APROXIMACIÓN DE UNA RED MULTICAST DE RESOLUCIÓN CON NETWORK CODING UTILIZANDO EL ALGORITMO DE DINIC

ING. LILIANA ANDREA GONZALEZ HERNANDEZ

UNIVERSIDAD DEL NORTE
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
BARRANQUILLA
2019

APROXIMACIÓN DE UNA RED MULTICAST DE RESOLUCIÓN CON NETWORK CODING UTILIZANDO EL ALGORITMO DE DINIC

ING. LILIANA ANDREA GONZALEZ HERNANDEZ

Monografía para optar al título de
Magister en Ingeniería de Sistemas y Computación

Directores:
José Márquez Díaz

UNIVERSIDAD DEL NORTE
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
BARRANQUILLA
2019

Aprobado por el Profesorado de la División de Ingenierías en cumplimiento de los requisitos exigidos para otorgar el título de MAGÍSTER EN INGENIERÍA DE SISTEMAS Y COMPUTACIÓN.

Ing. LILIANA ANDREA GONZALEZ HDEZ

Ing. JOSÉ MARQUEZ DIAZ Ph.D
Director de Proyecto.

Ing. Miguel Jimeno Ph.D.
Coordinador de la Maestría en Ingeniería de Sistemas y Computación.

Barranquilla, 15 de Agosto de 2019

AGRADECIMIENTOS

A Jesús fuente de todas las cosas, sin El nada de esto fuera posible.

A mi papá José González por inspirarme y demostrarme que nunca es demasiado tarde para aprender, descubrir y crear.

A mi mamá Aidé Hernández por enseñarme con el ejemplo que nada es imposible para alguien que entrega su 100%. Por ella entendí y aprendí que organizando mi tiempo puedo lograr hacer todo lo que me proponga.

A mi amigo y tutor José Márquez, por tener la paciencia, el compromiso, la entrega, por creer más en mí incluso en momentos donde yo dudaba, por motivarme a no rendirme y empujarme a cerrar esta etapa de mi vida.

A mis buenos amigos y líderes Karen y Jonathan porque con sus vidas y sus oraciones me han dado fuerzas para avanzar y concluir ciclos importantes y transcendentales.

A mi segunda mamá Luz Marina, por entenderme, apoyarme y creer que este tiempo sería el indicado para subir un peldaño más.

A mi segunda familia Mary P y Rosario por ayudarme, apoyarme y estar ahí para mi durante este proceso, sus oraciones fueron el motor que no me dejaba desfallecer.

A mi buena amiga Laura B, por orar por mí.

A mi familia que nunca dejó de confiar y recordarme que soy capaz de alcanzar todo lo que me proponga.

Gracias a todos los que me han tenido paciencia, a los que dejé de ver y saludar. A aquellos que me esperaron y están muy pendientes de mi avance en este proyecto.

Gracias Espíritu Santo por darme tu sabiduría y la fuerza para avanzar.

Gracias Dios por ser el mejor padre y mostrarme tu misericordia, tu amor me sostuvo en mis momentos más difíciles.

Y hoy me celebro, porque, aunque pensé rendirme, acá estoy una vez convenciéndome que no hay nada imposible para aquel que cree.

TABLA DE CONTENIDOS

TABLA DE FIGURAS

TABLA DE ALGORITMOS

TABLA DE TABLAS

ABREVIATURAS

NC	Network Coding
LNC	Linear Network Coding
VLNC	Vector Linear Network Coding
NLNC	Non Linear Network Coding
TNC	Triangular Network Coding
FF	Ford-Fulkerson
EK	Edmonds-Karp
BFS	Breadth-First Search

RESUMEN

No es un secreto para nadie, que cada vez es más usual e indispensable el uso de Internet, además, la tendencia es que cualquier dispositivo electrónico se pueda conectar a la red para transmitir y/o recibir datos. Esto nos hace pensar que cada vez es más necesario el uso de técnicas o métodos que mejoren y potencialicen el rendimiento, eficiencia y escalabilidad de los canales por donde se están enviando todo este cúmulo de paquetes de información.

Uno de los problemas frecuentes consiste en la pérdida de paquetes; esto, debido a errores en los enlaces que conectan a los distintos nodos. El mecanismo tradicional para atacar este inconveniente, se realiza a través de técnicas de reenvío de paquetes perdidos desde el punto de inicio; es decir, desde el terminal o fuente que genera la información hasta su destino final. Esto traduce en que el canal estará transmitiendo no solo los datos nuevos que recibe, sino que tendrá datos que volverán a repetirse ya que no llegaron a su destino.

Una solución inminente a esta problemática, consiste en la implementación de la técnica matemática de Network Coding (NC). Usando esta técnica se otorga capacidad de procesamiento de datos a los nodos intermedios de una red, con el fin de permitirle realizar operaciones lógicas sobre los paquetes. El principal objetivo de la técnica es la recuperación de datos perdidos sin tener que recurrir a la retransmisión.

En la teoría y en los ensayos realizados con esta técnica, en redes con condiciones ideales, el concepto funciona a la perfección; pero yendo un poca más a la realidad, en modelos y redes multicast o con más de un sumidero no es tan sencilla la implementación como se plantea. En este punto, es necesario garantizar que las redes cumplan con ciertas características para ser solucionables por sistemas lineales de ecuaciones, esto debido a que Network Coding es una alternativa que nació a partir de una solución matemática.

Este trabajo de investigación busca como objetivo principal generar un grafo aproximado de red de transmisión multicast, para la utilización de la técnica de Network Coding a través del uso iterativo del algoritmo de **Dinic** sobre una red general de comunicaciones, representada por un grafo. Esta red multicast uni-sesión contará con un nodo fuente y más de un nodo sumidero conocidos.

El camino a seguir consiste, en primera medida, encontrar los flujos máximos independientes para cada uno de los nodos sumideros, y haciendo las reducciones necesarias para que cumpla las mínimas características de una red multicast que permita la entrega simultánea de paquetes hasta el flujo máximo aplicando la técnica de **Network**

Coding. Además, se obtendrá una matriz de flujos generales que represente la red multicast de única sesión para varios ejemplos de redes, la cual se pondrá a prueba con la técnica de Network Coding, y se concluirá para cuales redes de comunicaciones, el algoritmo de **Dinic**, provee por lo menos una solución de red multicast uni-sesión.

ABSTRACT

It is not a secret for anyone, that the use of the Internet is becoming more usual and indispensable, in addition, the tendency is that any electronic device can connect to the network to transmit or receive data. This makes us think that it is increasingly necessary to use techniques or methods that improve and increase the performance, efficiency and scalability of the channels through which all this cluster of information packages are being sent.

One of the frequent problems is the loss of packages; this, due to errors in the links that connect to the different nodes. The traditional mechanism to attack this problem is done through techniques for forwarding lost packets from the starting point; that is, from the terminal or source that generates the information to its final destination. This means that the channel will be transmitting not only the new data it receives, but will have data that will be repeated because they did not reach their destination.

An imminent solution to this problem is the implementation of the mathematical technique of Network Coding (NC). Using this technique, data processing capacity is granted to the intermediate nodes of a network, in order to allow it to perform logical operations on the packets. The main objective of the technique is the recovery of lost data without resorting to retransmission.

In the theory and in the tests carried out with this technique, in networks with ideal conditions, the concept works perfectly; but going a little more to reality, in multicast models and networks or with more than one sink, the implementation is not as simple as it is proposed. At this point, it is necessary to ensure that networks meet certain characteristics to be solvable by linear systems of equations; this is because Network Coding is an alternative that was born from a mathematical solution.

This research work aims to generate an approximate graph of multicast transmission network, for the use of the Network Coding technique through the iterative use of the Dinic algorithm over a general communications network, represented by a graph. This unicast multicast network will have a source node and more than one known sink node.

The way forward is, firstly, to find the maximum independent flows for each of the sink nodes, and making the necessary reductions so that it meets the minimum characteristics of a multicast network that allows simultaneous delivery of packets up to the maximum flow by applying Network Coding technique. In addition, a matrix of general flows will be obtained representing the single session multicast network for several network examples, which will be tested with the Network Coding technique, and will be concluded for which communications networks, the Dinic algorithm, It provides at least one multicast network connection solution.

1. INTRODUCCIÓN

Network Coding es una técnica matemática que nació como campo de investigación hacia finales de los 90 e inicio la década los 2000. Sin embargo, este concepto se venía utilizando desde antes. En 1978 un grupo de personas que intentaban comunicarse entre sí a través de un satélite, combinaron sus dos flujos de datos de la siguiente manera: hicieron una suma entre ellos y luego el resultado fue dividido entre 2, este nuevo flujo fue enviado al satélite y al recibirlo pudieron decodificar el otro flujo utilizando su propia información. [1]

Podríamos, con este ejemplo, decir que Network Coding es una técnica de red que le otorga la capacidad de procesamiento de datos a los nodos intermedios, con el fin de permitirle realizar operaciones lógicas sobre los paquetes. El objetivo principal de esta técnica es la recuperación de datos perdidos sin tener que recurrir a la retransmisión, además de aumentar el rendimiento de la red, reducir los retrasos y hacer que la red sea más robusta.

Por lo anterior, cuando se desea implementar Network Coding, es indispensable contar con nodos codificadores/decodificadores que permitan la construcción de combinaciones lineales de los paquetes que están siendo transportados. Estas combinaciones terminan siendo un solo paquete basado en el principio de la suma módulo 2 o XOR de paquetes bit a bit, los cuales generan nuevos paquetes distintos de los originales y con el mismo tamaño de éstos.

Los nuevos paquetes codificados son reenviados por cada uno de los nodos de la red hasta llegar al destino, y es ahí donde los nodos sumideros actúan como receptores de paquetes y también como nodos decodificadores. En este punto, los sumideros son capaces de volver a generar la información original y entregarla a los dispositivos destinatarios de la red sin ningún problema.

Imaginemos una ciudad grande y avanzada económicamente, dicha ciudad está repleta de los mejores y más lujosos carros, con los motores más potentes, pero con un colosal problema en sus calles y vías principales (las cuales fueron construidas hace más de 50 años), estas vías no cuentan con señalización apropiada, no posee un sistema de semáforos eficiente y los agentes de tránsito hacen lo que quieren. Aunque alguien esté en un Ferrari, llegar a su destino podría ser toda una odisea ya que mientras encuentra la ruta perfecta y logra vencer las inclemencias del flujo vehicular, se percataría que toma casi el mismo tiempo que alguien que hace la misma ruta en bicicleta.

Este es el mismo contexto de las redes que utilizan un protocolo estándar de enrutamiento, cuyo cuello de botella se centra en la capacidad de optimizar y poder darle

un uso adecuado y eficiente al ancho de banda, al momento de transmitir paquetes entre nodos.

Analizando la realidad de este momento, estamos viviendo en un mundo digital que crece abismalmente, donde todo lo que imaginamos o no imaginamos está produciendo datos, enviando y/o recibiendo información; por esta razón, es necesario comenzar a estudiar estrategias para mejorar el comportamiento de la red y una de esas iniciativas consiste en aplicar el concepto de Network Coding. Podríamos pensar en algo tan natural en este siglo como una videoconferencia o una llamada grupal.

El ejemplo anterior, nos ubica en un entorno de red multicast. Cuando nos referimos a multicast hablamos de la entrega de datos de forma simultánea a un grupo de nodos receptores como destinatarios, desde un nodo emisor como origen.

Tradicionalmente UDP (User Datagram Protocol) permite aprovechar las comunicaciones multicast, que contrario a unicast, proporcionan soporte de envíos simultáneos. Pero las desventajas más destacables, si se usa UDP en implementaciones multicast, son la pérdida de paquetes y la congestión. [2]

Con este trabajo se pretende desarrollar un método para la obtención de un grafo multicast uni-sesión a partir de una red general de comunicaciones. Este grafo multicast debe permitir la entrega del máximo flujo común de paquetes simultáneamente en los nodos sumideros, mediante la implementación de Network Coding en los nodos codificadores intermedios y/o en los decodificadores en los nodos sumideros. La estrategia consiste en aproximar el grafo de red de comunicaciones a un grafo unicast uni-sesión desde el nodo fuente hasta cada nodo sumidero, de tal manera que cumpla las características que debe tener una red para la correcta y precisa aplicación de la técnica matemática de Network Coding, con el fin de entregar el máximo flujo común de paquetes a los nodos sumideros. Con esto se esperaría poder obtener un mejor rendimiento al entregar simultáneamente los paquetes enviados desde una fuente sin aumentar el ancho de banda, así evitando los retardos en las interfaces de salida y disminuyendo la pérdida de los mismos.

La aproximación del grafo de red que cumple los requisitos para aplicar Network Coding, se logra a través de la implementación del algoritmo de Dinic sobre el grafo de red de comunicaciones original. Con este algoritmo se halla el flujo máximo general para cada nodo sumidero. Teniendo en cuenta lo expuesto en [3], se homogenizan los flujos llevándolos todos al mínimo de los flujos máximos del conjunto de sumideros. Con los grafos homogenizados en el número de flujos, se construye un grafo de red multicast uni-sesión que permite la transmisión y entrega simultánea del número máximo común de paquetes al conjunto de nodos sumideros.

Es importante anotar que esta aproximación puede que no siempre esté en un estado satisfactorio para la aplicación de la técnica matemática de Network Coding, por lo que se presentarán los resultados del método sobre una cantidad considerable de redes distintas para determinar su grado de eficacia.

2. GENERALIDADES

2.1. **JUSTIFICACIÓN**

"Best effort" (BE)" lo más posible, lo antes posible" es una de las premisas de las redes IP. Los paquetes con este tipo de servicio tienen la misma expectativa de tratamiento mientras transitan por la red.

Cuando se envían paquetes, el enrutador sólo mira su cabecera, busca en la tabla de ruteo y define el próximo salto. Si llegase a ocurrir congestión, los paquetes enviados se retardan o en el peor de los casos se descartan. Este comportamiento es aceptable para aplicaciones como mail, ftp o navegar por una página web, pero no pasa lo mismo para otras aplicaciones que no toleran retardos variables o pérdida de datos, como es el caso de servicios de voz y video en tiempo real.

Se podría pensar que la mejor solución es agregar más ancho de banda, pero esto no es suficiente, ya que el tráfico es típicamente en ráfagas, produciendo congestiones temporales, retardos y pérdidas. Por lo tanto, la clave está en dotar a la red de una mayor "inteligencia", por medio de mecanismos para obtener Calidad de Servicio (QoS). El objetivo de la Calidad de Servicio en una red es cuantificar el tratamiento que un paquete debe esperar a medida que circula por los nodos y enlaces.

Con Network Coding, aplicado a la Calidad de Servicio, se obtiene un aumento considerable del ancho de banda al lograr transportar por un mismo enlace más de un paquete simultáneamente, lo cual no se conseguiría con una técnica tradicional.

Además, el hecho de otorgarle capacidad de procesamiento de datos a los nodos intermedios, hace que la recuperación de datos perdidos sea una de sus fortalezas sin tener que recurrir a la retransmisión de los mismos.

Sin la implementación de Network Coding en una red multicast, un enrutador tendría que enviar por un mismo enlace los paquetes en forma secuencial en el orden en que determine cómo deben emerger, realizando su función básica de almacenamiento y reenvío. Esto conlleva un consumo elevado del ancho de banda, ya que cada paquete es una unidad que consume parte de este limitado recurso, creando un cuello de botella en el enlace saliente. Por ejemplo, para un enrutador con dos enlaces entrantes y un enlace saliente, tendría que entregar los paquetes del mismo tamaño que ingresan en dos tiempos distintos, consumiendo dos unidades de ancho de banda.

Con la implementación de Network Coding, un enrutador combina linealmente los paquetes entrantes con destino hacia los mismos nodos sumideros y, luego envía el

paquete único resultante codificado a través del enlace de cuello de botella. Esta técnica permite una mayor utilización del ancho de banda, ya que solo se envía un solo paquete por el enlace, utilizando una sola unidad de transmisión, a diferencia del caso donde no se utiliza Network Coding (enrutamiento clásico). De igual manera, al aplicar la técnica de Network Coding sobre los paquetes entrantes, se disminuye el retardo por reenvío de los paquetes entrantes y, por ende, la variación del retardo o jitter [5].

Considerando todos los beneficios que ofrece Network Coding, se infiere que es posible dar solución a un escenario más complejo que los expuestos y solucionados con el modelo Butterfly [6], tal como es una red multicast de única sesión con un nodo fuente y un grupo de sumideros.

Entendiendo, que, para lograr este reto, es indispensable en primera medida, la aplicación de un método que pueda transformar la red inicial a una que sea apta para la aplicación de Network Coding; por lo tanto, el camino a seguir consiste en determinar el grafo unicast de flujo máximo desde el nodo fuente hasta cada uno de los nodos sumideros que conforman la red multicast. A partir de esto, se determinan los caminos disyuntos que constituyen el flujo máximo común para cada grafo unicast. La mezcla de los grafos unicast resulta en la aproximación de la red multicast apta para aplicar Network Coding. Todo este proceso y foco de la investigación es a través de la creación de un método cuyo fundamento es el algoritmo Dinic, que calcula el grafo unicast de flujo máximo entre el nodo fuente y un sumidero.

2.2. **OBJETIVOS**

2.2.1. **OBJETIVO GENERAL**

Crear un método de aproximación a una red multicast uni-sesión mediante el algoritmo de cálculo de flujos máximos Dinic a partir de una red de comunicaciones con múltiples enlaces y nodos, donde se especifiquen el nodo fuente y los nodos sumideros con el fin de determinar si es posible aplicar la técnica matemática de Network Coding para la entrega simultánea de paquetes.

2.2.2. **OBJETIVOS ESPECÍFICOS**

- Encontrar los flujos máximos de la red respectivamente para cada uno de los nodos sumideros con ayuda del algoritmo de Dinic.
- Encontrar el flujo máximo común de la red multicast siendo el mínimo de entre todos los flujos máximos encontrados en el conjunto de nodos sumideros.
- Obtener una matriz de flujos para la red multicast que puede o no ser una solución para Network Coding.

- Demostrar con casos de prueba que la propuesta presentada es una solución para hallar grafos multicast uni-sesión que permitan la transmisión simultánea de paquetes utilizando Network Coding.

2.3. DESCRIPCIÓN DEL PROBLEMA DE INVESTIGACIÓN

Dada una red de comunicaciones, con un grafo de representación del flujo G, donde se conoce el nodo fuente s y los nodos sumideros T, el problema consiste en hallar el grafo que representa el enrutamiento multicast uni-sesión G', con el mínimo flujo máximo entre los caminos disyuntos partiendo de s hacia cada nodo t en T. El grafo multicast uni-sesión debe permitir el envío y recepción, en forma simultánea, de paquetes hasta el mínimo del flujo máximo entre el nodo fuente y cada sumidero, este grafo debe ser apto para la utilización de la técnica de Network Coding.

3. MARCO TEÓRICO

3.1. TÉCNICA DE NETWORK CODING

Network Coding se puede resumir así: es una técnica de red que le otorga la capacidad de procesamiento de datos a los nodos intermedios, con el fin de permitirle realizar operaciones lógicas sobre los paquetes. Esta definición contradice a los esquemas de enrutamiento tradicionales almacenamiento-y-reenvío, en los que la función de estos nodos se limita a recibir un mensaje, buscar su siguiente destino en la red y reenviarlo.

Este concepto se puede entender de mejor manera al revisar la Figura 3.1.1 y observar los dos comportamientos. En la parte (a) se muestra el enrutamiento tradicional basado en almacenamiento-y-reenvío; la fuente recibe tres paquetes “A”, “B” y “C”, y estos son retransmitidos por un nodo intermedio R_i tal cual como los recibe. En la solución de Network Coding en (b), el nodo C_i genera una salida “nueva” a partir de los paquetes que le llegan. Para ello, utilizará combinaciones lineales a partir de las cuales se construirá un mensaje “D”.

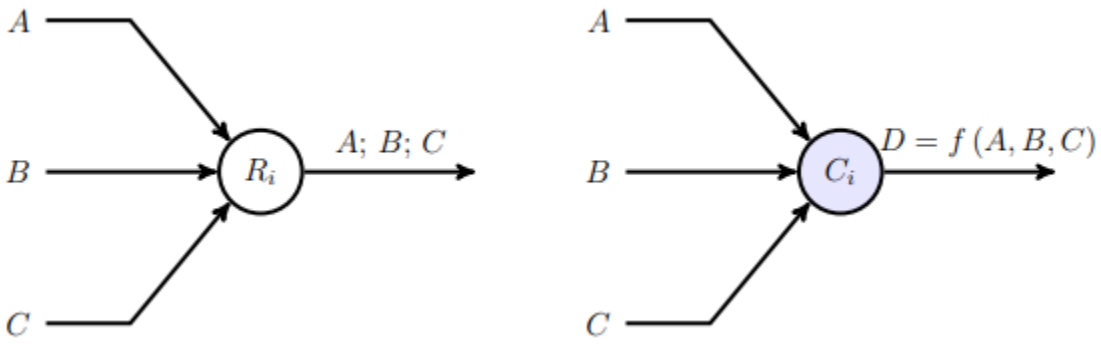


Figura 3.1.1 Esquema de enrutamiento de paquetes de red. (a) Esquema tradicional almacenamiento-y-reenvío (b) Esquema Network Coding

El ejemplo más clásico para ilustrar el comportamiento de la técnica de Network Coding se representa en la Figura 3.1.2. Alice y Bob desean comunicarse, pero como no pueden hacerlo directamente, tienen que recurrir a un nodo intermedio, en este ejemplo se utilizará un enrutador y tendrá la función de reenviar los mensajes a cada destino.

La forma tradicional funciona como se ilustra en la Figura 3.1.2a, primero cada nodo deberá luchar por tener acceso al canal para transmitir sus paquetes: en el ejemplo, ‘a’ y ‘b’. Cuando ya consigan esto, el enrutador almacenará ambos paquetes recibidos, comprobará en su tabla de rutas la dirección del siguiente salto y los reenviará. En total,

se habla de mínimo cuatro, las transmisiones “físicas” necesarias para terminar el proceso.

Por otra parte, haciendo uso de las técnicas de Network Coding en la Figura 3.1.2b, luego que se envíen los paquetes, el enrutador los recibirá, este tendrá la capacidad de combinar la información, generando uno nuevo a partir de la operación XOR (a ⊕ b) de ambos. Cuando esta información llegue a su destino, los nodos finales utilizarán los paquetes que tenían almacenados localmente (‘a’ en el caso de Alice y ‘b’ en el de Bob) para recuperar la información dirigida a ellos. Así, si un nodo dispone del paquete nativo ‘a’, puede realizar la misma operación XOR con el codificado y recuperar ‘b’: a ⊕ (a ⊕ b) = b.

En definitiva, son necesarias tres transmisiones, en lugar de las cuatro del esquema clásico ideal, donde ningún paquete se pierde, consiguiendo un ahorro del 25 % del total de transmisiones necesarias.

Esto a su vez le otorga otras ventajas adicionales, como un incremento en la tasa promedio de éxito en la entrega de un mensaje sobre un canal de comunicación, una reducción de la energía consumida por todos los dispositivos de la red, así como un incremento en la seguridad de la transmisión, ya que la información viaja “cifrada” o encriptada.

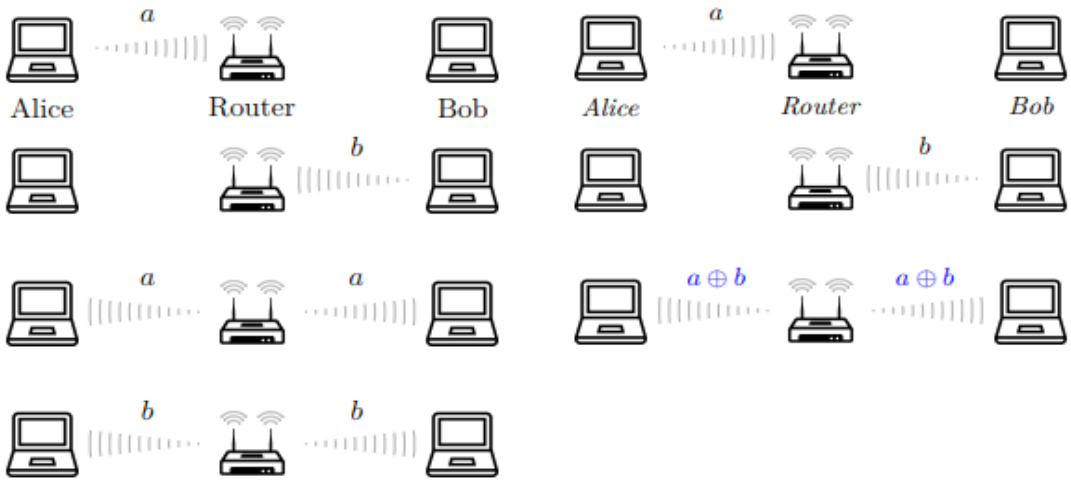


Figura 3.1.2 Esquema de enrutamiento entre Alice y Bobo. (a)Enrutamiento tradicional (b)Network Coding

3.2. **MEJORA EN LAS REDES CON NETWORK CODING**

Existen grandes ventajas asociadas al uso de Network Coding, tales como, mejoras en la seguridad de la transmisión, rendimiento, velocidad y capacidad del canal o ancho de banda, de acuerdo con las investigaciones realizadas por Tracey Ho Desmond y S. Lun. En [5] estas mejoras se resumen en cuatro puntos.

3.2.1. **Mejoras en Throughput o tasa de transferencia efectiva**

Cuando se utiliza Network Coding los envíos de paquetes son más eficientes, es decir, se logra hacer más entregas de información en un menor número de transmisiones.

Esta ventaja se entiende de manera más clara, al revisar el ejemplo más famoso del uso de esta técnica, que fue dado por Ahlswede en [6], y quien consideró el problema multicast en redes cableadas. Este ejemplo es comúnmente conocido como red mariposa. En la Figura 3.2.1.1, se puede ver el ejemplo clásico de una red multicast que cuenta con una sola fuente y varios nodos sumideros, en este caso dos. Ambos sumideros desean poder conocer la información que fue enviada desde el nodo fuente.

De acuerdo con los autores, una red capacitada con Network Coding, será aquella donde alguno de sus nodos intermedios tiene la habilidad de realizar una operación de codificación, rompiendo con el paradigma de enrutamiento tradicional de redes, donde se permiten nodos intermedios solo para hacer copias de paquetes recibidos y luego reenviarlos.

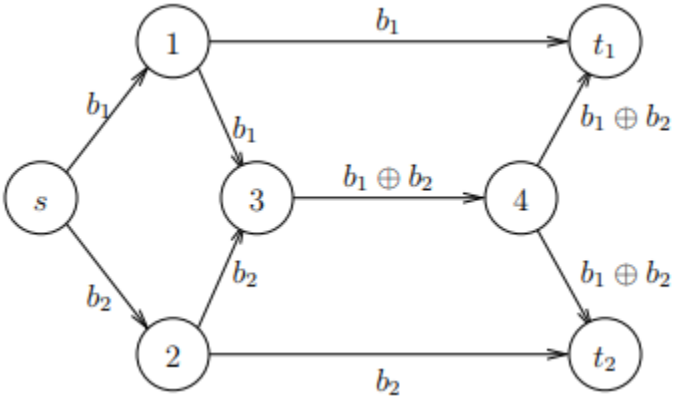


Figura 3.2.1.1 Modelo de Mariposa en Network Coding

En la Figura 3.2.1.1 el nodo 3 recibe dos paquetes, forma un nuevo paquete al tomar la suma binaria, o XOR, de los dos paquetes, y genera el paquete resultante. Por lo tanto,

si el contenido de los dos paquetes recibidos son los vectores b_1 y b_2, cada uno compuesto de bits, entonces el paquete que se envía desde 3 hacía 4 es $b_1 \oplus b_2$, formado a partir del XOR bit a bit de b_1 y b_2.

Los sumideros decodifican realizando operaciones de codificación adicionales con los paquetes que recibe cada uno. El sumidero t_1 recupera b_2 tomando el XOR de b_1 y $b_1 \oplus b_2$, y lo mismo para el sumidero t_2 que recupera b_1 tomando el XOR de b_2 y $b_1 \oplus b_2$.

En conclusión, las nueve transmisiones de paquetes que se utilizan en la red mariposa comunican el contenido de dos paquetes. Si el nodo intermedio 3 no tuviera esta capacidad, estas nueve transmisiones no serían suficientes para comunicar el mismo número de paquetes, y sería necesario hacer más transmisiones (por ejemplo, una transmisión adicional del nodo 3 al nodo 4); con esto vemos claramente cómo se incrementa la tasa de transferencia efectiva

3.2.2. **Mejoras en la Robustez**

Una de las principales desventajas de las comunicaciones multicast, en especial en redes inalámbricas es la presencia de errores causados por la pérdida de paquetes. En el esquema Network Coding, los nodos intermedios adquieren un papel más protagonista, lo que les permitirá mitigar el impacto de estos errores aleatorios. Esto se logra creando códigos o reglas de codificación/decodificación en los nodos intermedios que le otorgan la posibilidad de recuperar paquetes en caso de que estos se pierdan sin necesidad de volver a retransmitir.

3.2.3. **Mejoras en la Complejidad**

El uso de ciertas soluciones de Network Coding sobre topologías de red complejas permitirá mejorar significativamente el comportamiento del sistema global, recurriendo a soluciones subóptimas.

3.2.4. **Mejoras en la Seguridad**

Desde un punto de vista de seguridad, la codificación de red puede ofrecer ventajas en el tema, consideremos nuevamente la red mariposa de la Figura 3.1.1.1, supongamos que un intruso logra obtener solo el paquete $b_1 \oplus b_2$. Como solo obtuvo este paquete, el adversario no puede obtener ni b_1 ni b_2; así tenemos un posible mecanismo de comunicación segura.

Como se ha venido describiendo, son muchas ventajas que esta técnica aporta a una red, a su vez esas ventajas generan otras de manera simultánea. Como estas cuatro ventajas, a su vez, se podría pensar también en la reducción del consumo energético

como queda evidenciado en [7], que se traduce en mejor uso en dispositivos móviles o portátiles, y también en dinero.

3.3. **REDES MULTICAST**

Al hablar de redes multicast, debemos trasladar la mirada al momento tecnológico y cultural en que se encuentra la humidad. En otros tiempos pensar en que muchas personas estuvieran consumiendo servicios como Netflix, Skype, los “en vivo” de las redes sociales, o el simple hecho de hacer una conferencia con personas de todo el mundo, era solo ciencia ficción. Partiendo de esta premisa, entonces se pueden definir una red multicast como la capacidad de una red de comunicaciones para generar un mensaje simple desde una aplicación, y entregar copias del mensaje a múltiples receptores en diferentes localizaciones [8].

Las redes multicast nacen de la necesidad de enviar un mensaje desde una fuente a diferentes receptores, por lo que en 1990 Deering propuso el IP multicast, como una extensión al modelo de servicio unicast, con el fin de lograr este gran reto [9]. Se podría decir que se puede enviar información a un grupo de interesados, a través del envío de diferentes mensajes unicast a cada uno de los hosts destinos, pero esta solución no es óptima, por lo tanto, es preferible utilizar multicast para realizar esta tarea; todo esto debido a que al usar multicast se logra disminuir notablemente la carga en la red.

Uno de los grandes problemas que manifiestan en [10] para el enrutamiento multicast general, consiste en determinar la red que representa una sesión para la entrega confiable de los paquetes originados en un nodo fuente s a un conjunto de nodos sumideros $T \subset V\backslash\{s\}$ atravesando un grupo de nodos intermedios. Aquí V representa el conjunto de nodos del grafo de comunicaciones.

3.4. **FLUJO MAXIMO EN GRAFOS**

El flujo máximo se puede describir como la cantidad máxima de flujo que una red es capaz de transportar desde el nodo fuente hasta el nodo sumidero.

El valor del flujo se define como $|f| = \sum_{v \in V} f(s, v)$, denotando el valor del flujo total que sale de la fuente.

El flujo dentro del grafo de la red debe cumplir con las siguientes condiciones:

- Para los nodos intermedios, el flujo de entrada debe ser igual al flujo de salida.
- El flujo que pasa por cada uno de los caminos entre nodos, no debe ser mayor a la capacidad de ese camino.
- El flujo total desde el nodo fuente al nodo sumidero debe ser igual.

Asumiendo que un grafo dirigido $G = (V,\ E)$ es una red, con $s, t\ \in V$ con capacidad C en cada uno de los enlaces que forman los nodos $a, b, c, d\ \in E$, el flujo máximo del grafo en la Figura 3.4.1a se define como:

$$|f| = \sum_{n \in E} f_n$$

La Figura 3.4.1b muestra el grafo resultante luego de aplicar alguno de los métodos para calcular el flujo máximo, dejando como flujo máximo $|f| = 6$.

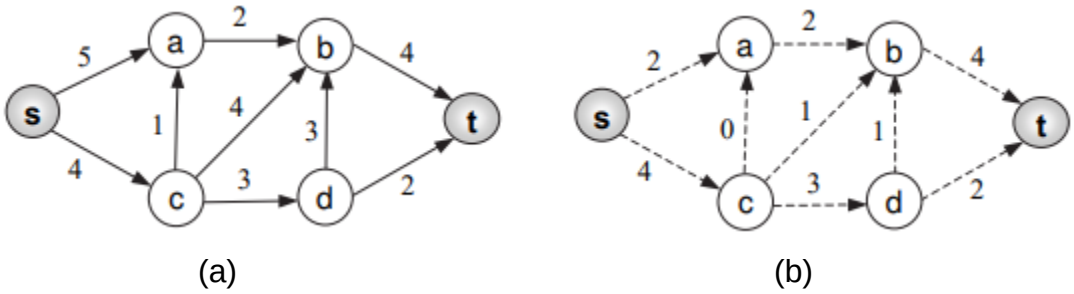


Figura 3.4.1 Problema de flujo máximo en redes. (a) Grafo de capacidades máximas de los canales. (b) Solución del problema, grafo de flujos reales.

Existen muchas implementaciones para obtener la solución al problema de flujo. Este trabajo de investigación solo se centrará en Dinic. Es importante saber que existen otras alternativas para acometerlo. A continuación, se enumeran los más conocidos y los más usados para esto.

- Algoritmo de Ford-Fulkerson.[11]
- Algoritmo de Dinic. [12]
- Algoritmo de Edmonds-Karp. [13]
- Algoritmo de bloqueo de fluo de Dinic. [14]
- Algoritmo MPM (Malhorta-Pramodh-Maheshwari). [15]
- Algoritmo de empuje y re-etiquetar (push-relabel). [16]
- Algoritmo de empuje y re-etiquetar con FIFO (First In, First Out). [17]
- Algoritmo de KRT (King-Rao-Tarjan). [18]

3.5. **DINIC**

Dinic es un algoritmo de Flujo máximo en redes de flujo, este algoritmo fue creado en 1970 por el científico de la computación Yefim Dinitz, un israelí de origen soviético. El

algoritmo está basado en el Algoritmo de Edmonds-Karp, pero a diferencia de este, Dinic utiliza elementos diferenciadores, como son las trayectorias de aumento más cortas e introduce el concepto de nivel del grafo y bloqueo de flujo, que es lo que define el rendimiento del algoritmo.

3.5.1. **Definiciones**

Sea $G = (V, E)$ un grafo, con $|V|$ vértices, $|E|$ enlaces y donde cada enlace (u, v) tiene una capacidad no negativa $c(u, v)$ y un flujo $f(u, v)$, se busca maximizar el valor del flujo desde una fuente s hasta un sumidero t, definido entonces así: $G = ((V, E), c, s, t)$

La **capacidad residual** es un mapeo $c_f: V \times V \rightarrow R^+$ definida así:
$1. Si\ (u, v) \in E$,
$2. c_f(u, v) = c(u, v) - f(u, v);\ c_f(v, u) = f(u, v)$
$3. c_f(u, v) = 0, de\ otra\ manera$

El grafo residual es el grafo $G_f = \big((V, E_f), c_f|E_f, s, t\big)$ donde
$E_f = \{(u, v) \in V \times V | c_f(u, v) > 0\}$

La **trayectoria de aumento** es una ruta $s - t$ en el grafo residual G_f.

Se define $dist(v)$ como la **longitud del camino** más corto desde s hasta v en G_f.

El **nivel del grafo** de G_f es el grafo
$G_L = \big((V, E_L), c_f|E_L, t\big)$, donde $E_L = \{(u, v) \in E_f : dist(v) = dist(u) + 1\}$

El **bloqueo del flujo** es una ruta $s - t$, y flujo f de manera tal que el grafo $G' = \big((V, E_L^i), s, t\big)$ con $E_L^i = \{(u, v) | f(u, v) < c_f|E_L(u, v)\}$ no tiene ninguna ruta $s - t$.

3.5.2. **Algoritmo**

Se puede demostrar que el número de arcos en cada bloqueo de flujo es incrementado en al menos una unidad cada vez, y por lo tanto, hay al menos $n - 1$ bloqueos de flujo en el algoritmo, donde n es el número de vértices en la red. El nivel del grafo G_L es construido por el algoritmo de profundidad en un tiempo de $O(E)$ y un bloqueo de flujo en cada nivel del grafo puede ser encontrado en un tiempo de $O(VE)$. Por lo tanto, el tiempo del algoritmo de Dinic es de $O(V^2E)$. En el Algoritmo 1 se puede ver la definición general de Dinic.

Algoritmo 1: Dinic

Entrada: Red $G = ((V, E), c, s, t)$, nodo fuente s, nodos sumideros t
Salida: Un flujo $s - t$, f de valor maximizado

1. Se tiene $f(e) = 0\, para\, cada\, e \in E$;
2. Construir G_L desde G_f de G. Si $dist(t) = \infty$, detener y retornar resultado de f;
3. Encontrar un bloque de flujo de f' y en G_L;
4. Aumentar el flujo f por f' y volver al paso 2;

Algoritmo 1 Seudocódigo Dinic General

3.5.3. **Ejemplo**

Según la Figura 3.7.3.1, se parte del grafo G, que tiene un nodo origen s y un sumidero t, se puede observar la capacidad de los enlaces y el flujo inicial en 0.

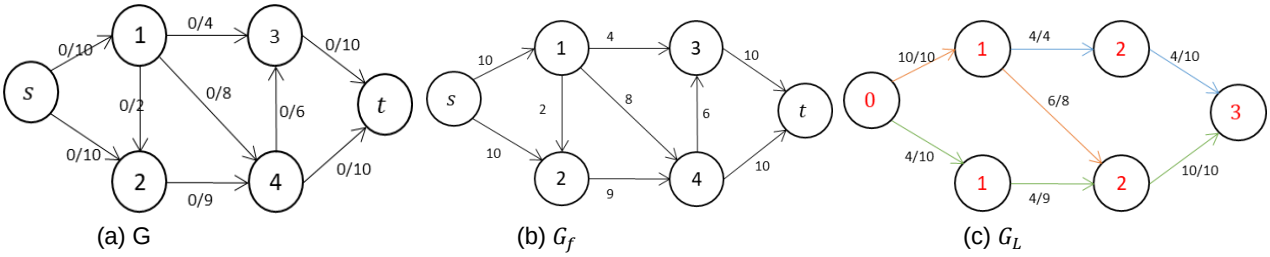


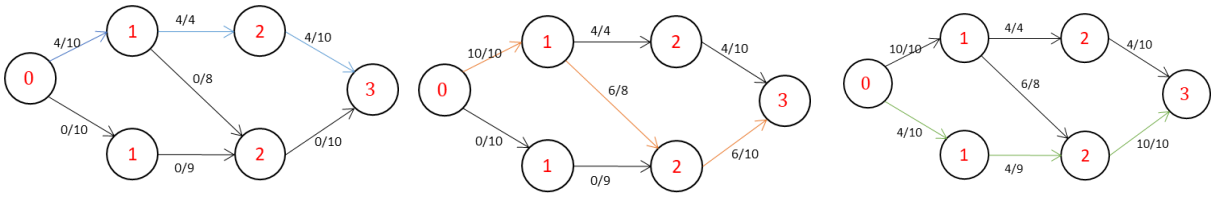
Figura 3.5.1 Simulación del algoritmo de Dinic. , (a) Grafo original, (b) Grafo residual y (c) Grafo de Nivel. En el nivel del grafo G_f,los nodos marcados en rojo son los valores dist(v) . Las rutas en colores forman el bloqueo de flujo. Primera iteración.

El grafo residual G_f posee la capacidad residual, $c_f(u, v) = c(u, v) - f(u, v)$;
$c_f(s, 1) = 10 - 0 = 10$
$c_f(1, s) = f(s, 1) = 0$
$c_f(s, 2) = 10 - 0 = 10$
$c_f(2, s) = f(s, 2) = 0$
$c_f(1,2) = 2 - 0 = 2$
$c_f(2,1) = f(1,2) = 0$

Finalmente, el grafo de nivel muestra en sus nodos los niveles del grafo original luego de haber calculado el bloqueo de flujo.

En la Figura 3.5.3.2 se puede observar el resultado de la primera iteración, y los diferentes caminos que se generan para hallar el flujo máximo.

1. $\{s, 1,3, t\}$ con flujo máximo 4, (Figura 3.5.3.2a)
2. $\{s, 1,4, t\}$ con flujo máximo 6, (Figura 3.5.3.2b)
3. $\{s, 2,4, t\}$ con flujo máximo 4, (Figura 3.5.3.2c)

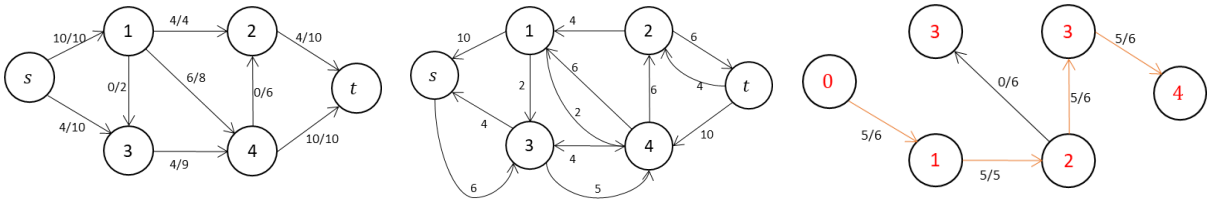


(a) *(b)* *(c)*

Figura 3.5.2 Primera iteración algoritmo de Dinic ,(a) Grafo de nivel del camino {s,1,3,t} , (b) Grafo de nivel del camino {s,1,4,t} y (c) Grafo de nivel del camino {s,2,4,t}.

Por lo tanto, el bloqueo del flujo es de 14 unidades y el valor del flujo $|f|$ es 14.Para este ejemplo, solo toma estos 3 bloqueos de flujo debido a que de acuerdo con la definición de nivel, se debe ir a un nivel $u + 1$. En este caso quedaron por fuera $\{s, 1,2,4, t\}$, $\{s, 1,2,4,3, t\}$, $\{s, 2,4,3, t\}, \{s, 1,4,3, t\}$.

Iteración 2: Se repite el proceso con el resultante del grafo y queda reda registrado en la Figura 3.7.3.3.



(a) *(b)* *(c)*

Figura 3.5.3 Simulación del algoritmo de Dinic., (a) Grafo G, (b) Grafo residual y (c) Grafo de Nivel. En el nivel del grafo G_f, los nodos marcados en rojo son los valores dist(v) . Las rutas en naranja forman el bloqueo de flujo. Segunda iteración.

El grafo residual G_f queda así para esta iteración, $c_f(u, v) = c(u, v) - f(u, v)$;

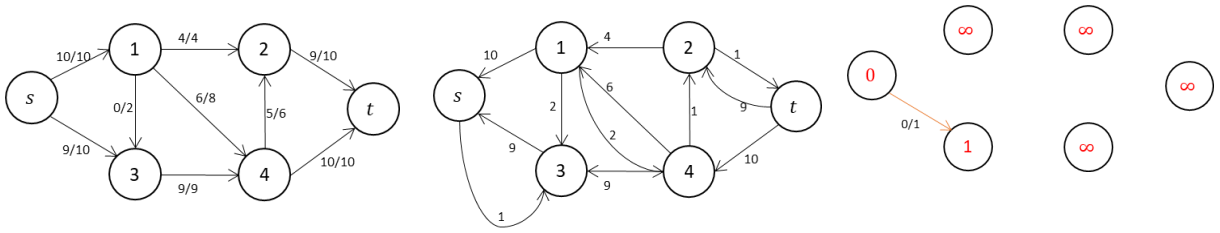
$c_f(s,1) = 10 - 10 = 0$
$c_f(1,s) = f(s,1) = 10$
$c_f(s,2) = 10 - 4 = 6$
$c_f(2,s) = f(s,2) = 4$
$c_f(1,2) = 2 - 0 = 2$
$c_f(2,1) = f(1,2) = 0$

Finalmente, el grafo de nivel muestra en sus nodos los niveles del grafo luego de haber calculado el bloqueo de flujo del grafo.

1. $\{s, 2, 4, 3, t\}$ con flujo máximo 5

Por lo tanto, el bloqueo del flujo es de 5 unidades y el valor del flujo $|f|$ es 14 + 5 = 19.

Iteración 3. Repetimos el proceso con el resultante del grafo de la Figura 3.7.3.4.



(a) *(b)* *(c)*

Figura 3.5.4 Simulación del algoritmo de Dinic. (a) Grafo G, (b) Grafo residual y (c) Grafo de Nivel. En el nivel del grafo G_f, los nodos marcados en rojo son los valores dist(v) . Las rutas en naranja forman el bloqueo de flujo. Tercera iteración.

Desde s no puede ser alcanzado en t en G_f. El algoritmo termina y retorna un valor máximo de flujo de 19.

Dinic está basado en el Algoritmo de Edmonds-Karp, pero incluye las mejoras en cuanto a bloqueo de flujo y niveles de grafo, que hace que cada ruta de aumento en el flujo de bloqueo tenga caminos más cortos.

Dinic al igual que Edmons-Karp requiere del Algoritmo de Búsqueda en Anchura para encontrar todos los caminos posibles hasta el nodo sumidero, con la restricción de nivel, que es definida por la Búsqueda en Profundidad, por lo tanto, es necesario comprender como funciona cada uno de estos algoritmos.

3.5.4. **Breadth-First Search (BFS) o Búsqueda en Anchura**

BFS es un algoritmo de búsqueda en anchura para grafos y árboles. Este algoritmo es usualmente utilizado para resolver problemas en el área de la teoría de grafos [7], como encontrar el camino más corto entre dos nodos (GPSs, rutas de aviones, etc), construcción de recolectores de basura en sistemas de información y el problema que más interesa en este trabajo: encontrar caminos más cortos y óptimos para aplicar la solución del problema de flujo máximo con el algoritmo Dinic.

Este algoritmo funciona empezando desde el nodo fuente del grafo, de ahí visita a sus vecinos más cercanos y los clasifica como nodos con la misma distancia al nodo fuente. Como ya el nodo fuente sabe cuáles son sus vecinos más cercanos y se encuentran debidamente clasificados, se marca el nodo fuente como un nodo ya visitado.

Se toma el siguiente nodo dentro de la bolsa de los clasificados, como nodo origen, se visitan todos los nodos adyacentes a este nodo y son debidamente clasificados. Si se encuentra un nodo que ya se visitó anteriormente, este no se re-clasifica. Así, sucesivamente se mueve entre todos los nodos necesarios, visitándolos, por lo menos, una sola vez y clasificando los nodos adyacentes a él. Esto se puede observar en la Figura 3.5.3.4

Dinic utiliza estos caminos clasificados para poder hallar el flujo máximo, sin antes verificar que el camino que tenga como candidato, sea un camino con el menor flujo de bloqueo a través de una Búsqueda en Profundidad (DFS, por su acrónimo en inglés).

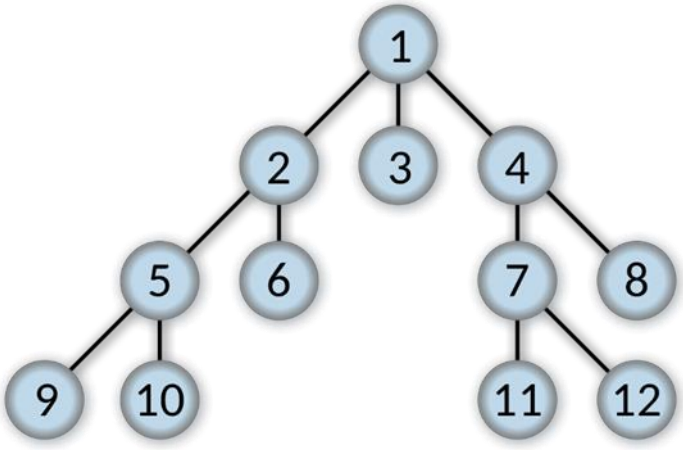


Figura 3.5.5 Representación numérica de búsqueda en anchura de un grafo

3.5.5. **Depth First Search (DFS) o Búsqueda en Profundidad**

DFS también es un algoritmo utilizado para resolver problemas en el área de la teoría de grafos, el cual realiza una búsqueda en profundidad en grafos y árboles, a diferencia de BFS, este algoritmo, aunque inicia la búsqueda en el nodo fuente del grafo, expande cada uno de los nodos que visita de manera recurrente hasta que no encuentre ningún otro camino. Mientras esto ocurre, clasifica a los nodos encontrados como visitados, al momento de no encontrar más nodos, se devuelve de modo que repite el mismo proceso con cada uno de los nodos adyacentes al nodo ya procesado. La Figura 3.5.3.4 muestra una representación gráfica de lo que sería una búsqueda en profundidad de manera más clara.

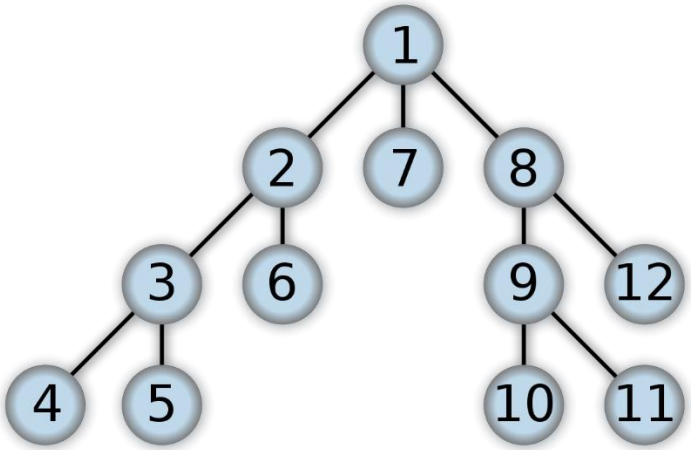


Figura 3.5.6 Representación numérica de búsqueda en profundidad de un grafo

4. ESTADO DEL ARTE

4.1. NETWORK CODING

El término Network Coding fue originalmente postulado por Ahlswede *et. al.* en [19]. Este grupo de personas, fueron los pioneros en hacer el cambio de paradigma de comunicación de paquetes a diferencia de como se venía haciendo tradicionalmente. En este trabajo, se demostró matemáticamente que si una red se modela como un grafo dirigido $G\ (V, E)$, una fuente $s \in V$, puede comunicarse con un conjunto de receptores $T \subseteq V$, alcanzando la tasa broadcast h (siendo h el valor del mínimo corte entre s y $t \in T$), siempre y cuando se permita la codificación de los paquetes en los nodos intermedios, según el teorema conocido como “Max Flow-Min Cut” [20].

Este primer paso, generó toda una ola de investigación respecto al tema, donde Li, Yeung y Cai [21] demostraron que no es necesario recurrir a complejas operaciones para conseguir un rendimiento óptimo, sino que es suficiente con utilizar sencillas combinaciones lineales.

Surgieron otros estudios [22] y [23] que demuestran que, para tráfico multicast, los códigos lineales alcanzan las cotas superiores de rendimiento, pudiendo llevar a cabo las tareas de codificación y decodificación en tiempo polinomial. Dando un paso más allá, mostraron en [24] que la elección aleatoria de estos coeficientes lineales dentro de un cuerpo finito de Galois proporciona también la máxima capacidad en esquemas multicast con múltiples fuentes.

Tras esta fase inicial, se empiezan a conformar dos grandes grupos en cuanto al criterio a utilizar al momento de la codificación: en el primero de ellos, se conoce como inter-flujo o inter-sesión, acá se combina la información procedente de diferentes flujos en los nodos intermedios. La propiedad fundamental de este tipo de combinaciones es que se aumenta el throughput gracias a la reducción del número de transmisiones necesarias para intercambiar la misma cantidad de información.

Por otra parte, se tiene el grupo de intra-flujo o intra-sesión, donde cada una de los flujos de datos será tratado de forma independiente. En este caso, el primer proceso de codificación lo lleva a cabo el nodo fuente, mientras que los elementos intermedios recombinan los mensajes a medida que atraviesan la red.

Existe una tercera alternativa, más reciente, que trata de explotar las virtudes de las anteriores, combinándolas en una única solución Network Coding. Este tipo de

esquemas, se definen como híbridos, cuyo propósito es tomar lo mejor de los primeros grupos.

Dentro del Séptimo Programa Marco de la Comisión Europea, N-CRAVE10, finalizado en diciembre del 2010, se expuso un proyecto completamente dedicado a mejorar la robustez del Network Coding en redes inalámbricas. Dando inicio al Network Coding simulator (NECO) [25], una herramienta de simulación de alto rendimiento para la evaluación del comportamiento de diferentes protocolos de Network Coding. Microsoft también ha mostrado interés por este tema, teniendo activo el proyecto Avalanche [26], generando un prototipo para la distribución de archivos en una red peer-to-peer con Network Coding como elemento central de la arquitectura.

Existen investigaciones de naturaleza más práctica, que tratan de encontrar los nichos en los que se pueda sacar mayor provecho al uso de Network Coding, pasando de resultados analíticos a implementaciones reales. En la Figura 4.1.1 se recogen algunas de las principales líneas de trabajo más importantes sobre el tema.

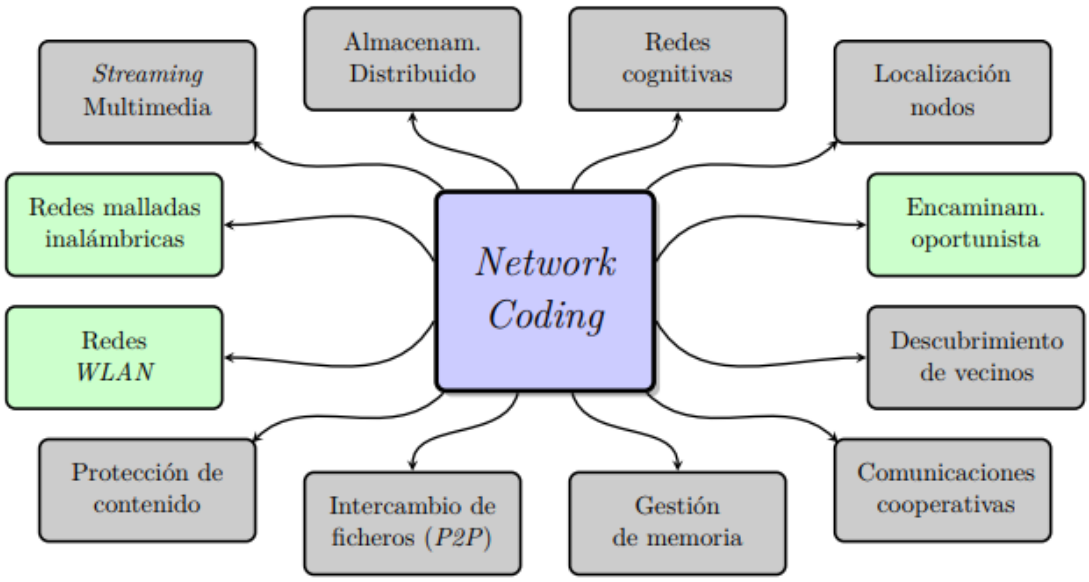


Figura 4.1.1 Aplicaciones con soluciones basadas en Network Coding. En verde se encuentran aquéllas que presentan una mayor relación con el contenido que se va a tratar en esta Tesis

Como puede verse en el diagrama, hay una gran heterogeneidad. Por citar algunos ejemplos de aplicación de Network Coding, se encuentran esquemas para la transmisión multicast de tráfico multimedia [27], optimizando la utilización del ancho de banda en redes inalámbricas y mejorando la Calidad del Servicio sobre canales hostiles. También tiene una gran relevancia en técnicas de almacenamiento “en la nube” (Cloud

Computing) [28]; así, aprovechando la redundancia distribuida, se incrementa la fiabilidad del servicio. Otro de los grupos a destacar son los esquemas de intercambio de archivos a través de redes Peer-to-Peer (P2P) [29], en los que se pretende demostrar la validez de una de las premisas iniciales del uso de Network Coding: puede maximizar la tasa de los flujos en sesiones multicast sobre grafos dirigidos acíclicos.

Por último, con el lanzamiento de la nueva red 5G se han venido adelantando investigaciones dado que la filosofía de esta nueva tecnología es tener una comunicación D2D (Device to Device). Se pretende con Network Coding, lograr casi que duplicar el rendimiento de la red, al tiempo que aumentar su robustez y disminuir el tiempo total para difundir mensajes masivos entre los dispositivos como se observa en [30].

4.2. ALGORITMOS PARA HALLAR FLUJO MAXIMO Y NETWORK CODING

En una investigación realizada por Wu y Curran [31] en el 2010 dentro de este campo de estudio, los autores decidieron tomar un esquema de una red basado en la combinación con el teorema del flujo máximo. Dentro de este trabajo, se hizo la propuesta de un conjunto de reglas con las cuales se toman los flujos máximos de una red multicast y se transforman en una arquitectura de red para aplicar la técnica de Network Coding. Además del conjunto de reglas, también se propone que, dentro de la arquitectura de red resultante, los nodos codificadores, aquellos nodos que solo retransmiten, y los múltiples nodos sumideros; sean ajustados para también aplicar reglas de enrutamiento. En conjunto, estas propuestas proveen la construcción de una red práctica para un esquema que tiene solución aplicando Network Coding.

Después de aplicar este conjunto de reglas en el trabajo propuesto por Wu y Curran, pudieron obtener un framework para la construcción de un esquema de red con Network Coding y además introdujeron dentro de la técnica, otros conceptos de nodos: los nodos multicast y los nodos de retransmisión [31]. Con este framework fueron capaces de identificar como las redes construidas a partir de este tipo de reglas, disminuyen el uso de recursos que se requieren para Network Coding.

Esta investigación es también muy similar al programa propuesto en este trabajo, ya que utiliza los beneficios de los algoritmos para hallar el flujo máximo en redes multicast con el fin de construir un esquema de red a la cual se le pueda aplicar la técnica de Network Coding. La diferencia de ese trabajo con el presentado en este proyecto es el algoritmo de flujo máximo que utilizan para encontrar los caminos óptimos en la red y varios detalles técnicos internos. Además, se incluyen los conceptos de enrutamiento y otro tipo de nodos que ayudan a disminuir los recursos que necesita la red para la retransmisión de paquetes.
Aunque son muchos los trabajos e investigaciones sobre Network Coding que se encuentran publicados alrededor del mundo, son muy pocos los trabajos relacionados

con los problemas de flujo máximo en grafos y su aplicación en Network Coding. La mayoría de los trabajos que incluyen temas sobre el flujo máximo, aunque son parecidos en hipótesis, no resuelven los mismos problemas que se propone resolver en este trabajo.

Es por esto, que el trabajo realizado en [10] se convierte en el eje principal para esta investigación, debido a que los problemas que se tratan van de la mano con toda la esencia de lo que desea demostrar con esta disertación, incluso se siguieron muchos de los lineamientos planteados por el autor, el cual se centra en dos aspectos a solucionar: en primer lugar, la reducción a un grafo dirigido y acíclico multicast uni-sesión de mínimo flujo máximo, que permite la resolución de un sistema lineal de ecuaciones bajo la técnica de Network Coding; y en segundo lugar, la obtención del esquema de codificación, que determina la posibilidad de resolver la organización y envío de los paquetes simultáneamente, desde el nodo fuente hacia los nodos sumideros en un grafo multicast de mínimo flujo máximo. Esto se convierte entonces en nuestro punto de partida.

5. METODOLOGÍA

5.1. MÉTODO DE INVESTIGACIÓN

Este proyecto se basa en el método inductivo, por lo que encontraremos dentro de esta investigación los cuatro pasos esenciales: la observación de los hechos para su registro; la clasificación y el estudio de estos hechos; la derivación inductiva que parte de los hechos y permite llegar a una generalización; y la contrastación.

El punto de partida de esta investigación, consiste en un esquema de red general de comunicación con los nodos fuente y sumideros multicast predefinidos. A esta red se le aplica el concepto de teoría de flujo máximo en grafos; el algoritmo escogido para aplicar esta teoría es Dinic.

Por lo anterior, se sabe que se parte de una red general de comunicaciones que posee múltiples enlaces y nodos, descritos de la siguiente manera: un nodo fuente, n nodos sumideros y m nodos intermedios. El centro de esta investigación consiste en encontrar una aproximación de esta red, que la convierta en una red multicast uni-sesión (con un nodo fuente y varios sumideros) que posea un flujo máximo común y que permita aplicar la teoría de Network Coding para aprovechar mejor el ancho de banda y la entrega simultánea de paquetes.

Al aplicar la técnica de Network Coding, se pretende evaluar el comportamiento de Dinic para hallar una red multicast. Además, los resultados obtenidos, permiten la comparación entre los mecanismos de eliminación de rutas redundantes para llegar al mínimo flujo máximo, y finalmente se obtienen las conclusiones de esta investigación.

El desarrollo experimental de este trabajo se lleva a cabo probando diferentes ejemplos de redes de comunicación representadas en grafos con múltiples enlaces y nodos intermedios, donde se intenta obtener la aproximación de la red multicast uni-sesión, para luego probarla sobre el programa de asignación y ordenamiento de paquetes [32]. De este ejercicio, se puede concluir si la red tiene una solución multicast donde se pueda aplicar Network Coding.

Durante el proceso de investigación y formulación de este trabajo, se encontró que aplicando métodos o algoritmos de flujo máximo basados en Dinic, se podía llegar a una posible solución de red multicast uni-sesión. Sobre este grafo multicast uni-sesión, se determina si con la implementación de Network Coding en los nodos intermedios, se podría enviar el mínimo flujo máximo de paquetes simultáneos al conjunto de nodos sumideros. A partir de este hecho, se elaboró la hipótesis en la cual se establece si con

el apoyo del algoritmo Dinic, se podría ejecutar una aproximación de red multicast uni-sesión a partir de un grafo general de comunicaciones.

5.2. **HIPÓTESIS**

Se parte de un grafo dirigido correspondiente a una red de comunicaciones general, esta red cuenta con múltiples nodos y enlaces, con un nodo fuente y n sumideros identificados con anticipación, los cuales forman la red multicast. A este grafo se le aplica el algoritmo unicast de flujo máximo Dinic para cada uno de sus nodos sumideros, seguidamente se verifica que todos los caminos resultantes sean disyuntos entre ellos; con estos caminos disyuntos se obtiene el flujo máximo de cada sumidero. A partir del conjunto de valores de flujos máximos, se obtiene el mínimo flujo máximo común de todos los grafos unicast. Al unir todos los caminos se obtiene el grafo con la aproximación de red multicast uni-sesión que, bajo la técnica de Network Coding, podría llevar a cabo la codificación/decodificación de paquetes. Lo anterior, posibilita la entrega de estos paquetes que originalmente fueron emitidos por el nodo fuente a los nodos sumideros.

Este grafo, podría resultar efectivo o no para entregar de forma correcta y ordenada los paquetes originales emitidos por el nodo fuente. Esta es entonces, la razón fundamental para asegurar que la propuesta solo ofrezca una aproximación a la construcción de un grafo de red multicast que permita entregar simultáneamente a los nodos sumideros cada uno de los paquetes enviados mediante la aplicación de Network Coding.

El primer paso para el desarrollo de la hipótesis fue escoger el algoritmo a utilizar, se tomó como referente el algoritmo de Dinic; esto, porque de acuerdo con su origen, es una mejora de EK, todo gracias a que los tiempos de ejecución son menores, ya que al utilizar trayectorias de aumento más cortas y manejar los conceptos de nivel de grafo y bloqueo de flujo, se convierten en sí en las características esenciales y fundamentales para darle un mejor rendimiento a este algoritmo.

El algoritmo de Dinic, como se mencionó anteriormente, puede ser aplicado sobre un grafo dirigido, con un nodo fuente y un solo nodo sumidero, resolviendo un problema unicast de flujo máximo. En el modelo propuesto, este algoritmo se aplica sobre el grafo general de comunicaciones para encontrar el flujo máximo y los caminos disyuntos entre cada par de nodos formado por el nodo fuente y cada uno de los nodos sumideros. Luego, los grafos de flujo máximo individuales, se unifican en uno solo, el cual será el grafo de la red multicast con flujo máximo común determinado por el teorema de Network Coding [26].

5.3. ELABORACIÓN DE LA SOLUCIÓN

Se escribió una solución donde se aplica el algoritmo de Dinic tantas veces como número de nodos sumideros estén presentes en el grafo de la red multicast de flujo máximo.

Ejemplo, para el grafo de la Figura 5.3.1 con un nodo fuente y tres nodos sumideros, se debe aplicar el algoritmo de Dinic tres veces. Para cada una de estas iteraciones, el nodo fuente es el mismo de la red inicial pero el sumidero debe ir cambiando hasta completar la totalidad existente de nodos sumideros. De esta manera, se obtendría un flujo máximo particular para el segmento de grafo unicast constituido entre el nodo fuente, cada sumidero y los enlaces que los conectan.

Al hallar los posibles caminos para cada segmento de grafo, se debe asegurar que estos sean caminos disyuntos; es decir, que no presenten nodos en común o nodos repetidos. La Figura 5.3.1 presenta este tipo de casos, donde se nota que el nodo 5 genera caminos en común desde 1 para llegar al sumidero 8, que será el nodo al cual queremos llegar.

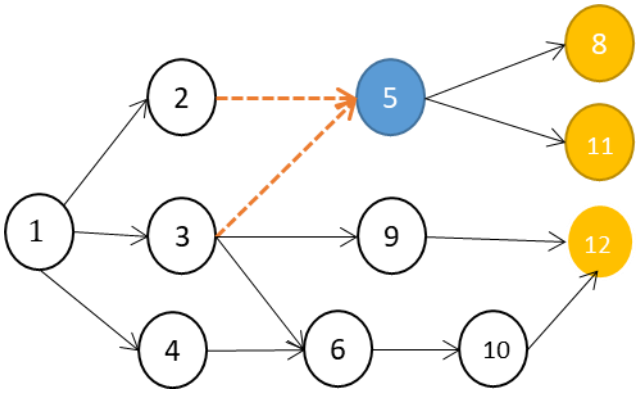


Figura 5.3.1 Segmento de grafo de 1 a 8 con nodo 5 común

En el primer segmento de grafo con nodo fuente 1 y nodo sumidero 8 de la Figura 5.3.1b , se encuentran dos caminos posibles:

- Primer camino: $1 \rightarrow 2 \rightarrow 5 \rightarrow 8$
- Segundo camino: $1 \rightarrow 3 \rightarrow 5 \rightarrow 8$

Se observa que los caminos 1 y 2 tienen en común el nodo 5; es decir, que dos nodos pertenecen a distintos caminos para llegar al nodo sumidero.

Esto no es óptimo para la solución que se necesita construir, por lo tanto, se necesita hacer ajustes al segmento de grafo para que el algoritmo de Dinic encuentre otro camino donde no se presente esta situación; es decir, un camino disyunto.

En la solución, cuando se encuentra que un camino no es disyunto, se identifica cual es el nodo común, se deja como apto el primer camino que es hallado, luego se bloquea el nodo en común, eliminando el enlace que se conectaba con este nodo en el nuevo camino encontrado, así:

- Primer camino: $1 \rightarrow 2 \rightarrow 5 \rightarrow 8$
- Segundo camino: $1 \rightarrow 3 \times 5 \rightarrow 8$

Seguidamente, se ejecuta el algoritmo de búsqueda nuevamente para que siga verificando que todos los caminos sean disyuntos. En este caso, como no encuentra otro camino de 1 a 8, el camino $1 \rightarrow 2 \rightarrow 5 \rightarrow 8$ es guardado como camino apto y estará presente en el nuevo grafo que cumple con las necesidades de este trabajo.

El resultado de esta iteración se verá como se muestra en la Figura 5.3.2 donde fue eliminado el segundo camino hallado.

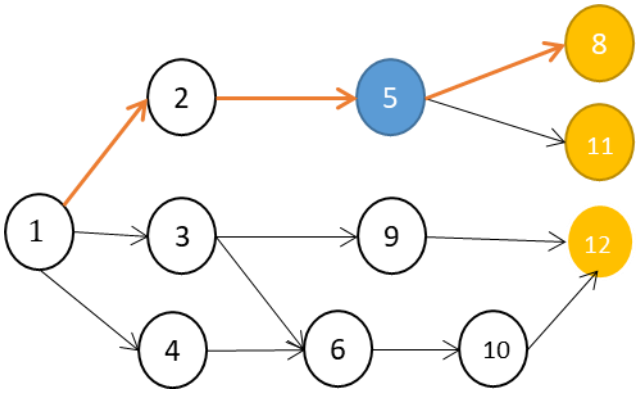


Figura 5.3.2 Segmento de grafo de 1 a 8 con camino disyuntos

Una vez obtenidos los flujos máximos y los grafos unicast de flujo máximo particulares de todos los nodos sumideros, aun no se cuenta con todas las condiciones para construir una aproximación de red, ya que se debe establecer el valor del mínimo flujo máximo global para el grafo completo.

El valor del mínimo flujo máximo para todo el grafo en general, se toma como el menor valor numérico de todos los flujos máximos encontrados [25]; es decir, si para el grafo con cuatro nodos sumideros, los flujos máximos son 3, 4, 3 y 5 respectivamente, entonces se toma como flujo máximo global el flujo máximo particular de menor valor, 3.

En este punto, se ha determinado el mínimo flujo máximo global de la red de comunicación, pero se seguirá teniendo en algunos segmentos del grafo un flujo máximo mayor al flujo máximo global, por lo que será necesario realizar una última tarea: se debe garantizar que todos los segmentos tengan este flujo.

Para realizar este ajuste, se debe tomar cada segmento de grafo unicast con flujo máximo mayor que el flujo máximo global y eliminar los caminos que sean innecesarios para igualar el flujo máximo global. Por ejemplo, para un segmento de grafo donde el algoritmo de Dinic terminó con flujo máximo particular 4, y teniendo en cuenta que el flujo máximo global de todo el grafo es 3, entonces se debe eliminar un camino de este segmento de grafo para igualar el flujo máximo global.

Esta eliminación se hace probando tres formas distintas, con el fin específico de analizar cuanto impacta la eliminación de los caminos, si tienen algunas características especiales algunos caminos o si es inherente a la solución eliminar estas rutas en el grafo. Las formas de eliminación que se implementaron en este trabajo fueron: el camino más largo, el primer camino hallado y por último, elegir un camino al azar entre los diferentes caminos que presentan esta condición.

Ya con todos los segmentos del grafo con el mismo flujo máximo, se deben unir los caminos encontrados para cada uno de los segmentos y guardarlos en una matriz que contendrá al grafo final cuya representación concluiría con una aproximación de red a la cual sería posible aplicarle la técnica de Network Coding.

Esta solución y el algoritmo desarrollado, serán explicados con detalle y ejemplos en la siguiente sección.

5.4. EVALUACIÓN DE LOS RESULTADOS

Después de haber obtenido la aproximación de red otorgada por el programa basado en Dinic, es necesario verificar que estos resultados sean positivos o negativos para la propuesta de este trabajo, por esta razón se utiliza un programa de verificación externo, el cual determina que la red obtenida por el programa basado en Dinic, logra o no una solución multicast utilizando la técnica de Network Coding [32].

Este programa de verificación recibe como entrada la aproximación de red multicast uni-sesión en forma de matriz cuadrada que el programa basado en Dinic proporciona; este confirma si la red multicast uni-sesión tiene solución. Esta solución puede ser en dos formas:

- Paquetes individuales: el nodo fuente retransmite por los enlaces de salida los paquetes originales correspondientes a la cantidad que representa el flujo máximo de la red. Estos paquetes se envían etiquetados con letras del alfabeto español. En este caso, el nodo fuente actúa como un nodo de retransmisión solamente.

- Paquetes codificados: el nodo fuente tiene la posibilidad de codificar los paquetes individuales generando una combinación lineal de estos y retransmitiéndolos por sus enlaces de salida.

Por ejemplo, si el flujo máximo de la red obtenida con el algoritmo propuesto en este trabajo es 3, según la primera forma de retransmitir, se deben enviar los 3 paquetes individuales correspondientes a los etiquetados con las letras A, B, y C; y según la segunda forma de retransmitir se podrían enviar 7 paquetes (3 que son los paquetes originales y 4 que son las combinaciones lineales de ellos).

6. SOLUCIÓN PROPUESTA

El objetivo principal de este trabajo es tomar una red general de comunicaciones donde se especifican los nodos fuente y sumideros de una red multicast uni-sesión. Esta red es representada como un grafo dirigido en el cual se aplica un método determinístico para obtener una aproximación a una red multicast uni-sesión de flujo máximo común entre los nodos sumideros. La red resultante debe ser apta para aplicar la técnica de Network Coding, todo esto con el fin de que a través de ella pueda hacerse la entrega simultánea del número de paquetes correspondientes al flujo máximo. Network Coding se aplica a través de la codificación y decodificación de los paquetes que simultáneamente ingresan a un nodo codificador/decodificador mediante la aplicación de la suma módulo 2 o el XOR entre los bits correspondientes a las mismas posiciones en los paquetes entrantes. Esta operación genera un único paquete saliente hacia los siguientes nodos adyacentes y, logrando con esto mejorar el rendimiento del tráfico de red y evitar cuellos de botella en los enlaces de salida del nodo.

Como se describió previamente, se utilizó el algoritmo de flujo máximo de Dinic, ajustándolo para poder aplicarlo a un grafo de comunicaciones dirigido con múltiples nodos sumideros, con el fin de obtener la representación matricial de un grafo de red multicast uni-sesión donde se pueda aplicar Network Coding posteriormente.

Las redes de comunicaciones generales iniciales que se usarán como parámetros de entrada (aquellas de las cuales se obtendrá la aproximación de red multicast uni-sesión), están representadas por grafos. Estos grafos deben ser dirigidos (de nodo a nodo tienen un enlace que los une en una sola dirección), cada enlace debe tener asociado un valor denominado capacidad, y para el caso de este proyecto, este valor siempre será 1 (el ancho de banda de cada enlace es de 1 paquete por unidad de tiempo, no importando su tamaño), y la representación del grafo de comunicaciones tiene la forma de matriz de adyacencia cuadrada. Si G es la matriz de adyacencia de la red de comunicaciones, $G(i,j) = 1$, significa que existe un enlace de comunicaciones entre el nodo i y j con un ancho de banda de 1; si $G(i,j) = 0$, esto quiere decir que no existe un enlace entre los nodos i y j.

A continuación, se explicará detalladamente paso a paso como se obtuvo el resultado esperado mediante un programa escrito en C++, el cual espera una matriz de entrada que representa el grafo de la red y un arreglo con los índices de los nodos sumideros del grafo. Se asume que el nodo etiquetado con 0 es el nodo fuente.

6.1. MODO DE USO DE LA SOLUCIÓN

El programa fue desarrollado en C++ ya que este lenguaje provee un completo soporte nativo para muchas estructuras de datos, incluyendo los arreglos y vectores que se

utilizaron para manejar las matrices que representan los grafos. Más importante, y la razón principal para la selección de este lenguaje, es que es más veloz haciendo operaciones a bajo nivel, lo que brinda la oportunidad de utilizar sus métodos para generar aproximaciones de redes incluso en grafos considerablemente grandes.

6.1.1. **Parámetros de entrada**

La Figura 6.1.1.1 muestra la información de archivo de texto *.txt* de entrada que recibe el programa desarrollado en este trabajo, los parámetros están distribuidos de la siguiente manera, la primera línea contiene el número de nodos del grafo, la segunda línea contiene el arreglo de nodos sumideros de la red y de la tercera línea hasta el final del archivo, está la representación matricial del grafo de la red general de comunicaciones.

Ejemplo:

Nombre del archivo: 7grafo2fm1v2des.txt

Contenido del archivo:

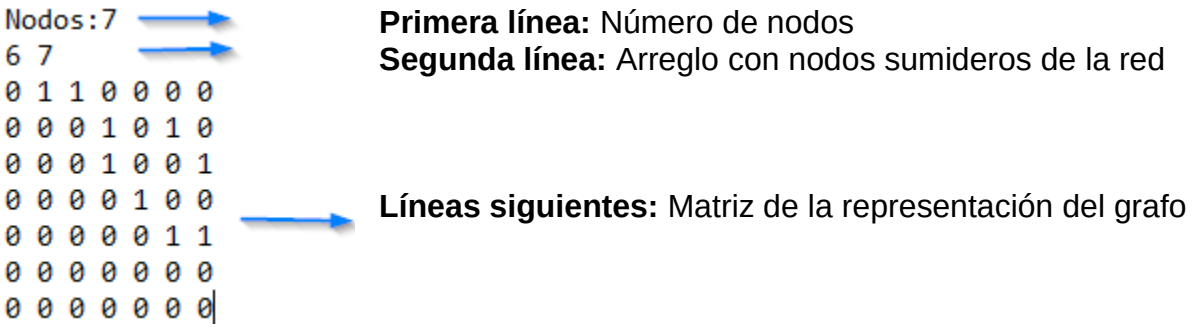


Figura 6.1.1.1 Archivo de entrada para algoritmo Dinic

6.1.2. **Ejecución del programa**

Para ejecutar el programa, es necesario ir a la terminal del computador y escribir el comando $aproximacionDinic < archivo\ de\ grafo >$.
La primera parte del comando de ejecución corresponde al nombre del programa binario, y la segunda, es el nombre del archivo de texto que se necesita como parámetro de entrada. Una vez ejecutado el programa en la terminal, aparecerá un mensaje de terminación exitosa.

6.1.3. **Parámetros de salida**

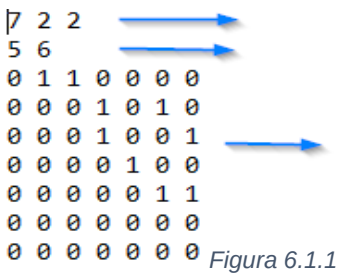
El programa, al ser iniciado, ejecutará una serie de comandos que concluirán en una representación matricial del grafo de la red multicast unisesión aproximada generada.

Esta matriz es guardada en un archivo de texto con la extensión .ffm, el cual sirve de entrada al programa de verificación y solución de la red multicast uni-sesión para determinar la configuración de salida del número de paquetes que corresponden al flujo máximo. La Figura 6.1.1.2 contiene la estructura del archivo de salida, cuya distribución está definida así: la primera línea contiene un arreglo de tres valores, donde la primera posición es el número de nodos, la segunda, el mínimo flujo máximo de la red multicast; y la última posición tiene el número de nodos sumideros. La línea 2, contiene un arreglo con los nodos codificadores. De la línea 3 en adelante está la matriz de adyacencia que representa el grafo multicast uni-sesión.

Ejemplo:

Nombre del archivo de salida: matriz_final_df_7grafo2fm1v2des.ffm

Contenido del archivo:



Primera línea: Núm. de nodos, Flujo máximo, Núm. de sumideros
Segunda línea: Nodos sumideros

Líneas siguientes: Matriz de la representación del grafo

Figura 6.1.1.6.1.2 Archivo de salida de algoritmo Dinic

Como se utilizan tres métodos diferentes para la eliminación de grafos, el programa genera tres archivos con la misma estructura, cada uno con la respectiva información resultante después de aplicar la eliminación, estos archivos se diferencian uno del otro por un prefijo en el nombre.

El prefijo ***df*** se utiliza para los nombres de archivos donde se hace el borrado por el primer camino encontrado, el prefijo ***dr*** para los nombres de archivos donde se lleva a cabo el borrado de los caminos en forma aleatoria, y el prefijo ***dl*** para los nombres de archivos resultantes de utilizar el método de eliminación por el camino más largo.

Ya en este punto, el programa de verificación [32] puede determinar si la aproximación de la red proporcionada por el método propuesto en este proyecto tiene o no solución

para Network Coding y también determina si la solución se logra con la salida de paquetes individuales o paquetes codificados.

6.2. EXPLICACIÓN DETALLADA DEL PROGRAMA DE LA SOLUCIÓN

Paso a paso se explica la solución propuesta para la problemática anteriormente planteada, describiendo el programa desarrollado y explicando los ajustes hechos a la implementación del algoritmo de Dinic para manejar redes multicast uni-sesión.

6.2.1. Primera parte: Lectura del archivo de entrada

Como se mencionó en el punto 6.1.1, el programa recibe un archivo de texto con el número de nodos, los nodos sumideros, y una matriz de adyacencia cuadrada como representación del grafo de comunicaciones.

Esta primera parte del algoritmo consta de la lectura del archivo de entrada, que se realiza a través de un desarrollo básico que lee y guarda en variables la información necesaria para ejecutar el programa.

De esta parte de la implementación se obtiene:

- La variable *sumideros,* la cual es definida como un vector en C++ y que posee los nodos sumideros.
- La variable *matrix,* la cual es definida como un vector de vectores en C++ y que tendrá la representación matricial del grafo.

6.2.2. Segunda Parte: Preparación para ejecutar algoritmo de Dinic

Como eje principal de esta investigación se encuentra el algoritmo de Dinic, el cual resuelve el problema de flujo máximo en un grafo dirigido con un nodo fuente y un nodo sumidero. Como en este proyecto se está trabajando con redes multicast (dos o más nodos sumideros), se necesita ejecutar el algoritmo de Dinic tantas veces como nodos sumideros estén presentes. Para esto se recorre el vector *sumideros* en un bucle*,* por ejemplo, si *sumideros* posee 3 nodos, el algoritmo de Dinic será ejecutado 3 veces.

El Algoritmo 2 muestra el seudocódigo del algoritmo Dinic con las pre y pos condiciones del desarrollo y se observan las variables más relevantes del método.

Como se ve en la Figura 6.2.2, el Algoritmo 2 tiene como variables principales la estructura de datos `Path`, en esta estructura se almacena, el flujo máximo de la red, y los caminos para cada uno de los segmentos del grafo correspondientes a cada nodo

sumidero. La variable `Matrix_final` contiene la información de todos los caminos validados, que conforman la aproximación de red.

Seudocódigo:

Algoritmo 2

Dinic: Función que contiene el segmento de código donde se ejecuta Dinic en todos los nodos sumideros, recibe diferentes parámetros y devuelve una matriz con la información final.

Entrada: **Disyuntos** (**true** o **false**), nodo fuente **s**, nodo origen **s1**,nodo destino ***t1***,Archivo de carga ***archivo***, nodo sumidero ***t*** .
Salida: Estructura ***Path***.

```
1 Sea n el grado de sumideros;
2 Sea Path la estructura asociada al resultado de Dinic;
3 Vector <Camino> Caminos
4 Para i = 1 hasta n
5   Camino Path=Dinic(false, 0, 0, 0, archivo,sumideros[i]);
6    Caminos=agregaraCaminos(Path);
7 Fin Para
8 Matrix_final=generarMatrix(Caminos)
```

Algoritmo 2 Algoritmo propuesto Dinic parte 1

En el Algoritmo 2, sección *Entradas* se encuentran estas variables: **Disyuntos** con valores true o false, es decir funciona como una bandera y se utiliza para eliminar los caminos no disyuntos, si está en ***true*** borra del camino que va desde el **nodo origen s1** al **nodo destino t1**; **nodo fuente s**, es el nodo fuente de la red multicast; **nodo origen s1** y **nodo destino t1,** forman el enlace que debe ser eliminado del grafo al estar incluido en (presenta una intersección) en otro camino, con lo cual se logran caminos disyuntos; **archivo** tiene el nombre y la ruta del archivo físico que contiene las estructuras de datos necesarias para ejecutar Dinic; y **nodo sumidero t**, es el nodo sumidero destino a donde se pretende enviar los paquetes.

Después de declarar las variables, se llama al algoritmo de Dinic para que devuelva una estructura compuesta por el flujo máximo y los caminos de cada segmento de grafo, todo esto se guarda en la variable `Path`. En este punto se obtiene un conjunto de caminos guardado en un arreglo.

El **nodo sumidero t** es representado por cada iteración de la variable *sumideros* dentro del bucle. El resultado final de la aproximación de la red multicast se almacena en una matriz llamada *Matrix_final*.

Para finalizar, en el Algoritmo 2 se observan dos funciones, la primera es la función `agregarCaminos`, que añade al vector Caminos todos los caminos obtenidos luego de ejecutar Dinic en cada nodo sumidero, y la función `generarMatrix` que genera la matrix final donde se encuentra la representación de la red multicast final.

6.2.3. **Tercera Parte: Estructura de salida del Algoritmo de Dinic**

Como los resultados de Dinic son un conjunto de variables, se necesita una estructura que lo soporte; C++ tiene dentro de sus componentes, objetos que permiten este tipo de uso llamados ***struct***, para este desarrollo se incorporó una variable **struct** y se le dio el nombre de ***Camino***, donde cada instancia tendrá los parámetros para generar la salida tras la ejecución de Dinic. En el Algoritmo 3 se puede ver la definición de esta estructura.

Algoritmo 3
Estructura Camino:

```
1 Camino  Path=Dinic(false, 0, 0, 0,source,sumideros[i]);
2 Estructura Camino
3   vector camino;
4   entero maxFlow;
5 Fin
```

Algoritmo 3 Algoritmo propuesto parte 2

La estructura `Camino` posee dos parámetros:

- `maxFlow` : Un entero que guarda el flujo máximo del segmento de grafo.
- `Camino`: Un vector de vectores que representa el camino encontrado. Por ejemplo, si el algoritmo de Dinic encuentra un camino del nodo 0 al 7: 0-3-5-7, en este vector se guarda [0,3,5,7]. Este vector tendrá más de un valor de este tipo.

6.2.4. **Cuarta Parte: Dentro del algoritmo de Dinic**

Entrando un poco más profundo al método Dinic mencionado en el Algoritmo 2, en el Algoritmo 4 se muestran los diferentes pasos que son llevados a cabo dentro de la mejora realizada para este trabajo de investigación.

Seudocódigo:

Algoritmo 4
Método Dinic Detallado

Entrada: ***Disyuntos*** (***true*** o ***false)***, nodo fuente **s**, nodo origen **s1**, nodo destino **t1**, Archivo de carga ***archivo***, nodo sumidero ***t***.
Salida: Estructura ***Path***.

```
1  Dinic(BuscarDisy,nFuente,nDisyF,nDisyD,source,sumidero)
2    llenarEstructuras (BuscarDisy,nDisyF,nDisyD,source);
3    Camino camino;
4    entero flow = 0;
5    mientrasQue sea verdadero
6         Si BFS es falso rompa el ciclo;
7         para i = 1 hasta sumidero
8           upTo[i] = 0;
9           mientrasQue sea verdadero
10              entero currFlow = DFS(nFuente, sumidero);
11              si currFlow es igual a 0 rompa el ciclo;
12              flow += currFlow;
13              path= agregaraCamino();
14          Fin mientraQue
15          Camino.camino = camino;
16          Camino.maxFlow=flow;
17         validarDisyunto();
18        Fin Para
19       Fin SI
20  Fin mientrasQue;
21  devuelva Camino;
22  FIN
```

Algoritmo 4 Algoritmo propuesto parte 3

Esta es una de las partes más importantes del proyecto, ya que se modificó el algoritmo de Dinic para poder conseguir el resultado esperado. Usualmente, el algoritmo de Dinic solo retorna el flujo máximo, pero en este caso se necesita, además del flujo máximo, los caminos que se encontraron con una búsqueda en anchura de todo el grafo con el algoritmo de BFS. Si existen caminos, el siguiente paso es verificar a través de una búsqueda en profundidad con el algoritmo DFS si el nivel de los caminos encontrados cumple con la definición dada en Dinic: $E_f: dist(v) = dist(u) + 1$. Si esto ocurre, los caminos serán guardados, sino, serán desechados; es decir, no serán incluidos caminos con un nivel menor que su nivel padre.

El flujo máximo depende de los caminos encontrados, pero esta no es la única condición, también se debe tener en cuenta si estos caminos son válidos para aplicar Network Coding, por lo tanto, los caminos encontrados por los algoritmos de BFS y DFS deben ser disyuntos. Se dice que dos caminos son disyuntos cuando no tienen vértices en común; por esta razón, en el programa se realiza la eliminación de estos caminos a través del bloqueo o la eliminación de los vértices comunes, para que al correr BFS y DFS otra vez, Dinic no los tome en cuenta. Todo esto se logra utilizando el método `validarDisyunto()`.

6.2.5. **Quinta Parte: Conclusión después de obtener resultados separados por segmento de grafo**

La forma de obtener el mínimo flujo máximo de toda la red, es tomando todos los flujos máximos encontrados, luego hallando el menor flujo, y a través de los diferentes métodos de eliminación descritos en este trabajo, igualar todos los flujos a este mínimo obtenido. En el Algoritmo 5, se muestran los diferentes métodos de eliminación usados para esta investigación.

Ejemplo:

maxFlows = [4,3,4,3], este vector contiene los flujos máximos correspondiente a cada sumidero de la red. Se comparan los valores de los flujos entre ellos y se escoge el menor, que es 3, pero no basta con decir que el flujo máximo general es 3, sino que, además, se deben ajustar las matrices obtenidas por cada segmento de grafo, para que aquellas que tengan un flujo máximo mayor que el seleccionado tengan un flujo igual al mínimo flujo máximo general.

El criterio para la eliminación se dividió en tres tipos, esto con el fin de evaluar el impacto de la eliminación de caminos. El algoritmo 5 contiene el seudocódigo de estas eliminaciones.

1 Eliminar el camino más largo
2 Eliminar el primer camino que encuentre
3 Eliminar un camino aleatorio

Algoritmo 5

Segmento de código para buscar el mínimo flujo y podar el grafo de acuerdo con los criterios de eliminación. 1: Borrar el primero, 2: Borrar el más largo, 3: Borrar aleatorio

```
1 matrix_final=generarMatriz();
2  int min=minMaxFlow(matrix_final);
3  needPoda(1, min);
4  needPoda(2, min);
5  needPoda(3, min);
6  funcion needPoda(int opcionBorrado,int min) {
7    si(opcionBorrado == 1)//Elimine el primero
8        deleteFirst(min);
9    sino
10      si(opcionBorrado == 2)//elimine el camino más largo
11         deleteLongest(min);//elimine aleatoriamente
12      sino
13       deleteRamdom(min);
15     fin si
15   fin si
16 matrix_final=generarMatriz();
17 GenerarArchivoSalida(matrix_final)
```

Algoritmo 5 Algoritmo propuesto parte 4

Se observa que después de realizar la eliminación de caminos no necesarios, se genera la matriz final con la unión de todos los caminos resultantes; en esta matriz se guarda la representación de una red multicast uni-sesión a la cual sería posible aplicarle la técnica de Network Coding.

Para finalizar, el método GenerarArchivoSalida, definido en el algoritmo 5 prepara y crea el archivo de salida mencionado en el punto 6.2.3 que será el insumo para el algoritmo descrito en la investigación [32] con el fin de verificar que la red es apta para la aplicación de Network Coding.

7. EJEMPLOS Y RESULTADOS

En este capítulo se presentan los ejemplos y los resultados luego de ejecutar el algoritmo de Dinic extendido. Para esto se necesita una red de comunicaciones en el formato mencionado en 6.1.1, con la cual se obtiene la aproximación de red multicast uni-sesión, que después se verifica si tiene solución o no para la aplicación de Network Coding.

En esta sección se muestran tres ejemplos con sus respectivas representaciones de entrada y de salida, y se explican los resultados del algoritmo de verificación.

7.1. EJEMPLO RED 10grafo2fm1v3des

Número de nodos: 10. En la Figura 7.1.1 se observa la red de entrada con la que se resolverá este ejemplo, Las figuras 7.1.2, 7.1.3 y 7.1.4 representan los caminos desde el nodo fuente hacia los diferentes sumideros luego de haber aplicado el algoritmo de Dinic. La figura 7.1.6 muestra el grafo con la aproximación de red multicast uni-sesión resultante.

Red de entrada:

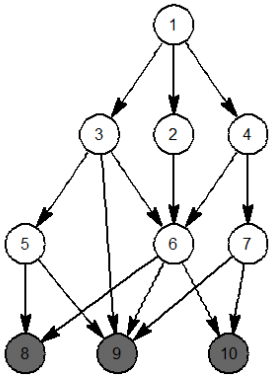


Figura 7.1.1 Red de entrada 10grafo2fm1v3des

En las Figuras 7.1.2, 7.1.3 y 7.1.4, se puede observar el orden como se ejecuta el programa desarrollado en este trabajo para calcular la aproximación de red. Se observa que el algoritmo encuentra los caminos para llegar del nodo fuente hasta el primer nodo sumidero (Figura 7.1.2). De igual forma, se realiza para el segundo sumidero (Figura 7.1.4) y, por último, para el tercer nodo sumidero (Figura 7.1.3). Con lo anterior, se obtienen tres segmentos de grafos, uno por cada sumidero, los cuales serán el insumo para la formación de la red multicast.

Redes intermedias:

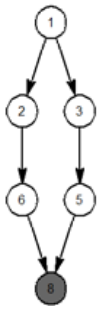


Figura 7.1.2 10grafo2fm1v3des: Red unicast hacia el nodo 8

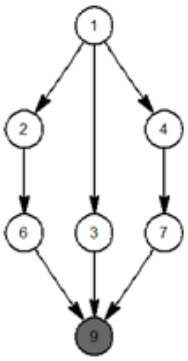


Figura 7.1.3 10grafo2fm1v3des: Red unicast hacia el nodo 9

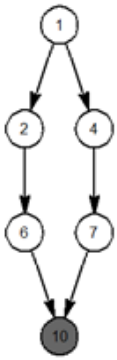


Figura 7.1.4 10grafo2fm1v3des: Red unicast hacia el nodo 10

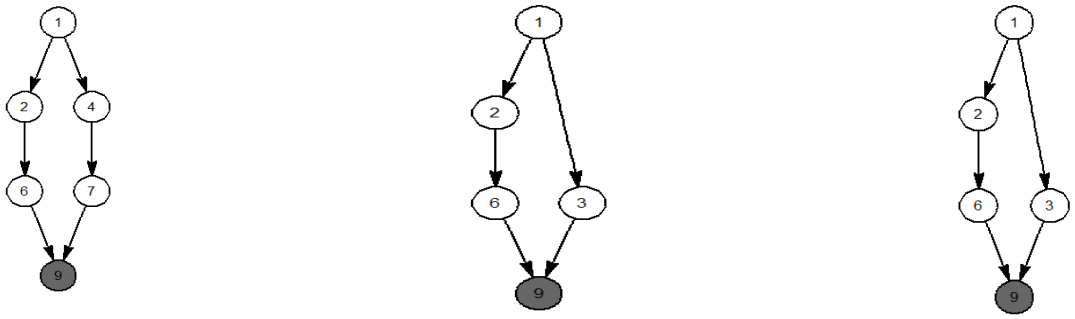
En esta red se encuentra un comportamiento particular. En la sección 6.2.5, donde se explicó una parte del programa desarrollado en este proyecto, se menciona el caso en el

cual los flujos máximos obtenidos para cada segmento de grafo no son iguales; como es notorio en la figura 7.1.3, donde se observa un flujo máximo diferente a los otros segmentos. Es decir, se puede encontrar que, para el primer segmento de grafo en la Figura 7.1.2, el flujo máximo es 2; para el segmento en la Figura 7.1.4, el flujo máximo es 2 y para el segmento en la Figura 7.1.3, el flujo máximo es 3.

Cuando esto ocurre, se deben igualar los flujos máximos en los segmentos de grafo al mínimo valor encontrado, en este caso 2 (valor del mínimo flujo máximo).

Por lo tanto, el siguiente paso, será eliminar tantos caminos como se necesiten para poder igualar el flujo máximo global.

De acuerdo con el trabajo elaborado se consideran tres tipos de eliminaciones diferentes, como se observa en la Figura 7.1.5. La Figura 7.1.5a muestra la eliminación del primer camino que encuentra el algoritmo en el grafo de 7.1.3, la Figura 7.1.5b muestra la eliminación del camino más largo y la Figura 7.1.5.c muestra la eliminación aleatoria de los caminos necesarios para alcanzar el flujo máximo común.



(a) (b) (c)

Figura 7.1.5 Eliminación de caminos mayores al flujo máximo general de 10grafo2fm1v3des: Red unicast hacia el nodo 9 - (a) Eliminar el primero, (b) Eliminar el más largo, (c) Eliminar aleatorio

Los caminos eliminados fueron de 1→3→9 en la Figura 7.1.6 (a) y 1→4→7→ 9 en las Figuras 7.1.6 (b) y 7.1.6 (c), dejando todos los segmentos de grafo con el mismo flujo máximo que el mínimo encontrado.

En la figura 7.1.6 se muestran las aproximaciones de red multicast uni-sesión y la aplicación de Network Coding para la entrega simultánea de los paquetes originados en el nodo fuente 1.

Red multicast uni-sesión:

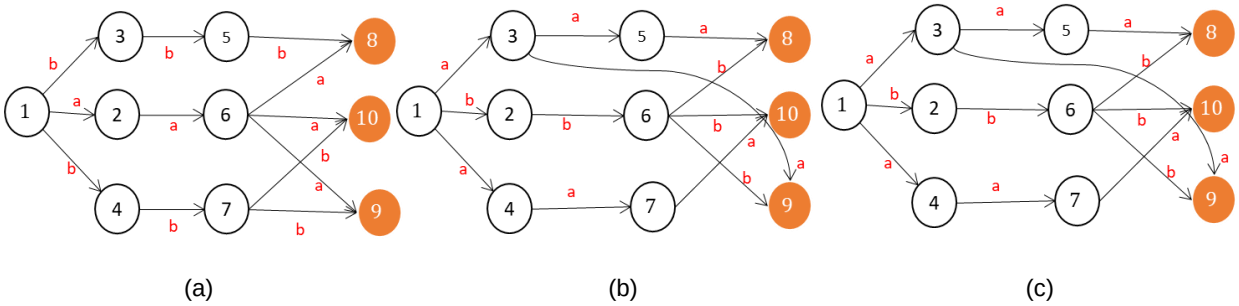


Figura 7.1.6 Red multicast de mínimo flujo máximo para 10grafo2fm1v3des (a) Eliminar el primero, (b) Eliminar el más largo, (c) Elimina aleatorio

Con el ejercicio desarrollado, se puede observar que el programa de verificación concluyó que esta es una aproximación de red funcional para aplicación de Network Coding, llevando así a un caso exitoso para la propuesta desarrollada de este trabajo. Esto se puede comprobar observando la Figura 7.1.6, donde se especifican los paquetes que se envían por cada enlace y los paquetes que se reciben al final. Los tres nodos sumideros reciben a y b con la técnica de Network Coding.

Podemos entonces decir, que esta es una aproximación de red a la que el algoritmo de verificación respondió que la red multicast uni-sesión se soluciona para Network Coding permitiendo la entrega simultánea de los paquetes enviados desde el nodo fuente. La solución se construyó sobre la base de entrega de paquetes individuales desde el nodo fuente. De la misma manera, esta red multicast también permite la solución con paquetes codificados.

7.2. **EJEMPLO RED 11grafo2fm3v3des**

Número de nodos: 11. En la Figura 7.2.1 se observa la red de entrada con la que se realizará este ejemplo, Las Figuras 7.2.2, 7.2.3 y 7.2.4 representan los caminos desde el nodo fuente hacia los diferentes sumideros luego de haber aplicado el algoritmo de Dinic. La Figura 7.2.6 muestra el grafo con la aproximación de red multicast uni-sesión resultante.

Red de entrada:

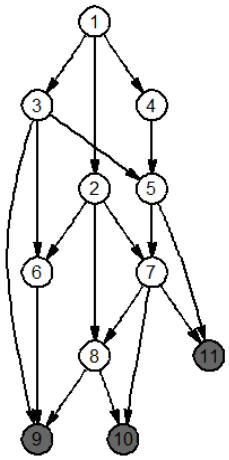


Figura 7.2.1 Red de entrada 11grafo2fm3v3des

En las Figuras 7.2.2, 7.2.3 y 7.2.4, se puede observar el orden como se ejecuta el programa desarrollado en este trabajo para calcular la aproximación de red. Se observa que el algoritmo encuentra los caminos para llegar del nodo fuente hasta el primer nodo sumidero (Figura7.2.2). De igual forma, se realiza para el segundo sumidero (Figura7.2.3) y, por último, para el tercer sumidero (Figura7.2.4). Con lo anterior, se obtienen tres segmentos de grafos, uno por cada sumidero, los cuales serán el insumo para la formación de la red multicast.

Redes Intermedias:

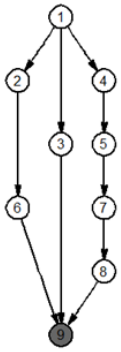


Figura 7.2.2 2 11grafo2fm3v3des: Red unicast hacia el nodo 9

Figura 7.2.3 11grafo2fm3v3des: Red unicast hacia el nodo 10

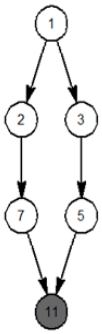


Figura 7.2.4 11grafo2fm3v3des: Red unicast hacia el nodo 11

En la Figura 7.2.3 se observa que el flujo máximo del grafo es 1, y este se convierte en el mínimo flujo máximo de la red, este comportamiento no se esperaba, por lo que será necesario recrear el algoritmo para entender la respuesta obtenida.

A partir del grafo G de la Figura 7.2.5a, se buscan los caminos desde 1 hasta 10, el primer camino encontrado es, 1→2→7→10, como se observa en el grafo G_L de la Figura 7.2.5b coloreado en naranja. Siguiendo los pasos de Dinic, se construye el árbol de niveles G_L de la Figura 7.2.5c, verificando que este camino hallado efectivamente es apto. Luego al realizar la segunda iteración ilustrada en la figura 7.2.6, se observa que en el grafo de niveles G_L ningún otro enlace llega hasta 10 desde el nivel 0, esto debido a que en Dinic solo es posible ir a un nivel superior de navegación. Lo anterior conlleva afirmar que solo existe un solo camino de $1 \rightarrow 10$.

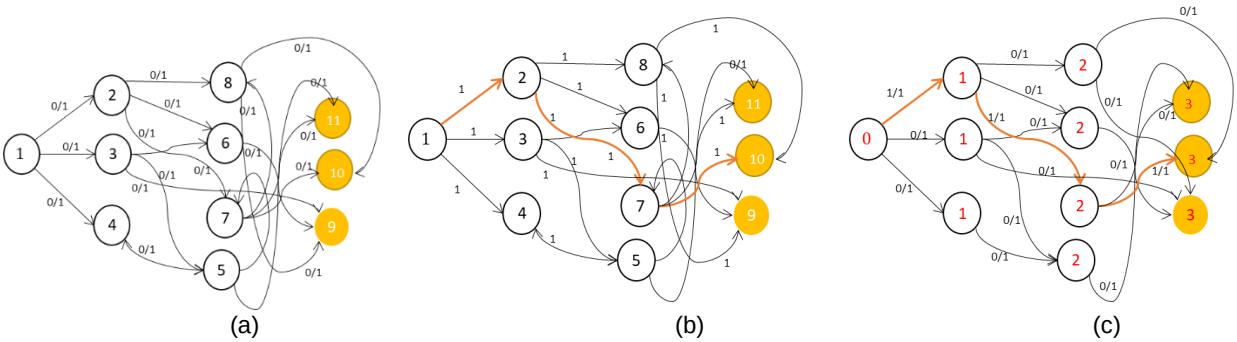


Figura 7.2.5 Red 11grafo2fm3v3des: (a) Grafo original, (b) Grafo de trayectoria de aumento, (c) Grafo de Nivel, primera iteración

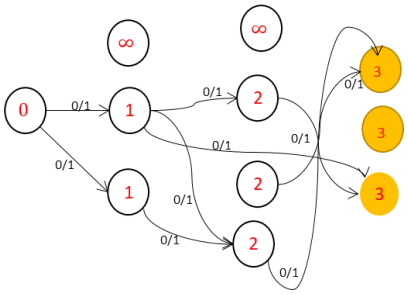


Figura 7.2.6 Grafo de niveles de Red unicast hacia el nodo 10, segunda iteración

El concepto de bloqueo de flujo hace que el comportamiento de Dinic sea diferente, y es posible que esto sea una desventaja para la aproximación de la red en comparación con otros algoritmos que tienen el mismo objetivo. En esta solución, el flujo máximo es 1, dado que el grafo de nivel no encuentra otra ruta apta para aplicar el algoritmo como es evidente en la Figura 7.2.6, y debido a que en Network Coding es necesario que llegue más de un paquete para hacer las operaciones de decodificación, la solución mostrada no tendrá validez para la aplicación de la técnica

Se observa en las redes de las Figuras 7.2.2, 7.2.3 y 7.2.4 que los tres segmentos de grafo obtenidos tienen un flujo máximo diferente. En el primer segmento de grafo en la Figura 7.2.2, el flujo máximo es 3; en el segundo segmento en la Figura 7.2.3, el flujo máximo es 1; y en la Figura 7.2.4, el flujo máximo es 2.

Cuando esto ocurre, se deben igualar los flujos máximos en los segmentos de grafo al mínimo valor encontrado, en este caso 1 (valor de flujo máximo mínimo).

Por lo tanto, el siguiente paso, será eliminar tantos caminos como se necesiten para poder igualar el flujo máximo global, y por último llevar a cabo la mezcla de los grafos

unicast resultantes para obtener la red multicast de mínimo flujo máximo como se observa en la Figura 7.2.7.

Red multicast uni-sesión:

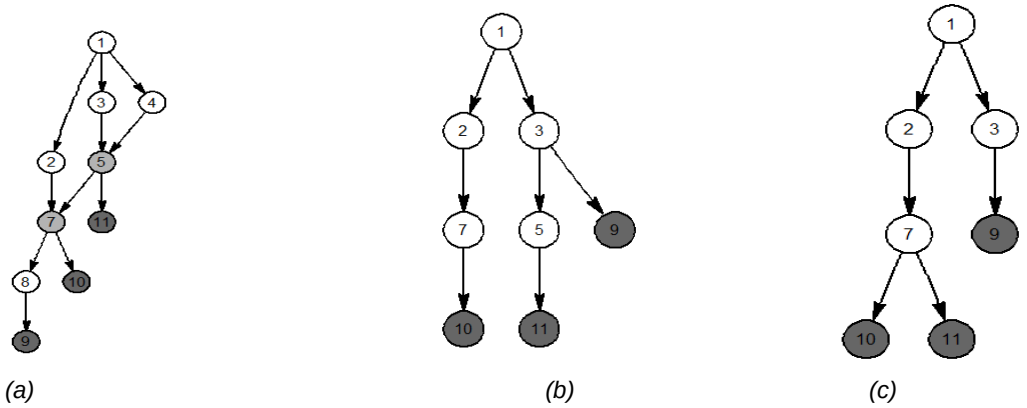


Figura 7.2.7 Red multicast de mínimo flujo máximo para 11grafo2fm3v3des (a) Elim. Primer Camino; (b) Elim camino más largo; (c) Elim camino aleatorio

En la red multicast uni-sesión de la Figura 7.2.7, se puede observar que el flujo máximo permitido es 1, por lo tanto, solo será posible recibir un paquete por sumidero; en efecto, esta es una red mejorada, pero no es útil para el propósito de esta investigación.

7.3. **EJEMPLO RED 12grafo2fm1v3des**

Número de nodos: 12. En la Figura 7.3.1 se observa la red de entrada con la que se resolverá este ejemplo, Las Figuras 7.3.2, 7.3.3 y 7.3.4 representan los caminos desde el nodo fuente hacia los diferentes sumideros luego de haber aplicado el algoritmo de Dinic. La Figura 7.3.6 muestra el grafo con la aproximación de red multicast uni-sesión resultante.

Red de entrada:

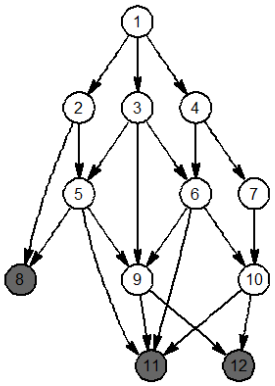


Figura 7.3.1 Red de entrada 12grafo2fm1v3des

En las Figuras 7.3.2, 7.3.3 y 7.3.4, se puede observar el orden como se ejecuta el programa desarrollado en este trabajo para calcular la aproximación de red. Se observa que el algoritmo encuentra los caminos para llegar del nodo fuente hasta el primer nodo sumidero (Figura 7.3.2). De igual forma, se realiza para el segundo sumidero (Figura 7.3.3) y, por último, para el tercer nodo sumidero (Figura 7.3.4). Con lo anterior se obtienen tres segmentos de grafos, uno por cada sumidero, los cuales serán el insumo para la formación de la red multicast.

Redes intermedias:

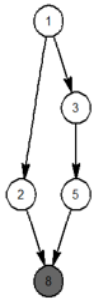


Figura 7.3.2 12grafo2fm1v3des: Red únicast hacia el nodo 8

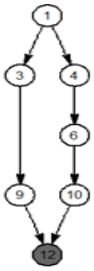


Figura 7.3.3 12grafo2fm1v3des: Red unicast hacia el nodo 12

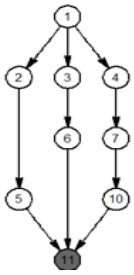


Figura 7.3.4 12grafo2fm1v3des: Red unicast hacia el nodo 11

Se observa que las redes de las Figuras 7.3.2 y 7.3.4 tienen un flujo máximo de 2, diferente al segmento de red de la Figura 7.3.3 que tiene un flujo máximo de 3.

En el grafo de la Figura 7.3.3 se necesita, entonces, hacer una poda o eliminar un camino para que resulte en un grafo con el mínimo flujo máximo general. En la Figura 7.3.5 se muestran los resultados luego de utilizar los tres métodos propuestos en esta investigación para la eliminación de segmentos.

En la Figura 7.3.5, se muestra el grafo resultante después de la eliminación del primer camino, la Figura 7.1.5b muestra el grafo resultante después de la eliminación del camino más largo, y la Figura 7.1.5c muestra que el grafo resultante fue, coincidencialmente, el mismo de la Figura 7.1.5b

Se puede ver en estas dos figuras que hay un nodo en común entre las redes generadas, al cual le llegan paquetes de nodos distintos, el nodo 6. Quiere decir que por este nodo pasarán los paquetes enviados desde dos nodos simultáneamente y, que este nodo tendrá que encargarse de la codificación de paquetes para poder hacer el re-envío del paquete resultante en un solo instante de tiempo en forma codificada y no dos, como en el enrutamiento clásico.

Red multicast uni-sesión:

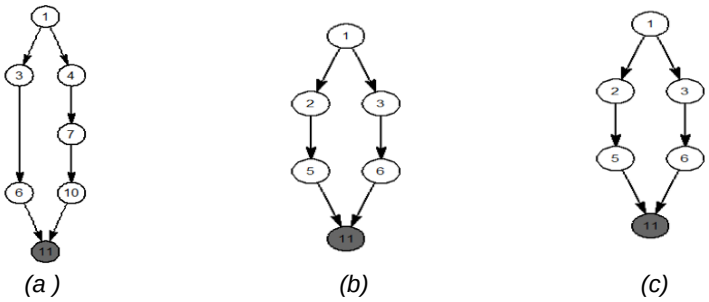


Figura 7.3.5 12grafo2fm1v3des: Red unicast hacia el nodo 11(a) Eliminación primer camino, (b) Eliminación Camino más largo y (c) Eliminación aleatoria.

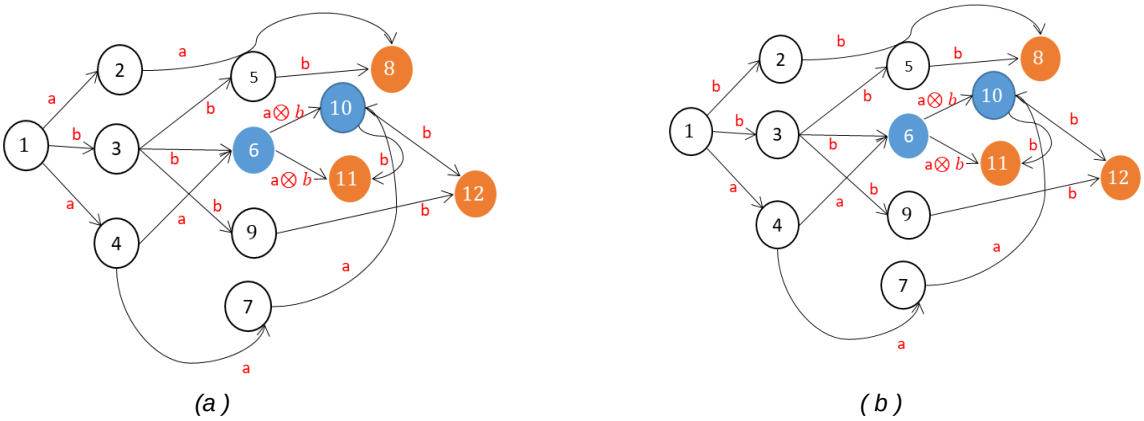


Figura 7.3.6 Red multicast de mínimo flujo máximo para 12grafo3fm1v3des (a) Eliminación por primer camino encontrado, paquetes enviados [a,b,a]. (b) Eliminación por primer camino encontrado, paquetes enviados [b,b,a].

En la Figura 7.3.7 se observa que esta aproximación de red multicast uni-sesión, calculada por el programa, no cumple con los requisitos para aplicar Network Coding, debido a que los paquetes no llegan completos a los nodos sumideros. Por lo tanto, no es apta y se considera que no tiene solución.

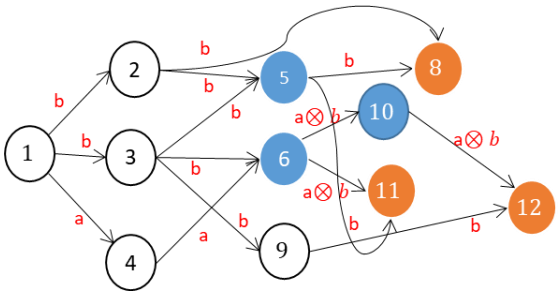


Figura 7.3.7 Red calculada 12grafo3fm1v3des con el grafo resultante de eliminar el camino más largo y camino aleatorio, paquetes enviados [b,b,a].

7.4. **RESULTADOS GENERALES**

En este trabajo se utilizaron como ejemplos de pruebas, 37 redes de comunicaciones representadas en grafos de diferentes formas (artículos científicos y redes construidas propias dentro del proyecto), sobre las cuales se ejecutó el programa desarrollado, concluyendo con una aproximación de red multicast uni-sesión.

La red multicast uni-sesión resultante tiene menos enlaces que la red inicial de carga, lo que significa que se han encontrado optimizaciones en los caminos para envío de paquetes en forma más rápida y directa, también que los caminos que no son necesarios

han sido eliminados de la red, en algunos casos la red queda con nodos codificadores, desde donde se pueden enviar más de un paquete a la vez, al ser combinados linealmente los paquetes entrantes al nodo.

Con estas aproximaciones de redes obtenidas, se utilizó el programa de verificación para clasificarlas. Estas fueron clasificadas de la siguiente manera:

- Solución con paquetes individuales (incluyen solución con paquetes codificados).
- Solución con paquetes codificados exclusivamente.
- Sin solución.

En la Tabla 7.4.1 se puede encontrar el resultado de aplicar el algoritmo de verificación para cada una de las aproximaciones de redes multicast obtenidas.

Esta tabla, contienen los resultados que provee el algoritmo de verificación, el cual responde para cuáles aproximaciones de redes se encuentra una solución a la aplicación de la técnica de Network Coding sobre redes multicast uni-sesión. También para cuáles aproximaciones de redes fue posible encontrar una solución con paquetes individuales; es decir, sin el nodo fuente como nodo codificador; y cuáles aproximaciones de red resultaron con solución con paquetes codificados; es decir, aquellas redes que utilizan el nodo fuente como nodo codificador.

También se encontraron aproximaciones de redes a las cuales fue imposible hallar una solución.

	Dinic_PRI			Dinic_ML			Dinic_RD		
Archivo txt	**S**	**C**	**Co**	**S**	**C**	**Co**	**S**	**C**	**Co**
7grafo2fm1v2des	X	x	1	X	x	1	x	x	1
7grafo2fm2v3des		x	0		x	0		x	0
7grafo2fm3v3des	X	x	0		x	0		x	0
9grafo2fm1v3des	X	x	1	X	x	1	x	x	1
10grafo2fm1v3des	X	x	0	X	x	0	x	x	0
11grafo2fm1v2des	X	x	1	X	x	1	x	x	1
11grafo2fm1v3des		x	2	X	x	2	x	x	2
11grafo2fm1v6des			0			0			0
11grafo2fm2v3des	X	x	1	X	x	1	x	x	1
11grafo2fm3v3des		x	2	X	x	3	x	x	2
11grafo2fm4v3des	X	x	0	X	x	0	x	x	0
11grafo4fm1v2des	X	x	1	X	x	1	x	x	1
12grafo2fm1v3des			1		x	3		x	3
12grafo2fm2v3des	X	x	0	X	x	1	x	x	1
12grafo3fm1v3des	X	x	0	X	x	0	x	x	0
13grafo1fm2v3des	X	x	0		x	0	x	x	0
13grafo2fm1v5des	X	x	0	X	x	0	x	x	0
13grafo3fm1v4des		x	0		x	0	x	x	0
13grafo3fm2v4des	X	x	0	X	x	0	x	x	0
15grafo2fm1v4des	X	x	1	X	x	1	x	x	1
18grafo2fm1v6des		x	1		x	1	x	x	1
18grafo2fm2v6des	X	x	1	X	x	1		x	1
20grafo2fm1v4des	X	x	1	X	x	1	x	x	1
20grafo4fm1v3des	X	x	4	X	x	4	x	x	3
20grafo4fm2v3des		x	5			4	x	x	3
20grafo5fm1v3des	X	x	3	X	x	3	x	x	3
23grafo6fm1v2des	X	x	3	X	x	3	x	x	3
27grafo4fm1v5des	X		9		x	8		x	9
31grafo2fm1v7des		x	3	X	x	0		x	1
35grafo2fm1v6des			4		x	3			4
38grafo2fm1v6des			6			3		x	4
38grafo3fm1v7des		x	6		x	4			5
45grafo6fm1v3des			4			4			4
46grafo6fm1v4des			9			9			9
47grafo6fm1v5des			10			10			10
54grafo3fm1v7des		x	9		x	9	x	x	8
Total	**20**	**28**	**89**	**20**	**30**	**82**	**23**	**30**	**83**

Tabla 7.4.1 Resumen de resultados

Para la solución de borrar el primer camino hallado:

- El porcentaje de aproximaciones de redes que se encontraron con solución con paquetes individuales fue del 54.05%.

- El porcentaje de aproximaciones de redes que se encontraron con solución con paquetes codificados exclusivamente fue del 21.62%.

- El porcentaje de aproximaciones de redes que se encontraron sin solución fue del 21.62%.

- El promedio de nodos codificadores fue de 2.4

Por tanto, de las 37 redes probadas, el 75.7% tiene por lo menos una solución con Network Coding después de aplicar Dinic.

Para la solución de borrar el camino más largo hallado:

- El porcentaje de aproximaciones de redes que se encontraron con solución con paquetes individuales fue del 54.05%.

- El porcentaje de aproximaciones de redes que se encontraron con solución con paquetes codificados exclusivamente fue del 18.9%.

- El porcentaje de aproximaciones de redes que se encontraron sin solución fue del 18.9%.

- El promedio de nodos codificadores fue de 2.2

Por tanto, de las 37 redes probadas, el 81.1% tiene por lo menos una solución con Network Coding después de aplicar Dinic.

Para la solución de borrar los caminos aleatoriamente.

- El porcentaje de aproximaciones de redes que se encontraron con solución con paquetes individuales fue del 62.16%.

- El porcentaje de aproximaciones de redes que se encontraron con solución con paquetes codificados exclusivamente fue del 18.9%.

- El porcentaje de aproximaciones de redes que se encontraron sin solución fue del 21.62%.

- El promedio de nodos codificadores fue de 2.2

Por tanto, de las 37 redes probadas, el 81.1% tiene por lo menos una solución con Network Coding después de aplicar Dinic.

8. CONCLUSIÓN

La solución propuesta, parte de redes de comunicaciones con múltiples enlaces y nodos y, con conocimiento de los nodos fuente y sumideros de un grupo multicast uni-sesión, y a partir de esto, genera redes multicast uni-sesión que son verificadas para determinar si son capaces de dar una solución para la correcta aplicación de la técnica de Network Coding.

Como se ha mencionado anteriormente, los posibles resultados que se pueden obtener de las aproximaciones de redes son revisiones realizadas por un algoritmo de verificación. De los resultados de la revisión, se puede concluir que la red multicast uni-sesión generada por el programa, tiene solución con paquetes individuales, solución con paquetes codificados o no tiene solución.

Después de ejecutar el programa desarrollado en este trabajo sobre una cantidad considerable de redes de comunicaciones, con base en los resultados, se puede concluir que:

- El algoritmo de Dinic ajustado demuestra que permite obtener redes multicast uni-sesión a partir de redes de comunicaciones, en un alto porcentaje de solución de transmisiones de paquetes, desde el nodo fuente hasta el conjunto de nodos sumideros aplicando la técnica de Network Coding.

- Se demuestra que, con el programa propuesto, se tiene una alternativa importante para aproximar redes de comunicaciones a redes multicast uni-sesión con un alto porcentaje de solución al envío de paquetes individuales y por consiguiente al envío de paquetes codificados.

- El número de soluciones correspondientes al envío de paquetes codificados exclusivamente es inferior al de paquetes individuales. Es casi la mitad.

- El algoritmo de DFS dentro del algoritmo ajustado de Dinic, permite encontrar caminos óptimos dentro del segmento red unicast dado. Estos caminos después de ser verificados, dan aviso al algoritmo de BFS de si se necesita otro camino, ya que alguno no cumplió con las características necesarias. Concluyendo que la unión DFS y BFS proporcionan caminos, muchos de ellos disyuntos desde la primera iteración, dando lugar a generar caminos óptimos con alta probabilidad de aplicar técnicas de Network Coding.

- En general, el programa desarrollado, ofrece aproximaciones de red, que según el algoritmo de verificación, poseen distintas soluciones hasta en un 81% de los casos presentados, y solo no ofrece solución en el 19% de los casos.

- Cuando se implementó la técnica de borrado aleatoriamente, se obtuvo mejor resultado que con las otras técnicas de borrado, porque se puede concluir que este camino es el más acertado en el experimento.

- Los métodos de eliminación de ruta más larga y aleatoriamente tienen menos nodos codificadores, estos presentan mejor rendimiento debido a que el tiempo que deben invertir en la codificación menor.

La importancia de este trabajo radica en haber hallado un método basado en un algoritmo de flujo máximo para redes unicast sobre el algoritmo de Dinic, que permita calcular o reducir una red de comunicaciones general a una red multicast uni-sesión y que sea solucionable para la entrega de paquetes simultáneos a través de la técnica matemática de Network Coding.

Para poder comparar este trabajo con otros de la misma naturaleza, se propone ejecutar proyecto similar donde se apliquen otros algoritmos de flujo máximo en redes únicast.

También es importante trabajar sobre la forma de mejorar el algoritmo propuesto para poder aumentar el número de redes solucionables con NC.

BIBLIOGRAFÍA

- [1] M. Celebiler; G. Stette "On Increasing the Down-Link Capacity of a Regenerative Satellite Repeater in Point-to-Point Communications". *in Proceedings of the IEEE.* vol. 66, pp.98–100, Jan. 1978.
- [2] H. P. Rubén., H. H. Gonzalo.,*Comunicaciones multicast* [Online]. Available: https://www.uaeh.edu.mx/scige/boletin/huejutla/n9/r1.html
- [3] C H. Papadimitriou, K Steiglitz “Combinatorial Optimization: Algorithms and Complexity. Upper Saddle River”, in PrenticeHall, Inc., 1982.
- [4] M. Medard, A. Sprintsonm, *Network Coding: Fundamentals and Applications.* Massachusetts: Academic Press, 2012.
- [5] R. W. Yeung, "Network Coding: A Historical Perspective," in *Proceedings of the IEEE*, vol. 99, no. 3, pp. 366-371, March 2011.
- [6] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, “Network information flow,” IE ” IEEE Trans. Inform. Theory, vol. 46, no. 4, pp. 1204–1216, July 2000.
- [7] A. Paramanathan, M.V. Pedersen, D.E. Lucani, F.H.P. Fitzek y M. Katz. «Lean and mean: network coding for commercial devices». En: Wireless Communications, IEEE 20.5 (oct. de 2013), págs. 54-61. issn: 1536-1284. doi: 10.1109/MWC.2013.6664474.
- [8] L. H. Sahasrabuddhe y B. Mukherjee, «Multicast routing algorithms and protocols: A tutorial», IEEE Netw., vol. 14, n.o 1, pp. 90-102, 2000
- [9] Deering, *Multicast routing in internetworks and extended LANs*, vol. 18. ACM, 1988.
- [10] J. Márquez, “Códigos De Red Aplicados Sobre Múltiples Flujos De Datos En Transmisión Multicast,” 2018 *Universidad del Norte.*
- [11] L.R Ford,Jr, and Fulkerson, "Maximum flow through a network", in Canadian Journal of Mathematics 8, 399-404.
- [12] Dinic, E.A. (1970). "Algorithm for solution of a problem of maximum flow in networks with power estimation", Soviet Mathematics Doklady 1 1, 1277-1280
- [13] Edmonds, J., and Karp, R.M. (1972), Theoretical improvements in algorithmic efficiency for network flow problems, Journal of ACM 19, 248-264.
- [14] E. A. DINIC, Algorithm for solution of a problem of maximum flow in a network with power estimation. Soviet Math. Dokl., 11 (1970), pp. 1277-1280.]
- [15] V. M. MALHOTRA, M. P. KUMAR AND S. N. MAHESHWARI, An O(\ K|3) algorithm for finding maximum flows in networks. Inform. Process Lett., 7 (1978), pp. 277-278.
- [16] A. V. Goldberg y R. E. Tarjan, «A new approach to the maximum-flow problem», J. ACM JACM, vol. 35, n.o 4, pp. 921–940, 1988.
- [17] B. V. Cherkassky y A. V. Goldberg, «On implementing the push—relabel method for the maximum flow problem», Algorithmica, vol. 19, n.o 4, pp. 390-410, 1997.

- [18] Goldberg, A.V., and Tarjan, R.E. (1986), "A new approach to the maximum flow problem", Proceedings of the 18th ACM Symposium on the Theory of Computing, 136-146. Full paper in Journal of ACM 35 (1988), 921-940.
- [19] R. Ahlswede, Ning Cai, S.-Y.R. Li y R.W. Yeung. «Network information flow». En: Information Theory, IEEE Transactions on 46.4 (jul. de 2000), págs. 1204-1216. issn: 0018-9448. doi: 10.1109/18.850663.
- [20] Christos H. Papadimitriou y Kenneth Steiglitz. Combinatorial Optimization: Algorithms and Complexity. Upper Saddle River, NJ, USA: PrenticeHall, Inc., 1982. isbn: 0-13-152462-3.
- [21] S.-Y.R. Li, R.W. Yeung y Ning Cai. «Linear network coding». En: Information Theory, IEEE Transactions on 49.2 (feb. de 2003), págs. 371-381. issn: 0018-9448. doi: 10.1109/TIT.2002.807285.
- [22] Peter Sanders, Sebastian Egner y Ludo Tolhuizen. «Polynomial Time Algorithms for Network Information Flow». En: Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures. SPAA '03. San Diego, California, USA: ACM, 2003, págs. 286-294. isbn: 1-58113-661-7. doi: 10.1145/777412.777464. url: http://doi.acm. org/10.1145/777412.777464.
- [23] Tracey Ho, Ralf Koetter, Muriel Médard, David R. Karger y Michelle Effros. «The benefits of coding over routing in a randomized setting». En: In Proceedings of 2003 IEEE International Symposium on Information Theory. 2003.
- [24] Tracey Ho, Ralf Koetter, Muriel Médard, David R. Karger y Michelle Effros. «The benefits of coding over routing in a randomized setting». En: In Proceedings of 2003 IEEE International Symposium on Information Theory. 2003.
- [25] D. Ferreira, L. Lima y J. Barros. «NECO: NEtwork COding simulator.» En: SimuTools. Ed. por Olivier Dalle, Gabriel A. Wainer, L. Felipe Perrone y Giovanni Stea. ICST, 15 de mayo de 2009, pág. 52. isbn: 978-963-9799-45-5. url: http://dblp.uni-trier.de/db/conf/ simutools/simutools2009.html#FerreiraLB09.
- [26] Raymond W. Yeung. «Avalanche: A Network Coding Analysis». En: Communications in Information and Systems (2007), págs. 353-358.
- [27] Tin-Yu Wu, S. Guizani, Wei-Tsong Lee y Po-Chang Huang. «An enhanced structure of layered forward error correction and interleaving for scalable video coding in wireless video delivery». En: Wireless Communications, IEEE 20.4 (ago. de 2013), págs. 146-152. issn: 1536-1284. doi: 10.1109/MWC.2013.6590062.
- [28] A.G. Dimakis, P.B. Godfrey, Y. Wu, M.J. Wainwright y K. Ramchandran. «Network Coding for Distributed Storage Systems». En: Information Theory, IEEE Transactions on 56.9 (sep. de 2010), págs. 4539-4551. issn: 0018-9448. doi: 10.1109/TIT.2010.2054295.
- [29] Baochun Li y Di Niu. «Random Network Coding in Peer-to-Peer Networks: From Theory to Practice». En: Proceedings of the IEEE 99.3 (mar. de 2011), págs. 513-523. issn: 0018-9219. doi: 10.1109/JPROC. 2010.2091930.
- [30] Network Coding for 5G Network and D2D Communication

- [31] L. Wu and K. Curran, "Practical Network Coding Scheme Based on Maximum Flow Combination and Coding Node Identification," *2010 International Conference on Internet Technology and Applications*, Wuhan, 2010, pp. 1-4.
- [32] J. Márquez, I. Gutiérrez, S. Valle, M. Falco, “Packet Output and Input Configuration in a Multicasting Session Using Network Coding,” 2017 *Universidad del Norte.*