UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

CÓMPUTO DISTRIBUIDO CON HEURÍSTICAS ADAPTATIVAS, ANÁLISIS DE LA CONVERGENCIA DEL PROTOCOLO BGP

T E S I S

QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN CIENCIAS (COMPUTACIÓN)

PRESENTA:
GERARDO CARDELAS GÓMEZ

TUTOR
DR. SERGIO RAJSBAUM GORODEZKY
INSTITUTO DE MATEMÁTICAS

MÉXICO, D.F. FEBRERO 2016

Agradecimientos

Doy gracias a Dios porque me ha permitido vivir para lograr uno más de mis sueños, me ha reconfortado en los momentos difíciles y me ha acompañado siempre en mi sendero. Le doy gracias por la bendición de vivir con unos padres extraordinarios, Olivia Gómez y Andrés Cardelas, a quienes tanto amo y que siempre han creído en mi. Padres, les agradezco su amor, esfuerzo, valentía y entereza con que me enseñan a vivir cada día.

Deseo aprovechar este espacio para agradecer al Dr. Sergio Rajsbaum Gorodezky por sus enseñanzas e invaluables consejos para la elaboración de esta tesis. De igual manera, agradezco a cada uno de los sinodales que a través de sus revisiones y observaciones, contribuyeron al desarrollo de éste trabajo.

Gracias al personal del Posgrado en Ciencia e Ingeniería de la Computación de la Universidad Nacional Autónoma de México así como al Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas (IIMAS) por hacer de estos años una experiencia inolvidable.

Por último, agradezco el apoyo del Consejo Nacional de Ciencia y Tecnología (CONACYT), el cual sigue creyendo en la educación como motor que impulsa y fortalece el desarrollo de México.

Resumen

En los últimos años se ha realizado investigación de la interacción de las ciencias de la computación, teoría de juegos y teoría económica motivados por el desarrollo de sistemas como Internet [27].

Mientras que los estudios sobre el comportamiento de Internet se han basado en modelos de gráficas, en los cuales los elementos que la conforman sólo siguen un conjunto de acciones definidas, el énfasis en investigaciones recientes considera a los elementos constituyentes como tomadores de decisiones y por ende susceptibles a ser estudiados con elementos de la teoría de juegos [12], [2].

El presente trabajo presenta una introducción al estudio de los sistemas distribuidos con heurísticas adaptativas. Tales heurísticas adaptativas son reglas de comportamiento que describen la forma de reaccionar de agentes tomadores de decisiones antes cambios en su entorno [15].

Se presenta cuando la convergencia a un punto de equilibrio es garantizada en dichos sistemas cuando estos operan en ambientes asíncronos y como los resultados pueden utilizarse para analizar la convergencia del protocolo BGP, el cual es un pilar del funcionamiento de Internet.

De igual manera, se presentan resultados previos sobre la convergencia del protocolo BGP, la complejidad computacional que tiene el determinar si un sistema converge y condiciones que garantizan convergencia, de tal manera que sea posible comparar los enfoques anteriores con el presentado por el cómputo distribuido con heurísticas adaptativas.

Índice general

Capítulo 1

Introducción

El ser humano, a lo largo de su historia, ha desarrollado diversas tecnologías, motivado por las inquietudes y necesidades de un momento en particular. Hoy día, la recopilación, procesamiento y distribución de la información, son un elemento fundamental de sus inquietudes.

La historia de la computación, conlleva en cierta medida el deseo por resolver dichas inquietudes. Si bien, su historia es relativamente reciente, su crecimiento no tiene semejanza con ningún otra industria. Los avances en las tecnologías de construcción de equipos de cómputo, han permitido pasar de sistemas centralizados, que ocupaban cuartos enteros, a sistemas que caben en la punta de un alfiler.

El deseo de intercambiar información entre diversas computadoras, condujo a la interconexión de las mismas. Al sistema resultante, se le conoce, como *red de computadoras*. Cada uno de los elementos que conforman dicha red, son elementos autónomos, conectados por una tecnología en común que permite el intercambio de información [34].

Diversos avances en las tecnologías de transmisión han ocasionado cambios radicales en las redes de computadoras. Durante la década de 1970 el estándar de la red proveía un ancho de banda del orden de kilobits por segundo (kbps) mientras que hoy día las velocidades se encuentran en los cientos de gigabits por segundo (gbps) [33].

Tales avances, aunados a la continua demanda de nuevos servicios por parte de los usuarios y la creación de otros para proporcionar nuevos mecanismos de comunicación, han cambiado de manera radical la forma en que los usuarios vemos el mundo hoy en día, logrando conectar equipos que se encuentran en puntos opuestos del planeta.

1.1. Antecedentes y contexto del trabajo

Las redes se presentan en diferentes tamaños y formas. Usualmente se encuentran conectadas entre ellas para formar redes más grandes, en la cual *Internet* es uno de los ejemplo más conocido de una red de redes.

El uso de Internet es de suma importancia para el conjunto de actividades que desarrolla el ser humano día con día. Permite a las personas el comunicarse, colaborar e interactuar en diversas formas, acceder a páginas web, hablar usando teléfonos IP (siglas en inglés de Internet Protocol), participar en video conferencias, competir en juegos interactivos, completar cursos en línea, consultar información con fines educativos o recreativos, hasta el manejo de millones de transacciones económicas en segundos; pensar en un mundo sin este soporte tecnológico resulta difícil ya que ahora dependemos mucho de estos sistemas.

Internet es una red de computadoras que interconecta cientos de millones de dispositivos computacionales. Aunque en algún momento tales dispositivos eran computadoras de escritorio, estaciones de trabajo y servidores, en la actualidad podemos incluir a los teléfonos inteligentes, tabletas electrónicas, televisiones y refrigeradores, entre otros más.

Para lograr una comunicación entre dispositivos, se requiere establecer un conjunto de reglas, de forma que la información intercambiada sea entendible por los participantes. Un *protocolo* se encarga de definir el formato y orden de los mensajes intercambiados entre dos o más dispositivos, también conocidos como *anfitriones* o *sistemas finales* [19].

Dentro de las tareas de un protocolo de red, se encuentran por ejemplo los métodos utilizados para codificar la información, mecanismos para detectar errores, métodos de retransmisión en caso de pérdidas y métodos para regular el flujo de información entre iguales (*peers*, por su nombre en inglés) para evitar el desbordamiento de información [33].

Los protocolos de red son definidos de manera funcional, es decir, solamente se especifica las funciones y servicios que deben llevar a cabo, sin indicar como debe ser su implementación en algún sistema en particular.

Los elementos físicos sobre los cuales se implementan los protocolos se ven influenciados por los requerimientos de los mismos, y de igual forma los avances en la tecnología son tomados en cuenta por los protocolos, de forma que puedan aprovechar al máximo las nuevas características disponibles. Internet se beneficia del desarrollo de nuevos protocolos y tecnologías, pero sin perder de vista el problema fundamental, el envío de información.

1.1.1. Tarea esencial: el envío de información

Una de los problemas fundamentales que deben de resolver redes como Internet es cómo enviar mensajes desde un punto conocido como origen a un punto final llamado destino. Por sencilla que parezca la descripción anterior, el problema es bastante complejo y se debe tener en cuenta que Internet esta constituida por redes mucho más pequeñas, las cuales a su vez, engloban a otras redes. La solución debe considerar, por tanto, dos problemas principales: heterogeneidad y escalamiento.

Cualquier solución debe de considerar la diferencia que existe entre los sistemas y desarrollar mecanismos apropiados para su correcta comunicación. De igual forma, la solución debe poder escalarse, ya que el crecimiento de Internet no parece tener fin y día con día se agregan más elementos a la misma.

Aunque los factores mencionados son cruciales en el diseño de una solución, también se debe considerar la existencia de factores impuestos por las aplicaciones que se ejecutaran, tales como requerimientos de velocidad, costo, desempeño, fiabilidad y consumo de energía, entre otros.

Para considerar que la operación del sistema de red es correcta, se espera que se adhiera a los protocolos utilizados, pero además, existen elementos no funcionales que perciben los usuarios del sistema y que resultan deseables, por ejemplo, que la degradación de la red sea grácil, lo que permite percibir una transición más sutil cuando la red presenta una alta carga de tráfico o falla en los componentes, de la misma manera se espera que el sistema siga operando aún cuando el desempeño no sea el óptimo.

Los protocolos, como se ha mencionado con anterioridad, se definen por medio de la funcionalidad proporcionada e independiente de la plataforma sobre la que serán ejecutados. Estas características permite el desarrollo de sistemas de gran escala de forma económica, donde diversas empresas manufactureras pueden desarrollar sistemas que ejecuten los mismos protocolos en diferentes plataformas pero con diferentes características de desempeño y confiabilidad.

El *Protocolo de Puerta Exterior* (BGP, por sus siglas en inglés), es uno de los protocolos diseñados con el fin de resolver los problemas de heterogeneidad y escalamiento de Internet. Si bien su uso es fundamental, existen cuestiones por resolver y que motivan un estudio más profundo de su funcionamiento.

1.1.2. El protocolo BGP y el problema de la convergencia

El protocolo BGP facilita el intercambio de información acerca de las redes entre entidades conocidas como *sistemas autónomos* (AS, por sus siglas en inglés). El intercambio de información permite que una parte de Internet sepa como enviar información a otra parte de la misma [25].

Cada AS agrupa a redes más pequeñas, que se encuentran bajo un mismo control administrativo. El intercambio de información entre AS se lleva a cabo por medio de una sesión de comunicación entre AS vecinos.

Si imaginamos a cada AS como un único elemento y la información de conectividad entre los mismos como enlaces físicos, es posible utilizar la teoría de gráficas para modelar el problema de enviar un mensaje entre AS. De tal modo que el envío de un mensaje será posible siempre que sea posible establecer una trayectoria entre los AS origen y destino.

Una de las preguntas que surgen en el envío de mensajes es la siguiente: ¿siempre es posible el encontrar una trayectoria entre los AS origen y destino?. La respuesta es afirmativa de manera trivial si la red es estática, conectada y confiable, pero en el caso de Internet donde la falla de algún enlace y la creación de nuevos enlaces puede ocurrir en todo momento, la respuesta no resulta fácil de contestar.

Aunado a los problemas anteriores, cada AS puede establecer *políticas* que determinan que trayectorias son aceptadas y cuales no. Entonces, aún cuando exista una trayectoria física entre un AS origen o destino, las políticas de alguno o varios AS a lo largo de la trayectoria podrían ocasionar que tal trayectoria no fuera aceptada, impidiendo el envío de mensajes entre los AS origen y destino.

En el caso de BGP, la información de intercambiada entre los AS y las políticas de cada uno son utilizadas para construir en cada AS una *tabla de ruteo*. Tal construcción indica de manera general, que AS son alcanzables y la trayectoria a seguir para llegar a ellos.

Puesto que la información se actualiza frecuentemente y que la red puede presentar fallas, los datos que contienen las tablas de ruteo pueden cambiar varías veces antes de estabilizarse. Si tal estabilización es posible, decimos que el sistema *converge* y en caso contrario el sistema diverge, es decir, los AS intercambian información de manera indefinida sin llegar nunca a un conjunto de trayectorias estables.

El problema de la convergencia de BGP trata de determinar si es posible

el garantizar que el sistema no diverja. Estudios tales como [22, 21, 16], exponen patologías en BGP que hacen posible conjeturar que no siempre es posible dar esta garantía. Y de hecho, Griffin [13] muestra que tal conjetura no es del todo errónea.

El análisis de la convergencia del protocolo BGP dista de ser sólo un ejercicio académico y tiene implicaciones para el usuario común, véase por ejemplo [21, 16].

Para entender el funcionamiento del protocolo BGP y realizar inferencias sobre su comportamiento se han desarrollado modelos. El propósito de tales modelos es centrarse en las características fundamentales, con el objeto de responder a preguntas como las antes planteadas. Un ejemplo es el formalismo del *Protocolo Simple de Trayectorias de Vector* (SPVP, por sus siglas en inglés), el cual busca capturar la semántica subyacente de cualquier protocolo de trayectoria vector, tal como BGP [11].

El SPVP realiza una descripción abstracta del protocolo BGP por medio una gráfica con las trayectorias posibles de cada nodo y una función de ordenación para las trayectorias. La función de ordenación es una representación abstracta de las políticas de cada AS.

Otro de los modelos de BGP, es el desarrollado por una área, relativamente nueva, conocida como *cómputo distribuido con heurísticas adaptativas*. En el modelo, los AS se consideran tomadores de decisiones. Cada AS elige la opción que considera mejor y maximice su *ganancia*, es decir, trata de elegir la mejor trayectoria en términos de las políticas que se encuentren en uso y la información que le ha sido proporcionada por los demás AS [15].

Es a través del enfoque de esta área y de su modelo que se dará respuesta a algunas de las preguntas de BGP planteadas anteriormente, principalmente la que concierne a la convergencia.

1.2. Objetivo y contribuciones

El presente trabajo tiene como objetivo estudiar el área conocida como cómputo distribuido con heurísticas adaptativas, la cual se encuentra en la frontera de las ciencias de la computación y la teoría de juegos, presentando su aplicación en un problema concreto: el análisis de la convergencia del protocolo BGP.

Se presentarán enfoques utilizados previamente para el análisis del protocolo BGP y los resultados obtenidos. Posteriormente se introducirá el marco teórico necesario para introducir ésta nueva área y estudiar el problema de la convergencia.

El estudio de la convergencia del protocolo BGP, permite sintetizar mucha de la información y resultados que se han generado en los últimos años. Además prepara el terreno para la presentación del modelo utilizado por [15] para el análisis de la convergencia de BGP.

Una de las directrices en la selección del material usado y la presentación del mismo ha sido la claridad. La información es reciente y diversa, por lo cual aún no esta escrita en forma accesible a un publico amplio.

Se ha buscado que la información aquí presentada sea asequible a aquellos que tienen contacto por primera vez con el área y tienen interés en profundizar en su estudio.

De los principales resultados que se presentan se encuentra un modelo para el análisis de las heurísticas adaptativas en ambientes asíncronos, el concepto de independencia de decisiones y un resultado de imposibilidad de convergencia cuando los nodos participantes en el sistema cumplen ciertas características.

Es importante recalcar que la aplicación de esta nueva área no esta limitada al estudio de protocolos, sino que es más amplia, pero se prefirió presentar su potencial en un problema concreto, esperando que aquel lector interesado encuentre un punto de partida apropiado y una fuente de inspiración de nuevas ideas para los problemas que encuentre.

1.3. Organización de la tesis

La presentación de la información se da de manera incremental. En el capítulo 2, se presenta lo que es un protocolo de ruteo, la clasificación de los algoritmos de ruteo y el porqué de su importancia.

Una vez presentados los conceptos de ruteo y de algoritmo de ruteo, el capítulo 3 presenta el funcionamiento del protocolo BGP y un concepto fundamental de este trabajo, la convergencia.

El capítulo 4 presenta qué es el cómputo distribuido con heurísticas adaptativas, introduciendo un modelo de cómputo para el estudio de las heurísticas adaptativas en ambientes asíncronos, el concepto de independencia de decisiones y su implicación en la convergencia del protocolo BGP.

Por último, el capitulo final presenta las conclusiones, implicaciones y trabajo futuro de un área con gran potencial.

Capítulo 2

Ruteo

Las redes de computadoras permiten el intercambio de información entre sistemas que pueden encontrarse en lugares diametralmente opuestos del planeta. Internet es un ejemplo de este tipo de redes, la cual interconecta cientos de millones de dispositivos a través del planeta. A los dispositivos que conecta se les conoce como anfitriones o sistemas finales.

Para reducir la complejidad en el diseño de sistemas de gran escala como Internet, se recurre al diseño en capas. El diseño es tal que cada capa ofrece ciertos servicios a la capa inmediatamente encima de ella. El número, nombre, propósito y servicio de cada una, varía dependiendo del tipo de red [34].

Uno de los beneficios del diseño en capas es que permite una separación de las responsabilidades y servicios ofrecidas en cada una de ellas. Una capa utiliza los servicios otorgados por la capa inferior y los aumenta. Lo cual da como resultado un diseño flexible y fácil de extender.

Aunque Internet es un sistema de dimensiones considerables consta de pocas capas. Las capas que lo conforman, comenzando con la inferior son: física, enlace, red, transporte y aplicación.

La capa de red es esencial para el funcionamiento de Internet. Una de sus funciones es determinar la trayectoria que debe recorrer la información, de modo que dos sistemas finales pueden comunicarse entre ellos.

Para cumplir tal labor, la capa de red se auxilia de los protocolos de ruteo[1]. Un protocolo tiene como objetivo el facilitar el intercambio de in-

[1]Las palabras ruteador y ruteo no existen en el diccionario de la Real Academía Española. Ruteador y ruteo, sin embargo, se usan frecuentemente como traducciones de las palabras inglesas *router* y *routing* en algunas publicaciones, véase [3, 26]. En esta obra utilizaremos el término ruteador para designar al dispositivo encargado de calcular las

formación de forma estandarizada. El protocolo de ruteo, a su vez, emplea algoritmos de ruteo para la determinación de la trayectoria entre sistemas finales.

Los algoritmos de Dijkstra y Bellman-Ford, usados para el cálculo de la trayectoria más corta en la teoría de gráficas, son una base fundamental de los algoritmos de ruteo. Por tanto, se presentará un repaso de ellos al igual que las pruebas de correctez correspondientes.

2.1. Ruteo y Reenvío

Los sistemas finales se encuentran conectados por una red de *enlaces de comunicación* y *conmutadores de paquetes*.

Cuando un sistema final[2] envía información a otro, el sistema enviante segmenta la información en unidades conocidas como *paquetes*. A cada paquete se le agrega un encabezado que contiene la dirección del sistema final al cual será enviado (el encabezado contiene otros atributos adicionales). Esta dirección es única y es proporcionada por el protocolo IP (siglas en inglés de *Internet Protocol*), el cual esta incluido en la capa de red. Una vez segmentada la información, ésta puede transitar por la red que interconecta los sistemas finales.

A la secuencia de enlaces y encaminadores de paquetes que atraviesa un paquete desde un sistema final que envía a uno que recibe se le conoce como *ruta* o *trayectoria* [19].

Los conmutadores de paquetes son diversos, pero los dos tipos preponderantes son los *ruteadores* y los *conmutadores de la capa de enlace*. Los ruteadores se encuentran en la capa de red.

Como se había mencionado antes, la tarea de la capa de red es el mover paquetes de un anfitrión a otro. Para llevar a cabo esta tarea, dos funciones principales pueden ser identificadas [19]:

1. **Reenvío**. Cuando un paquete llega a uno de los enlaces de entrada del ruteador, éste debe de encargarse de mover el paquete al enlace de salida apropiado.

trayectorias y el término ruteo para designar la acción.

[2]En esta obra utilizaremos los términos sistema final y anfitrión de manera intercambiable.

2. **Ruteo**. La capa de red debe ser capaz de determinar la ruta o trayectoria a seguir por los paquetes mientras estos son enviados desde el origen al destino. A los algoritmos que se encargan de realizar esta tarea se les conoce como *algoritmos de ruteo*.

Obsérvese que el reenvío y el ruteo son dos procesos diferentes, el primero se refiere a la tarea local que tiene un ruteador de transferir un paquete desde su enlace de entrada a un enlace de salida dentro del mismo, mientras que el segundo corresponde al proceso de determinar de manera sistemática como enviar paquetes desde un anfitrión llamado origen a otro llamado destino en base a su dirección [29].

Cada ruteador contiene una *tabla de reenvío* (*forwarding table* en inglés). Cuando un paquete es recibido por el ruteador, su encabezado es examinado en busca de un campo, el cual a su vez es utilizado para buscar un registro en la tabla de reenvío. El registro contiene el enlace de salida que necesita utilizarse para reenviar el paquete.

Una de las preguntas que surge es cómo se crea la tabla de reenvío en primer lugar y cómo se modifican los registros una vez creados. Ésta tarea es responsabilidad los algoritmos de ruteo, adicional a la que tienen del cálculo de trayectorias y su estudio nos permitirá aclarar las cuestiones antes planteadas.

2.2. Algoritmos de ruteo

De manera general una red de comunicación esta conformada por nodos y enlaces. Lo que representa un nodo depende de la red que estemos hablando, en el caso de Internet puede representar anfitriones, conmutadores de paquetes, ruteadores u otras redes. Un enlace se encarga de conectar dos nodos, en redes como Internet, dicho enlace es algunas veces llamando *tronco IP* (*IP trunk* en inglés) o enlace IP, mientras que el extremo del enlace que sale del ruteador se conoce como *interfaz* [25].

Usualmente un sistema anfitrión se encuentra unido a un ruteador, al que se conoce como *ruteador por defecto* o también como el *ruteador del primer brinco* (*first-hop router*, en inglés). Cuando el anfitrión envía un paquete, el paquete se manda al ruteador por defecto, al cual se le nombrará el ruteador origen y al ruteador por defecto del destino como el ruteador destino. El problema de un algoritmo de ruteo se reduce entonces, a determinar la

trayectoria de ruteadores y enlaces por los cuales debe enviarse el paquete para que llegue a su destino [19].

Puesto que cabe la posibilidad que existan múltiples trayectorias posibles, el algoritmo debe contar con criterios para seleccionar una de todas las posibles. Usualmente una "buena"trayectoria es aquella que tiene el menor costo posible, aunque en el mundo real existen otras cuestiones que deben de tomarse en cuenta por los algoritmos de ruteo, tales como las *políticas* (por ejemplo, que el ruteador de una organización no reenvíe paquetes que no sean generados por ella) o las restricciones impuestas por las aplicaciones que se ejecutarán en la red [19, 25].

Si consideramos la representación de nodos y enlaces dada al inicio, el uso de la teoría de gráfica para la representación de problemas de ruteo resulta apropiada. Por ejemplo, considere la gráfica de la figura 2.1, la cual es la representación de una red

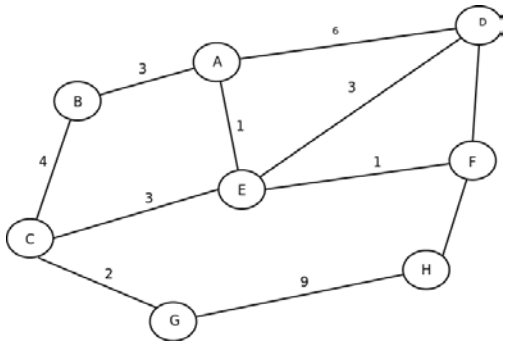

Figura 2.1: La red representada como una gráfica

Las letras A hasta la H representan los nodos de la red, los cuales como se había mencionado antes pueden corresponder a anfitriones, conmutadores de paquetes, ruteadores u otras redes. Las aristas de la gráfica corresponden a enlaces de comunicación por los cuales es posible enviar mensajes de datos, dichos enlaces pueden ser cables de cobre, fibra óptica o mecanismos inalámbricos. Cada arista de la gráfica (enlace de comunicación) contiene un número, este número representa un *costo* asociado al envío de un mensaje por medio de dicha arista.

El costo de un enlace es una medida genérica que puede representar la longitud, velocidad o el retardo en el envío de un paquete por dicho enlace. No estamos interesados en como se determina este valor, sino que asumiremos que ya nos es dado.

Cada nodo puede en algún momento tomar el rol del nodo origen, a partir del cual se enviarán los mensajes a cualquier otro nodo de la gráfica. El problema básico del ruteo es entonces el encontrar la trayectoria con el menor costo posible entre dos nodos, donde el costo de la trayectoria es igual a la suma de los costos de todos los enlaces que conforman la trayectoria.

La cantidad de información que se requiere para encontrar la trayectoria más corta difiere entre algoritmos y da lugar a una clasificación general que se abordará en la siguiente sección.

2.2.1. Clasificación de los algoritmos de ruteo

Existen diversos criterios para la clasificación, uno de ellos se basa en si son centralizados o descentralizados [19].

- **Algoritmo de ruteo global**. Calculan la trayectoria de menor costo entre dos nodos usando un conocimiento completo de la red. Esto requiere que el algoritmo obtenga la información de todos los nodos que conforman la red, así como también el costo de los enlaces que los unen. Los cálculos pueden ser ejecutados en un sólo sitio (un algoritmo *centralizado global*) o replicado en múltiples sitios. La característica principal de este tipo de algoritmos es que tienen información completa acerca de la conectividad de los nodos de la gráfica que representa la red así como los costos asociados de los enlaces. Los algoritmos que cuentan con esta información son conocidos como *Algoritmos de Estado-Enlace* (LS, siglas en inglés de *link-state*), ya que deben estar atentos de los costos de cada enlace en la red.

- **Algoritmo de ruteo descentralizado**. El cálculo de la trayectoria con el menor costo se realiza de manera iterativa, en forma distribuida. Ningún nodo cuenta con información completa acerca de los costos de todos los enlaces en la red. En un inicio, cada nodo comienza sólo con la información del costo de los enlaces a los que se encuentra unido de forma directa. Después de un proceso de cálculos iterativos y de intercambio de información con sus vecinos (esto es, nodos que se encuentran al otro extremo de los enlaces a los que se encuentra directamente conectado), un nodo gradualmente calculará la trayectoria con el menor costo posible a un nodo destino o conjunto de nodos. Un algoritmo de ruteo descentralizado que se presentará más adelante, es

llamado *algoritmo distancia vector* (DV, siglas en inglés de *Distance-Vector*), porque cada nodo mantiene un vector de costos (distancias) estimados a los demás nodos en la red.

Una segunda clasificación considera si el cálculo de trayectorias ocurre de manera automática al registrarse un cambio en la topología de la red o no. En un *algoritmo de ruteo estático* las rutas cambian de manera inusual y usualmente por intervención humana. En los *algoritmos de ruteo dinámico* el cambio de las trayectorias ocurre cuando hay un cambio en la red o en el costo de los enlaces.

Una tercera clasificación se basa en si los algoritmos son sensibles o insensibles a la carga de la red. En un *algoritmo sensible a la carga*, el costo de los enlaces refleja su nivel de congestión. Si un enlace tiene demasiado tráfico de paquetes en un instante dado, el algoritmo buscará rutas alternativas para el envío de mensajes.

Que un algoritmo se encuentre en determinada clasificación no lo excluye de estar incluido en otra, por ejemplo, un algoritmo de estado enlace puede ser insensible a la carga y dinámico. Las clasificaciones sirven como una guía general solamente y no deben verse como excluyentes la una de la otra.

2.2.2. Algoritmo De Ruteo De Estado-Enlace

Una de las características en un algoritmo de estado enlace es que cada nodo tiene un mapa completo de la red, en el cual se encuentran todos los nodos y los costo de los enlaces que la conforman.

Para crear este mapa, se presupone que cada nodo sabe como alcanzar a sus vecinos, el estado de los enlaces (activo, inactivo), así como sus costos. Una vez que un nodo tiene esta información, la disemina a sus vecinos, quienes a su vez repiten este proceso. Eventualmente cada nodo tendrá toda la información necesaria para llevar a cabo su tarea.

En esta sección y la siguiente asumiremos que la red se modela como una gráfica dirigida $G = (V, E)$, donde V representa los nodos y E los enlaces de la misma. La gráfica G asociada a la red tiene un *costo* para cada una de las aristas que la conforman, el cual es dado por una *función de costo* $c : E \to \mathbb{R}$. El costo de la arista es igual al que existe en el enlace de red que modela.

Una *trayectoria* es una gráfica no vacía $P = (V, E)$ de la forma

$$V = x_0, x_1, \ldots, x_k \qquad E = x_0x_1, x_1x_2, \ldots, x_{k-1}x_k \tag{2.1}$$

donde todas las x_i son distintas. Los vértices x_0 y x_k están *vinculados* por P y son llamados sus *vértices finales* o *extremos*; los vértices $x_1, \ldots, x_{k-1}$ son los vértices *internos* de P [6]. El costo de la trayectoria P, que denotaremos $c(P)$, es la suma del costo de todas las aristas en P.

Sea $\mathbb{P} \neq \emptyset$ el conjunto de todas las trayectorias $\forall u, v | u, v \in V$, el costo de la trayectoria más corta, que denotaremos como OPT_{uv}, se define como

$$OPT_{u,v} = \begin{cases} min\{c(p) : p \in \mathbb{P}\} & \text{si existe una trayectoria de } u \text{ a } v, \\ \infty & \text{en otro caso} \end{cases}$$

La trayectoria de menor costo del vértice u al vértice v en una gráfica dirigida G con función de costo, es cualquier trayectoria $p \in \mathbb{P}$ de u a v con costo $c(p) = OPT_{u,v}$.

Con las definiciones anteriores se presentará un algoritmo de estado enlace conocido como el algoritmo de Dijkstra. El algoritmo es iterativo y tiene la propiedad de que después de la k-ésima iteración del algoritmo, las trayectorias de menor costo de k nodos son conocidas, y entre las trayectorias de menor costo a todos los nodos destinos, estas k trayectorias tendrán los k costos más pequeños.

A partir de un vértice inicial s, el algoritmo de Dijkstra calcula las trayectorias de menor costo a los demás vértices. El algoritmo consiste de un paso de inicialización, InicializarFuenteUnica 2.3. Para realizar los cálculos, se mantiene un conjunto S de vértices u para los cuales se ha determinado la trayectoria de menor costo $d(u)$. En cada iteración se busca el vértice $v \in V - S$ para el cual su costo $d(v)$ es el menor. Una vez encontrado v se agrega al conjunto S y se llama al procedimiento relajar 2.2.

```
Dijkstra(G,s)
(1)  InicializarFuenteUnica(G, s)
(2)  S = ∅
(3)  Q = V
(4)  while Q ≠ ∅
(5)     u = mín_{j∈Q}(costo_s[j])
(6)     S = S ∪ {u}
(7)     Q = Q \ {u}
(8)     for each v ∈ vecinos_u
(9)        RELAJAR(u, v, arista_⟨u,v⟩)
```

Listado 2.1: Algoritmo de Dijkstra

```
Relajar(u, v, arista⟨u,v⟩)
(1)  if costo_s[v] > costo_s[u] + c(arista⟨u,v⟩)
(2)     costo_s[v] = costo_s[u] + c(arista⟨u,v⟩)
(3)     routing_s[v] = u
```

Listado 2.2: Procedimiento relajar

```
InicializarFuenteUnica(G,s)
(1)  for each v ∈ V
(2)     costo_s[v] = ∞
(3)     routing_s[v] = NIL
(4)  costo_s[s] = 0
```

Listado 2.3: Procedimiento inicializar fuente única

El procedimiento InicializarFuenteUnica utiliza dos arreglos auxiliares: $costo_s$ y $routing_s$. El primero de ellos es un estimado del costo de la trayectoria del vértice s origen a cada uno de los demás vértices, mientras que el segundo indica el siguiente vértice a partir de s en la trayectoria más corta hacia cada uno de los demás vértices.

Al finalizar el algoritmo de Dijkstra, el arreglo $costo_s$ tendrá el costo de la trayectoria más corta para cada uno de los vértices a partir de s o tendrá el valor de ∞ en caso de que no exista una trayectoria. De igual forma, el arreglo $routing_s$ tendrá el siguiente vértice en la trayectoria más corta de s a cada uno de los vértices a los que puede llegar, o el valor *NIL* en caso de que no exista una trayectoria.

Mientras el algoritmo se ejecuta, recopila información acerca de las trayectorias más cortas que conectan el vértice origen con los demás vértices encontrados en los arreglos $costo_s$ y $routing_s$. Al actualizar tal información, se mejoran las inferencias de las trayectorias más cortas. El procedimiento 2.2 utiliza la *relajación de arista*, la cual se define del modo siguiente: *relajar* una arista (v, w) significa el probar si la mejor forma de llegar de s a w es ir de s a v y después seguir la arista (v, w), si es el caso, se actualizan los arreglos $costo_s$ y $routing_s$. Si el costo de la trayectoria más corta de s a w no puede ser mejorado, los arreglos permanecen sin cambio.

La operación de relajación se realiza para cada una de las aristas del vértice recién agregado al conjunto S. Esta operación y el orden en que son elegidos los vértices que son agregados al conjunto S permiten de manera incremental, el encontrar las trayectorias más cortas para cada vértice. Para

realizar la prueba de correctez del algoritmo de Dijkstra utilizaremos varios elementos, comencemos con el siguiente teorema [32].

Teorema 2.2.1 (Optimalidad de las trayectorias más cortas)**.** *Sea $G = (V, E)$ una gráfica dirigida con función de costo $c : E \to \mathbb{R}$, con un vértice s designado como origen en G y un arreglo $costo_s[v]$ para todo $v \in V$, tal que, para cada v alcanzable a partir de s, el valor de $costo_s[v]$ representa el costo de alguna trayectoria de s a v. Los valores del arreglo $costo_s[]$ representan el costo de las trayectorias más cortas, sí y sólo si para cada $costo_s[w] \leq costo_s[v] + c(arista_{\langle v,w \rangle})$ para todo $v, w \in V$ [32].*

Demostración. Supongamos que $costo_s[w]$ es el costo de la trayectoria más corta de s a w. Si $costo_s[w] > costo_s[v] + c(arista_{\langle v,w \rangle})$ para algún $v \in V$, entonces, la trayectoria que va de s a v y sigue la $arista_{\langle v,w \rangle}$ tiene un menor costo que $costo_s[w]$, una contradicción. Ahora bien, en el caso general considérese que w es alcanzable a partir de s por medio de una secuencia de aristas $s = v_0v_1, v_1v_2, \ldots, v_{k-1}v_k = w$, y que es la trayectoria más corta de s a w de peso $OPT_{s,w}$. Para i de 1 a k, denotaremos la arista $v_{i-1}v_i$ como e_i. Por la condición de optimalidad de la trayectoria más corta se tienen las siguientes desigualdades:

$$\begin{aligned} costo_s[w] = costo_s[v_k] &\leq costo_s[v_{k-1}] + c(e_k) \\ costo_s[v_{k-1}] &\leq costo_s[v_{k-2}] + c(e_{k-1}) \\ &\cdots \\ costo_s[v_2] &\leq costo_s[v_1] + c(e_2) \\ costo_s[v_1] &\leq costo_s[s] + c(e_1) \end{aligned}$$

Simplificando estas desigualdades y eliminando $costo_s[s] = 0$ se tiene que

$$costo_s[w] \leq c(e_1) + \cdots + c(e_k) = OPT_{sw}$$

Ahora bien, $costo_s[w]$ es el costo de alguna trayectoria de s a w, pero tal costo no puede ser menor que el de la trayectoria más corta y por tanto debe ser una igualdad

$$OPT_{sw} \leq costo_s[w] = OPT_{sw}$$

□

El teorema 2.2.1 nos indica las condiciones que deben cumplir las entradas del arreglo $costo_s[]$ para saber que contienen el costo de las trayectorias más cortas. Una de las cuestiones por resolver es ¿cómo reconstruir la trayectoria más corta de cada vértice cuyo costo es $costo_s[v]$ para todo $v \in V$?. Para reconstruir la trayectoria, cada vez que se actualice una de las entradas en $costo_s[v]$ se registra la arista encargada de producir el cambio en el arreglo $routing_s[v]$ para todo $v \in V$. La trayectoria P, puede recuperarse comenzando por el vértice final de la misma y siguiendo el enlace indicado en $routing_s[]$, el cual dará el vértice penúltimo de la trayectoria, y siguiendo de manera consecutiva se llegará al vértice inicial s.

Los siguientes lemas describen como se ven afectados los estimados de las trayectorias más cortas después de realizar el procedimiento de inicialización y ejecutar varias secuencias de relajación [5].

Lema 2.2.2 (Desigualdad del triángulo)**.** *Sea $G = (V, E)$ una gráfica con función de costo $c : E \to \mathbb{R}$ y con un vértice s designado como origen. Entonces, para todas las aristas $(u, v) \in E$, se tiene*

$$OPT_{s,v} \leq OPT_{s,u} + c(arista_{\langle u,v \rangle})$$

Demostración. Supongamos que p es la trayectoria más corta del vértice s al vértice v. Entonces p no tiene un costo mayor que cualquier otra trayectoria de s a v. En concreto, la trayectoria p no tiene un costo mayor que aquella formada al seguir la trayectoria más corta de s a u y después toma la arista (u, v). □

Lema 2.2.3 (Propiedad del límite superior)**.** *Sea $G = (V, E)$ una gráfica con función de costo $c : E \to \mathbb{R}$. Sea $s \in V$ un vértice designado como el origen, y el arreglo $costo_s[]$ inicializado por el procedimiento InicializarFuenteUnica 2.3. Entonces, $costo_s[v] \geq OPT_{sv}$ para todo $v \in V$, y esta invariante se mantiene sobre cualquier secuencia de pasos de relajación en las aristas de G. Más aún, una vez que $costo_s[v]$ alcanza el valor de OPT_ssv, este valor nunca cambia.*

Demostración. Se prueba la invariante $costo_s[v] \geq OPT_{sv}$ para todos los vértices $v \in V$ por inducción sobre el número de pasos de relajación.

Para el caso base, $costo_s[v] \geq OPT_{s,v}$ es cierto después de la inicialización ya que $costo_s[v] = \infty$ implica $costo_s v \geq OPT_{s,v}$ para todo $v \in V - s$, y además $costo_s[s] = 0 \geq OPT_{s,s}$.

Para el paso inductivo, considérese la relajación de la arista (u, v). Por la hipótesis inductiva $costo_s[x] \geq OPT_{s,x}$ para todo $x \in V$ antes de la relajación. El único valor d que puede cambiar es $costo_s[v]$. Si cambia se tiene

$$\begin{aligned} costo_s[v] &= costo_s[u] + c(arista_{\langle u,v \rangle}) \\ &\leq OPT_{s,u} + c(arista_{\langle u,v \rangle}) && \text{por la hipótesis inductiva} \\ &\leq OPT_{s,v} && \text{por la desigualdad del triángulo} \end{aligned}$$

y la invariante se mantiene.

Para ver que el valor de $costo_s[v]$ nunca cambia una vez que $costo_s[v] = OPT_{s,v}$, nótese que una vez que ha alcanzado su límite inferior, $costo_s[v]$ no puede decrecer porque se ha mostrado que $costo_s[v] \geq OPT_{s,v}$, y no puede incrementar porque los pasos de relajación solo decrementan los valores de $costo_s[]$. □

Ahora estamos listos para presentar la prueba de la correctez del algoritmo de Dijkstra, para ello utilizaremos el siguiente teorema.

Teorema 2.2.4 (Correctez del algoritmo de Dijkstra)**.** *La ejecución del algoritmo de Dijkstra en una gráfica dirigida $G = (V, E)$ con una función de costo c y un vértice s nombrado origen, termina con $costo_s[v] = OPT_{s,u}$ para todos los vértices $v \in V$ [5].*

Demostración. Para realizar la prueba utilizaremos una invariante del ciclo del algoritmo 2.1:

> Al comienzo de cada iteración de el ciclo **while** de las líneas 4-8 del, $costo_s[v] = OPT_{s,v}$ para cada vértice $v \in S$

Al inicio, $S = \emptyset$, y por tanto la invariante es trivialmente verdadera. Para el propósito de contradicción, sea u el primer vértice para el cual $costo_s[u] \neq OPT_{s,u}$ cuando u es agregado al conjunto S y llamemos p la trayectoria de s a u. Debe ser el caso que $u \neq s$ ya que s es el primer vértice que es agregado al conjunto S y $costo_s[s] = OPT_{s,s} = 0$. Debido a que $u \neq s$, también se tiene que $S \neq \emptyset$ justo antes de que u fuera agregado a S. Debe existir una trayectoria de s a u, de otra forma $costo_s u = \infty$.

Antes de agregar u a S debe de existir un vértice en S que conecta a $u \in V - S$, llamémosle t. Ahora consideremos otra trayectoria p' entre s y u, que es de menor costo. Sea y el primer vértice en la trayectoria p' que se encuentra en $V - S$, y sea $x \in S$ el nodo que se encuentra justo antes de y.

Esto se ilustra en la figura siguiente.

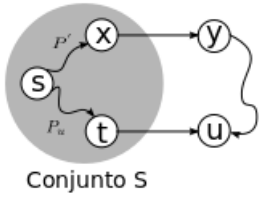

Figura 2.2: La trayectoria P' es al menos igual de costosa al momento de dejar el conjunto S que la trayectoria P_u.

Puesto que t se encuentra en S, $costo_s[t] = OPT_{s,t}$. De igual manera, el costo de x es $costo_s[x] = OPT_{s,x}$, ya que se encuentra en S. Ahora p' no puede ser menor que P_u porque p' tiene al menos el mismo costo que p en el momento que deja el conjunto S. De hecho, debe ser el caso que el algoritmo consideró agregar el vértice y en la línea 5 del algoritmo por medio de la arista (x, y) y la rechazo en favor del vértice v. Esto significa que no existe una trayectoria de s a u a través de x que sea menor que p y por tanto la trayectoria p es la de menor costo, una contradicción.

Al terminar el ciclo $Q = \emptyset$ que junto a la invariante $Q = V - S$, implica que $S = V$. Por tanto, $costo_s[v] = OPT_{s,v}$ para todo $v \in V$ y por el teorema de optimalidad 2.2.1 representa el costo de las trayectorias más cortas. □

Complejidad

¿Cuál es la complejidad computacional del algoritmo? El algoritmo realiza los pasos de inicialización 2.3 y la creación del conjunto Q, ambos de complejidad $|V|$. Después itera sobre el conjunto Q un total de $|V|$ ocasiones, en cada uno buscando el vértice $u \in Q$ para el cual el $costo_s[u]$ es mínimo. Ya que cada vértice es agregado exactamente una vez al conjunto S, cada arista es relajada exactamente una vez. Ya que el número total de aristas es

$|E|$, el ciclo **for each** de la línea 8 es ejecutado un total de $|E|$ veces, que es el mismo número de veces que es ejecutado el procedimiento relajar 2.2.

El tiempo de ejecución depende de como se implemente la operación $u = \min_{j \in Q}(costo_s[j])$. Si se implementa esta operación con un arreglo la operación relajar 2.2 toma tiempo $O(1)$, pero buscar el elemento más pequeño toma $O(V)$ (ya que se tiene que recorrer todo el arreglo), lo que da un tiempo total de $O(|V|^2 + |E|)$.

Podemos mejorar el tiempo de ejecución si en lugar de mantener los estimados de los costos en un arreglo utilizamos un montículo binario mínimo (*binary min-heap* en inglés). La operación para buscar el mínimo toma tiempo $O(lg|V|)$, y la operación relajar toma $O(lg|V|)$ si se puede acceder a los elementos del montículo en tiempo $O(1)$. El tiempo total es en este caso $O((|V| + |E|)lg|V|)$, lo cual es $O(|E|lg|V|)$ si la gráfica es dispersa, es decir, $|E| = o(|V|^2/lg|V|)$.

2.2.3. El Algoritmo de Ruteo de Distancia-Vector (DV)

En un algoritmo *Distancia Vector* cada ruteador mantiene una tabla, la cual contiene la mejor distancia estimada a los demás ruteadores que conforman la red. La información en la tabla se actualiza al intercambiar información con sus vecinos [34].

A diferencia del algoritmo Estado-Enlace presentado en la sección anterior, que contiene un mapa global de la red, el algoritmo distancia vector sólo contiene el costo de los enlaces a los cuales se encuentra directamente conectado y obtiene la demás información que necesita por medio del intercambio de información con sus vecinos directos.

El algoritmo es distribuido y asíncrono ya que cada (ruteador) recibe información de uno o más de sus vecinos a los que se encuentra unido de manera directa, realiza los cálculos correspondientes independiente de los demás y comunica sus resultados de vuelta. También es iterativo ya que cada uno de los ruteadores continua hasta que no hay más información intercambiada entre ellos [19].

El algoritmo de ruteo distancia vector es algunas veces conocido por otros nombres, usualmente el algoritmo distribuido Bellman-Ford [34]. Antes de presentar la versión distribuido, comenzaremos con la versión centralizada.

2.2.4. Algoritmo Bellman-Ford centralizado

El algoritmo de Dijkstra que se presento previamente funciona sólo cuando los costos de las aristas son positivos, comenzando con un conjunto S el cual tiene la propiedad de que las trayectorias menos costosas de s a cualquier nodo en S es conocida. En cada iteración el algoritmo busca la arista menos costosa que conecta un vértice en S con un vértice $i \in V - S$. Una vez que la encuentra, agrega el vértice i al conjunto S, relaja las aristas conectadas a i y repite el procedimiento.

Una de los requerimientos en la correctez del algoritmo de Dijkstra es que el costo de las aristas no puede ser negativo. Esto garantiza que cualquier otra trayectoria de un vértice en S a un vértice $i \in V - S$ es al menos de costo igual a la seleccionada por el algoritmo en el momento que deja al conjunto S. Sin embargo, cuando el costo puede ser negativo, tal garantía se pierde y la prueba de correctez deja de ser válida [17]. Esta situación se muestra en el ejemplo de la figura siguiente

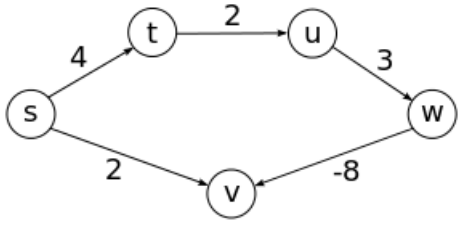

Figura 2.3: El algoritmo de Dijkstra elige la arista de menor costo que deja el conjunto S a un vértice en $V - S$. Pero cuando las aristas son negativas, puede darse el caso que tal arista no de el camino con el menor costo.

El tratar de compensar el costo de las aristas negativas sumando un valor positivo lo suficientemente grande no funciona, ya que esto cambia el costo de la trayectoria más corta. El ejemplo siguiente muestra esta situación

El algoritmo de Bellman-Ford por otra parte funciona aún cuando existen ciclos y aristas con costo negativo, siempre y cuando no existan ciclos negativos [3]

[3]Un *ciclo negativo* en una gráfica dirigida con pesos, es un ciclo dirigido cuyo costo (la suma del costo de las aristas que lo conforman) es negativo [32].

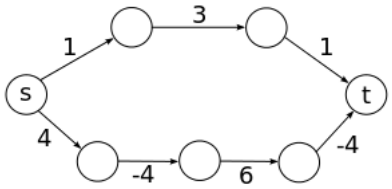

Figura 2.4: Cuando el costo de las aristas puede ser negativo, el algoritmo de Dijkstra puede fallar. En la gráfica que se presenta, la trayectoria inferior tiene un menor costo. Cuando se agrega la cantidad de 4 a cada arista, el costo de cada una es positivo, pero desafortunadamente, ésto cambia la trayectoria más corta de s a t, donde ahora la trayectoria más corta es la superior.

Una propiedad esencial que tiene una gráfica G dirigida con función de peso, es la siguiente:

Lema 2.2.5. *Si G no tiene ciclos negativos, entonces existe una trayectoria de s a t que es simple (es decir, no repite nodos), y por lo tanto tiene a los más $n-1$ aristas [17].*

Demostración. Ya que el costo de cada ciclo es positivo, la trayectoria más corta P de s a t con el menor número de arista no repite ningún vértice en v. Si P repitiera un vértice v, se podría remover la porción de P entre porciones consecutivas de v, resultando en una trayectoria que no tiene un costo mayor y que contiene menos aristas. □

Por notación, utilicemos $OPT(v, i)$ para designar el costo mínimo de la trayectoria $v - t$ que utiliza i aristas a lo más. Por 2.2.5, el problema de encontrar la trayectoria con menor costo entre s y t, se puede expresar como $OPT(s, n-1)$.

Utilizando el paradigma de programación dinámica es posible expresar $OPT(i, v)$ utilizando problemas más pequeños del siguiente modo [17]:

- Si la trayectoria P utiliza a lo más $i-1$ aristas, entonces $OPT(v, i) = OPT(v, i-1)$
- Si la trayectoria P usa i aristas, y la primera aristas es (v, w), entonces $OPT(i, v) = c(arista_{\langle v,w \rangle}) + OPT(w, i-1)$.

Lo cual conduce a la fórmula recursiva

Si $i > 0$ entonces

$$OPT(v,i) = min(OPT(v,i-1), min_{w \in V}(OPT(w,i-1) + c(arista_{\langle v,w \rangle}))$$

El implementar la recurrencia anterior, utilizando programación dinámica, requeriría de espacio $O(n^2)$ debido a la matriz utilizada para almacenar los estimados para cada pareja v, i, sin embargo, puede realizarse utilizando espacio $O(n)$. Para ello, en lugar de almacenar cada pareja v, i, sólo se almacena el mejor estimado encontrado en el arreglo $costo_s[v]$, realizando un total de $n-1$ iteraciones, con lo cual nuestra ecuación quedaría del siguiente modo

$$costo_s[v] = min(costo_s[v], \min_{w \in V}(c(arista_{\langle v,w \rangle}) + costo_s[w]))$$

Esta última forma nos permite utilizar la operación Relajar del procedimiento 2.2 para cada uno de los vecinos $w \in V$ del vértice v.

Ahora podemos presentar el algoritmo Bellman-Ford, el cual se muestra enseguida [5]

```
BellmanFord(G,s)
(1) InicializarFuenteUnica(G, s)
(2) for i = 1 to |V| - 1
(3)     for each arista_<u,v> ∈ E
(4)             Relajar(u, v, arista_<u,v>)
(5) for each arista_<u,v> ∈ E
(6)     if costo_s[v] > costo_s[u] + c(arista_<u,v>)
(7)         return FALSE
(8) return TRUE
```

Listado 2.4: Algoritmo Bellman Ford

Para probar la correctez del algoritmo, se mostrará el efecto que tiene el relajar todas las aristas en la gráfica, después se mostrará que si no existen ciclos negativos en la gráfica G, el algoritmo calcula correctamente el costo de las trayectorias menos costosas para todos los vértices alcanzables desde el vértice origen.

Lema 2.2.6 (Las subtrayectorias de trayectorias más cortas son también trayectorias más cortas)**.** *Dada una gráfica dirigida $G = (V, E)$ con función de costo $c : E \to \mathbb{R}$, sea $p = v_0, v_1, \ldots, v_k)$ una trayectoria más corta del vértice v_0 al vértice v_k, y para cualquier i y j tal que $0 \leq i \leq j \leq k$, sea $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$ una subtrayectoria de p del vértice v_i al vértice v_j. Entonces, p_{ij} es una trayectoria más corta de v_i a v_j [5].*

Demostración. Si se descompone la trayectoria p en $v_0 \overset{p_{0i}}{\rightsquigarrow} v_i \overset{p_{ij}}{\rightsquigarrow} v_j \overset{p_{jk}}{\rightsquigarrow} v_k$, entonces se tiene que $c(p) = c(p_{0i}) + c(p_{ij}) + c(p_{jk})$. Ahora asúmase que existe una trayectoria p'_{ij} de v_i a v_j con costo $c(p'_{ij}) < c(p_{ij})$. Entonces, $v_0 \overset{p_{0i}}{\rightsquigarrow} v_i \overset{p'_{ij}}{\rightsquigarrow} v_j \overset{p_{jk}}{\rightsquigarrow} v_k$ es una trayectoria de v_0 a v_k cuyo costo $c(p_{0i}) + c(p'_{ij}) + c(p_{jk})$ es menor que $w(p)$, lo cual contradice nuestra suposición de que p es la trayectoria más corta de v_0 a v_k. □

Lema 2.2.7 (Propiedad relajación de trayectoria)**.** *Sea $G = (V, E)$ una gráfica dirigida, con función de costo $c : E \to \mathbb{R}$, y sea $s \in V$ un vértice designado como el origen. Considérese una de las trayectorias más cortas $p = (v_0, v_1, \ldots v_k)$ de $s = v_0$ a v_k. Si G es inicializada por InicializarFuenteUnica(G,S) y después una secuencia de pasos de relajación ocurren que incluyen, en orden, la relación de las aristas $(v_0, v_1), (v_1, v_2), \ldots (v_{k-1}, v_k)$, entonces $costo_s[k] = OPT_{s,v_k}$ después de estos pasos y se mantiene en todos los momentos posteriores. La propiedad se mantiene no importa que otras operaciones de relajación sean realizadas sobre las aristas $\notin p$, incluso aquellas que son intercaladas con la relajación de las aristas de p [5].*

Demostración. Por inducción, para el caso base $i = 0$, y antes de que cualquier arista en p haya sido relajada, por el procedimiento de inicialización $costo_s[v_0] = costo_s[s] = 0 = OPT_{s,s}$. Por el lema de la propiedad superior 2.2.3, el valor $costo_s[s]$ no cambia después de la inicialización. Para el paso inductivo, asumimos que $costo_s[v_{i-1}] = OPT_{s,v_{i-1}}$, y examinaremos que ocurre cuando se relaja la arista (v_{i-1}, v_i).

Véase que $costo_s[v_i] \leq costo_s[v_{i-1}] + c(arista_{\langle v_{i-1}, v_i \rangle})$ justo después de relajar la arista $arista_{\rangle v_{i-1}, v_i \langle}$. Ya que si $costo_s[v_i]$ fuera mayor, al relajar la arista $costo_s[v_i] = costo_s[v_{i-1}] + c(arista_{\langle v_{i-1}, v_i \rangle})$ y en caso de que fuera menor la desigualdad se mantendría.

Ahora bien, por nuestra hipótesis inductiva $costo_s[v_{i-1}] = OPT_{s,v_{i-1}}$ y por tanto $costo_s[v_i] \leq OPT_{s,v_{i-1}} + c(arista_{v_{i-1}, v_i})$ y ya que las subtrayectorias de las trayectorias menos costosas, son a su vez menos costosas (lema 2.2.6) $costo_s[v_i] = OPT_{s,v_i}$. □

Lema 2.2.8. *Sea $G = (V, E)$ una gráfica dirigida, con origen s y con una función de costo $c : E \to \mathbb{R}$, y asúmase que G no contiene ciclos negativos que sean alcanzables desde s. Entonces, después de $|V| - 1$ iteraciones del ciclo* ***for*** *de las líneas 2-4 del algoritmo 2.4, se tiene que $v.d = OPT_{s,v}$ para todos los vértices v que son alcanzables desde s [5].*

Demostración. Considérese cualquier vértice v que es alcanzable desde s, y sea $p = \langle v_0, v_1, \ldots, v_k \rangle$, donde $v_0 = s$ y $v_k = v$, sea cualquier trayectoria con la ruta más corta de s a v. Debido a que las trayectorias más cortas son simples, p tiene a lo más $|V| - 1$ aristas, y por tanto $k \leq |V| - 1$. Cada una de las $|V| - 1$ iteraciones de el ciclo **for** de las líneas 2-4 relaja las $|E|$ aristas. De las aristas relajadas en la i-ésima iteración, para $i = 1, 2, \ldots, k$, esta (v_{i-1}, v_i). Por la propiedad de relajación de trayectoria, se tiene, $v.d = v_k.d = OPT_{s,v_k} = OPT_{s,v}$. □

Teorema (Correctez del algoritmo Bellman-Ford) 1. *Sea el algoritmo BELLMAN-FORD ejecutado en una gráfica ponderada $G = (V, E)$ un vértice s considerado origen y función de costo $c : E \to \mathbb{R}$. Si G no contiene ciclos con pesos negativos que sean alcanzables desde s, entonces el algoritmo regresa $TRUE$, y se tiene $costo_s[v] = OPT_{s,v}$ para todos los vértices $v \in V$. Si G contiene ciclos con negativos alcanzables desde s, entonces el algoritmo regresa FALSE [5].*

Demostración. Supóngase que la gráfica G no contiene ciclos negativos que sean alcanzables desde el origen s. Se probará primero que al terminar, $costo_s[d] = OPT_{s,v}$ para todos los vértices $v \in V$. Si el vértice v es alcanzable desde s, entonces el lema 2.2.7 y el teorema 2.2.1 prueban esta aseveración. Si v no es alcanzable desde s $costo_s[v] = \infty$. Por lo tanto, la aseveración esta probaba. Al terminar, se tiene para todas las aristas $(u, v) \in E$,

$$\begin{aligned} costo_s[v] &= OPT_{s,v} \\ &\leq OPT_{s,u} + c(arista_{\langle u,v \rangle}) \qquad \text{(por la desigualdad del triángulo)} \\ &= costo_s[u] + c(arista_{\langle u,v \rangle}) \end{aligned}$$

y ninguna de las pruebas de la línea 6 del algoritmo 2.4 regresa FALSE. Por lo tanto, regresa TRUE.

Ahora supóngase que la gráfica G contiene un ciclo negativo que es alcanzables desde el origen s; sea $c = \langle v_0, v_1, \ldots, v_k \rangle$ este ciclo, donde $v_0 = v_k$. Entonces,

$$\sum_{i=1}^{k} w(v_{i-1}, v_i) < 0 \tag{2.2}$$

Asúmase para el propósito de contradicción que el algoritmo Bellman-Ford regresa TRUE. Entonces, $costo_s[v_i] \leq costo_s[v_{i-1}] + c(arista_{\langle v_{i-1}, v_i \rangle})$ para $i = 1, 2, \ldots, k$. Sumando las desigualdades alrededor del ciclo c se tiene

$$\begin{aligned}\sum_{i=1}^{k} costo_s[v_i] &\leq \sum_{i=1}^{k} (costo_s[v_{i-1}] + c(arista_{\langle v_{i-1}, v_i \rangle})) \\ &= \sum_{i=1}^{k} costo_s[v_{i-1}] + \sum_{i=1}^{k} c(arista_{\langle v_{i-1}, vi \rangle})\end{aligned}$$

Ya que $v_0 = v_k$, cada vértice en c aparece exactamente una vez en cada una de las sumatorias $\sum_{i=1}^{k} costo_s[v_i]$ y $\sum_{i=1}^{k} costo_s[v_{i-1}]$ y por tanto

$$\sum_{i=1}^{k} costo_s[v_i] = \sum_{i=1}^{k} costo_s[v_{i-1}]$$

Más aún, por el corolario $costo_s[v_i]$ es finito para $i = 1, 2, \ldots, k$. Por lo tanto

$$0 \leq \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

lo cual contradice la desigualdad 2.2. Concluimos que el algoritmo Bellman-Ford regresa TRUE si la gráfica G no contiene ciclos con pesos negativos alcanzables desde el origen, y FALSE de otra forma.

□

Complejidad El algoritmo 2.4 ejecuta un paso de inicialización en tiempo

$O(|V|)$. Después del paso de inicialización, el algoritmo relaja todas las aristas de la gráfica un total de $|V| - 1$ veces, por tanto la complejidad del algoritmo es $O(|V||E|)$.

2.2.5. Algoritmo Bellman-Ford distribuido

La versión que se presentó del algoritmo Bellman-Ford es apropiada cuando la red en la cual se encuentra se mantiene estática y no se presentan fallas. Cuando alguno de los enlaces se ve afectado por algún desperfecto, éste debe de notificarse a los demás ruteadores, de tal manera que pueda recalcular la trayectoria de menor costo, en caso de ser necesario.

Para hacer frente a este escenario se muestra la versión distribuida asíncrona del algoritmo [30]. La cual permite que cada ruteador ejecute el algoritmo de manera independiente a los demás y que reaccione a los cambios en los costos de las trayectorias ocasionados por fallas en los enlaces.

```
BellmanFord(G,s)
when START() is received
(1)  for each j ∈ neighbors_i
(2)        send UPDATE(length_i) to p_j

when UPDATE(length) is received from p_j
(3)   updated_i ← false
(4)   for each k ∈ {1,...n} \ {i}
(5)       if (costo_i[k] > c(arista_⟨i,j⟩) + costo_j[k])
(6)           costo_i[k] ← c(arista_⟨i,j⟩) + costo_j[k];
(7)           routing_i[k] ← j;
(8)           updated_i ← true
(9)  if (updated_i)
(10)    for each j ∈ neighbors_i
(11)       send UPDATE(length_i) to p_j
```

Listado 2.5: Algoritmo Bellman Ford distribuido

Inicialmente cada nodo (ruteador) conoce solamente el número e identidad de los procesos que conforman la gráfica. Adicionalmente, cada nodo mantiene el arreglo $costo_i[v]$ para todo $v \in V$, el cual tiene el costo de alguna trayectoria del nodo i al nodo v para todo $i, v \in V$. Inicialmente $costo_i[v] = \infty$ para todo $i \neq v$ y $costo_i[i] = 0$ para todo $i \in V$.

Al igual que en la versión centralizada, se permite que existan ciclos, pero se supone que no la suma de los costos de los enlaces que lo conforman es positiva. También se asume que la gráfica esta fuertemente conectada.

Para el análisis del algoritmo se asume que existe un tiempo t_0 en el cual el sistema se mantiene estable, después de un conjunto de cambios que

ocurren antes de t_0.

Uno de los hechos que puede notarse del algoritmo es el siguiente:

Lema 2.2.9 (Actualización constante)**.** *Los nodos nunca dejan de enviar información de actualización mientras existan cambios en sus estimados de las trayectorias más cortas [1].*

Demostración. Por el propósito de contradicción supóngase que existen cambios en los estimados de las trayectorias y que el algoritmo deja de enviar dichos cambios. La variable $updated_i$ es inicializada a *false* en la línea 3 del algoritmo 2.5. Si el algoritmo deja de enviar notificaciones, entonces la prueba de línea 9 falla, lo cual implica que la variable no cambio su estado después de la inicialización, sin embargo, hubo un cambio en el estimado de las trayectorias más cortas (suposición inicial), por lo cual la línea 8 es ejecutada y la variable $updated_i$ cambia a *true* y no existe otro punto en el cual la variable sea alterada, por tanto, la prueba de la línea 9 es afirmativa y se envían los cambios, una contradicción. □

Lema 2.2.10 (Permanencia actualizaciones)**.** *La información de actualización enviada por un nodo i tiene una permanencia finita en el sistema. Para un tiempo $t' > t_0$ existe un tiempo $\bar{t} > t'$ tal que los estimados $costo_i[]$ calculados en el nodo i antes del tiempo $\bar{t}$ no son recibidos por ningún nodo $v \neq i$, para todo $i \in V$ después del tiempo $\bar{t}$ [1].*

Demostración. Véase que el algoritmo envía $costo_i[]$ para todo $i \in V$ si y sólo si los estimados del nodo i cambian debido a algún vecino (líneas 5-6, programa 2.5. De ser el caso, el nodo i envía sus nuevos estimados a cada uno de sus vecinos (líneas 10-11). Si los vecinos no pueden mejorar sus estimados, la actualización termina ya que la prueba de la línea 9 falla. En caso contrario, el vecino j de i mejorará todos los estimados en los cuales $costo_j[k] > c(arista_{\langle j,i \rangle} + costo_i[k])$. En el peor caso, j tendrá que actualizar todo su arreglo $costo_j[k]$ y propagará la nueva información. Eventualmente todos los nodos tendrán la información de actualización en el tiempo $\bar{t}$, pero cuando alguno de ellos reciba esta información nuevamente $costo_j[k] > c(arista_{\langle j,i \rangle} + costo_i[k])$ será falso, ya que fue actualizada en un tiempo $t' < \bar{t}$, y la propagación terminará. □

Teorema 2.2.11 (Correctez algoritmo Bellman-Ford distribuido)**.** *Si el algoritmo Bellman Ford Distribuido es ejecutado en una gráfica $G(V, E)$ fuertemente conectada, con función de costo $c : E \to \mathbb{R}$ y sin ciclos negativos, el*

algoritmo termina con $costo_i[v] = OPT_{i,v}$ *para todo* $i, v \in V$ *para un* $\bar{t} > t_0$ *suficientemente grande, donde* t_0 *representa el tiempo en el cual se llama la función START.*

Demostración. El algoritmo supone la existencia de un tiempo inicial t_0 a partir del cual el sistema no cambia. Inicialmente para cada nodo $costo_i[v] = \infty$ para todo $i, v \in V$ y $costo_i[i] = 0$. El valor de $costo_i[v]$ representa el estimado de alguna trayectoria de i a v. Puesto que la gráfica es conectada, el valor de $costo_i[v]$ es finito. El valor de $costo_i[v]$ a lo largo del algoritmo es monotónicamente decreciente, ya que el valor sólo puede cambiar cuando $costo_i[v] > c(arista_{\langle i,v \rangle}) + costo_v[k]$. Por tanto, sólo puede disminuir un número finito de ocasiones (hasta que alcance su valor óptimo). Una vez que cada $v \in V$ alcance su valor óptimo, $costo_i[v] = OPT_{i,v}$ para todo $i, v \in V$ y por el teorema 2.2.1, estos representan el costo de las trayectorias más cortas. □

Mientras que existe un tiempo finito $\bar{t} > t_0$ para el cual el algoritmo 2.5 obtiene los costos de las trayectorias más cortas, los nodos no tienen forma de conocer este valor. Para ello podría utilizarse una versión síncrona del algoritmo, sin embargo, no se ahondará más en el tema, el lector interesado puede consultar [30] para mayor detalle.

2.3. Protocolo de Información de Ruteo (RIP)

El Protocolo de Información de Ruteo (RIP, siglas en inglés de *Routing Information Protocol*), es uno de los protocolos de dominio interno más utilizados y esto se debe en gran medida a que fue incluido en el sistema operativo Berkeley Software Distribution (BSD) [19].

RIP es un ejemplo de protocolo distancia-vector, la versión especificada en el RFC 1058 utiliza el número de brincos como función de costo. En nuestro análisis el costo asignado era entre los enlaces que unían diferentes nodos (ruteadores) [19]. En RIP los costos son por otra parte entre subredes. En RIP si a partir de un ruteador origen se puede alcanzar una red, el número de brincos es 1, si se requiere atravesar una red intermedia, el número de brincos es 2 y así de manera sucesiva.

El costo máximo que puede tener una trayectoria en RIP antes de que se considere inalcanzable es 15 saltos, limitando su uso en redes más grandes [19]. Recordando el funcionamiento de un algoritmo distancia-vector,

cada nodo envía sus costos estimados a los demás nodos a cada uno de sus vecinos, lo que les permite mejorar sus estimados. En RIP, esto se lleva por medio de actualizaciones de ruteo, las cuales ocurren cada 30 segundos aproximadamente.

Las actualizaciones permiten a cada ruteador el ejecutar el algoritmo Distancia-Vector y mejorar sus costos en caso de ser necesario. Los mensajes son automáticos, pero pueden enviarse en respuesta a algún cambio en la red, por ejemplo, que un enlace deje de funcionar.

Para ver el funcionamiento de RIP utilizaremos la figura siguiente

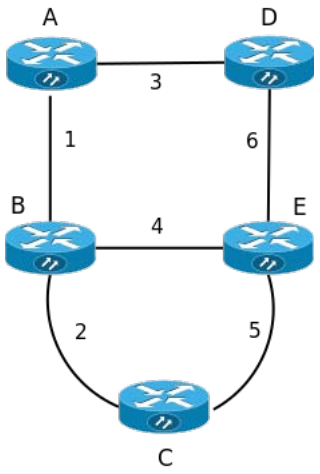

Figura 2.5: Protocolo RIP

Como puede observarse en la figura 2.5 cada uno de las aristas representa el costo de alcanzar una red en particular. Cada nodo tiene una vista parcial de la red, de acuerdo a los vecinos que puede alcanzar, la vista local del nodo A es la siguiente

Destino	Costo	Siguiente salto
B	1	-
D	3	-

Cuadro 2.1: Tabla de ruteo inicial para el nodo A

Como puede verse en la Tabla 2.1 el nodo A sólo sabe que puede alcanzar las redes B y D, pero desconoce la existencia de otras redes. Como se mencionó antes, en el protocolo cada nodo envía su información a los demás

nodos, en nuestro ejemplo, después de que A recibe la información de ruteo del nodo B actualiza su tabla de ruteo quedando como en la tabla 2.2

Destino	Costo	Siguiente salto
B	1	-
C	3	B
D	3	-
E	5	B

Cuadro 2.2: Tabla de ruteo del nodo A después de recibir la información del nodo B

Como puede observarse después que el nodo A recibió la información del nodo B, sabe que puede alcanzar el nodo C con un costo de 3 (el nodo B puede alcanzar al nodo C con un costo de 2, A puede alcanzar a B con un costo de 1, por lo cual el costo total es la suma de los valores anteriores) a partir de B y también puede alcanzar el nodo E.

Repitiendo este procedimiento para cada uno de los nodos se obtiene la Tabla 2.3 de costos siguiente

	A	B	C	D	E
A	0	1	3	3	5
B	1	0	2	4	4
C	3	2	0	6	5
D	3	4	6	0	6
E	5	4	5	6	0

Cuadro 2.3: Información de ruteo de los nodos

La versión 2 del protocolo definida en el RFC 2453, es compatible con la versión 1 y agrega las características de autenticación y uso de mascaras de red [4].

El funcionamiento de RIP se puede describir de manera general como sigue:

[4]La versión 1 del protocolo asumía el uso de direcciones con clase, lo cual agrupaba las redes en tipo A, B, C, D y E. Hoy día la estrategia de asignación de direcciones es sin clase y es conocida como Ruteo Interdominio sin Clases (CIDR, siglas en inglés de *Classless Interdomain Routing*)

- Cada nodo mantiene una tabla con una entrada para cada destino D en la red
- Se almacena la métrica M (distancia) y el siguiente salto N para cada D en la tabla
- De manera periódica cada nodo envía la información que tiene a cada uno de sus vecinos (el vector distancia).
- Para cada actualización que proviene del nodo vecino N' ejecutar el algoritmo Bellman-Ford y actualizar los costos estimados de ser el caso.

2.4. El protocolo de ruteo OSPF

El Protocolo Abierto de Primero la Ruta Más Corta (OSPF, siglas en inglés de *Open Shortest Path First Protocol*), es un protocolo de estado enlace cuyo funcionamiento general puede enunciarse de manera sencilla. La idea básica es que cada nodo conoce como alcanzar a sus vecinos directos y que esta información la puede transmitir al resto de los nodos de la red. Esto permite que cada uno pueda construir un mapa de la red y a partir de este mapa puede utilizar el algoritmo de Dijkstra para calcular las trayectorias menos costosas a los demás nodos [29].

OSPF no indica como deben de ser establecidos los costos de los enlaces, lo cual queda como responsabilidad del administrador de red. Los costos pueden ajustarse de acuerdo a los requerimientos que se busquen, por ejemplo, podría buscarse que las trayectorias menos costosas sean aquellas que utilizan los enlaces con mayor ancho de banda o algún criterio diferente [19].

Para transmitir la información de cada nodo OSPF utiliza el mecanismo de inundación. La idea básica es que cada nodo envía su información a sus vecinos directos, una vez que ellos la reciben, la transmiten a sus vecinos correspondientes a excepción del nodo del cual recibieron la información, y así de manera sucesiva. Al final todos los nodos tendrán la misma información.

El mecanismo de inundación tal como se describe debe de ajustarse para funcionar de manera apropiada. OSPF no utiliza el Protocolo de Control de Transmisión (TCP siglas en inglés de *Transmission Control Protocol*) y por tanto debe proveer su propio mecanismo de envío confiable. Por otra parte los mensajes no deben de permanecer indefinidamente en la red, por lo que se

agrega el campo de tiempo de vida (TTL siglas en ingles de *Time To Live*). Otra cuestión es como diferenciar entre un mensaje reciente y uno que llega tarde, para lo cual se agrega un número de secuencia a los mensajes enviados. Las características mencionadas son agregadas (entre otras) al mensaje de actualización que se envía, el cual se conoce como *Paquete de Estado Enlace* (LSP siglas en inglés de *Link-State Paquet*) [29].

Los mensajes de actualización no sólo se envían la primera vez que los nodos son activados, sino también cuando se detectan cambios en los enlaces de los vecinos a los cuales se encuentran conectados. También por cuestiones de robustez la información es enviada cada cierto tiempo de manera periódica aún cuando no existan cambios en los enlaces [19].

A continuación se presentan algunas de las características del protocolo OSPF:

- *Seguridad.* Los intercambios entre ruteadores que implementan OSPF puede realizarse utilizando autenticación. Esta capa adicional de seguridad evita que ruteadores maliciosos participen en el protocolo OSPF. La autenticación es opcional y debe ser habilitada por el administrador de red.

- *Múltiples trayectorias del mismo costo.* El algoritmo de Dijkstra tal como se presento sólo elige un camino aún cuando exista otro con el mismo costo, con OSPF es posible el uso de trayectorias múltiples si estas tienen el mismo costo, lo cual mejora la distribución del tráfico en la red.

- *Soporte para jerarquías dentro de un dominio único de ruteo.* OSPF provee funcionalidad para particionar un dominio interno (sistema autónomo) en diferentes subdominios o áreas como normalmente se conocen.

Con el diseño jerárquico, cada área ejecuta su propio algoritmo OSPF de estado-enlace. Dentro de cada área, uno o más *ruteadores de frontera de área* son responsables de encaminar paquetes fuera del área. Por último, exactamente un área OSPF en el AS es configurada como el área *backbone*. La actividad principal del área backbones es el encaminar tŕafico entre otras áreas en el AS. El backbone siempre contiene todos los ruteadores de frontera de las áreas en el AS y puede contener también ruteadores que no son de frontera. Por tanto, para enviar un paquete entre diferentes áreas, el paquete primero tiene que llegar al ruteador de frontera del área en la cual

fue originado, atravesar el backbone y llegar al ruteador frontera del área destino, el cual se encargará de enviarlo a su destino final.

OSPF tiene más características de las aquí descritas, el lector interesado puede consultar la sección de notas y lecturas recomendadas para ahondar en los temas expuestos.

2.5. Notas y lecturas recomendadas

Los algoritmos en gráficas son variados y se aplican en diversas áreas, para ver aplicaciones concretas puede consultarse las obras de Cormen [5] y Tardos [17]. Estudios más detallados pueden encontrarse en [6].

El teorema de optimalidad de las trayectorias más cortas se presenta en [32]. La desigualdad del triángulo y la propiedad del límite superior que se presentan aquí son adaptaciones de las presentadas en [5].

Las pruebas de correctez de Dijkstra y Bellman-Ford son adaptaciones de las presentadas en [5], [17] y [32].

La versión distribuida del algoritmo Bellman-Ford fue adaptada de la presentada en [30]. La prueba de correctez que se presenta de esta versión, esta inspirada por los trabajos de [1] y [30].

Los protocolos de ruteo RIP y OSPF son sólo algunos de los muchos que existen y la presentación ha sido somera. Los trabajos de [19], [29] y [34], presentan con mayor detalle lo que son las redes, las dificultades en su implementación y diferentes protocolos utilizados en ellas. Para ahondar en el tema de ruteo, los algoritmos y protocolos utilizados, se puede consultar la obra de [25].

Capítulo 3

El protocolo BGP

Para manejar la complejidad de redes como Internet, los recursos que la conforman como los huéspedes, ruteadores y mecanismos de transmisión se agrupan en dominios administrativos conocidos como sistemas autónomos (AS). Por cuestiones económicas, un dominio proveedor podría permitir solamente el paso de ciertas clases de tráfico a través de su infraestructura e impidiendo otras.

El Protocolo de Puerta Exterior (BGP, siglas en inglés de *Border Gateway Protocol*) fue diseñado para diseminar la información de alcanzabilidad entre diferentes AS y para proporcionar la flexibilidad necesaria a los AS de indicar que tráfico desean permitir y bajo que circunstancias.

A lo largo de este capítulo veremos que el protocolo BGP no es del todo estable y que dependiendo de diversos factores puede presentar problemas de convergencia u oscilaciones, esto nos permitirá plantearnos diversas preguntas y nos preparará para abordar el tema del capítulo siguiente.

3.1. Introducción

El protocolo BGP tiene como principal objetivo el comunicar información acerca de las redes que se encuentran dentro de un sistema autónomo a los demás sistemas autónomos que conforman Internet. Para llevarlo a cabo, cada sistema autónomo define uno o varios ruteadores de frontera, los cuales se comunican por medio de una sesión TCP utilizando el puerto 179.

Se requiere que la comunicación entre los ruteadores de frontera de los AS se mantenga activa, con objeto de enviar las actualizaciones correspondientes.

Si la sesión TCP falla por algún motivo, se requiere que los AS involucrados dejen de utilizar los datos aprendidos por medio de la sesión.

Por lo que hemos visto hasta ahora, sabemos que la diseminación de información de alcanzabiliad puede lograrse utilizando un algoritmo estado-enlace o un algoritmo distancia vector. El primero, presenta problemas en cuanto a escalabilidad se refiere ya que cada AS requiere tener un mapa completo de la red, lo cual resulta prohibitivo en redes como internet. Por otra parte, el algoritmo estado-enlace presenta patologías como ciclos o conteos a infinito que lo hacen poco deseable [19]. [29].

Debido a ello, BGP utiliza un protocolo de ruteo *vector trayectoria*, el cual es semejante al protocolo distancia-vector pero permite evitar los ciclos y donde el intercambio de información se realiza por medio de sesiones confiables de comunicación. En la siguiente sección de ahondará en éste protocolo.

En un protocolo vector-trayectoria cada nodo recibe el vector de distancias de sus vecinos a cierto destino, pero además de ello, recibe la trayectoria que recorre cada vecino para llegar a su destino. La información de la trayectoria es útil ya que permite la detección de ciclos [25].

En BGP la métrica utilizada para asignar los costos es el número de saltos, es decir, el número de redes que se requiere atravesar para llegar al destino y no el número de ruteadores, el cual puede ser mucho mayor.

El término *red* en el contexto de BGP tiene un significado más específico, y en realidad se refiere a *redes definidas por prefijo IP*. En el esquema de asignación de direcciones de Internet conocido como Ruteo de Interdominio Sin Clase (CIDR, siglas en inglés de *Classless Interdomain Routing*), las direcciones IP de 32 bits se escriben en notación decimal de la forma $a.b.c.d/x$, donde x indica el número de bits de la primera parte de la dirección y es a lo que se conoce como el prefijo de red [19]. En nuestro estudio de BGP utilizaremos el término red, en lugar de prefijo IP, por claridad.

Utilizando terminología de BGP, a la comunicación que establecen dos AS por medio de sus ruteadores de frontera, se le conoce como *sesión BGP*. Y a tales ruteadores también se les conoce como *iguales BGP* (en inglés, *BGP peers*).

Aunque sólo se ha mencionado la comunicación entre AS, también existe una de manera interna cuando existe más de un ruteador de frontera dentro del AS, la cual permite diseminar la información de ruteo a cada uno de ellos. A la primera se le conoce como *sesión externa BGP* (eBGP), y a la segunda como *sesión interna BGP* (iBGP) [19].

Para diferenciar a cada AS se utiliza un identificador único global de 16

bits, conocido como el *número del sistema autónomo*. El valor del identificador puede encontrarse entre 1 a 65535, pero sólo se ocupa el rango de 1 a 64511 de manera pública, mientras que los valores restantes se reservan como números privados del AS [25].

Aunque uno podría visualizar cada AS como un Proveedor de Servicios de Internet (ISP, siglas en inglés de Internet Servide Provider), esto no es siempre el caso. Un ISP puede contener varios AS y de manera inversa un AS puede estar conformado por varios prefijos IP proporcionados por diferentes proveedores [25].

3.2. Operación de BGP

Para poder llevar a cabo su tarea, BGP utiliza mensajes y temporizadores. Los primeros permiten indicar cuando agregar una nueva información o cuando descartar la que ya no es válida, los segundos por su parte, permiten verificar el estatus de la conexión y ayudan descongestionar el tráfico de red con los mensajes del protocolo.

El protocolo BGP define cuatro tipos de mensajes: OPEN, UPDATE, KEEPALIVE y NOTIFICATION, además de un mensaje opcional ROUTE-REFRESH [25].

El mensaje OPEN es el primero que se envía después de establecer la sesión TCP entre los ruteadores BGP de frontera, el cual tiene como objetivo el establecer varias parámetros de operación y descubrir las capacidades de los participantes.

Por medio del mensaje UPDATE dos ruteadores BGP frontera intercambian la información de los prefijos IP. Cuando alguno de los ruteadores tiene una nueva ruta que anunciar, el mensaje es generado y enviado a su igual BGP (*bgp peer*). Una de las responsabilidades que tienen los ruteadores frontera cuando notifican de una nueva ruta a alguno de sus iguales BGP, es la de indicar cuando esta ruta deja de estar disponible, de forma que el ruteador que se encuentra al otro extremo de la comunicación pueda retirarla de su tabla de ruteo.

El mensaje KEEPALIVE se envía de manera periódica por medio de los temporizadores y permite a los participantes de la sesión eBGP el saber que la sesión sigue activa, el tiempo de envío entre mensajes se establece por medio del mensaje OPEN.

El mensaje NOTIFICATION es enviado para cerrar una sesión BGP, ya

sea que exista algún error o debido a que el tiempo de espera entre mensajes UPDATE o KEEPALIVE exceda el tiempo de espera.

Por último el mensaje ROUTE-REFRESH es opcional y tiene como objetivo el solicitar todos los prefijos IP de su igual BGP. Debido a que no es obligatorio, el mensaje OPEN debe preguntar si es soportado antes de iniciar con el envío de los demás mensajes.

3.3. Atributos de trayectoria y proceso de selección de rutas

Como se mencionó previamente, un protocolo vector trayectoria no sólo envía el costo a cada destino visto por cada nodo en sus mensajes de actualización, sino que además envía la trayectoria utilizada.

Adicionalmente a estos valores, BGP agrega varios atributos de trayectoria en los mensajes de actualización. Tales atributos son utilizados en el proceso de selección de rutas y son éstos los que ocasionan que el protocolo no se defina como un protocolo de ruta más corta. A continuación se listan los atributos de trayectorias [25]:

- *Origin* Indica el mecanismo por el cual un prefijo IP es anunciado en BGP.
- *AS-PATH* Contiene la secuencia de identificadores AS que el mensaje a recorrido al anunciar la ruta.
- *NEXT-HOP* Dirección IP del ruteador del siguiente brinco hacia el destino cuyo prefijo IP fue anunciado en el mensaje UPDATE.
- *MULTI-EXIT-DISCRIMINATOR (MED)* Utilizado cuando existe más de un enlace de comunicación hacia un AS vecino. El enlace con el menor valor de MED es preferido.
- *LOCAL-PREF* Usado de manera interna en el AS. Puede ser utilizado para que todos los ruteadores BGP elijan el mismo enlace de salida hacia un prefijo IP en particular.

Los atributos que se muestran no son todos los posibles, ya algunos pueden ser opcionales al momento de generar los mensajes de actualización o no todas las implementaciones de BGP los soporten.

Una vez que un ruteador frontera recibe información de prefijos IP, utiliza un proceso de selección en el cual toma en consideración sus políticas de importación, filtrado, determinación de mejor ruta y de exportación.

En conjunto, todas estas reglas explican como las rutas a ciertos prefijos IP son seleccionados y cuales son notificados a otros ruteadores de frontera, y es aquí donde se utilizan los atributos de trayectoria que se mencionaron anteriormente.

El proceso de selección es el siguiente [25]:

1. Si el prefijo IP no es deseado debido a políticas de importación, descartar la ruta.
2. Aplicar la política local preconfigurada o elegir aquella con el valor más alto de LOCAL-PREF.
3. Si existe más de una ruta a un prefijo IP, seleccionar la ruta que fue originada de manera local.
4. Si aún existe más de una ruta hacia el prefijo IP, seleccionar aquella con el menor número de AS listados en el atributo AS-PATH.
5. Si aún existe más de una ruta, seleccionar aquella con el valor más pequeño del atributo ORIGIN.
6. Si aún existe más de una ruta, seleccionar la ruta con el valor más pequeño del atributo MULTI-EXIT-DISCRIMINATOR.
7. Si aún existe más de una ruta, seleccionar la ruta recibida por eBGP sobre iBGP.
8. Si aún existe más de una ruta, seleccionar la ruta con el costo más pequeño del atributo NEXT-HOP.
9. Si aún existe más de una ruta, seleccionar la ruta aprendida por un vecino eBGP con el valor más pequeño de identificador BGP.
10. Si aún existe más de una ruta, selecciona la ruta del vecino iBGP con el identificador más pequeño.

Las mejores rutas hacia un destino IP resultan del proceso descrito y son las almacenadas por cada uno de los ruteadores BGP.

3.4. Convergencia

La *convergencia* de un sistema de red podemos entenderla como el estado de no presencia de fluctuaciones en las tablas de ruteo y donde los cálculos de selección de rutas se mantienen constantes siempre que no existan cambios en la topología del sistema [29].

En el caso de BGP, hasta hace poco tiempo se asumía que el mecanismo de cálculo de rutas siempre convergía, sin importar los criterios de selección de rutas que presentaran los AS involucrados. Sin embargo, los estudios realizados por Varadhan [16] muestran que este no es el caso y que por el contrario existen criterios que ocasionan oscilaciones persistentes, los cuales se presentan aún cuando no existan cambios en la topología de red.

Aunque los resultados son reproducibles sólo de manera experimental y se deben a un proceso de retroalimentación"(gráficas de retorno), las pruebas presentadas en su artículo no dependen del tiempo de propagación de rutas o la velocidad del cálculo de rutas, sino que están basados en la topología y políticas de selección de rutas. Incluso en casos donde existe una asignación de rutas estables se puede presentar el caso de oscilaciones debido a la configuración inicial.

Las políticas basadas en la distancia más corta o en siguiente brinco (*hop*) por otra parte son seguras y siempre convergen, sin embargo, son poco expresivas y por ende poco deseables.

Varadhan sugiere que el análisis estático de las políticas de ruteo podría evitar problemas con la convergencia, pero no da resultados al respecto. Griffin [13] por su parte investiga la complejidad computacional de un análisis estático en las políticas de ruteo con el fin de determinar problemas de convergencia.

Entre las preguntas que Griffin busca responder son aquellas de alcanzabilidad, de resolución, tolerancia a fallas de los enlaces y unicidad. Desafortunadamente los resultados que presenta muestran que la complejidad computacional de los resultados es NP o NP-completo [1].

[1]De manera general, la teoría de complejidad computacional busca clasificar los algoritmos en aquellos que pueden ser resueltos de manera "eficiente" y aquellos que no. La eficiencia puede considerar diversos aspectos como tiempo o espacio, aunque generalmente el factor que se toma en cuenta es el tiempo. Un algoritmo se considera, de manera general, eficiente si esta acotado por alguna función polinomial que depende del tamaño de la instancia del problema que desea resolverse, si no existe tal función se considera que el algoritmo tiene tiempo exponencial. Un algoritmo de tiempo exponencial puede tardar

En un trabajo posterior Griffin [12] presenta el problema de las trayectorias estables y el protocolo simple de vector trayectoria (SPVP, siglas en inglés de *simple path vector protocol*) como un algoritmo distribuido para resolverlo. La intención de SPVP es capturar la semántica de BGP para el análisis de su convergencia.

Sus resultados muestran que la existencia de una solución única que converge depende que la instancia del SPVP no contenga una rueda de disputa, la cual representa políticas que entran en conflicto y ocasionan oscilaciones en el sistema. No obstante los resultados, la complejidad computacional de decir si una instancia de SPP es segura o robusta queda como problema abierto en su artículo.

Sin embargo, una de los puntos importantes de su artículo, se encuentra en la intuición de modelar el problema de las trayectorias con la teoría de juegos. Intuitivamente cada nodo puede modelarse como un jugador con un espacio de acción del cual puede elegir una trayectoria, que sea la mejor posible de acuerdo a sus políticas. Un tema que se ahondará en el capítulo siguiente.

3.4.1. Prácticas que garantizan convergencia

Cuando Internet tiene divergencia se presentan situaciones poco deseables como oscilaciones en el cálculo de rutas o problemas de conectividad, donde aún cuando existe una trayectoria entre dos AS, problemas en las políticas de los AS impiden a BGP el descubrir tal trayectoria. Tales fallas afectan por ende a los usuarios, quienes ven una degradación en el desempeño de la red y en las aplicaciones que se ejecutan sobre ella.

Dos problemas principales existen en el análisis de las políticas de los AS para determinar conflictos, el primero de ellos es ocasionado por el ocultamiento de información entre AS por cuestiones comerciales, ya que la información de las políticas no se publica de manera abierta, el segundo de ellos es más complejo, ya que como se mencionó en la sección anterior un análisis

días, años, centurias o más, por lo cual no se considera práctico. Para ayudar en el estudio de la complejidad computacional, se utiliza una clasificación, en ella podemos encontrar las clases P, NP y NP-difícil (entre otras más). La clase P engloba aquellos algoritmos que pueden resolverse en un tiempo polinomial, mientras que NP y NP-difícil engloban aquellos problemas de los cuales no se conoce un algoritmo de orden polinomial. El tema es extenso y aquí sólo se expone de manera somera para poner en contexto los resultados presentados. El lector interesado puede encontrar más información en [9]

estático es un problema NP-completo.

Una solución factible es por aquella que otorga flexibilidad en la cantidad de información que se publica de las políticas de cada AS y que no requiere un análisis estático de las políticas involucradas. Gao y Rexford [8] proponen un método constructivo que garantiza estabilidad y convergencia para el ruteo en Internet.

Su método se basa en las relaciones que pueden presentar los AS entre sí. Las relaciones pueden ser de consumidor-proveedor, igual-igual y relaciones de soporte en caso de fallas (en inglés *backup*).

Con objeto de establecer preferencias entre los diversos tipos de relaciones, el modelo de BGP que presentan es similar al utilizado por Griffin [12] pero aumentado para considerar el atributo MED y el intercambio de información por medio de iBGP.

De manera general el método consiste en que cada AS selecciona primero la trayectoria exportada por algún AS consumidor que la exportada por alguno de sus iguales (*peer* en inglés). En caso de no tener iguales y tener que seleccionar una de entre un consumidor y un proveedor, seleccionar primero la de un consumidor. De igual manera la relaciones deben poder modelarse por medio de una gráfica dirigida acíclica (DAG por sus siglas en inglés).

En caso de que los AS tengan enlaces auxiliares para el soporte de fallas, los lineamientos anteriores tienen preferencia. Si no es posible el elegir una trayectoria, la preferencia establecida de la trayectoria auxiliar se establece por medio de un valor fijado previamente.

Con el método de Gao, los AS no tienen que publicar sus políticas de selección de ruta y sólo se pide el indicar las relaciones que cumplen con otros AS. También el modelo es resistente a fallas (por medio de las trayectorias auxiliares) y garantiza la convergencia en este caso.

Otro método constructivo desarrollado por Cittadini et. al. [4] busca una solución heurística para el análisis estático de las trayectorias de los AS que garantizan convergencia, para ello utiliza una estrategia glotona (*greedy* en inglés) que construye de manera iterativa las trayectorias de cada AS a un prefijo IP determinado.

De manera general, se parte de un conjunto S que se sabe estable y que siempre converge, después los AS vecinos seleccionan de su conjunto de trayectorias posibles a aquella que tiene como su siguiente brinco a un elemento del conjunto S y que garantice convergencia, en cada iteración el algoritmo descarta las trayectorias que entran en conflicto (reduciendo el espacio de búsqueda) y selecciona para la siguiente etapa sólo aquellas

trayectorias que no puedan ser expresadas por las ya presentes en S.

Como ya se menciono anteriormente el análisis estático es un problema NP-completo, sin embargo, Cittadini menciona que su método de manera experimental muestra un desempeño aceptable y por tanto se propone utilizarlo como un auxiliar en la creación de políticas de ruteo.

Los métodos de Gao y Cittadini son constructivos y aunque demuestran que es posible la construcción de una red de AS que utilizando BGP sea estable y siempre converja, tienen algunas limitantes. En el caso de Gao, sólo se consideran tres tipos de relaciones que pueden cumplir los AS: consumidor-proveedor, igual-igual y de respaldo en caso de falla. Tanto Gao como Cittadini por otra parte modelan de manera implícita cada AS como un sólo nodo, pero esto no siempre es el caso tal como lo muestra Roughan et. al. [31], por lo cual existen situaciones que pueden estar fuera de los métodos indicados.

La utilización de un modelo de BGP sobre otro obedece a los criterios que desean estudiarse y siempre se busca un compromiso entre un modelo detallado pero complejo y uno que contenga los elementos necesarios para el estudio en cuestión pero que posiblemente no contemple algunos casos. Es importante señalar que en el caso del análisis de la convergencia, los resultados presentados no se ven afectados por el modelo de AS presentado y por tanto siguen siendo válidos [31].

3.5. Notas y lecturas recomendadas

Si bien la descripción de BGP que se presento fue somera, contiene los elementos esenciales para su estudio. Una explicación más detallada se puede encontrar en los textos de [19], [29], [25] y [18].

Una descripción más detallada sobre los problemas de convergencia puede encontrarse en el trabajo de Varadhan [16] y los trabajos de Griffin [11], [13] y [12].

Como se menciono, la convergencia tiene repercusiones para el usuario común y además de los problemas que pueden generarse por causa de políticas en conflicto, el tiempo de convergencia puede verse afectado por la topología como lo muestra Labovitz [22] y [21].

Además de los trabajos de Gao y Cittadini que estudian como garantizar políticas que sean estables y garanticen convergencia, se puede consultar los trabajos de Wang et. al. [23] y Mahajan [24]. Para un descripción de lo que es y cómo funciona una estrategia glotona se puede consultar los textos de Cormen [5] y Tardos [17].

Capítulo 4

Cómputo Distribuido con Heurísticas Adaptativas

El desarrollo de Internet ha sido de suma importancia y sus constantes retos presentan un campo fértil para el estudio de nuevos algoritmos y desarrollo de nuevas técnicas para su estudio. Papadimitriou ya había señalado con anterioridad que el uso de la teoría de juegos y la teoría económica podría proveer de herramientas útiles para llevar a cabo esta tarea [28].

La intuición que subyace en la aplicación de estas disciplinas en el análisis de redes como Internet, se encuentra en los factores que determinan su complejidad tanto como su objetivo, que en muchos casos obedecen a situaciones económicas. Como se menciono en el capítulo 3, el protocolo BGP no utiliza la métrica de la distancia más corta para el cálculo de las trayectorias, sino que aplica un conjunto de políticas para elegir la mejor de ellas.

Las políticas aplicadas permiten una gran flexibilidad en el proceso de selección y varias de ellas obedecen a factores económicos en lugar de problemas técnicos, por ejemplo, que el tráfico que se permita sea sólo de aquellos AS con los cuales se tiene un contrato o convenio de colaboración. Cuestiones que dan lugar a lo que Papadimitriou refiere como "complejidad socio-económica" [28]. En consecuencia, el uso de herramientas matemáticas que estudian cuestiones económicas y competencia entre diferentes entidades parece lo más apropiado.

En éste capítulo se muestra un enfoque al estudio de sistemas como Internet, cuyo comportamiento puede modelarse utilizando las disciplinas mencionadas previamente y a partir de ello se presentará un resultado que permite caracterizar la convergencia de tales sistemas a un punto de equilibrio.

4.1. Introducción

La existencia de ambientes dinámicos conformados por nodos que realizan cómputos o toman decisiones no es atípica, ejemplos de ellos son sistemas como Internet o las arquitecturas multiprocesador. La acción de cada nodo es simple, natural y miope en muchas de las ocasiones, es decir, aplica una heurística, pero también es adaptativa, ya que responde de manera constante y autónoma a las acciones de los demás [15].

Tales heurísticas adaptativas son simples y espontáneas debido a restricciones impuestas en el diseño de los ambientes, que requieren respuestas rápidas y limitadas por los recursos que pueden disponer. La teoría de juegos ha realizado estudios al respecto, pero de manera general los resultados obtenidos se fundamentan en un comportamiento síncrono de los nodos que conforman el sistema.

La asincronía introduce dificultades que no se presentan en ambientes síncronos, por ejemplo, la falla de algún nodo puede impedir que la computación realizada por el sistema termine [7]. Tales dificultades agregan por tanto complejidad al estudio de los sistemas.

En éste capítulo se mostrará que una clase grande y natural de heurísticas adaptativas puede fallar en converger a un punto de equilibrio en ambientes asíncronos, aún si se garantiza que los nodos y los canales de comunicación se encuentran libres de fallas, para ello se presentan algunos de los resultados que se encuentran en el trabajo de Jaggard et. al. [15], seleccionando aquellos relevantes al tema de la convergencia y que se exponen con mayor detalle que en el trabajo original.

4.2. El Modelo

Se necesita un modelo general que sea capaz de describir a los ambientes antes mencionados, y para ello se requiere un conjunto de definiciones. Se tiene un *sistema de interacción* el cual esta conformado por un conjunto de n *nodos computacionales*. Cada nodo $i \in 1, \ldots, n$ tiene un conjunto de acciones que puede ejecutar, al cual se se conoce como *espacio de acción* A_i. El conjunto de todos los espacios de acción del sistema, denotado como A es igual a $A = \times_{i\in[n]} A_i$, donde $[n] = \{1, \ldots, n\}$. El espacio de acción del sistema que no contiene el espacio de acción del nodo i se expresa por medio de $A_{-i} = \times_{j\in[n]\setminus\{i\}} A_j$. Cada elemento del espacio de acción del nodo i puede

ser elegido con cierta probabilidad, $\delta(A_i)$ representa el conjunto de todas las distribuciones de probabilidad sobre las acciones en A_i.

Un *planificador* indica en que momentos son activos los nodos computacionales a lo largo del tiempo, formalmente un planificador es una función σ que mapea cada $t \in \mathbb{N}_+ = \{1, 2, \dots\}$ a un conjunto no vacío de nodos computacionales: $\sigma(t) \subseteq [n]$. Decimos que un planificador es justo si para una secuencia infinita de acciones, cada nodo i se activa infinitamente. Decimos que un planificador σ es $r - justo$ (r-fair), si cada nodo es activado al menos en cada secuencia consecutiva de r pasos.

Las interacciones de los nodos a lo largo del tiempo definen la *historia* del sistema, lo cual se representa por medio de H_t. Sea $H_0 = \emptyset$, y sea $H_t = A^t$ para cada $t \geq 1$. H_t representa una historia posible de interacción de los nodos en el paso de tiempo t. Para cada nodo i, existe una secuencia infinita de funciones $f_i = (f_{(i,1)}, f_{(i,2)}, \dots, f_{(i,t)}, \dots, \dots)$ tal que, para cada $t \in \mathbb{N}_+$, $f_{(i,t)} : H_t \to \Delta(A_i)$; a tal f_i le llamaremos la *función de reacción* del nodo i. La función f_i tiene como propósito el capturar la forma de responder del nodo i a la historia de interacción en cada paso de tiempo.

Las funciones de reacción pueden presentar varias restricciones, aquí se presentan cinco de ellas: determinismo, auto-independencia, recuerdo acotado, estacionariedad y sin historia.

1. **Determinismo:** una función de reacción f_i es determinista si, para cada entrada, f_i da como resultado una sola acción (esto es, una función de probabilidad donde una única acción en A_i tiene probabilidad 1).

2. **auto-independencia:** una función de reacción f_i es *auto-independiente* (*self-independent* en inglés) si las propias acciones del nodo i (pasadas o presentes) no afectan el resultado de f_i.

3. **k-memoria y estacionariedad**: un nodo i tiene *k-memoria* si su función de reacción f_i sólo depende a lo más de los k pasos de tiempo más recientes. Decimos que una función de reacción con k-memoria es *inmutable* (*stationary*, en inglés) si el contador t no tiene importancia (esto es, no importa en que instante de tiempo t se decidan tomar las últimas k-muestras).

4. **Sin historia:** Una función de reacción f_i es sin historia (historylessness) si f_i es 1-recuerdo y estacionaria, esto es, si f_i sólo depende en las acciones más recientes de i y de los vecinos de i.

Con las definiciones anteriores es posible definir la *dinámica* del sistema, la cual representa la manera en que el sistema evoluciona a lo largo del tiempo. De manera informal, existe un estado inicial (historia de interacción) a partir del cual la interacción del sistema se desenvuelve, y para cada paso discreto de tiempo, algún subconjunto de los nodos reacciona a la historia de interacción pasada.

Sea $s^{(0)}$, que será llamado el *"estado inicial"*, un elemento en H_w, para algún $w \in \mathbb{N}_+$. Sea σ un planificador. Se describirá el significado de "$(s^0, \sigma) - dynamics$". La evolución del sistema comienza al tiempo $t = w + 1$, donde cada nodo $i \in \sigma(w + 1)$ de manera simultánea elige una acción de acuerdo como responde su función de reacción al tiempo $(w+1)$. Sea $s^{(1)}$ el elemento en H^{w+1} para el cual las primeras w coordenadas (n-tuplas con las acciones de los nodos) son como en s^0 y las últimas coordenadas son la n-tupla de las acciones de los nodos que fueron activados (*realized* nodes) al final del paso de tiempo $t = w+1$. Siguiendo el mismo argumento, para cada paso de tiempo $t > w + 1$, cada nodo en $\sigma(t)$ actualiza sus acciones de acuerdo a su función de reacción $f_{(i,t)}$, basado en la historia pasada $s^{(t-w-1)}$ y las acciones de los nodos activados en el tiempo t, combinadas con $s^{(t-w-1)}$, definen la historia de interacción al final del paso de tiempo t, s^{t-w}.

Decimos que las acciones de los nodos *convergen* bajo la dinámica de (s^0, σ) si existe algún $t_0 \in \mathbb{N}_+$, y algún conjunto de acciones $a = (a_1, \ldots, a_n)$, tal que, para todo $t > t_0, s^{(t)} = a$. Si este es el caso, se dice que la dinámica del sistema *converge* a a, y a es llamado un "*estado estable*"(para la dinámica de $(s^{(0)}, \sigma)$. La definición anterior nos dice que existe un estado en el sistema que una vez alcanzado no cambia.

Decimos que el sistema de interacción es *convergente* si, para todos los estados iniciales $s^{(0)}$ y planificadores *justos* (fair schedule) σ, la dinámica de $(s^{(0)}, \sigma)$ converge. Decimos que el sistema es *r-convergente* si, para todos los estados iniciales $s^{(0)}$ y planificadores *r-justos* (*r-fair schedules*) σ, la dinámica de $(s^{(0)}, \sigma)$ converge.

Obsérvese que en el modelo presentado, las acciones de los nodos son *observables inmediatamente* por los demás nodos al final de cada paso de tiempo. Resulta de especial interés el caso en que todas las funciones de reacción son determinísticas y sin historia. Si cada función de reacción f_i es determinística y sin historia, entonces puede ser especificada por una función $g_i : A \to A_i$. Sea $g = (g_1, \ldots, g_n)$. Obsérvese que el conjunto de todos los estados estables (para todas las dinámicas posibles) es precisamente el conjunto de todos los puntos fijos de g.

4.3. Resultado de No-Convergencia

Ahora se presentará un resultado general de imposibilidad para la convergencia de las acciones de los nodos en ambientes computacionales distribuidos bajo dinámicas de recuerdo limitadas (bounded-recall) [15].

Teorema 4.3.1 (No-convergencia)**.** *Si cada función de reacción tiene recuerdos limitados y es auto-independiente entonces, la existencia de múltiples estados estables implica que el sistema no es convergente.*

Para realizar la prueba del teorema, se definirá lo que significa un sistema asíncrono en términos de un conjunto de axiomas (evitando mencionar detalles operacionales como lo son el envío y recibo de mensajes), un enfoque utilizado por Taubenfeld [35] para demostrar que no existen protocolos de consenso resilientes. A diferencia del trabajo de Taubenfeld aquí se permiten corridas infinitas [15].

Después de definir lo que significa un sistema asíncrono, se definirá el coloreado de corridas, lo cual corresponderá a los diversos resultados que pueden obtenerse a partir de una configuración inicial del sistema, y se definirá la propiedad de Independencia de Decisiones (IoD), lo cual permitirá crear ejecuciones que siempre pueden extenderse sin llegar a un resultado. Por último se mostrará que los protocolos que satisfacen la hipótesis del teorema 4.3.1 también satisfacen IoD.

El concepto de IoD es similar a la idea presentada por Fisher et. al. [7] en la demostración de imposibilidad de consenso en caso de la falla de un proceso, la cual utiliza el concepto de estados bivalentes, en nuestro caso es la existencia de estados policromáticos la que permitirá el extender las ejecuciones.

4.3.1. Eventos, Corridas y Protocolos

Un *protocolo* es un conjunto C (posiblemente infinito) de corridas y un conjunto finito P de procesos. Una *corrida* es un conjunto finito o infinito de elementos e conocidos como eventos. Cada evento afecta a un conjunto de procesos $S \subseteq P$ (por ejemplo como receptor o emisor de algún mensaje), mientras que los procesos restantes $P - S$ son incapaces de ver el evento. Si S es el conjunto de procesos afectados por el evento e lo denotamos por e_S. Véase que no se asigna ningún significado operacional a las procesos, corridas y eventos.

Un evento e es *habilitado* en la corrida x de un protocolo si y sólo si $\langle x; e\rangle$ (la secuencia de eventos en x seguidos por el evento e) es también una corrida en ese protocolo.

Para una corrida x, sea $M \subseteq P$, los eventos e que afectan al subconjunto M son todos aquellos eventos e_S tal que $S \cap M \neq \emptyset$ (recuerde que cada evento afecta un conjunto $S \subseteq P$) y lo denotaremos por medio de $x_{\langle M\rangle}$. Decimos que dos corridas x e y son *equivalentes respecto a* M, denotado por $x[M]y$, si y sólo si $x_{\langle M\rangle} = y_{\langle M\rangle}$. Decimos que la corrida y *incluye* a x si $x_{\langle\{i\}\rangle}$ es un prefijo de $y_{\langle\{i\}\rangle}$ para todo $i \in P$ (esto no significa que la secuencia de eventos x es un prefijo de la secuencia de eventos y, eventos que afectan conjuntos disjuntos de P pueden ser reordenados sin afectar la relación de inclusión).

Los sistemas asíncronos de paso de mensajes son caracterizados por tres propiedades que cualquier protocolo que opere en tales ambientes debe satisfacer.

Un *protocolo asíncrono* es un protocolo cuyas corridas satisfacen las propiedades

- P_1 Cada prefijo de una ejecución es una ejecución.
- P_2 Sean $\langle x; e_S\rangle$ e y ejecuciones. Si y incluye a x y $x[S]y$, entonces $\langle y : e_S\rangle$ es también una ejecución.
- P_3 Sólo una cantidad finita de eventos son habilitados en una corrida.

Por propiedad P1 la secuencia vacía es también una corrida, la propiedad P2 significa que si un evento e_S ocurre en algún punto en el tiempo, entonces este evento puede volver a ocurrir en un tiempo posterior. P3 corresponde a la suposición en Fischer [7] que un proceso puede enviar una cantidad arbitraria pero finita de mensajes a otros procesos.

Otro concepto que necesitaremos es el de una *secuencia justa*, el cual Taubenfeld [35] utilizó para definir lo que es un protocolo de consenso. Una secuencia justa es una secuencia de eventos donde (1) cada prefijo de la secuencia es una corrida y (2) si la secuencia es finita, entonces ningún evento es habilitado en la secuencia, en otro caso (cuando la secuencia es infinita) cada evento que es habilitado en cada prefijo exceptuando una cantidad finita de éstos aparece infinitamente seguido en la secuencia. Se define una *extensión justa* de una secuencia x (no necesariamente justa) a la secuencia finita de eventos $e_1, e_2, \ldots, e_k$ tal que e_1 es habilitado en x, e_2 es habilitado en $\langle x; e_1\rangle$, etc.

De acuerdo al modelo presentado por Fischer [7], una vez que un mensaje es enviado, el evento de recepción del mensaje es habilitado hasta que el mensaje es recibido. Por tanto, si interpretamos algunos de los eventos en el modelo presentado como eventos de envío y de recepción, una secuencia justa contiene la intuición de una ejecución donde todos los procesos tienen una oportunidad de proceder con su ejecución y todos los mensajes enviados son eventualmente entregados [35].

Si asignamos colores a cada secuencia, sujeto a ciertas restricciones llegamos a la definición de un protocolo asíncrono C-cromático. Formalmente sea C un conjunto (llamado el conjunto de *colores*) a partir del cual asignamos un conjunto de colores a las secuencias (la asignación puede ser una función parcial). Para tal C, decimos que el protocolo es *C-cromático* si satisface las siguientes propiedades.

- C_1 Para cada $c \in C$, existe una ejecución del protocolo de color $\{c\}$.
- C_2 Para cada ejecución x del protocolo de color $C' \subseteq C$, y para cada $c \in C'$, existe una extensión de x que tiene color $\{c\}$
- C_3 Si y incluye x y x tiene color C', entonces el color de y es un subconjunto de C'.

Decimos que una secuencia imparcial es *policromática* si el conjunto de colores asignado a ella tiene más de un elemento.

Finalmente, un protocolo C-cromático es llamado un *protocolo de decisión* si también satisface la siguiente propiedad:

- D Cada secuencia imparcial tiene un prefijo monocromático, es decir, un prefijo cuyo color es $\{c\}$ para algún $c \in C$.

4.3.2. Independencia de Decisiones (IoD)

Un protocolo satisface *Independencia de Decisiones* (IoD) si, siempre que

- Una ejecución es policromática y
- existe algún evento e que es habilitado en x y $\langle x; e\rangle$ es monocromático de color $\{c\}$

entonces

1. para cada $e' \neq e$ que es habilitado en x, el color de $\langle x; e' \rangle$ contiene c, y

2. para cada $e' \neq e$ que es habilitado en x, si $\langle \langle x; e' \rangle; e \rangle$ es monocromático, entonces el color es también $\{c\}$.

La primera condición nos indica que existe un evento e que decide el resultado del protocolo y que ningún otro evento puede descartar. La segunda condición es la que da nombre a la propiedad "Independencia de Decisiones", ya que si el evento e decide el resultado tanto antes o después del evento e', entonces la decisión es independiente ya sea que e' ocurra inmediatamente antes o después de e. La figura 4.1 muestra las condiciones antes mencionadas.

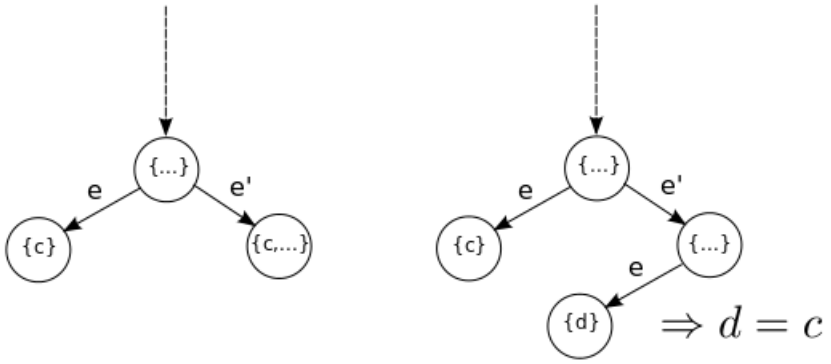

Figura 4.1: Condiciones que cumple la Independencia de Decisiones (IoD)

El siguiente lema será utilizado en pruebas posteriores.

Lema 4.3.2. *Si IoD se mantiene, entonces para cualquier dos eventos e y e' que son habilitados en la ejecución x, si tanto $\langle x; e \rangle$ como $\langle x; e' \rangle$ son monocromáticos, entonces dichos colores son los mismos.*

Demostración. Por IoD, el color de $\langle x; e' \rangle$ debe contener el color $\langle x; e \rangle$, y ambos conjuntos consisten de un solo elemento, por lo tanto son el mismo. □

4.3.3. Protocolos Conformes con IoD No Siempre Convergen

Se mostrará que aquellos protocolos que satisfacen IoD no siempre convergen, para ello se verá que una secuencia policromática siempre puede ser extendida de manera justa a otra secuencia policromática. Después de lo cual se probará que esto implica que existe una secuencia que nunca alcanza una decisión. Empecemos con el lema de extensión.

Lema 4.3.3 (El lema de Extensión Justa)**.** *En un protocolo de decisión policromático que satisface IoD, si una ejecución x es policromática, entonces x puede ser extendida por una extensión justa a otra ejecución policromática.*

Demostración. Asúmase para el propósito de contradicción que para algún C', existe una ejecución x de color C' que no puede ser extendida de manera imparcial a otra ejecución policromática. Debido a que $|C'| > 1$, debe existir algún evento que es habilitado en x, si no fuera así, contradeciríamos la propiedad D de la definición de protocolo C-cromático de decisión.

Considere la extensión de x que usa tantos eventos como es posible y que es policromática, y elija uno de estos y que minimiza el número de eventos que son habilitados en cada prefijo de y (después de que x ha sido ejecutado) pero que no aparecen en y. Si y no contiene eventos, entonces cada evento e que es habilitado en x es tal que $\langle x; e\rangle$ es monocromático. Por el lema 4.3.2, estos conjuntos unitarios deben tener el mismo color $\{c\}$; sin embargo esto significa que para todo $c' \in C'\backslash\{c\} \neq \emptyset$, x no tiene ninguna extensión cuyo color es c', contradiciendo la propiedad C_2 de la definición de protocolo C-cromático. Si y contiene uno o más eventos, entonces (debido a que no es una extensión justa de x) existe al menos un evento e que es habilitado en todas partes en la extensión, incluyendo en $\langle x; y\rangle$, pero que no aparece en ninguna parte en y. Debido a que y fue elegido en lugar de $\langle y; e\rangle$ (o alguna otra extensión con el mismo número de eventos distintos), el color de $\langle\langle x; y\rangle; e\rangle$ debe ser el conjunto unitario $\{c\}$. Debido a que $\langle x; y\rangle$ es policromático, tiene alguna extensión z que es (eventualmente) monocromática con color $\{d\} \neq \{c\}$; sea e' el primer evento en esta extensión. Debido a que IoD se satisface, el color de $\langle\langle x; y\rangle; e'\rangle$ también contiene c y es por lo tanto policromática. El evento e es de nuevo habilitado aquí (de lo contrario $\langle\langle x; y\rangle; e'\rangle$ hubiera sido elegido en lugar de y). Si $\langle\langle\langle x; y\rangle; e'\rangle e\rangle$ no es monocromático, entonces es una extensión policromática de x que usa más eventos distintos que aquellos utilizados por y, una contradicción. Si $\langle\langle\langle x; y\rangle; e'\rangle e\rangle$ es monocromático, entonces por

IoD tiene color $\{c\}$. Podemos movernos de manera inductiva a lo largo de la extensión z; después de que cada evento es agregado a z en la ejecución, la ejecución resultante es policromática (su conjunto de colores debe incluir d, pero si es monocromática su color debe ser el color $\{c\}$) y de nuevo habilita e (por nuestra elección de y). De nuevo, por nuestra elección de y, agregar e a esta ejecución debe producir una ejecución monocromática , la cual (debido a IoD) debe de tener color $\{c\}$. Procediendo a lo largo de z debemos eventualmente alcanzar una ejecución policromática en la cual e es habilitado (y produce una ejecución monocromática de color $\{c\}$) y que también habilita un evento diferente que da como resultado una ejecución monocromática de color $\{d\}$. Esto contradice el lema 4.3.2. □

Teorema 4.3.4. *Cualquier protocolo asíncrono que satisface IoD con un estado inicial policromático tiene una secuencia imparcial que comienza en este estado inicial y nunca alcanza una decisión, esto es, tiene una secuencia imparcial que no contiene un prefijo monocromático.*

Demostración. Comenzando con la ejecución vacía (policromática) y aplicando de manera iterativa el lema de extensión imparcial hasta obtener una secuencia policromática infinita. Si el evento e es habilitado en casi todos excepto alguna cantidad finita de prefijos en esta secuencia, entonces en todos excepto una cantidad finita de las extensiones imparciales, e es habilitado en cada paso de la extensión. Debido a que estas extensiones son justas (en el sentido definido anteriormente), e es activado en cada una de estas (infinitamente muchas) extensiones y por tanto aparece infinitamente de manera seguida en la secuencia, lo cual es por tanto justo. □

4.3.4. Protocolos 1-Recuerdo, Estacionales y Auto Independientes No Necesitan Converger

Para mostrar el teorema 4.3.1 basta con mostrar que el protocolo descrito en el mismo satisface IoD, para llevar a cabo lo anterior se mostrará que los protocolos 1-recuerdo, sin historia cuando son coloreados como se indica más adelante satisfacen IoD y por el teorema 4.3.4 no siempre convergen.

Definición (Coloreo estable) . En un protocolo definido como en 4.2, el *coloreo estable* de los estados del protocolo es el coloreado que tiene un color distinto para cada estado estable y que colorea cada estado en una ejecución con el conjunto de colores de los estados estables que son alcanzables a partir de dicho estado.

Para modelar la dinámica de protocolos 1-recuerdo y sin historia se utilizan dos tipos de acciones: la primera corresponde a la aplicación de las funciones de reacción de los nodos, donde e_i es la acción del nodo i comportándose como lo especifica f_i, y la segunda corresponde a una acción de "revelación"W. Los nodos que de acuerdo al planificador han sido seleccionados para reaccionar en el primer paso de tiempo, lo realizan secuencialmente, pero estas acciones no son aún visible a los demás nodos (de tal forma, que los nodos después del primero en la secuencia reaccionan al estado inicial y no a las acciones realizadas anteriormente en la secuencia). Una vez que tales nodos han reaccionado, la acción W es efectuada, esta acción revela las nuevas acciones que han sido llevadas a cabo a los demás nodos en la red. Los nodos seleccionados por el planificador para reaccionar en el siguiente paso de tiempo lo hacen en secuencia, seguida de otra acción W, y así de manera sucesiva. Esto convierte el modelo de acción simultánea de la sección 4.2 a uno en el cual las acciones son realizadas de forma secuencial, al cual nombraremos el modelo "actuar y decir". Hay que hacer notar que todas las acciones son habilitadas en cada paso de tiempo (esto es, e_i puede ser llevado a cabo múltiples veces entre acciones W; sin embargo, esto es indistinguible de una única acción e_i porque las ocurrencias adicionales no son vistas por los demás nodos, y ellas no afectan las acciones de i, las cuales son gobernadas por una función de reacción sin historia). El siguiente lema será utilizado más adelante.

Lema 4.3.5. *(Igualdades de color) . En un protocolo 1-recuerdo y sin historia (en el modelo actuar y decir):*

1. *Para cada par de ejecuciones x, y y para cada $i \in [n]$, el color de $\langle\langle x; e_i W e_i W\rangle; y\rangle$ es del mismo color que $\langle\langle x; W e_i W\rangle; y\rangle$.*

2. *Para cada par de ejecuciones x, y y para cada $i, j \in [n]$, el color de $\langle\langle x; e_i e_j\rangle; y\rangle$ es del mismo color que $\langle\langle x; e_j e_i\rangle; y\rangle$.*

De manera informal, la primera igualdad de color dice que, si todas las actualizaciones son anunciadas y después i es activado y después todas las actualizaciones son reveladas nuevamente (el resultado de i siendo lo único nuevo), no existe diferencia si i fue activado inmediatamente antes de la primera acción de revelación. La segunda igualdad de color dice que, siempre que no existan eventos de revelado intermedios, el orden en que los nodos computan sus resultados no tiene importancia (porque los nodos no tienen acceso a los resultados de sus vecinos hasta después de la acción de revelado).

Demostración. Para la primera igualdad de color, debido a que el protocolo es auto-independiente, la primera ocurrencia de e_i (después de x) en la ejecución $\langle\langle x; e_i W e_i W\rangle; y\rangle$ no afecta la segunda ocurrencia del evento e_i. Más aún, debido a que el protocolo tiene 1-recuerdo, los eventos siguientes a y no son afectados. La segunda definición de igualdad de color se verifica por la definición del modelo actuar y decir. □

Con la definición de coloreo estable podemos ahora presentar el siguiente lema.

Lema 4.3.6. *Si un protocolo es 1-recuerdo y sin historia, entonces el protocolo (con coloreo estable) satisface IoD.*

Demostración. Coloreamos cada estado de las ejecuciones del protocolo de acuerdo a los estados estables que pueden ser alcanzados a partir de él. Asúmase que x es una ejecución policromática (con color C') y que algún evento e es tal que $\langle x; e\rangle$ es monocromático (con color $\{c\}$). Sea e' otro evento (recuerde que todos los eventos son siempre habilitados). Si e y e' son dos eventos de nodos distintos e_i y e_j $(i \neq j)$, respectivamente, entonces el color de $\langle\langle x; e_j\rangle; e_i\rangle$ es el color de $\langle\langle x; e_i\rangle; e_j\rangle$ y por tanto el color (monocromático) de $\langle x; e_i\rangle$, que es $\{c\}$. Si e y e' son ambos W (acción de revelado) o son el mismo evento de nodo e_i, entonces la afirmación es trivial.

Si $e = e_i$ y $e' = W$ (véase la parte (a) de la figura 4.2), entonces podemos extender $\langle x; e_i\rangle$ por We_iW para obtener una ejecución cuyo color es de nuevo $\{c\}$. Por la segunda igualdad de color, esto es también el color de la extensión de $\{x; W\}$ por e_iW, por lo que el color de $\langle x; W_i\rangle$ contiene c y si la extensión de $\langle x; W\rangle$ por e_i es monocromática, su color debe ser $\{c\}$ también. Si por otra parte, $e = W$ y $e' = e_i$ (inciso (b) de la figura 4.2), podemos extender $\langle x; W\rangle$ por e_iW y $\langle x; e_i\rangle$ por We_iW para obtener ejecuciones de color $\{c\}$; por lo que el color de $\langle x; e_i\rangle$ debe contener c y utilizando un argumento como el antes presentado, si la extensión intermedia de $\langle\langle x; e_i\rangle; W\rangle$ es monocromática, su color debe de ser $\{c\}$. □

Lema 4.3.7. *Si una computación con 1-recuerdo y sin historia que siempre converge puede, para diferentes estados iniciales, converger a diferentes estados estables, entonces existe alguna entrada para la cual la computación puede alcanzar múltiples estados estables. En particular, bajo el coloreo estable, existe un estado policromático.*

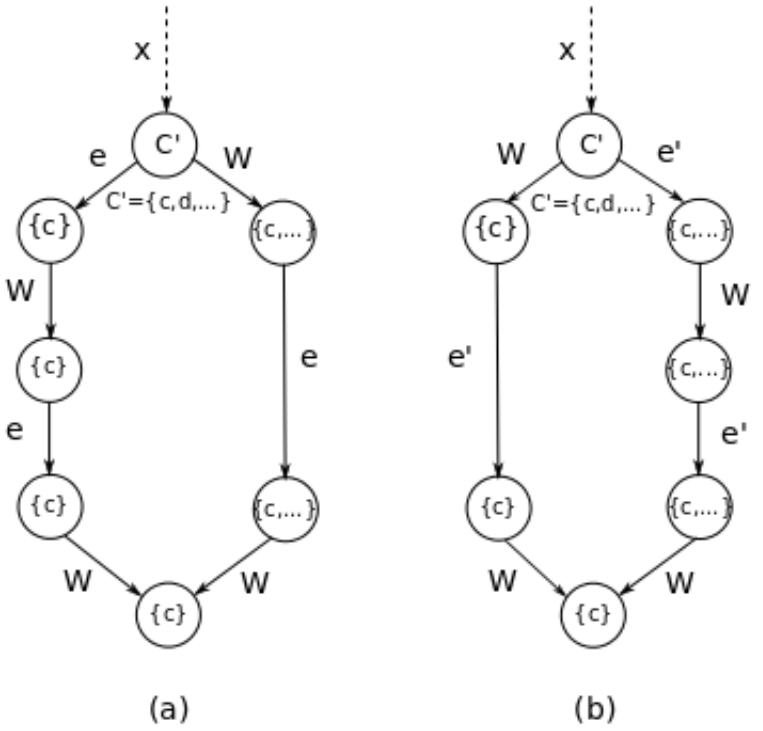

Figura 4.2: Argumentos utilizados en la prueba del lema 4.3.6

Demostración. Asúmase que hay (considerando el coloreo estable) dos estados de entrada monocromáticos diferentes para la computación, que las entradas difieren solo en un nodo v, y que la computación siempre converge (esto es, para cada planificador justo) para ambos estados de entrada. Considere un planificador imparcial que activa v primero y después prosigue arbitrariamente. Debido a que las entradas a la función de reacción de v son las mismas en cada caso, después del primer paso en cada computación, la dos redes resultantes tienen los mismos estados en los nodos. Esto significa que la computación se desenvolverá en la misma forma, en particular produciendo salidas idénticas. Si una computación sin historia que siempre converge puede producir dos diferentes salidas, entonces la aplicación consecutiva del argumento anterior conduce a una contradicción a menos que exista un estado

inicial policromático. □

4.3.5. Protocolos k-Recuerdo, Estacionales y Auto Independientes No Necesitan Converger

El teorema 4.3.1 trata sobre sistemas con recuerdo limitado, no con sistemas 1-recuerdo, por tanto se requiere extender la definición de igualdades de color del lema 4.3.5 para permitir k-recuerdo (para todo $k > 1$), para ello se presenta el siguiente lema:

Lema 4.3.8. *(Igualdades de color) . En un protocolo k-recuerdo, para $k > 1$ y sin historia (en el modelo actuar y decir):*

1. *Para cada par de ejecuciones x, y y para cada $i \in [n]$, el color de $\langle\langle x; e_i W(e_i W)^k\rangle; y\rangle$ es del mismo color que $\langle\langle x; W(e_i W)^k\rangle; y\rangle$.*

2. *Para cada par de ejecuciones x, y y para cada $i, j \in [n]$, el color de $\langle\langle x; e_i e_j\rangle; y\rangle$ es del mismo color que $\langle\langle x; e_j e_i\rangle; y\rangle$.*

Demostración. Para la primera igualdad de color, debido a que el protocolo es auto-independiente, la primera ocurrencia de e_i (después de x) en la ejecución $\langle\langle x; e_i W(e_i W)^k\rangle; y\rangle$ no afecta ninguna de las k-ocurrencias del evento e_i posteriores. Más aún, debido a que el protocolo tiene 1-recuerdo, los eventos siguientes a y no son afectados. La segunda definición de igualdad de color se verifica por la definición del modelo actuar y decir. □

Con lo cual se puede enunciar un lema similar al 4.3.6 para sistemas k-recuerdo, la prueba sería similar sólo que la extensión de las ejecuciones sería con $W(e_i W)^k$ o $(e_i W)^k$, el lema sería el siguiente:

Lema 4.3.9. *Si un protocolo es k-recuerdo (con $k > 1$) y sin historia, entonces el protocolo (con coloreo estable) satisface IoD.*

Demostración. Coloreamos cada estado de las ejecuciones del protocolo de acuerdo a los estados estables que pueden ser alcanzados a partir de el. Asúmase que x es una ejecución policromática (con color C') y que algún evento e es tal que $\langle x; e\rangle$ es monocromático (con color $\{c\}$). Sea e' otro evento (recuerde que todos los eventos son siempre habilitados). Si e y e' son dos eventos de nodos distintos e_i y e_j ($i \neq j$), respectivamente, entonces el color de $\langle\langle x; e_j\rangle; e_i\rangle$ es el color de $\langle\langle x; e_i\rangle; e_j\rangle$ y por tanto el color (monocromático)

de $\langle x; e_i \rangle$, que es $\{c\}$. Si e y e' son ambos W (acción de revelado) o son el mismo evento de nodo e_i, entonces la afirmación es trivial.

Si $e = e_i$ y $e' = W$ (véase la parte (a) de la figura 4.3), entonces podemos extender $\langle x; e_i \rangle$ por $W(e_iW)^k$ para obtener una ejecución cuyo color es de nuevo $\{c\}$. Por la segunda igualdad de color, esto es también el color de la extensión de $\{x; W\}$ por $(e_iW)^k$, por lo que el color de $\langle x; W_i \rangle$ contiene c y si la extensión de $\langle x; W \rangle$ por e_i es monocromática, su color debe ser $\{c\}$ también. Si por otra parte, $e = W$ y $e' = e_i$ (inciso (b) de la figura 4.3), podemos extender $\langle x; W \rangle$ por $(e_iW)^k$ y $\langle x; e_i \rangle$ por $W(e_iW)^k$ para obtener ejecuciones de color $\{c\}$; por lo que el color de $\langle x; e_i \rangle$ debe contener c y utilizando un argumento como el antes presentado, si la extensión intermedia de $\langle \langle x; e_i \rangle; W \rangle$ es monocromática, su color debe de ser $\{c\}$. □

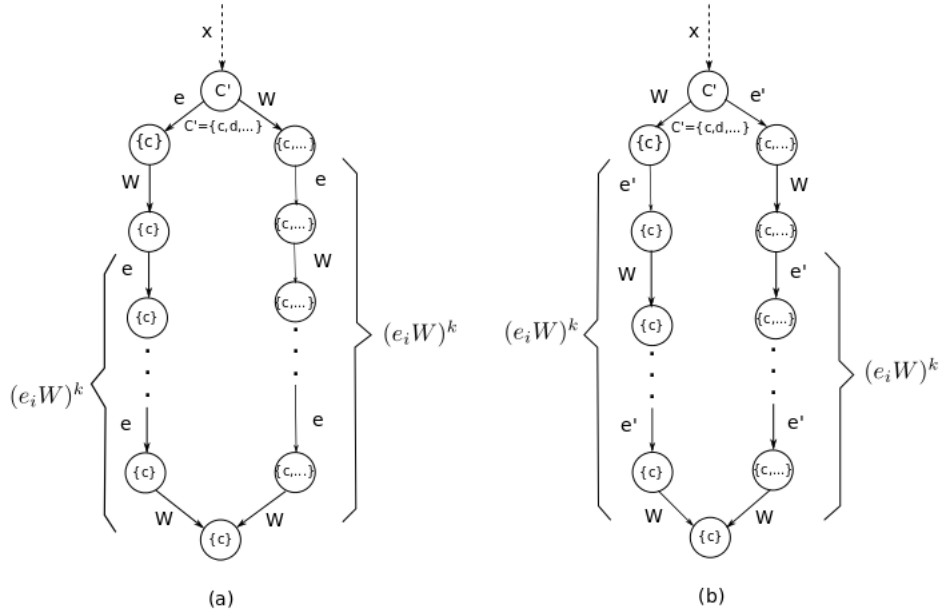

Figura 4.3: Argumentos utilizados en la prueba del lema 4.3.9

De igual manera, es posible establecer un lema similar al 4.3.7 para el caso de k-recuerdo del siguiente modo:

Lema 4.3.10. *Si una computación con k-recuerdo (con $k > 1$) y sin historia que siempre converge puede, para diferentes estados iniciales, converger a*

diferentes estados estables, entonces existe alguna entrada para la cual la computación puede alcanzar múltiples estados estables. En particular, bajo el coloreo estable, existe un estado policromático.

Demostración. Asúmase que hay (considerando el coloreo estable) dos estados de entrada monocromáticos diferentes para la computación, que las entradas difieren solo en un nodo v, y que la computación siempre converge (esto es, para cada planificador justo) para ambos estados de entrada. Considere un planificador imparcial que activa v primero un número k de veces y después prosigue arbitrariamente. Debido a que las entradas a la función de reacción de v son las mismas en cada caso, después del primer paso en cada computación, la dos redes resultantes tienen los mismos estados en los nodos. Esto significa que la computación se desenvolverá en la misma forma, en particular produciendo salidas idénticas. Si una computación sin historia que siempre converge puede producir dos diferentes salidas, entonces la aplicación consecutiva del argumento anterior conduce a una contradicción a menos que exista un estado inicial policromático. □

Con los lemas anteriores podemos probar ahora el Teorema 4.3.1.

Demostración. (prueba del teorema 4.3.1) Considérese un protocolo con k-recuerdo, auto-independiente, con estacionalidad, y que tiene dos estados estables diferentes. Si existe alguna ejecución del protocolo que no converge, entonces la red no es segura y hemos terminado.

Para el propósito de contradicción asúmase que todas las ejecuciones convergen. Se colorean todos los estados del protocolo de acuerdo al coloreo estable (4.3.4). El lema 4.3.10 implica que existe un estado policromático. Debido al lema 4.3.9, se satisface IoD y podemos aplicar el teorema 4.3.4. En este contexto (con coloreo estable) esto implica que existe una ejecución infinita en la cual cada estado puede alcanzar al menos dos estados estables, y la ejecución entonces no converge. □

4.4. Implicaciones en el Protocolo BGP

Recuérdese que BGP es el protocolo utilizado para "unir" los diferentes *Sistemas Autónomos* (AS por sus siglas en inglés) que conforman el internet. Existe una red de AS fuente que desean enviar tráfico a un único AS *destino d*. Cada AS i tiene una *función de ordenamiento*, la cual ordena las

diferentes opciones que tiene el AS para llegar a su destino en base a sus políticas de preferencias. En BGP cada AS selecciona de manera constante la "mejor" ruta que le es posible (véase el capítulo 3). Para poder establecer la convergencia del protocolo BGP estableceremos la relación de este con el modelo presentado en 4.2.

Nodos computacionales y espacio de acción. Los AS son los *nodos computacionales*. El *espacio de acción* de cada nodo i, A_i, es el conjunto de todas las rutas simples (sin ciclos) entre i y el destino d que son exportables a i y la ruta vacía $\emptyset$.

Funciones de reacción y dinámica del sistema. La *función de reacción* f_i del nodo i da como resultado, para cada vector α que contiene rutas hacia d de todos los vecinos de i, una ruta $(i,j)R_j$ tal que:

1. j es un vecino de i
2. R_j es la ruta de j en α, y
3. $R_j > R$ para todas las demás rutas R en α de acuerdo a la función de ordenamiento del nodo i.

Si no existe ruta R_j en α, entonces f_i regresa $\emptyset$. Obsérvese que la función f_i es determinística, auto-independiente y sin historia.

Ahora, para estudiar la convergencia del protocolo utilizaremos el siguiente teorema.

Teorema 4.4.1. *Si existen múltiples árboles estables de ruteo en una red, entonces BGP no es seguro en dicha red.*

Demostración. Utilizando el teorema 4.3.4 queda demostrado. □

4.5. Notas y lecturas recomendadas

El estudio y aplicación de la teoría de juegos en problemas de computación no es nuevo, Griffin [12] ya lo había sugerido en su estudio del problema de las trayectorias estables y Papadimitriou [28] antes que él, Tardos [2] por su parte, explica que la unión de las ciencias de la computación y la teoría económica resulta benéfico a ambas disciplinas.

Si bien el estudio de tales cuestiones se trata en la reciente área conocida como *Teoría de Juegos Algorítmica*, el trabajo de Jaggard [15] toma en consideración un elemento adicional en su estudio, la presencia de asincronía y la complejidad que ello implica. La comparación de los resultados y el modelo presentado por Jaggard [15] con aquellos encontrados en el área de Teoría de Juegos Algorítmica, como los encontrados en Nissan [27] resulta intrigante pero exceden el alcance del trabajo presentado y se dejan como trabajo futuro.

El uso del método axiomático utilizado por Jaggard [15] para la demostración de sus resultados fue utilizado previamente por Taubenfeld [35] en el estudio de protocolos de consenso resilientes, quien además lo compara con el método utilizado por Fischer [7], cuyas ideas de estados bivalentes son similares a las de los estados policromáticos expuestas en el capítulo.

Capítulo 5

Conclusiones

En este capítulo se presentarán implicaciones de los resultados obtenidos en el capítulo 4 así como algunas medidas desarrolladas para garantizar la convergencia del protocolo BGP. Se presentará una vista general de otras aplicaciones donde el modelo presentado en el capítulo anterior es aplicable así como algunas de las cuestiones que siguen abiertas, las cuales presentan un campo fértil de investigación.

5.1. Implicaciones

Mientras que los algoritmos de Dijsktra y Bellman-Ford, utilizados por los algoritmos de ruteo, son aplicables aún cuando exista más de una trayectoria de un origen dado hacia un destino determinado y siempre garantizan convergencia, el Teorema 4.4.1 nos dice que en el caso de BGP la convergencia no se garantiza. Una forma de mejorar la convergencia sería utilizar un mecanismo más restrictivo de selección de rutas por parte de los AS, de forma que cada AS sólo pueda generar un árbol de trayectorias para cada destino dado, sin embargo, esto significaría que para cada destino se tendría un único punto de falla, lo cual conlleva un riesgo mayor que aquel que intentamos corregir.

Dado que el estudio estático para determinar problemas de convergencia es un problema NP-Completo [13], otras alternativas se han propuesto para mejorar la convergencia del protocolo, por ejemplo, utilizando un mecanismo constructivo, basado en las relaciones jerárquicas y comerciales presentes en los sistemas AS [8], un enfoque que preserva la capacidad de cada AS para implementar políticas locales complejas mientras mantiene su habilidad de

ocultar sus configuraciones BGP de otros AS.

Otro tipo de enfoque se basa en una solución paliativa, es decir, se aplica después de detectar oscilaciones en la selección de rutas. El proceso es dinámico ya que responde al proceso de oscilación y opera cambiando las prioridades de selección de ruta en base a las oscilaciones que presenta, si una ruta tiene un puntaje alto pero es más inestable, se cambiará por un puntaje bajo, de forma que tenga pocas probabilidades de ser utilizada [23].

Utilizando un enfoque diferente al teórico, se ha buscado dilucidar desde una perspectiva más pragmática el origen de los problemas de configuración de BGP debidos a la inserción accidental de rutas en las tablas globales de BGP o a la propagación de rutas que debieron ser filtradas. Además de estudiar las consecuencias que esto tiene en la conectividad y en la carga de red [24]. El objetivo de tal estudio es entender como los errores humanos pueden ser minimizados con objeto de mejorar la confiabilidad del ruteo en Internet.

La importancia del estudio de la convergencia de BGP y mejoras posibles sigue siendo un tema de investigación, véase por ejemplo [10], lo cual demuestra que es un problema lejos de estar concluido. Cualesquiera que sean los mecanismos utilizados en la solución, esta debe de satisfacer las demandas a veces dispares entre AS, que pueden ser desde técnicas a económicas.

5.2. Conclusiones

Una de las implicaciones del teorema 4.4.1 es que BGP y protocolos basados en el mismo que sean de ruteo multitrayectoria (por ejemplo R-BGP y NS-BGP) [36, 20] cuyos nodos computacionales sean independientes de sus propias acciones pasadas y que tengan un recuerdo limitado de sus interacciones pasadas, tampoco son seguros.

Siendo el caso, la búsqueda de un protocolo que pueda sustituir a BGP debe basarse en otros criterios de selección, pero desafortunadamente esto va en contra del principio de compatibilidad y hace difícil la adopción del mismo ya que debe entrar a un proceso de estandarización para permitir que todos lo puedan ocupar.

Dado que el protocolo BGP es uno de los motores de Internet, la posibilidad de encontrar estados en los cuales el protocolo diverge, impide otorgar garantías a aplicaciones de tiempo real, creando la necesidad de diseñar nuevos protocolos y minando el proceso de estandarización, ya que cada apli-

cación buscará lograr sus metas sin consideración de su aplicación en otras áreas, debido a los costos que esto conlleva.

Aunque la solución al problema de convergencia sigue siendo un problema a resolver, el estudio de la interacción de la teoría de Sistemas Distribuidos y Teoría de Juegos puede otorgarnos nuevas herramientas, las cuales nos permitan comprender el problema desde una perspectiva diferente y a partir de la cual encontrar mejores soluciones.

5.3. Trabajo futuro

El modelo presentado en el capítulo 4, para estudiar las heurísticas adaptativas en ambientes asíncronos, puede utilizarse también para el estudio del efecto de asincronía en el comportamiento de los circuitos, en la difusión de tecnologías, entre otros problemas más [15], lo cual queda fuera del alcance del trabajo, no obstante provee de valiosa información en otras áreas.

El estudio presentado en el trabajo ha sido solamente de carácter introductorio y presenta solo un tipo de heurística adaptativa, queda por estudiar otras nociones de convergencia y equilibrio (por ejemplo, nociones distintas el equilibrio de Nash). De igual forma, no se tiene una caracterización de cuando es posible una convergencia asíncrona.

Así mismo, queda pendiente el estudio de otras áreas en las cuales el modelo sea apropiado y la relación que tiene con otras disciplinas que han demostrado ser de gran utilidad tales como la Topología [14]. El campo es fértil y lleno de retos, pero promete ser de gran utilidad.

Bibliografía

[1] D. P. Bertsekas and R. G. Gallager, *Data networks*, 2nd ed. Prentice Hall, 1992.

[2] L. Blume, D. Easley, J. Kleinberg, R. Kleinberg, and É. Tardos, "Introduction to computer science and economic theory," *Journal of Economic Theory*, vol. 156, pp. 1–13, 2015.

[3] Cisco, "Redistribución de protocolos de enrutamiento - ip : Routing ip," 2015. [Online]. Available: http://www.cisco.com/cisco/web/support/LA/7/73/73348_redist.html

[4] L. Cittadini, M. Rimondini, M. Corea, and G. Di Battista, "On the feasibility of static analysis for bgp convergence," in *Integrated Network Management, 2009. IM '09. IFIP/IEEE International Symposium on*, June 2009, pp. 521–528.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 3rd ed. Mit Press, 2009, pp. 643–662.

[6] R. Diestel, *Graph theory*, 4th ed. Springer, 2010.

[7] M. J. Fischer, N. A. Lynch, and M. S. Paterson, "Impossibility of distributed consensus with one faulty process," *J. ACM*, vol. 32, no. 2, pp. 374–382, Apr. 1985. [Online]. Available: http://doi.acm.org/10.1145/3149.214121

[8] L. Gao and J. Rexford, "Stable internet routing without global coordination," *Networking, IEEE/ACM Transactions on*, vol. 9, no. 6, pp. 681–692, Dec 2001.

[9] M. R. Garey and D. S. Johnson, *Computers and intractability A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[10] P. Godfrey, M. Caesar, I. Haken, Y. Singer, S. Shenker, and I. Stoica, "Stabilizing route selection in bgp," *Networking, IEEE/ACM Transactions on*, vol. 23, no. 1, pp. 282–299, Feb 2015.

[11] T. Griffin, F. Shepherd, and G. Wilfong, "Policy disputes in path-vector protocols," in *Network Protocols, 1999. (ICNP '99) Proceedings. Seventh International Conference on*, Oct 1999, pp. 21–30.

[12] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," *IEEE/ACM Trans. Netw.*, vol. 10, no. 2, pp. 232–243, Apr. 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=508325.508332

[13] T. G. Griffin and G. Wilfong, "An analysis of bgp convergence properties," *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 4, pp. 277–288, Aug. 1999. [Online]. Available: http://doi.acm.org/10.1145/316194.316231

[14] M. Herlihy, D. N. Kozlov, and S. Rajsbaum, *Distributed computing through combinatorial topology*. Morgan Kaufmann, 2013.

[15] A. Jaggard, M. Schapira, and R. Wright, "Distributed computing with adaptive heuristics," in *Proceedings of Innovations in Computer Science ICS (2011)*, 2011.

[16] R. G. K. Varadhan and D. Estrin, "Persistent route oscillations in interdomain routing," *ISI technical report 96-631, USC/Information Science Institute*, 1996.

[17] J. Kleinberg and E. Tardos, *Algorithm design*. Pearson/Addison-Wesley, 2006.

[18] N. Kocharians and T. Vinson, *CCIE routing and switching v5.0 official cert guide library volume 2*, 5th ed., 2015.

[19] J. F. Kurose and K. W. Ross, *Computer Networking A Top-Down Approach*, 6th ed. Pearson, 2013.

[20] N. Kushman, S. Kandula, D. Katabi, and B. M. Maggs, "R-bgp: Staying connected in a connected world," in *Proceedings of the 4th USENIX Conference on Networked Systems Design & Implementation*,

ser. NSDI'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 25–25. [Online]. Available: http://dl.acm.org/citation.cfm?id=1973430.1973455

[21] A. A. Labovitz, Craig, R. Wattenhofer, and S. Venkatachary, "The impact of internet policy and topology on delayed routing convergence," *Proc. IEEE INFOCOM*, vol. 1, pp. 537–546, Apr. 2001.

[22] C. Labovitz, G. R. Malan, and F. Jahanian, "Internet routing instability," *IEEE/ACM Trans. Netw.*, vol. 6, no. 5, pp. 515–528, Oct. 1998. [Online]. Available: http://dx.doi.org/10.1109/90.731185

[23] W. Lijun, W. Jianping, and X. Ke, "An adaptive mechanism to guarantee bgp routing convergence with policies conflict," in *Broadband Convergence Networks, 2007. BcN '07. 2nd IEEE/IFIP International Workshop on*, May 2007, pp. 1–12.

[24] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding bgp misconfiguration," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 3–16, Aug. 2002. [Online]. Available: http://doi.acm.org/10.1145/964725.633027

[25] D. Medhi and K. Ramasamy, *Network Routing: Algorithms, Protocols, and Architectures*, 1st ed. Elsevier, 2007.

[26] I. México, "Ipv6 méxico, capítulo mexicano del foro ipv6," 2015. [Online]. Available: http://www.ipv6.unam.mx/

[27] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic game theory*. Cambridge University Press, 2007.

[28] C. Papadimitriou, "Algorithms, games, and the internet," in *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*, ser. STOC '01. New York, NY, USA: ACM, 2001, pp. 749–753. [Online]. Available: http://doi.acm.org/10.1145/380752.380883

[29] L. L. Peterson and B. S. Davie, *Computer networks: A Systems Approach*, 4th ed. Morgan Kaufmann, 2007.

[30] M. Raynal, *Distributed algorithms for message-passing systems*. Springer, 2013.

[31] M. Roughan, W. Willinger, O. Maennel, D. Perouli, and R. Bush, "10 lessons from 10 years of measuring and modeling the internet's autonomous systems," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 9, pp. 1810–1821, October 2011.

[32] R. Sedgewick and K. D. Wayne, *Algorithms*, 4th ed. Addison-Wesley, 2011, pp. 646–652.

[33] D. N. Serpanos and T. Wolf, *Architecture of network systems*. Morgan Kaufmann, 2011.

[34] A. S. Tanenbaum and D. Wetherall, *Computer networks*, 5th ed. Pearson Prentice Hall, 2011.

[35] G. Taubenfeld, "On the nonexistence of resilient consensus protocols," *Inf. Process. Lett.*, vol. 37, no. 5, pp. 285–289, Mar. 1991. [Online]. Available: http://dx.doi.org/10.1016/0020-0190(91)90221-3

[36] Y. Wang, M. Schapira, and J. Rexford, "Neighbor-specific bgp: More flexible routing policies while improving global stability," in *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '09. New York, NY, USA: ACM, 2009, pp. 217–228. [Online]. Available: http://doi.acm.org/10.1145/1555349.1555375