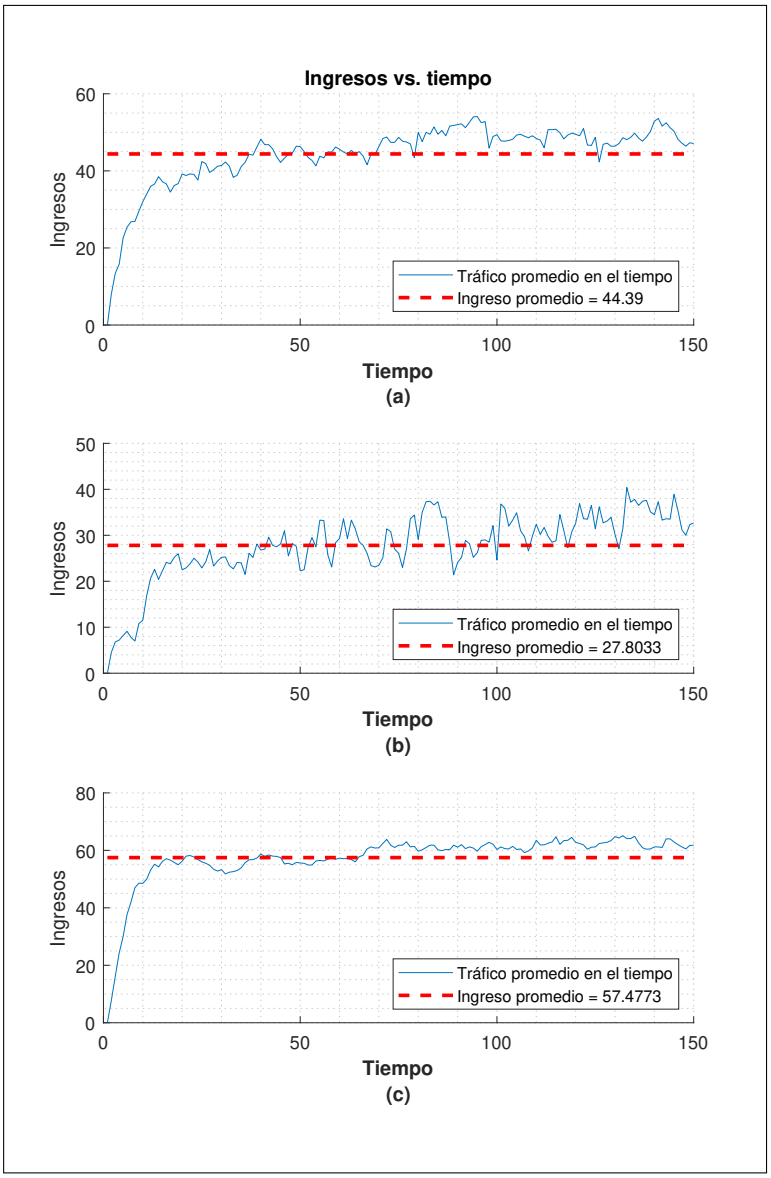


Figura 3.4: Uso de recursos (Revenue) en el tiempo para condiciones de tráfico moderado (a), poco tráfico (b) y tráfico alto (c).

La figura 3.5 presenta la tasa de aceptación de las solicitudes para servicio mMTC. En cada uno de las figuras aquí presentados se puede encontrar que, para este tipo de tráfico, se aceptan todas las solicitudes realizadas incluso cuando se hace una solicitud por cada time slot. Es decir, en cualquiera de los casos de flujo de tráfico se obtiene una tasa de aceptación acumulada del 100 %, esto debido a que en ningún momento se superan los recursos ofrecidos por la topología física, por lo que siempre habrá disponibilidad de recursos si solo se maneja este tipo de servicio únicamente. Se debe tener en cuenta que si las solicitudes realizadas se mantuvieran más tiempo virtualizadas dentro de la topología física, el tiempo de simulación fuera mucho más grande y siempre se realizaran solicitudes nuevas,

la topología física terminaría saturándose. Sin embargo, el número de solicitudes manejadas a la vez sería bastante grande.

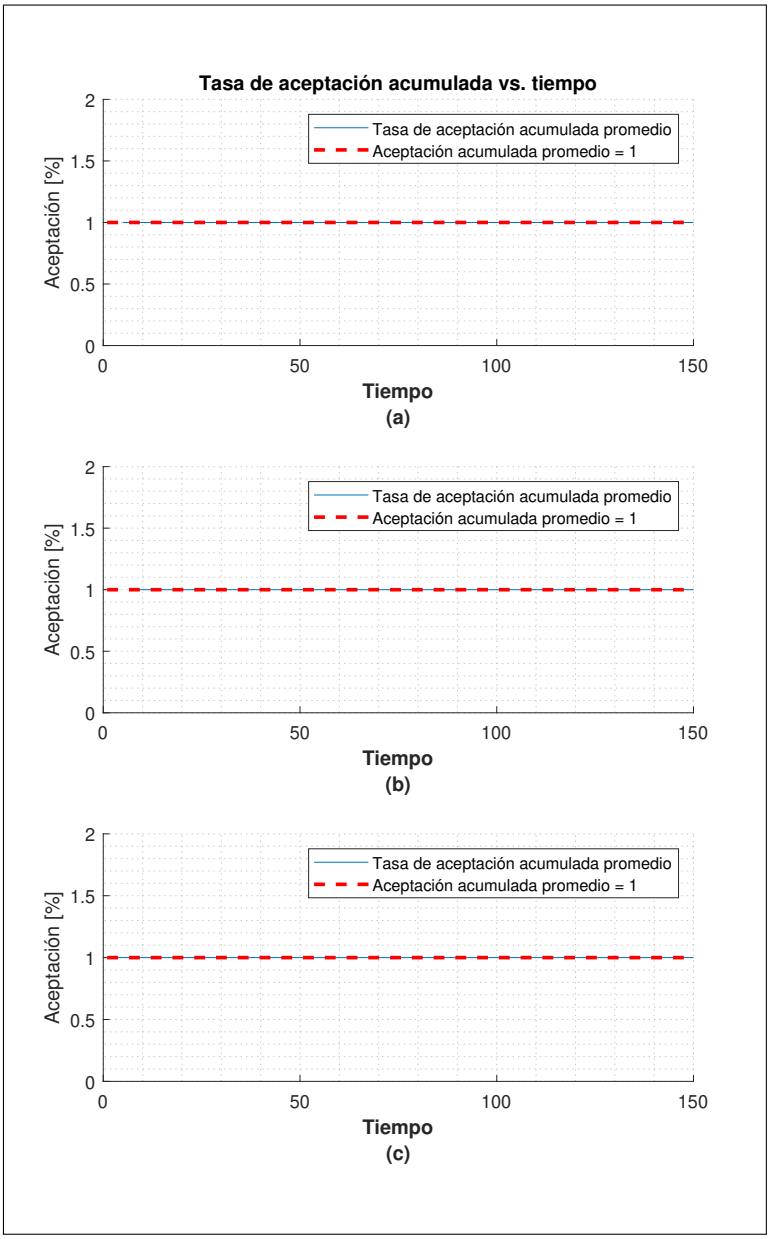


Figura 3.5: Tasa de aceptación acumulada en el tiempo para condiciones de tráfico moderado (a), poco tráfico (b) y tráfico alto (c).

En la figura 3.6 se presenta el número de solicitudes realizadas por unidad de tiempo para el servicio mMTC. En cada una de las figuras se puede apreciar que este tipo de servicio permite manejar hasta 6 solicitudes a la vez o una media de 5 solicitudes manejadas a la vez en el caso de mayor flujo de tráfico. Esto resalta que, para este tipo de tráfico, la topología física es capaz de brindar un alto nivel de disponibilidad.

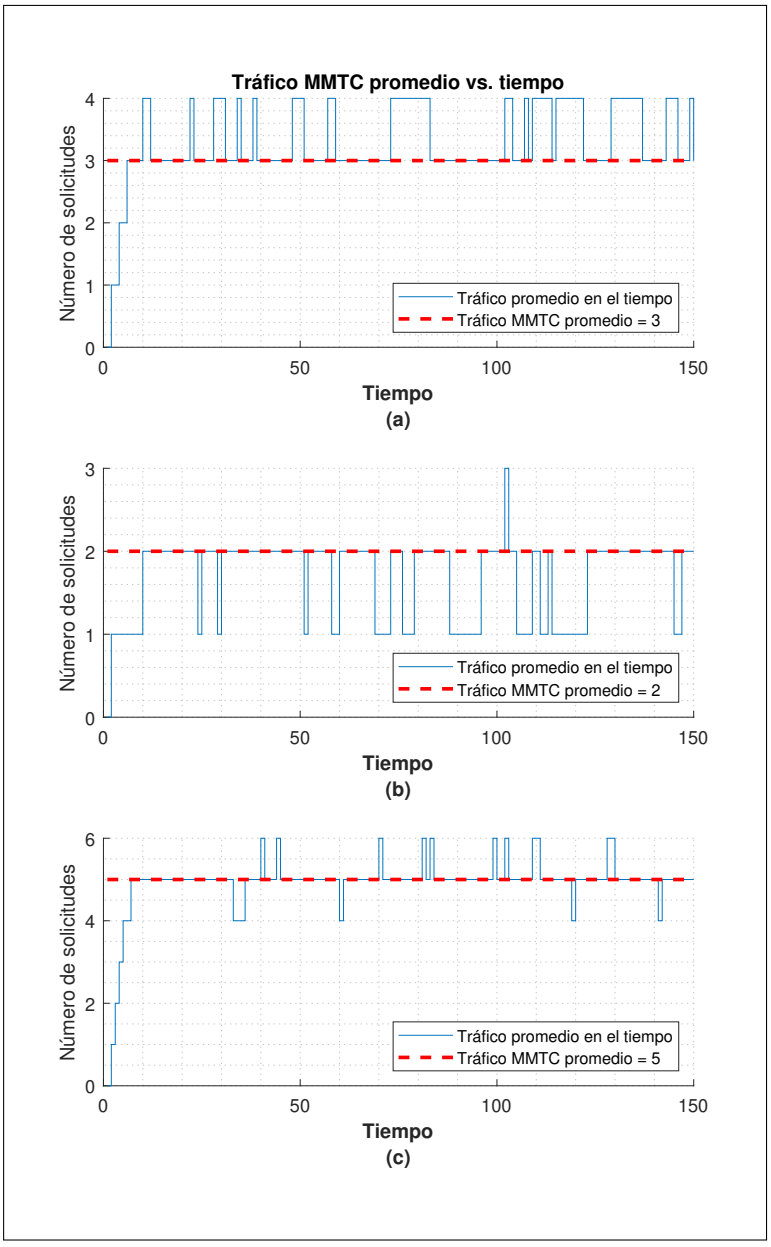


Figura 3.6: Número de solicitudes en el tiempo para condiciones de tráfico moderado (a), poco tráfico (b) y tráfico alto (c).

3.1.3. ANÁLISIS PARA TRÁFICO URLLC

La figura 3.7 muestra los ingresos realizados en la topología física ante la solicitud de tráfico con el tipo de servicio URLLC. En los resultados obtenidos se puede visualizar que el servicio URLLC genera un ingreso promedio que, para todos los casos, se encuentra entre los 200 y 300 ingresos por segundo, lo que se traduce en una alto uso de recursos por parte de las solicitudes realizadas. Debido a que los ingresos para la topología física ascienden a 200 o más durante los primeros 10 time slots para los casos de flujos de tráfico moderado, figura

3.7a y alto 3.7c, se puede deducir que la topología física se queda sin recursos disponibles rápidamente.

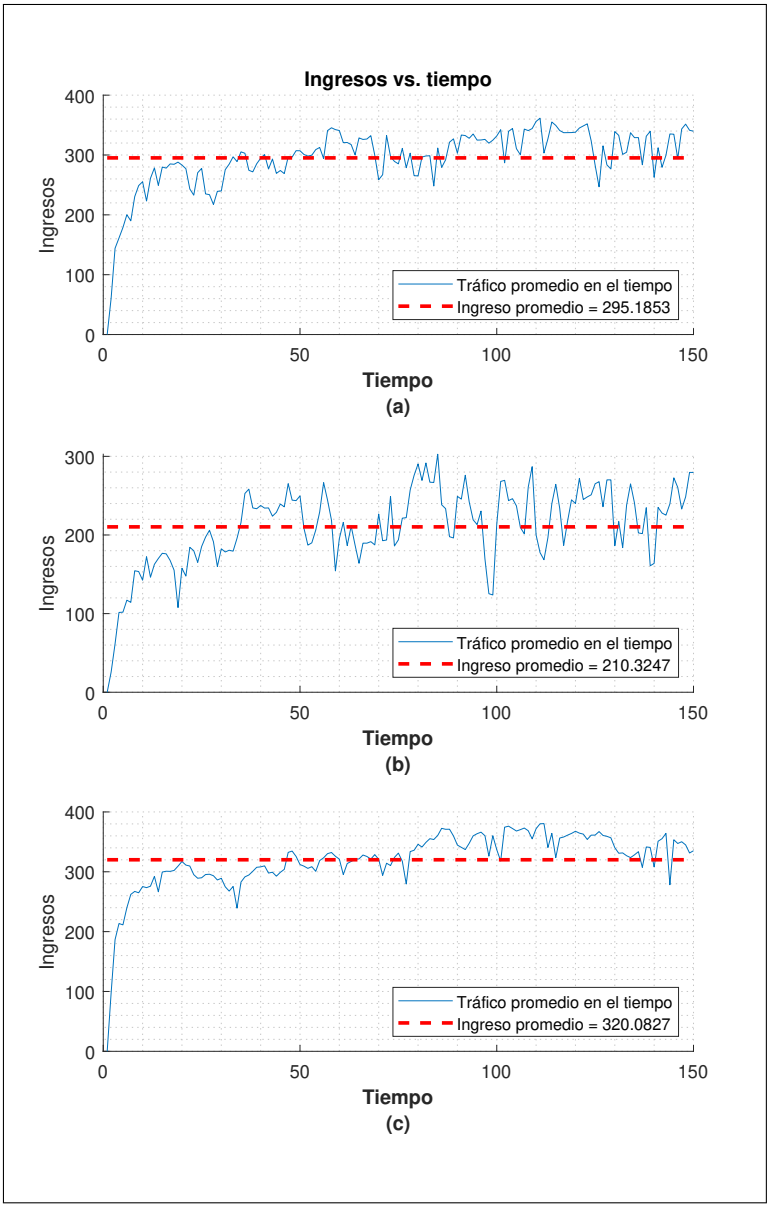


Figura 3.7: Uso de recursos (Revenue) en el tiempo para condiciones de tráfico moderado (a), poco tráfico (b) y tráfico alto (c).

La figura 3.8 presenta la tasa de aceptación de las solicitudes para el servicio URLLC. Al ser este un tipo de servicio que demanda bastantes recursos de la topología física (entre 30 % y 40 % para uso de CPU y entre 1 % y 25 % para uso de enlace por solicitud), se tienen tasas de aceptación bajas, estas varían de acuerdo al tipo de flujo de tráfico entre 46.45 % y 76.64 % siendo este tipo de servicio el que más solicitudes rechaza de todos los analizados. Por ejemplo, en la figura 3.8c, bajo un flujo de tráfico alto, se ve que solo el 40 % de todas las solicitudes realizadas son aceptadas. Es necesario indicar que este tipo de servicio coloca

a la topología física bajo un nivel de estrés alto debido con respecto al uso de recursos llevándola rechazar solicitudes nuevas en muchos de los casos.

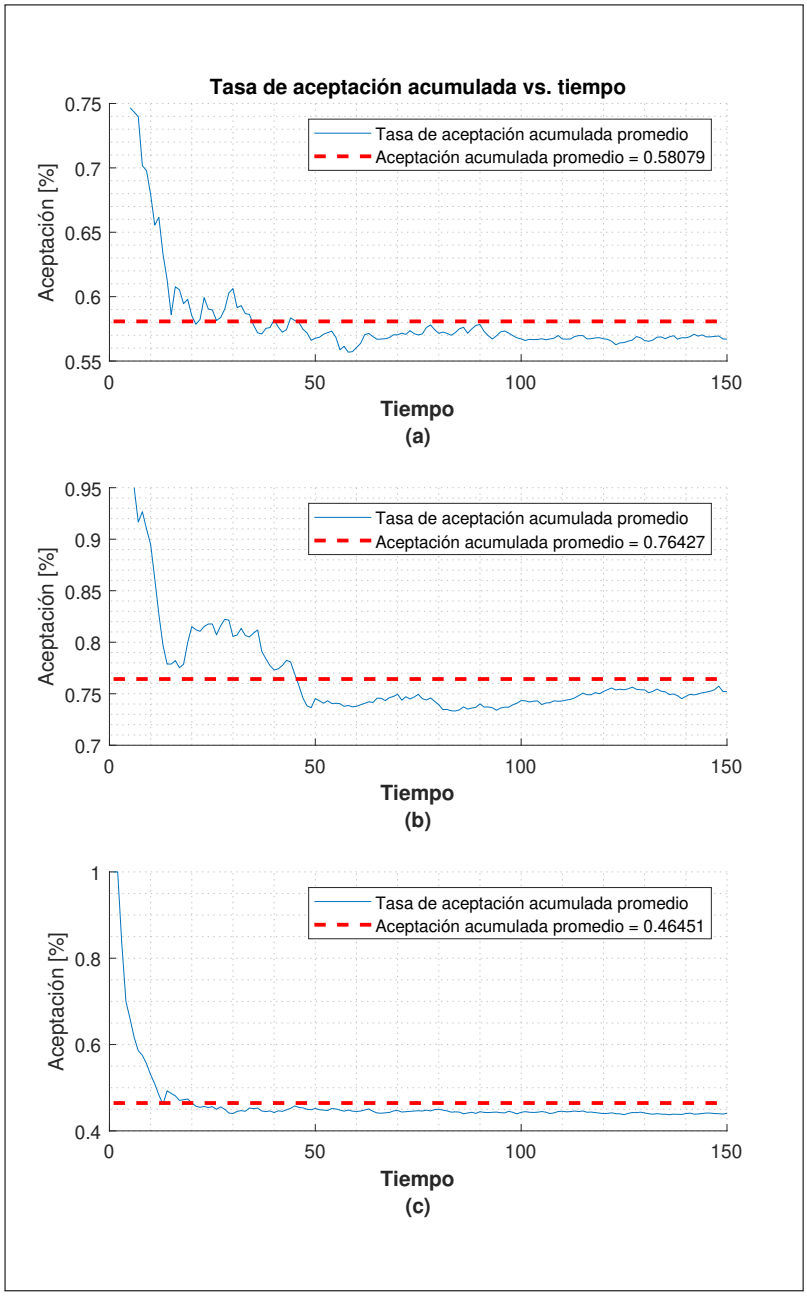


Figura 3.8: Tasa de aceptación acumulada en el tiempo para condiciones de tráfico moderado (a), poco tráfico (b) y tráfico alto (c).

En la figura 3.9 se muestra el número de solicitudes para el servicio URLLC manejadas a la vez por la topología física. El valor de esta métrica, para las condiciones de tráfico moderado y tráfico alto es similar (valor promedio igual a 2 solicitudes por time slot). Esto da cuenta que la topología física es capaz de manejar muy pocas de estas solicitudes de manera simultánea.

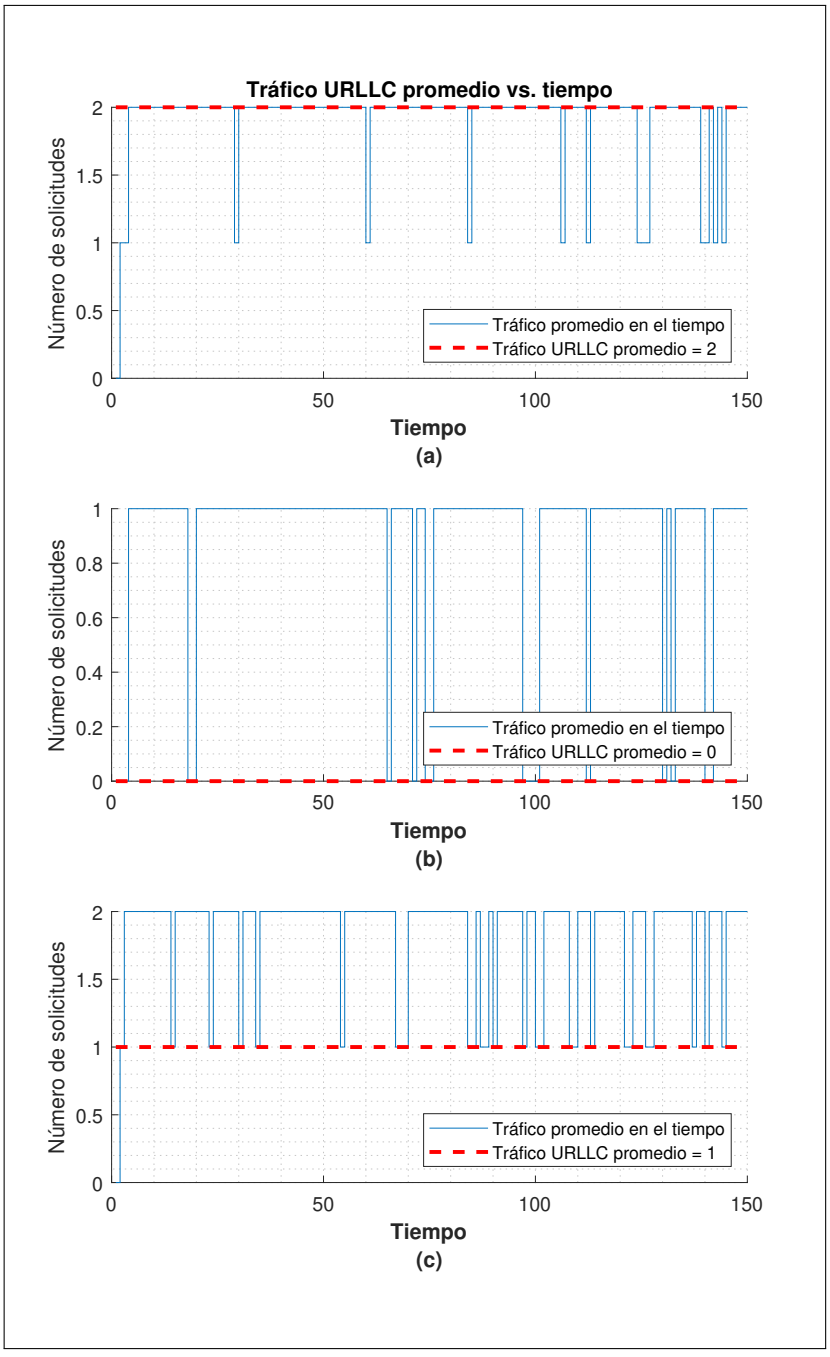


Figura 3.9: Número de solicitudes en el tiempo para condiciones de tráfico moderado (a), poco tráfico (b) y tráfico alto (c).

3.1.4. ANÁLISIS PARA TRÁFICO MIXTO

Con el fin de observar el desempeño del proceso de virtualización para un ambiente real se realizan solicitudes de los tres tipos de servicios (eMMC, mMTC, URLLC). En la figura 3.10 se presentan los ingresos realizados en la topología física ante la solicitud de todos los tipos de tráfico. Los ingresos para cada uno de los casos señalan que la topología física

esta siendo usada en su totalidad o cerca de su totalidad alcanzando un valor pico de 420 ingresos por segundo en el caso mostrado en la figura 3.10c.

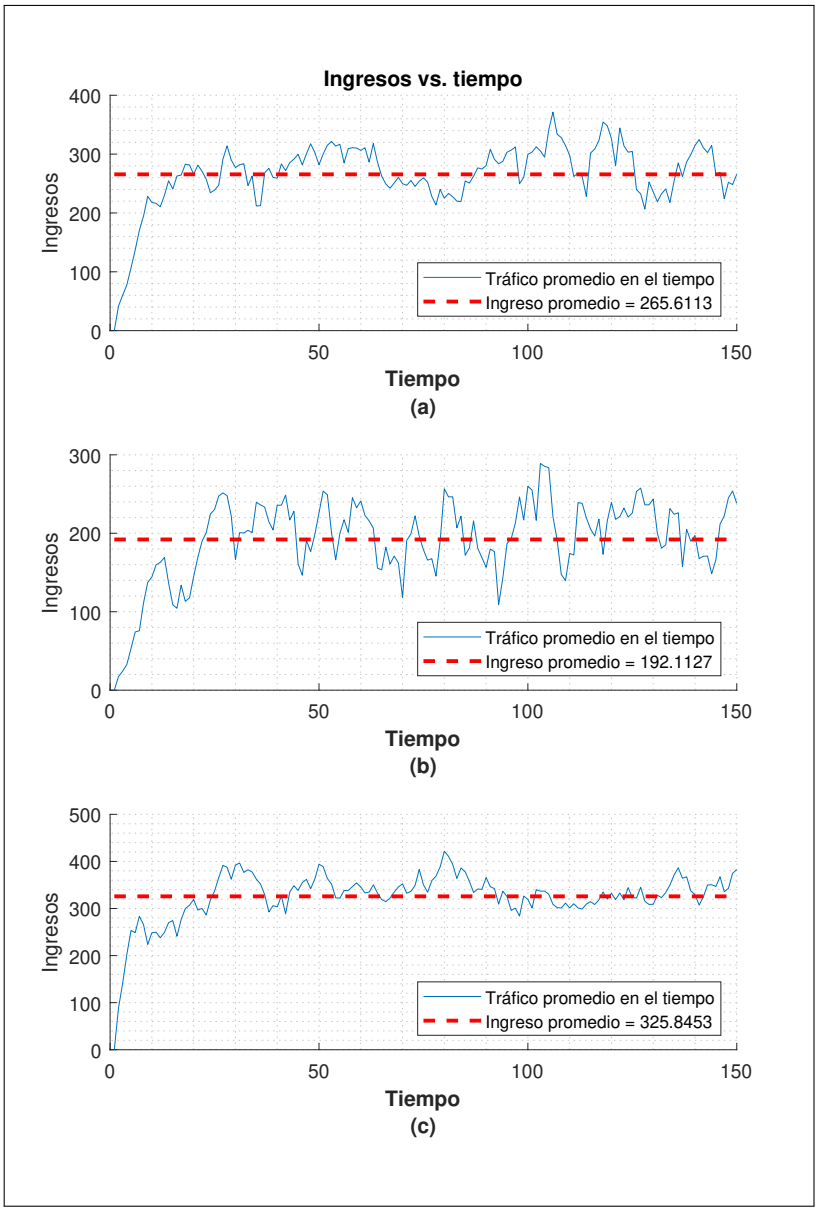


Figura 3.10: Uso de recursos (Revenue) en el tiempo para condiciones de tráfico moderado (a), poco tráfico (b) y tráfico alto (c).

La figura 3.11 muestra la tasa de aceptación de las solicitudes realizadas para todos los tipos de servicio. De acuerdo al tipo de flujo de tráfico se puede visualizar que el porcentaje de aceptación varía, teniendo como el flujo de tráfico con mejor porcentaje de aceptación al tráfico bajo, con una tasa de aceptación acumulada promedio de 96.69 % ; y como peor porcentaje se tiene al flujo de tráfico alto, con una tasa de aceptación acumulada promedio de 83.28 %. Esto se debe a la presencia y creación de solicitudes de varios tipos de servicio, por lo que aunque la topología física tenga varios de sus recursos saturados, podría aceptar

una solicitud de un tipo de servicio que no requiera muchos recursos, como una solicitud de tráfico mMTC. Dicho de otra forma, la variedad de servicios solicitados permite a la topología física distribuir mejor sus recursos.

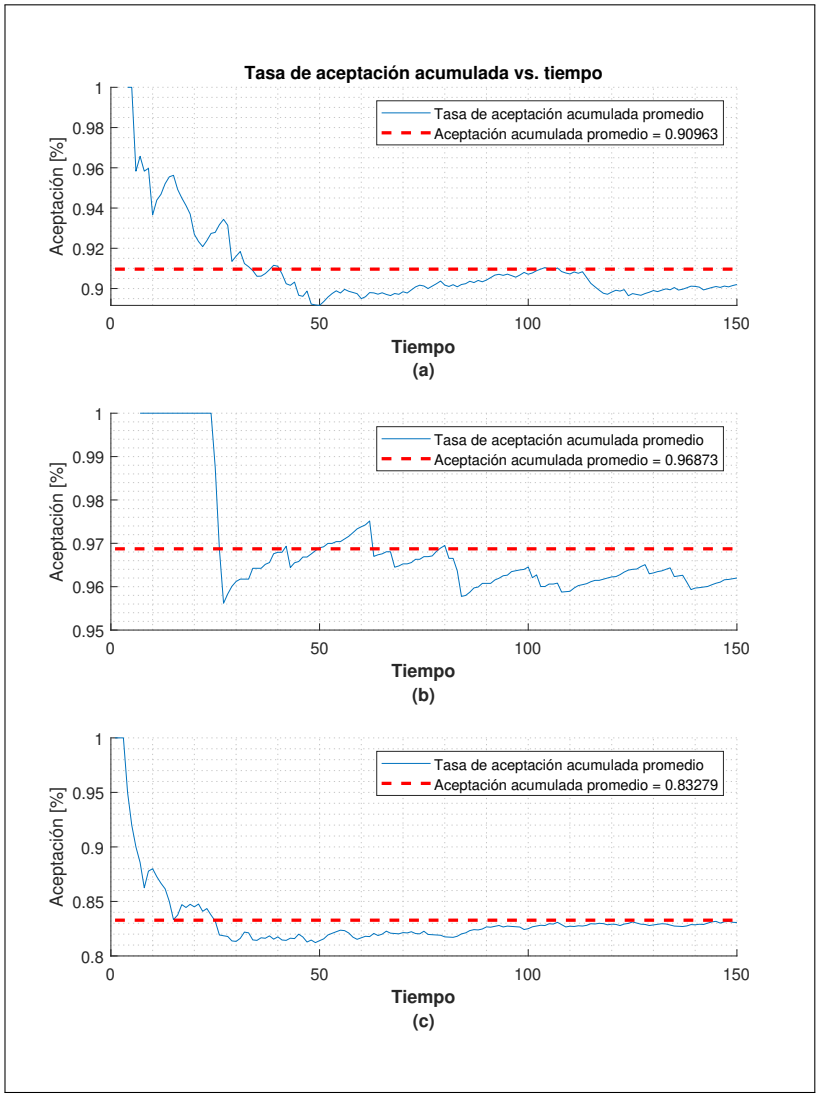


Figura 3.11: Tasa de aceptación acumulada en el tiempo para condiciones de tráfico moderado (a), poco tráfico (b) y tráfico alto (c).

Dentro del análisis del servicio EMMB manejado por la topología física, se puede visualizar en la figura 3.12 que se manejan pocas solicitudes de este tipo de tráfico a la vez, se infiere que esta situación se da debido a que este tipo de tráfico demanda recursos moderados (10 % a 20 % de recursos para cada CPU solicitado y 30 % y 40 % para cada enlace solicitado) de la topología física como se ha venido viendo en los casos anteriores. El número máximo de solicitudes manejadas a la vez es dos para los casos de las gráficas figura 3.12a y 3.12b.

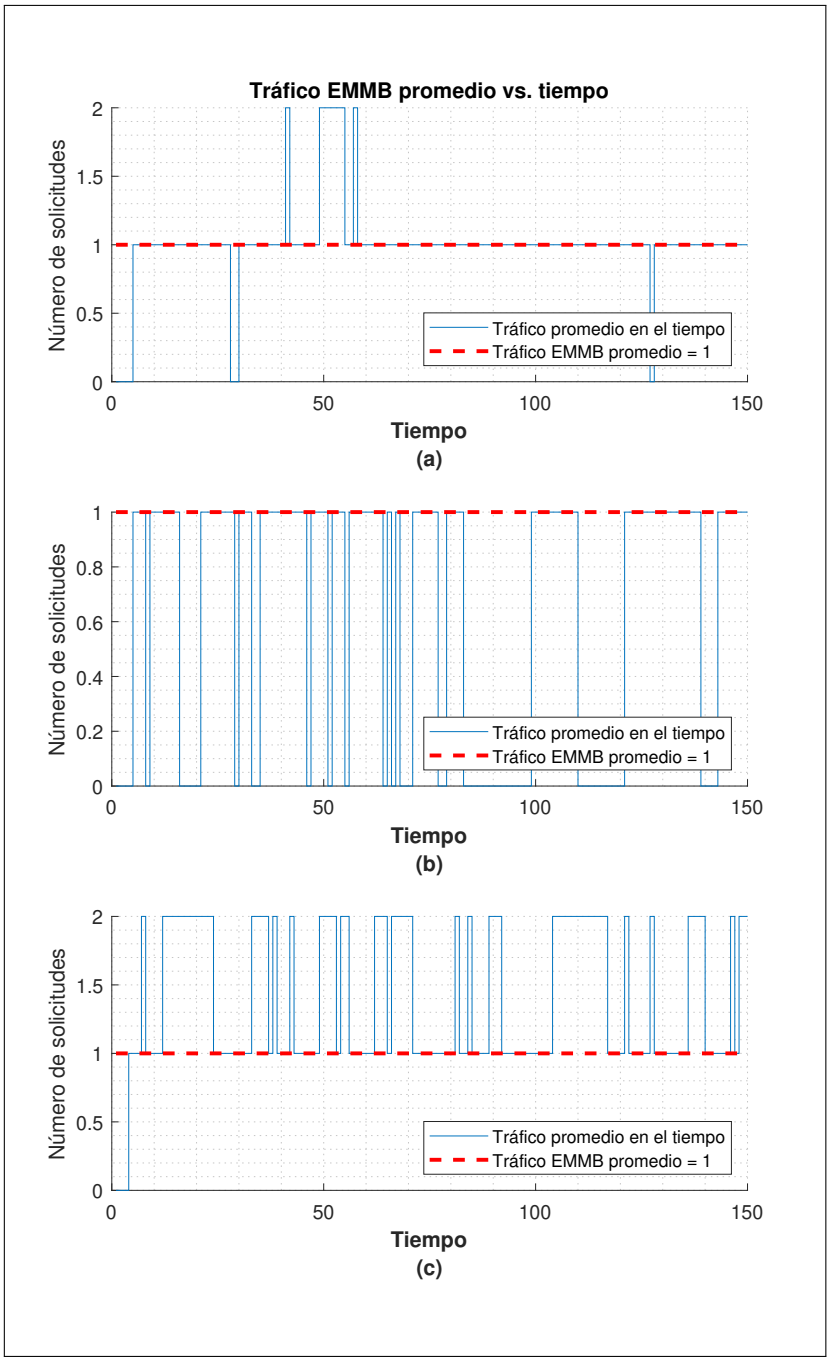


Figura 3.12: Número de solicitudes en el tiempo para tráfico EMMB en condiciones de tráfico moderado (a), poco tráfico (b) y tráfico alto (c).

Para el análisis del servicio mMTC manejado por la topología física, se puede visualizar en la figura 3.13 que el número promedio de solicitudes manejadas a la vez es de 2 en la figura 3.13c. Se infiere que esta situación se presenta debido a que este es el tipo de servicio que menos recursos demanda (entre 1 % a 5 % por cada CPU solicitado y 1 % y 2 % para cada enlace solicitado), pero existen otras solicitudes que requieren muchos más recursos de la topología física siendo procesados (tabla 2.1). Se debe tener en cuenta que bajo situaciones

de tráfico alto 3.13a y bajo 3.13b, el número promedio de solicitudes realizadas a la vez es de 1, situación que afecta al cumplimiento de los requerimientos (densidad de conexión) que debe cumplir el servicio mMTC [5].

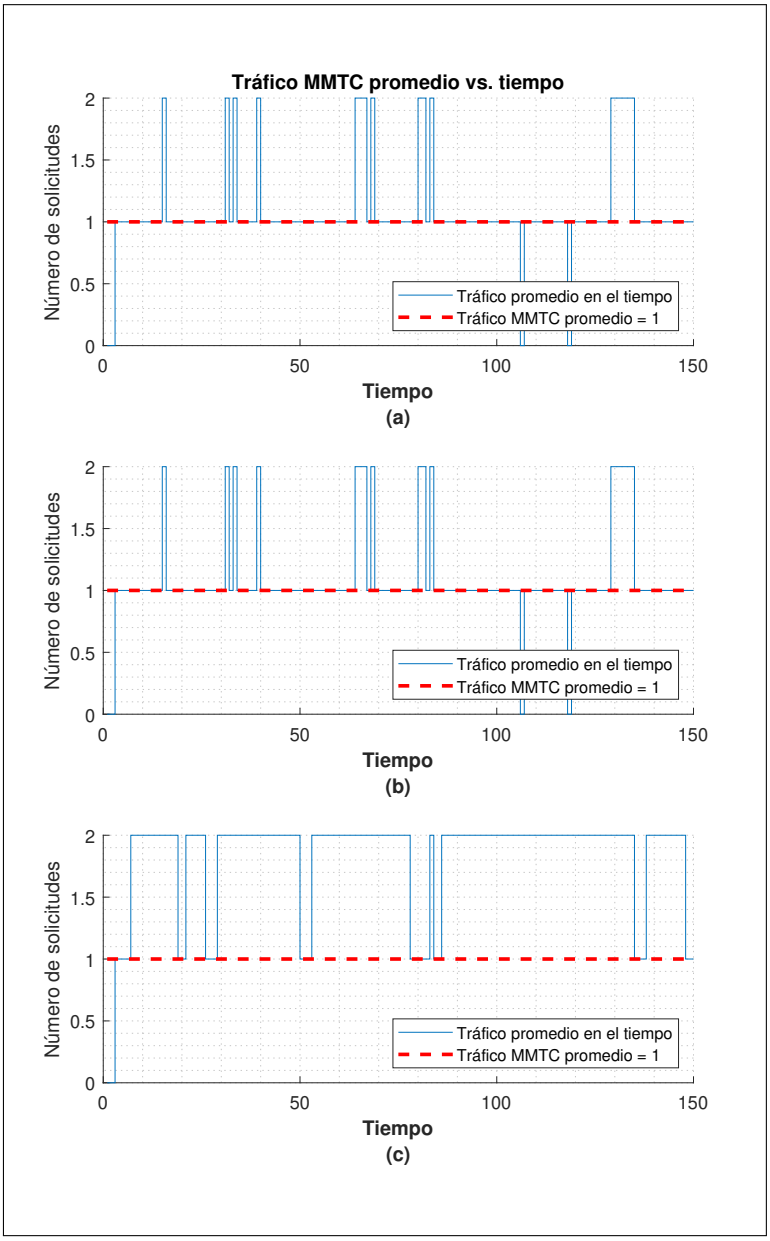


Figura 3.13: Número de solicitudes en el tiempo para tráfico MMTC en condiciones de tráfico moderado (a), poco tráfico (b) y tráfico alto (c).

Los resultados para el servicio URLLC se pueden observar en la figura 3.14, donde se evidencia que este es el tipo de servicio que se encuentra presente menos tiempo virtualizado a causa de las altas demandas que este realiza sobre la topología física, haciendole díficil a la topología física aceptar todas las solicitudes de este tipo de servicio mientras maneja el resto que ya se encuentran virtualizadas.

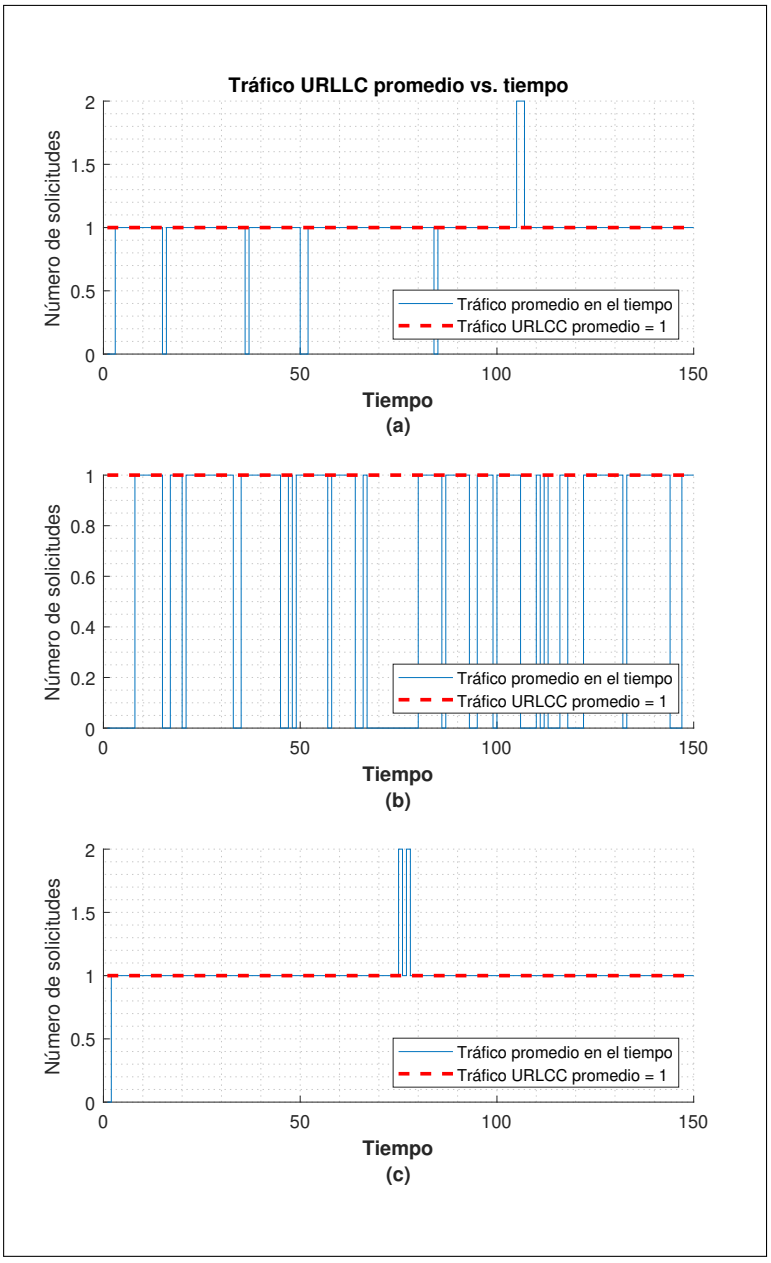


Figura 3.14: Número de solicitudes en el tiempo para tráfico URLLC en condiciones de tráfico moderado (a), poco tráfico (b) y tráfico alto (c)

3.2. Conclusiones

- Producto del análisis de los conceptos, características y funcionamiento de la tecnología de Virtual Network Embedding y de los sistemas de comunicaciones habilitados con UAVs, se logró conseguir una mejor apreciación y entendimiento de la tecnología Network Slicing en 5G. Una vez se estudió la teoría, se logró implementar un sistema considerando Network Slicing sobre una FANET compuesta por UAVs, el cual permite analizar los efectos de la virtualización de los diferentes tipos de servicio establecidos por la ITU (eMMB, URRLC y mMTC) sobre dispositivos que pueden ser transportados por UAVs.

- Matlab permitió la implementación de Network Slicing en un ambiente FANET, donde se pudo generar con éxito el proceso de creación y virtualización de solicitudes sobre una topología física, de acuerdo a los parámetros establecidos por los diferentes tipos de servicio definidos por la ITU para 5G. De igual modo, Matlab permitió desarrollar una interfaz gráfica con la capacidad de mostrar detalles sobre las solicitudes realizadas tales como los recursos de CPU y enlace requeridos de la topología física, las posiciones dentro de la topología física sobre las cuales se van a mapear los recursos de la solictud, el tiempo de simulación y demás elementos descritos en el capítulo dos de este trabajo.

- La implementación de Virtual Network Embedding facilitó la incorporación de la tecnología Network Slicing en un ambiente habilitado por UAVs, suministrando la base teórica suficiente para llevar a Network Slicing a la teoría de grafos y a partir de esta sustentar los paradigmas establecidos por Network Slicing en el capítulo uno. Asimismo, Virtual Network Embedding facilitó la abstracción de las topologías físicas y de las topologías solicitadas con el fin de llevar los conceptos de Network Slicing a un simulador.

- Las FANET son capaces de cumplir todos los requerimientos que el tipo de servicio mMTC demanda, ya que este es el tipo de servicio que menos recursos solicita al hardware de los UAVs. Esto se pudo evidenciar en las tasas de aceptación del 100 % que se tiene para este tipo de servicio, así como los bajos ingresos que mMTC requiere de la topología física.

- Los tipos de servicio eMMB y URLLC demandan muchos recursos (ingresos promedio que bordean los 300 ingresos por time slot) por parte de la topología física, haciendo

que esta sea incapaz de brindar un alto grado de disponibilidad a nuevas solicitudes ya sean de estos tipos de servicio u otros.

- El tipo de tráfico URLLC genera tasas de aceptación bajas debido a la gran cantidad de recursos (CPU y enlace) que este demanda. Por lo que afecta enormemente al desempeño de una red compuesta por UAVs en virtud de que estos poseen hardware limitado (tanto de energía como de capacidad de procesamiento) para el manejo de solicitudes.

- Esta simulación puede ser utlizada como punto de partida para implementar mecanismos de encolamiento para las solicitudes realizadas de manera que estas puedan tener la oportunidad de ser virtualizadas por la topología física y así, se pueda mejorar las tasas de aceptación de solicitudes.

- Finalmente, el presente trabajo busca estimular el estudio de tecnologías eficientes enfocadas al campo de las tecnologías celulares, permitiendo introducir el funcionamiento de Network Slicing y Virtual Network Embedding dentro de una simulación, que, a más de permitir visualizar el proceso de virtualización de solicitudes de acuerdo a los servicios establecidos por la ITU, permite al usuario visualizar las métricas básicas del desempeño de virtualización de recursos de acuerdo a Virtual Network Embedding.

3.3. Recomendaciones

- Debido a que los UAVs utilizan hardware limitado en potencia, es recomendable fijar el número de solicitudes del tipo URLLC que estos pueden manejar a la vez a un valor pequeño (entre uno y dos), de manera que estos no agoten su batería con rapidez o se queden con su procesamiento a la mayor capacidad y así evitar posibles incidentes, como la pérdida de un nodo de la topología debido a que este se quedó sin batería o porque este no se desempeñe normalmente por altas temperaturas provocadas por el alto nivel de procesamiento demandado.

- Para permitir a la red brindar un nivel de disponibilidad alto, se recomienda tener un conjunto de UAVs en espera lo suficientemente grande, de manera que pueda ser usados tan pronto los UAVs desplegados originalmente empiecen a presentar problemas de desempeño debido a escasez de energía o daños durante el vuelo.

- Con el fin de reforzar los requerimientos del tipo de tráfico mMTC, en el caso de que se espere tener una gran cantidad de solicitudes de este tráfico, se recomienda desplegar

un gran número de UAVs. De esta manera se espera mejorar la cobertura de la RAN y a la capacidad de manejar varias solicitudes de este tipo a la vez.

- En virtud de la necesidad de optimizar los recursos de la topología física, se recomienda incorporar algoritmos de mapeo de recursos que consideren el problema del nodo oculto.
- Se recomienda dimensionar los rangos de ocupación de recursos (CPU y enlace) de manera que se tenga la infraestructura disponible y a un costo económico favorable para cada proveedor de servicios. Esto debido a que de acuerdo al rango de ocupación definido por el proveedor de servicios, conforme al tipo de servicio que se solicite, puede ocupar más o menos recursos tanto de CPU como de enlace.

4. REFERENCIAS BIBLIOGRÁFICAS

[1] S. Chavhan, P. Ramesh, R. R. S. Chhabra, D. Gupta, A. Khanna, and J. J. P. C. Rodrigues, "Visualization and performance analysis on 5G network slicing for drones," in *Proceedings of the 2nd ACM MobiCom Workshop on Drone Assisted Wireless Communications for 5G and Beyond*, ser. DroneCom '20. New York, NY, USA: Association for Computing Machinery, Sep. 2020, pp. 13–19. [Online]. Available: https://doi.org/10.1145/3414045.3416208

[2] X. Sun, D. W. K. Ng, Z. Ding, Y. Xu, and Z. Zhong, "Physical Layer Security in UAV Systems: Challenges and Opportunities," *IEEE Wireless Communications*, vol. 26, pp. 40–47, Oct. 2019.

[3] J. Loo, J. Lloret Mauri, and J. Hamilton Ortiz, *Mobile Ad Hoc Networks: Current Status and Future Trends*, G. Williams, Ed. Taylor & Francis, 2011, accepted: 2020-07-28T18:51:22Z. [Online]. Available: https://library.oapen.org/handle/20.500.12657/41721

[4] *5G and Beyond*. [Online]. Available: https://link.springer.com/book/10.1007/978-3-030-58197-8

[5] K. Abbas, T. Ahmed Khan, A. Muhammad, and W.-C. Song, "Network Slice Lifecycle Management for 5G Mobile Networks: An Intent-Based Networking Approach," *IEEE Access*, vol. PP, pp. 1–1, May 2021.

[6] A. Fischer, J. Botero, M. Duelli, D. Schlosser, X. Hesselbach, and H. Meer, "ALEVIN - A Framework to Develop, Compare, and Analyze Virtual Network Embedding Algorithms," *ECEASST*, vol. 37, Jan. 2011.

[7] "5G Radio Access Network Architecture: The Dark Side of 5G | Wiley." [Online]. Available: https://www.wiley.com/en-ie/5G+Radio+Access+Network+Architecture%3A+The+Dark+Side+of+5G-p-9781119550884

[8] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network Slicing in 5G: Survey and Challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, May 2017, conference Name: IEEE Communications Magazine.

[9] P. Chavan, P. Patil, G. Kulkarni, R. Sutar, and S. Belsare, “IaaS Cloud Security,” in *2013 International Conference on Machine Intelligence and Research Advancement*, Dec. 2013, pp. 549–553.

[10] D. Irawan, N. R. Syambas, A. A. N. Ananda Kusuma, and E. Mulyana, “Network Slicing Algorithms Case Study:Virtual Network Embedding,” in *2020 14th International Conference on Telecommunication Systems, Services, and Applications (TSSA*, Nov. 2020, pp. 1–5.

[11] R. Gomes, D. Vieira, and M. Franklin de Castro, “Differential Evolution for VNE-5G Scenarios,” in *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Apr. 2021, pp. 1–6, iSSN: 2157-4960.

[12] A. Belbekkouche, M. Hasan, and A. Karmouch, “Resource Discovery and Allocation in Network Virtualization,” *IEEE Communications Surveys & Tutorials*, vol. 99, pp. 1–15, Feb. 2012.

[13] Z. X. Xu, L. Zhuang, M. Y. He, S. J. Yang, Y. Song, J. I. Guo, and W. C. Li, “An Edge-Based Approach for Virtual Network Embedding Based on the Graph Edit Distance,” In Review, preprint, Nov. 2021. [Online]. Available: https://www.researchsquare.com/article/rs-1029589/v1

[14] S. S. Epp, *Matemáticas discretas con aplicaciones*, cuarta edición ed.

[15] A. Azari, F. Ghavimi, M. Ozger, R. Jantti, and C. Cavdar, “Machine Learning assisted Handover and Resource Management for Cellular Connected Drones,” in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, May 2020, pp. 1–7, iSSN: 2577-2465.

[16] Y. Zeng, Q. Wu, and R. Zhang, “Accessing From the Sky: A Tutorial on UAV Communications for 5G and Beyond,” *Proceedings of the IEEE*, vol. 107, no. 12, pp. 2327–2375, Dec. 2019, conference Name: Proceedings of the IEEE.

[17] I. Bekmezci, O. K. Sahingoz, and S. Temel, “Flying Ad-Hoc Networks (FANETs): A survey,” *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, May 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870512002193

[18] “Graph with undirected edges - MATLAB - MathWorks América Latina.” [Online]. Available: https://la.mathworks.com/help/matlab/ref/graph.html

[19] J. F. Botero, X. Hesselbach, A. Fischer, and H. de Meer, “Optimal mapping of virtual networks with hidden hops,” *Telecommunication Systems*, vol. 51, no. 4, pp. 273–282, Dec. 2012. [Online]. Available: https://doi.org/10.1007/s11235-011-9437-0

[20] “Determine whether array is empty - MATLAB isempty - MathWorks América Latina.” [Online]. Available: https://la.mathworks.com/help/matlab/ref/isempty.html

[21] G. Romano, “IMT-2020 Requirements and Realization,” Dec. 2019, pp. 1–28.

[22] “Números aleatorios distribuidos de manera uniforme - MATLAB rand - MathWorks América Latina.” [Online]. Available: https://la.mathworks.com/help/matlab/ref/rand.html

[23] “Redondear al decimal o entero más próximo - MATLAB round - MathWorks América Latina.” [Online]. Available: https://la.mathworks.com/help/matlab/ref/round.html

[24] “Upper triangular part of matrix - MATLAB triu - MathWorks América Latina.” [Online]. Available: https://la.mathworks.com/help/matlab/ref/triu.html

[25] “Shortest path between two single nodes - MATLAB shortestpath - MathWorks España.” [Online]. Available: https://es.mathworks.com/help/matlab/ref/graph.shortestpath.html

[26] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, “Virtual Network Embedding: A Survey,” *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 1888–1906, 2013, conference Name: IEEE Communications Surveys Tutorials.

5. ANEXOS

Las funciones y demás elementos utilizados para la realización de este proyecto se encuentran disponibles en el siguiente repositorio:

https://github.com/jossue845/Network_Slicing_sim/