Capítulo 2 Descripción de Algoritmos de Ruteo

Es en este capítulo es donde se encuentra el marco teórico en el que se basa la tesis. Comienza con la explicación de un modelo punto a punto y sus principales tareas. Se prosigue con la explicación general del modelo *Open Systems Interconnection (OSI)* y de el conjunto de protocolos llamado *TCP/IP.* Se muestran las principales características del *Protocolo de Internet (IP),* los datagramas *IP,* y las direcciones *IP.* Después se profundiza en los protocolos de ruteo *Routing Information Protocol (RIP) y Open Shortest path First (OSPF).* Y se culmina con la explicación de los *Dijkstra* y *Bellman-Ford.* Para cada algoritmo se muestra un ejemplo su funcionamiento.

2.1 Introducción ¿Qué es una red de comunicaciones?

Antes de explicar los algoritmos de ruteo, es muy importante saber en donde se utilizan en una red de comunicaciones. Para ello analicemos una modelo de comunicaciones y algunas de las tareas principales requeridas para completar el intercambio de información desde una fuente hacia un destino.

El propósito fundamental de un sistema de comunicaciones es el intercambio de datos entre dos sistemas. En la Figura 2.1 se muestra el modelo punto a punto básico enfocado en un ejemplo de intercambio de datos entre una computadora personal y un servidor a través de la red telefónica. Los elementos de un modelo de comunicaciones son los siguientes [STA03]:

Figura 2.1 Modelo punto a punto

Fuente: Este módulo genera los datos que serán transmitidos. En el ejemplo de la Figura 2.1 la fuente es la computadora personal.

Transmisor: Usualmente, los datos generados por la fuente no son transmitidos directamente en la forma en la que fueron generados. En vez, un transmisor transforma y codifica la información de tal manera que genera señales electromagnéticas que pueden ser transmitidas a través de algún sistema de transmisión. En el ejemplo de la Figura 2.1 la computadora personal tiene un módem conectado. Éste último adquiere un tren de bits de la computadora personal y lo transforma en una señal analógica que pueda ser manejada por la red telefónica.

Sistema de Transmisión: Esta puede ser desde una simple línea de transmisión hasta una red compleja que enlaza a la fuente con el destino.

Receptor: El receptor acepta la señal del Sistema de Transmisión y la convierte en una forma en la cuál puede ser procesada por el módulo de destino. En el ejemplo de la Figura 2.1 el módem receptor acepta la señal analógica proveniente de la red telefónica y la convierte en un tren de bits.

Destino: Recibe los datos provenientes del receptor.

Esta simple idea de un modelo punto a punto en realidad contempla una complejidad técnica. Para tener una idea de esta complejidad, la Tabla 2.1, lista algunas de las tareas clave que deben funcionar en un sistema de comunicación de datos. La lista es algo arbitraria; hay elementos que pueden añadirse. Algunos puntos representan varias tareas que funcionan a diferentes niveles. Sin embargo esta lista es solamente para localizar el punto en el que los algoritmos de ruteo se llevan a cabo en una red de comunicaciones.

2.1.1 Tareas en una Red de Comunicaciones

El primer punto de la Tabla 2.1, **Utilización del sistema de transmisión**, se refiere a la necesidad de hacer eficiente el uso de los sistemas de transmisión que normalmente comparten una cantidad de dispositivos de comunicación. Varias técnicas son usadas para mantener la capacidad total de transmisión de un determinado número de usuarios estas técnicas son conocidas como *Técnicas de Multiplexado*. Técnicas de control de

congestionamiento pueden ser requeridas para asegurar que el sistema no está sobre saturado por una demanda excesiva de transmisión de servicios.

Para comunicarse, un dispositivo debe hacer **interfase** con el sistema de transmisión. Ciertas formas de comunicación dependen del uso de señales electromagnéticas que son propagadas sobre un medio de transmisión. Entonces, cuando la interfase es establecida, la **generación de señales** es requerida para la comunicación. Las propiedades de las señales, tales como forma e intensidad, deben ser tales para que la señal sea capaz de ser propagada a través de un sistema de transmisión y además deben ser interpretadas como datos por el receptor.

Tabla 2.1 Tareas de comunicación [STA03]

Tareas de comunicación	
1 Utilización del sistema de transmisión	*8 Direccionamiento*
2 Interfase	***9 Ruteo***
3 Generación de señales	*10 Recuperación*
4 Sincronización	*11 Formato de mensajes*
5 Administrador de intercambios	*12 Seguridad*
6 Detección y corrección de errores	*14 Administración de la red*
7 Control de flujo	

No solamente las señales deben ser generadas para conformar los requerimientos del sistema de transmisión y para el receptor, también debe haber una forma de de **sincronización** entre el transmisor y el receptor. El receptor debe ser capaz de determinar cuando una señal comienza y cuando termina. También debe conocer la duración de cada elemento de la señal.

Hay una variedad de requerimientos para una comunicación; a estos requerimientos los vamos a concentrar en la **administración de intercambios.** Si los datos deben ser intercambiados en ambas direcciones dentro de un periodo de tiempo, las dos partes deben cooperar. Por ejemplo, para enlazar dos partes en una conversación telefónica, una parte debe marcar el número de la otra, provocando señales que resultaran en el timbrado del teléfono de la otra. La parte llamada completa la conexión levantando el auricular. Para los dispositivos de procesamiento, será necesario más que una simple

conexión establecida; determinados requerimientos deben ser definidos. Estos requerimientos pueden incluir entre otros, si ambos dispositivos pueden transmitir simultáneamente o deben tomar turnos, la cantidad de datos a ser enviada al mismo tiempo, el formato de los datos y que hacer ante determinadas contingencias como la llegada de un error.

En todos los sistemas de comunicación hay un potencial de error; las señales transmitidas son distorsionadas antes de alcanzar su destino. La **Detección y corrección de errores** son requeridos en circunstancias en donde los errores no pueden ser tolerados. Este es el típico caso de los sistemas de procesamiento de datos, donde los datos enviados no pueden ser alterados en el proceso y tienen que llegar como fueron enviados al destino. El **Control de flujo** es requerido para asegurar que la fuente no sobresature al destinatario al enviar los datos más rápido de lo que pueden ser procesados o recibidos.

A continuación sigue lo relacionado a los conceptos de **direccionamiento** y **ruteo.** Cuando más de dos dispositivos comparten un sistema de transmisión, un sistema fuente debe indicar la identidad del destinatario. El sistema de transmisión debe asegurar que el sistema destinatario, y solo aquel sistema, reciba los datos. Además, el sistema de transmisión puede por si mismo ser una red a través de la cual se pueden tomar varios caminos. Una ruta específica de la red debe ser escogida. Es aquí donde los algoritmos *Dijkstra y Bellman-Ford* realizan su importante labor en las redes de comunicaciones: escoger la ruta optima por la cual la información será enviada para ir desde el punto fuente hacia el punto destino.

La **Recuperación** es un concepto diferente a la corrección de errores. Las técnicas de recuperación son necesarias en situaciones en donde en un intercambio de información, tales como una transacción de una base de datos o la transferencia de un archivo, son interrumpidos debido a una falla en alguna parte del sistema. El objetivo es que sea capaz de resumir la actividad en el punto de interrupción o cuando menos reestablecer el estado del sistema en la condición que se encontraba antes de ser interrumpido. El **formato de mensajes** tiene que ver con el consentimiento de las dos partes en la definición de la manera en que serán intercambiados o transmitidos los datos. Frecuentemente, es importante proveer algunas medidas de **seguridad** en los sistemas de comunicación de datos. El emisor de datos quiere estar seguro de que solo el receptor

elegido sea quién recibe los datos. Mientras que el receptor de datos desea estar seguro de que los datos recibidos no han sido alterados durante la transmisión y que los mismos vienen del emisor adecuado.

Finalmente, dado que el intercambio de datos es un sistema complejo que no se puede ejecutar por si mismo se requiere a un **administrador de la red**. La **administración de la red** configurar el sistema, monitorea su estado y reacciona a fallas y sobrecargas.

Así hemos concluido con un esquema muy general de las tareas que una red de comunicaciones tiene que realizar. Para continuar con nuestro estudio, prosigamos con la explicación del modelo *Open Systems Interconnection (OSI) y* el protocolo de comunicación *TCP/IP*.

2.2 Modelo *OSI*

El modelo *Open Systems Interconnection (OSI)* es un lineamiento funcional para tareas de comunicaciones y, por consiguiente, no especifica un estándar de comunicación para dichas tareas. Sin embargo, muchos estándares y protocolos cumplen con los lineamientos del modelo *OSI*.

OSI nace de la necesidad de uniformizar los elementos que participan en la solución del problema de comunicación entre equipos de cómputo de diferentes fabricantes. Estos equipos presentan diferencias en procesador central, velocidad, memoria, dispositivos de almacenamiento, *interfases* para Comunicaciones, códigos de caracteres, sistemas operativos, etc [DIA00].

Estas diferencias propician que el problema de comunicación entre computadoras no tenga una solución simple. Dividiendo el problema general de la comunicación, en problemas específicos, facilitamos la obtención de una solución a dicho problema.

La solución de cada problema específico puede ser optimizada individualmente. Este modelo persigue un objetivo claro y bien definido [DIA00]:

- Formalizar los diferentes niveles de interacción para la conexión de computadoras habilitando así la comunicación de las computadoras independientemente del fabricante, arquitectura, localización y sistema Operativo.

- Desarrollar un modelo en el cual cada nivel define un protocolo que realiza funciones específicas diseñadas para atender el protocolo de la capa superior. No especificar detalles de cada protocolo. Especificar la forma de diseñar familias de protocolos, esto es, definir las funciones que debe realizar cada capa.

Estructura del modelo *OSI*

El objetivo perseguido por *OSI* establece una estructura que presenta las siguientes particularidades [DIA00].

Estructura multinivel: Se diseñó una estructura multinivel con la idea de que cada nivel se dedique a resolver una parte del problema de comunicación. Esto es, cada nivel ejecuta funciones específicas.

El nivel superior utiliza los servicios de los niveles inferiores: Cada nivel se comunica con su similar en otras computadoras, pero debe hacerlo enviando un mensaje a través de los niveles inferiores en la misma computadora. La comunicación internivel está bien definida. El nivel N utiliza los servicios del nivel N-1 y proporciona servicios al nivel N+1.

Puntos de acceso: Entre los diferentes niveles existen *interfases* llamadas *puntos de acceso* a los servicios.

Dependencias de Niveles: Cada nivel es dependiente del nivel inferior y también del superior.

Encabezados: En cada nivel, se incorpora al mensaje un formato de control. Este elemento de control permite que un nivel en la computadora receptora se entere de que su similar en la computadora emisora esta enviándole información. Cualquier nivel dado, puede incorporar un encabezado al mensaje. Por esta razón, se considera que un mensaje esta constituido de dos partes: Encabezado e Información. Entonces, la incorporación de encabezados es necesaria aunque representa un lote extra de información, lo que implica que un mensaje corto pueda ser voluminoso. Sin embargo, como la computadora destino retira los encabezados en orden inverso a como fueron incorporados en la computadora origen, finalmente el usuario sólo recibe el mensaje original.

Unidades de información: En cada nivel, la unidad de información tiene diferente nombre y estructura, en la Tabla 2.2 se muestran los niveles del modelo *OSI.*

Niveles del modelo *OSI*

Tabla 2.2 Niveles del modelo OSI

Niveles del modelo *OSI.*
Aplicación.
Presentación.
Sesión.
Transporte.
Red.
Enlace de datos.
Físico.

A continuación se presentan los niveles o capas que componen al modelo *OSI*. Estos son siete niveles [LOZ01]:

Nivel Físico: Define el medio de comunicación utilizado para la transferencia de información, dispone del control de este medio y especifica bits de control, mediante:

- Definir conexiones físicas entre computadoras.
- Describir el aspecto mecánico de la interfase física.
- Describir el aspecto eléctrico de la interfase física.
- Describir el aspecto funcional de la interfase física.
- Definir la Técnica de Transmisión.
- Definir el Tipo de Transmisión.
- Definir la Codificación de Línea.
- Definir la Velocidad de Transmisión.
- Definir el Modo de Operación de la Línea de Datos.

Nivel Enlace de Datos: Este nivel proporciona facilidades para la transmisión de bloques de datos entre dos estaciones de red. Esto es, organiza los 1's y los 0's del nivel físico en formatos o grupos lógicos de información para:

- Detectar errores en el nivel físico.

- Establecer esquema de detección de errores para las retransmisiones o reconfiguraciones de la red.
- Establecer el método de acceso que la computadora debe seguir para transmitir y recibir mensajes. Realizar la transferencia de datos a través del enlace físico.
- Enviar bloques de datos con el control necesario para la sincronía.

En general controla el nivel y es la *interfase* con el nivel de red, al comunicarle a este una transmisión libre de errores.

Nivel de Red: Este nivel define el enrutamiento y el envío de paquetes entre redes. Es en este nivel donde los algoritmos de ruteo toman su lugar. Por lo consiguiente para los fines de este trabajo esta capa es la más importante. Dentro de sus funciones encontramos las siguientes:

- Es responsabilidad de este nivel establecer, mantener y terminar las conexiones.
- Este nivel proporciona el enrutamiento de mensajes, determinando si un mensaje en particular deberá enviarse al nivel 4 (Nivel de Transporte) o bien al nivel 2 (Enlace de datos).
- Este nivel conmuta, enruta y controla la congestión de los paquetes de información en una sub-red.
- Define el estado de los mensajes que se envían a nodos de la red.

Nivel de Transporte: Este nivel actúa como un puente entre los tres niveles inferiores totalmente orientados a las comunicaciones y los tres niveles superiores totalmente orientados al procesamiento. Además, garantiza una entrega confiable de la información.

Asegura que la llegada de datos del nivel de red encuentra las características de transmisión y calidad de servicio requerido por el nivel 5 (Sesión).

Este nivel define cómo direccionar la localidad física de los dispositivos de la red.

- Asigna una dirección única de transporte a cada usuario.
- Define una posible multicanalización. Esto es, puede soportar múltiples conexiones.
- Define la manera de habilitar y deshabilitar las conexiones entre los nodos.
- Determina el protocolo que garantiza el envío del mensaje.
- Establece la transparencia de datos así como la confiabilidad en la transferencia de información entre dos computadoras.

Nivel Sesión: proveer los servicios utilizados para la organización y sincronización del diálogo entre usuarios y el manejo e intercambio de datos.

- Establece el inicio y termino de la sesión.
- Recuperación de la sesión.
- Control del diálogo; establece el orden en que los mensajes deben fluir entre usuarios finales.
- Referencia a los dispositivos por nombre y no por dirección.
- Permite escribir programas que correrán en cualquier instalación de red.

Nivel Presentación: Traduce el formato y asignan una sintaxis a los datos para su transmisión en la red.

- Determina la forma de presentación de los datos sin preocuparse de su significado o semántica.
- Establece independencia a los procesos de aplicación considerando las diferencias en la representación de datos.
- Proporciona servicios para el nivel de aplicaciones al interpretar el significado de los datos intercambiados.
- Opera el intercambio.
- Opera la visualización.

Nivel Aplicación: Proporciona servicios al usuario del modelo *OSI*. Proporciona comunicación entre dos procesos de aplicación, tales como: programas de aplicación, aplicaciones de red, etc. Y finalmente también proporciona aspectos de comunicaciones para aplicaciones específicas entre usuarios de redes: manejo de la red, protocolos de transferencias de archivos, etc.

2.3 Protocolo *TCP/IP*

TCP/IP se llama así por dos de sus protocolos más importantes: ***Transmission Control Protocol*** (*TCP)* y el ***Internet Protocol*** (*IP*). También se le conoce como **protocolos de Internet** [CAR02].

El objetivo de *TCP/IP* fue construir una interconexión de redes que proporcionara servicios de comunicación universales: *Internet*. Cada red física tiene su propia interfase

de comunicaciones dependiente de la tecnología que la implementa. Las comunicaciones entre servicios las proporciona el software que se ejecuta entre la red física y la aplicación de usuario, y da a estas aplicaciones una interfase común, independiente de la estructura de la red física. De esta manera la arquitectura de las redes físicas es transparente al usuario.

Para poder interconectar dos redes, necesitamos un dispositivo que esté conectado a ambas redes y que pueda retransmitir paquetes de una a la otra; a este dispositivo se le llama *router*. Para ser capaz de identificar una computadora en la red, a cada una se le asigna una dirección, la *dirección IP*.

Basándonos en los protocolos que se han desarrollado, todas las tareas involucradas en la comunicación se pueden organizar en cinco capas relativamente independientes como se muestra en la Tabla 2.3.

Tabla 2.3 Capas del modelo TCP/IP [STA03]

Capas de TCP/IP
Aplicación
TCP (Transporte)
IP (Internet)
Acceso de Red
Física

El modelo de referencia TCP/IP consta principalmente de tres protocolos: *IP*, *UDP* y *TCP*. El protocolo básico es *IP* y permite enviar mensajes entre dos computadoras.

2.3.1 Protocolo IP

El protocolo IP es el más utilizado para la interconexión entre redes y cuando se diseñó ya se tuvo en cuenta la interconexión entre redes. Su trabajo es proporcionar un medio para el transporte de datagramas del origen al destino, sin importar si estas máquinas están en la misma red, o si hay otras redes entre ellas. IP está implementado en todas las computadoras y *routers*. Se preocupa de la retransmisión de los datos de una

computadora a otra, pasando por uno o varios *routers* nodo a nodo. No sabe de que aplicación son los paquetes, únicamente sabe de máquina son.

Los datos proporcionados por la capa de transporte son divididos en datagramas y transmitidos a través de la capa de red (capa *Internet*). Durante el camino puede ser fragmentado en unidades más pequeñas si deben atravesar una red o subred cuyo tamaño de paquete sea más pequeño. En la máquina destino, estas unidades son reensambladas para volver a tener el datagrama original que es entregado a la capa de transporte.

2.3.2 Datagramas IP

Para empezar con el estudio de la capa de red en Internet veamos el formato de los datagramas IP. El datagrama IP consiste en una parte de cabecera y en una parte de datos cuyo tamaño es variable.

Cabecera: En la cabecera hay una parte fija de 20 Bytes y una parte opcional de longitud variable. En la Tabla 2.4 se puede ver el formato de la cabecera IP.

Tabla 2.4 Cabecera del datagrama IP [STA03]

Cabecera del datagrama IP					
| ---------- 32 bits ----------|					
Versión	IHL	Tipo de Servicio	Longitud Total		
Identificación				DF	MF	Desplazamiento del fragmento
Tiempo de Vida		Protocolo	Suma de Comprobación		
Dirección del Origen					
Dirección del destino					
Opciones (desde 0 ó más palabras)					

A continuación se detallan los elementos que forman a la cabecera IP que se muestra en la Tabla 2.4 [STA03].

- **Versión** (4 bits): Indica el número de versión del protocolo al que pertenece el datagrama, lo que permitirá la evolución futura del protocolo y que la transición

entre las versiones se pueda hacer ejecutándose en unas máquinas la versión vieja y en otras la versión nueva.

- **IHL** (Internet Header length) (4 bits): Indica la longitud de la cabecera en palabras de 32 bits Este campo es necesario por no ser constante el tamaño de la cabecera.
- **Tipo de servicio** (8 bits): Permite que la computadora *(host)* especifique que clase de servicio quiere, pudiéndose combinar confiabilidad y velocidad. Para la voz digitalizada es mas importante realizar la entrega de forma rápida que precisa, mientras que para la transferencia de ficheros no importa a que velocidad se realiza la transferencia pero si que esté libre de errores.
- **Longitud total** (16 bits) en bytes que tendrá todo el datagrama, considerando tanto la cabecera como los datos. Hay que tener en cuenta que el tamaño máximo de un datagrama es de 65535 bytes.
- **Identificador** (16 bits): es un número de secuencia que junto a la dirección origen, la dirección destino y el protocolo de usuario, sirven para que la computadora destino determine a que datagrama pertenece el fragmento que ha recibido. Todos los fragmentos de un datagrama contienen el mismo valor en el campo identificador y este número debe ser único para la dirección origen, la dirección destino y el protocolo de usuario durante el tiempo en el que el datagrama permanece en el conjunto de redes.
- **Banderas** (3 bits): El primer bit no se utiliza actualmente. El indicador de mas fragmentos (**MF**) cuando vale 1 indica que este datagrama tiene mas fragmentos y toma el valor 0 en el último fragmento. El indicador de no fragmentar (**DF**) prohíbe la fragmentación cuando vale 1. Es una orden que se le da a los *routers* de que no fragmenten el datagrama cuando el destino es incapaz de reensamblarlo.
- **Desplazamiento del fragmento** (13 bits): Indica en que posición del datagrama original, medido en unidades de 8 bytes (64 bits), va el fragmento actual.
- **Tiempo de vida** (8 bits): Es un contador que sirve para limitar la vida de un paquete. Cuenta el número de saltos por el que pasa. Cuando el contador llega a cero, el paquete se descarta y se envía un paquete a la computadora origen

avisándole. Con este mecanismo se consigue que los datagramas no permanezcan indefinidamente en la red si, por ejemplo, se dañan las tablas de ruteo.

- **Protocolo** (8 bits): Se utiliza por la capa de red para saber a que protocolo de la capa de transporte le tiene que enviar el datagrama una vez lo ha reensamblado. Existen diferentes protocolos de transporte, entre ellos TCP y UDP.
- **Suma de comprobación** (16 bits): Sirve para verificar el contenido de la cabecera y es útil para la detección de errores generados durante la transmisión del datagrama.
- **Dirección origen** (32 bits): Indica el número de red y el número de la computadora que envía el datagrama.
- **Dirección destino** (32 bits): Indica el número de red y el número de la computadora al que se envía el datagrama.
- **Opciones** (variable): Contiene las opciones solicitadas por el usuario que envía los datos y se diseñó para que las versiones posteriores del protocolo pudieran incluir información no considerada originalmente. Hay seis opciones [CAR02]:
    - *Seguridad*
    - *Ruteo estricto desde el origen:* Es una secuencia de direcciones IP que sirve para indicar la trayectoria completa que debe seguir el datagrama desde el origen hasta el destino. Esta opción es usada sobre todo cuando los administradores de las computadoras envían paquetes de emergencia porque las tablas de ruteo se han corrompido o para hacer mediciones de tiempo.
    - *Ruteo libre desde el origen:* Es una secuencia de direcciones IP que sirve para indicar que el datagrama debe pasar obligatoriamente por algún *router* y en un cierto orden, pero también puede pasar por otros *routers.*
    - *Registrar la ruta:* Sirve para indicar a los *routers* que agreguen su dirección IP al campo de opción y de esta manera tener conocimiento de la ruta seguida por el datagrama. Se utiliza, por ejemplo, para poder determinar si los algoritmos de ruteo están funcionando correctamente. Los 40 bytes de tamaño máximo que puede tener el campo de opciones

sólo permite registrar 9 saltos, lo que puede ser en las redes actuales en muchos casos insuficiente.

- *Identificación de secuencia*
- *Marca de tiempo:* En este caso, además de registrar las direcciones de los *routers*, se utilizan 32 bits para guardar una marca de tiempo expresada en milisegundos. Esta marca es usada principalmente para buscar fallos en los algoritmos de ruteo.

2.3.3 Direcciones IP

Cada computadora y cada *router* tiene una dirección única cuya longitud será de 32 bits, que es utilizada en los campos dirección origen y dirección destino de la cabecera que se mostró en la Tabla 2.4 Esta dirección consta de un identificador de red y de un identificador de computadora. La dirección está codificada para permitir una asignación variable de los bits utilizados al especificar la red y la computadora. Este formato de direcciones permite mezclar las tres clases de direcciones en el mismo conjunto de redes. La dirección IP más pequeña es la 0.0.0.0 y la mayor es 255.255.255.255 [CAR02].

Existen tres clases de redes que se pueden clasificar teniendo en cuenta la longitud del campo de red y del campo de computadora *(host)*. La clase a la que pertenece una dirección puede ser determinada por la posición del primer 0 en los cuatro primeros bits. Las direcciones están codificadas para permitir una asignación variable de bits para especificar la red y la computadora en la Tabla 2.5 se muestran las clases de direcciones *IP*.

- **Clase A:** Pocas redes: 128, cada una con muchas computadoras: 16777216.
- **Clase B:** Un número medio de redes: 16384, cada una con un número medio de computadoras: 65536.
- **Clase C:** Muchas redes: 2097152, cada una con pocas computadoras: 256.
- **Clase D:** Permite hacer *multicasting* en la cual el datagrama se dirige a múltiples computadoras.
- **Clase E:** Reservado para el futuro.

Tabla 2.5 Direcciones IP[CAR02]

Direcciones IP				
| ---------- 32 bits ----------|				
Clase				Rango de Direcciones
A	0	Red	*Computadora (Host)*	1.0.0.0 127.255.255.255
B	1 0	Red	*Computadora (Host)*	128.0.0.0 191.255.255.255
C	1 1 0	Red	*Computadora(Host)*	192.0.0.0 223.255.255.255
D	1 1 1 0	Multicasting		224.0.0.0 239.255.255.255
E	1 1 1 1 1	Reservado para el futuro		240.0.0.0 247.255.255.255

2.4 Algoritmos de Ruteo

Una de las funciones básicas de IP es su habilidad para conectar distintas redes físicas. Esto es posible gracias a la flexibilidad de IP para utilizar para usar casi cualquier red física, y a sus algoritmos de ruteo. Al sistema que realiza esta función se le denomina *router*. Aunque ya lo hemos mencionado mucho anteriormente veamos ahora una definición formal de un *router*.

2.4.1 Router

Un ***router*** es un conmutador de paquetes que opera en el nivel de red del modelo *OSI* y en el protocolo IP del conjunto de protocolos *TCP/IP*. Los *routers* permiten interconectar tanto redes de área local *(LANs)* como redes de área extensa *(WAN)*. Otra de sus funciones es la de proporcionar un control del tráfico y funciones de filtrado a nivel de red, es decir, trabajan con direcciones de nivel de red, como por ejemplo, con direcciones IP.

Los *routers* son capaces de rutear dinámicamente, es decir, son capaces de seleccionar el camino que debe seguir un paquete en el momento en el que les llega, teniendo en cuenta factores como líneas más rápidas, líneas más baratas, líneas menos saturadas, etc.

Los *routers* son más sofisticados que los *switches*, sin embargo, esto los hace más caros. A diferencia de los *switches* y *bridges*, que sólo leen la dirección *Medium Access Control (MAC)*, los *routers* analizan la información contenida en un paquete de red leyendo la dirección de red. Los *routers* tienen la peculiaridad de que pueden almacenar datos en unas tablas, estas tablas se conocen como **tablas de ruteo.**

En general en esta tabla se pueden encontrar tres tipos rutas:

- Rutas directas, para redes conectadas localmente
- Rutas indirectas, para redes accesibles a través de uno o más *routers*

Los *routers* leen cada paquete y lo envían a través del camino más eficiente posible al destino apropiado, según una serie de reglas recogidas en sus tablas. Los *routers* se utilizan a menudo para conectar redes geográficamente separadas. El *router* es entonces la conexión vital entre una red y el resto de las redes. Un *router* también sabe cuándo mantener el tráfico de la red local dentro de ésta y cuándo conectarlo con otras *Local Area Networks (LAN)*, es decir, permite filtrar los *broadcasts* de nivel de enlace. Un *router* dispondrá de una o más interfases de red local, las que le servirán para conectar múltiples redes locales usando protocolos de nivel de red.

La función principal del nivel de Internet (nivel de Red) es hacer llegar los paquetes de una computadora a otra no importando cual sea el medio físico que utilicen ni los datos que estén transmitiendo; el enrutamiento es justamente eso.

2.4.2 Algoritmos de ruteo

Recapitulemos un poco lo que hemos visto: Cuando un paquete llega a un *router*, éste determina el enlace al cual el paquete debe de ser transferido y se indexa una tabla en la cual se guardan los valores de envío. Los algoritmos de ruteo, cuando operan en *routers* en redes de comunicaciones, tienen la peculiaridad de intercambiar y de procesar información entre *routers* de las tablas que utilizan para el envío de paquetes [TRI98].

La capa de red tiene la función de determinar el camino que los paquetes deben seguir desde un *router* inicial hasta uno final. El trabajo de ruteo es definir caminos desde el inicio hasta el final a través de una red de *routers.* Usualmente un *host* esta directamente relacionado uno a uno con un *router* en la red, a este *router* se le conoce

como *router* por predeterminado o por defecto. Siempre que se emite un paquete desde el *host* el paquete es enviado al *router* por defecto.

El propósito de un algoritmo de ruteo es simple: dado un grupo de *routers* con enlaces conectando a los mismos, un algoritmo de ruteo es aquel que busca un "buen" camino desde el *router* fuente hasta el *router* destino. Usualmente, un buen camino es aquel que presenta el menor coste de enlace. Sin embargo en la práctica, existen muchas políticas que entran en juego en la toma de decisiones en el ruteo de paquetes y que hacen el del simple concepto de algoritmos de ruteo en algoritmos complejos.

Un grafo es usado para formular problemas de ruteo. Un grafo $G = (N, E)$ donde N es un grupo de nodos, y E es una colección de ejes, en donde cada eje contiene un par de nodos contenidos en N. En el contexto de ruteo en redes de comunicaciones, los nodos en una gráfica representan a los *routers* (que son los puntos donde se toma la decisión de la ruta del envío del paquete) y los ejes que conectan a estos nodos representan a los enlaces físicos que conectan a dichos *routers*. Así que cuando se haga referencia a un nodo, en realidad estamos hacienda referencia a un *router* [KUR05].

En la Figura 2.2 se muestra una representación abstracta de una topología de seis nodos. Como podemos observar en el modelo, para cada eje o enlace se tiene un valor, dicho valor representa el costo o peso del enlace. Usualmente el peso del enlace es reflejo de la longitud física del enlace. Por ejemplo, un enlace trasatlántico puede tener un peso mayor que un simple enlace conectando a dos ciudades cercanas. El peso del enlace también es reflejo de varios factores como la velocidad del enlace, el costo monetario asociado al enlace, etc. Para los fines de esta tesis, nos limitaremos a saber que existe un costo de enlace sin preocuparnos a detalle cómo se determinó tal enlace. Simplemente daremos diferentes pesos a los enlaces representados por escalares. Para cualquier eje (x, y) dentro de E vamos a de denotar $c(x, y)$ para el costo del enlace entre los nodos x y y Si el par (x, y) no existe en E entonces definimos $(x, y) = \infty$.

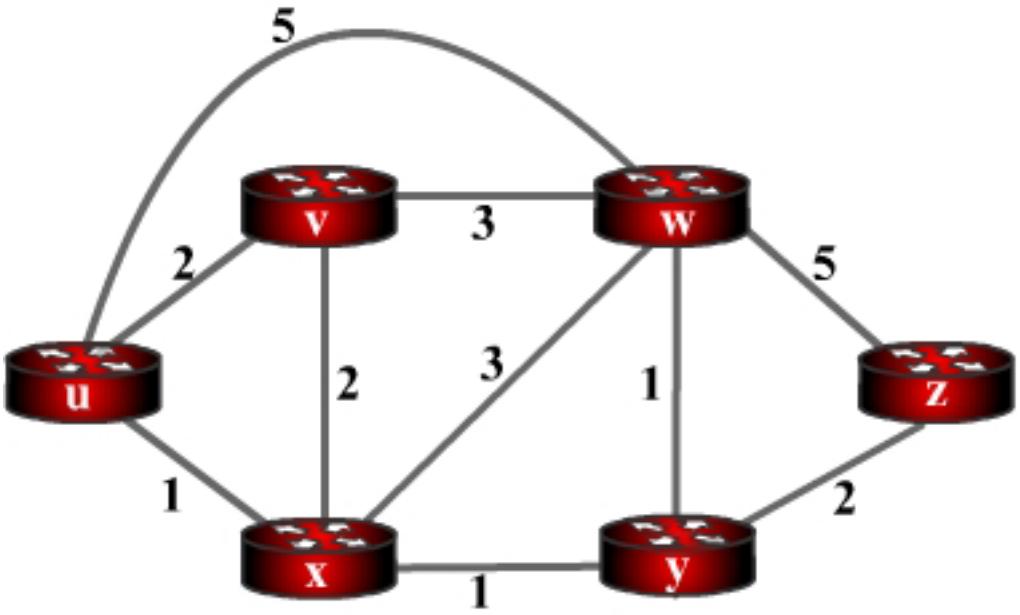


Figura 2.2 Ejemplo de una topología de seis nodos.

Un nodo y se dice que es *vecino* de un nodo x si (x, y) pertenece a E. Dados los costes de los ejes (o enlaces), el principal objetivo de un algoritmo de ruteo es identificar el camino con menor coste entre un nodo inicial (o fuente) y un nodo final (o destino). Para profundizar un poco en el tema definamos que el **camino** en un grafo $G = (N, E)$ es una secuencia de nodos $(x_1, x_2, ..., x_p)$ tales que cada par de nodos $(x_1, x_2), (x_1, x_3), (x_{p-1}, x_p)$ pertenezcan a los ejes E. El coste del camino $(x_1, x_2, ..., x_p)$ es simplemente la suma de todos los costes de los ejes a lo largo del camino, esto es $c(x_1, x_2) + c(x_1, x_3) + c(x_{p-1}, x_p)$. Dados cualquier par de nodos x y y, usualmente hay muchos caminos entre estos nodos, y cada camino con su respectivo coste asociado. Uno o más de estos caminos es el camino de menor coste. Así, el problema del camino del menor coste es claro: Encontrar un camino entre una fuente y un destino que tenga el menor coste. En la Figura 2.2 por ejemplo el camino con menor coste entre el nodo fuente u y el nodo destino w es (u, x, y, w) con un coste de 3. Cabe destacar que si el coste de los enlaces es igual para todos los enlaces, entonces el camino con el menor coste es igual al camino con menor longitud (o al camino con menor numero de enlaces entre el nodo fuente y el nodo destino) [KUR05].

Una manera de clasificar a los algoritmos de ruteo es de acuerdo a si estos son globales o descentralizados. A continuación se explica cada uno de ellos [KUR05].

Algoritmo de ruteo global: Un algoritmo de ruteo global calcula el camino con menor coste entre una fuente y un destino utilizando información global de la red de comunicaciones. Esto es, el algoritmo toma la conectividad entre todos los nodos y todos los costes como entradas. Esto luego requiere que el algoritmo de alguna manera obtenga esta información antes de realizar algún los cálculos requeridos. Los mismos cálculos pueden ser corridos en algún sitio (un algoritmo de ruteo global centralizado) o calculado en múltiples sitios. La clave para distinguirlos es que el algoritmo global posee la información completa acera de la conectividad y de los costos de cada enlace en la red de comunicaciones. En la práctica los algoritmos de información global son comúnmente referidos como **algoritmos de estado-enlace**, pues el algoritmo tiene que tener en cuenta del costo de cada enlace en la red de comunicaciones. Una explicación más detallada se encuentra en el apartado 2.5 Algoritmo de *Dijkstra* de esta tesis.

Algoritmo de ruteo descentralizado: El cálculo de el camino con menor coste es llevado a cabo a base iteraciones y de un modo distribuido. Ningún nodo en la red tiene información completa de los cosos de los enlaces de toda la red de comunicaciones. Por el contrario, cada nodo comienza solamente con el conocimiento de los costes de los enlaces de los nodos que están directamente conectados a si mismo. Entonces, a través de un proceso de iteraciones de cálculo y de intercambio de información con los nodos vecinos (o sea los nodos que están directamente conectados a si mismos). Un nodo gradualmente calcula el camino con menor coste hacia un destino o un grupo de destinos. Se profundizará más en la sección 2.6 Algoritmo de *Bellman-Ford*, donde analizaremos el **algoritmo de vector-distancia**. Es llamado así porque cada nodo mantiene un vector con las estimaciones de los costos hacia todos los demás nodos en la red.

Una segunda manera general de clasificar a los algoritmos de ruteo es de acuerdo a si estos presentan características dinámicas o estáticas. En los algoritmos de ruteo estáticos las rutas cambian muy lentamente con respecto al tiempo, a menudo como resultado de intervención humana (por ejemplo cuando alguien modifica las tablas de ruteo de manera manual). Los algoritmos de ruteo dinámicos cambian el camino según los cambios en el tráfico de la red o de la topología. Un algoritmo dinámico puede ser ejecutado ya sea de

manera periódica o de manera directa con respuesta a los cambios de topología o cambio en los costos de enlace. Mientras que los algoritmos de ruteo dinámicos responden más con los cambios en la red, también son más susceptibles a problemas tales como oscilaciones en las rutas, y lazos en las rutas.

Otra manera de clasificar a los algoritmos de ruteo es de acuerdo a si son sensibles a la carga o si son insensibles a la carga. En los algoritmos que son sensibles a la carga, los costos de los enlaces varían dinámicamente para reflejar el nivel actual de congestionamiento de la red. Si un coste alto es asociado con un enlace que actualmente esta congestionado, el algoritmo tratará de elegir rutas alternas para evitar pasar por el enlace congestionado. Hoy en día los algoritmos de ruteo para Internet (tales como *RIP*, *OSPF*, y *BGP*) son insensibles a la carga, pues en el pasado se encontró con muchos problemas al implementar los algoritmos sensibles a la carga.

2.4.2.1 OSPF (Open Shortest Path First)

OSPF se usa, como *RIP*, en la parte interna de las redes, es decir la parte que no está conectada al *backbone* de *Internet*. Su forma de funcionar es bastante sencilla pero a la vez algo complejo. Cada *router* conoce los *routers* vecinos y las direcciones que posee cada *router* de los vecinos. *OSPF* es un protocolo de ruteo complejo que esta basado en el algoritmo de *Dijkstra*. Un ejemplo basado en una topología de seis nodos se analiza en la sección 2.5 *Algoritmo de Dijkstra.* .Los beneficios de esta complejidad (sobre *RIP*) son los siguientes:

- Los *routers* OSPF convergerán mucho más rápido que los *routers* RIP tras cambios de topología. Este efecto se hace más pronunciado al aumentar el tamaño del AS.
- Incluye ruteo *Type of Service (TOS)* diseñado para calcular rutas separadas para cada tipo de servicio. Para cada destino, pueden existir múltiples rutas, cada una para uno o más TOSs.
- Proporciona balanceamiento de la carga ya que una *router* OSPF puede emplear varios caminos de igual coste mínimo.
- A cada ruta se le asocia una máscara de subred.

- Todos los intercambios entre *routers* se pueden autentificar mediante el uso de passwords.
- OSPF soporta rutas específicas de computadoras, redes y subredes.
- OSPF permite que las redes y las computadoras contiguas se agrupen juntos en *áreas* dentro de un sistema autónomo, simplificando la topología y reduciendo la cantidad de información de ruteo que se debe intercambiar. La topología de un área es desconocida para el resto de las áreas.
- Permite el intercambio de información de ruteo externa, es decir, información de ruteo obtenida de otro sistema autónomo.
- Permite el uso de enlaces punto a punto sin direcciones IP, lo que puede ahorrar recursos escaso en el espacio de direcciones IP.

2.4.2.2 RIP (Routing information protocol).

RIP es un protocolo de enrutado interno basando en el algoritmo *Bellman-Ford*. Es muy usado en sistemas de conexión a *Internet* en el que muchos usuarios se conectan a una red y pueden acceder por lugares distintos. Un ejemplo basado en una topología de seis nodos se analiza en la sección 2.6 *Algoritmo de Belman-Ford.*

Operaciones básicas

- Cuando RIP se inicia envía un mensaje a cada uno de sus vecinos pidiendo una copia de la tabla de ruteo del vecino. Los *routers* vecinos devuelven una copia de sus tablas de ruteo.
- Cuando RIP está en modo activo envía toda o parte de su tabla de ruteo a todos los vecinos. Esto se hace cada 30 segundos. La tabla de ruteo se envía como respuesta.
- Cuando RIP descubre que una métrica ha cambiado, la difunde a los demás *routers*.
- Cuando RIP recibe una respuesta, el mensaje se valida y la tabla local se actualiza si es necesario.
- Para mejorar el rendimiento y la fiabilidad, RIP especifica que una vez que un *router* a aprendido una ruta de otro, debe guardarla hasta que conozca una mejor

(de coste estrictamente menor). Esto evita que los *routers* oscilen entre dos o más rutas de igual coste.

- Cuando RIP recibe una petición, distinta de la solicitud de su tabla, se devuelve como respuesta la métrica para cada entrada de dicha petición fijada al valor de la tabla local de ruteo. Si no existe ruta en la tabla local, se pone a 16.
- Las rutas que RIP aprende de otros *routers* expiran a menos que se vuelvan a difundir en 180 segundos. Cuando una ruta expira, su métrica se pone a infinito, la invalidación de la ruta se difunde a los vecinos, y 60 segundos más tarde, se borra de la tabla.

Limitaciones

RIP no está diseñado para resolver cualquier posible problema de ruteo. Tiene las siguientes limitaciones:

- El coste máximo permitido en RIP es 16, que significa que la red es inalcanzable. De esta forma, RIP es inadecuado para redes grandes (es decir, aquellas en las que la cuenta de saltos puede aproximarse perfectamente a 16).
- En un mensaje RIP no hay ningún modo de especificar una máscara de subred asociada a una dirección IP.
- RIP carece de servicios para garantizar que las actualizaciones proceden de *routers* autorizados. Es un protocolo inseguro.
- RIP sólo usa métricas fijas para comparar rutas alternativas. No es apropiado para situaciones en las que las rutas necesitan elegirse basándose en parámetros de tiempo real tales como el retardo, la fiabilidad o la carga.

2.4.2.3 BGP (Border gateway protocol).

BGP es un protocolo muy complejo que se usa en la interconexión de redes conectadas por un *backbone* de *Internet*. Cuando el protocolo enlaza a dos *routers* en diferente *sistema autónomo* se le llama *exterior Border Gateway Protocol (eBGP),* si el enlace es dentro del mismo *sistema autónomo* se le llama *interior Border Gateway Protocol (iGBP)* [KUR05]. Este protocolo usa parámetros como ancho de banda, precio de la

conexión, saturación de la red, denegación de paso de paquetes, etc. para enviar un paquete por una ruta o por otra. Un *router BGP* da a conocer sus direcciones IP a los *routers BGP* y esta información se difunde por los *routers BGP* cercanos y no tan cercanos. *BGP* tiene sus propios mensajes entre *routers*, no utiliza *RIP*.

2.5 Algoritmo de *Dijkstra*

Recapitulemos: En un algoritmo de estado enlace, la topología de la red y todos los costes de enlaces son conocidos. Esto se puede obtener haciendo que cada nodo realice una transmisión de paquetes estado enlace hacia todos los demás nodos en la red. Cada paquete estado enlace contiene las identificaciones y los costes de los enlaces que se encuentran directamente enlazados a ese nodo. El resultado de la retransmisión es que todos los nodos obtienen información idéntica y completa de la red [KUR05].

El algoritmo de *Dijkstra* calcula el camino con menor coste desde un nodo fuente hacia todos los demás nodos en la red. El algoritmo de *Dijkstra* trabaja a base de iteraciones y tiene la propiedad de que después de la iteración k, el camino más corto es conocido para k nodos destino. Definamos las siguientes notaciones:

- $D(v)$: Coste del camino con menor coste desde el nodo fuente al nodo destino v.
- $p(v)$: Nodo previo (vecino v) a lo largo del actual camino con menor coste desde el nodo fuente v.
- N': Subgrupo de nodos; v esta en N' si el camino con menor coste desde el nodo fuente es conocido.

El algoritmo de ruteo global cosiste de un paso de inicialización seguido por un lazo. El número de veces que el lazo es ejecutado es igual al número de nodos en la red. Cuando el algoritmo termina éste habrá calculado el camino con menor coste desde el nodo fuente u hacia cada nodo en la red.

Algoritmo estado enlace para un nodo fuente u [KUR05]:

1: **Inicialización**

2: $N' = \{u\}$

3: para todo nodo v.

4: SI v es vecino de u

5: entonces $D(v) = c(u, v)$

6: SI no $D(v) = \infty$

7:

8: **Lazo**

9: encuentra w que no este en N' tal que $D(w)$ es el menor

10: añade w a N'

11: actualiza $D(v)$ para cada vecino v de w y que no este en N':

12: $D(V) = \min[D(v), D(w) + c(w, v)]$

13: /* Nuevo coste de v es o el antiguo coste de v o el costo del camino mas 14:
corto conocido a w más el costo desde w a v */

15: Hasta que $N' = N$

Como ejemplo, consideremos una red como se muestra en la Tabla 2.2 y calculemos el camino con menor coste desde u hacia todas las destinaciones posibles. Una tabla con todos los resultados de los cálculos del algoritmo se puede ver en la Tabla 4.03 donde cada línea de la tabla muestra los valores de las variables al final de cada iteración al ejecutar el algoritmo. Vamos a considerar los primeros pasos con mucho detalle.

Tabla 2.6 Ejecución del algoritmo enlace estado de la Figura 2.2

N'	*D(v),p(v)*	*D(w),p(w)*	*D(x),p(x)*	*D(y),p(y)*	*D(z),p(z)*
U	2,U	5,U	1,U	∞	∞
UX	2,U	4,X		2,X	∞
UXY	2,U	3,Y			4,Y
UXYV		3,Y			4,Y
UXYVW					4,Y
UXYVWZ					

En el paso de inicialización el actual valor del camino con menor coste conocido desde u hasta sus vecinos que se encuentran directamente enlazados a él: u, x y y son inicializados en 2,1 y 5 respectivamente. Note en particular que el coste a w es puesto en

5 pues este es el coste del enlace directo de u a w. Los costos de y a z son puestos en infinito porque no están directamente conectados a u.

En la primera iteración analizamos aquellos nodos que todavía no han sido añadidos al grupo N' y encontrar aquel nodo con el menor coste en el final de la iteración anterior. Ese nodo es x y su coste es 1, es por eso que x es añadido al grupo N'. La línea número 12 del algoritmo es ejecutada para actualizar $D(v)$ para todos los nodos v. Es así como obtenemos los resultados que se muestran en la segunda línea de la tabla mostrada en la Tabla 2.6 El coste del camino a v ya no será cambiado. El coste del camino a w (el cual era 5 al final del proceso de inicialización) a través del nodo x es igual a un coste de 4. El algoritmo selecciona este nuevo camino de coste más bajo y el predecesor de w a lo largo del camino más corto desde u es igual a x. Similarmente el coste a y a través de x según los cálculos es igual a 2 y la tabla es actualizada de acuerdo a estos resultados.

En la segunda iteración los nodos v y y resultan tener los caminos más cortos (2) y el algoritmo añade a y al grupo N' así N' ahora contiene a u, x y a y. El coste de los nodos restantes que todavía no están en el grupo N' (nodos v, w y z) son actualizados en la línea 12 del algoritmo mostrando los resultados como se muestran en la Tabla 2.6

El algoritmo continúa de manera similar las iteraciones restantes. Cuando el algoritmo termina tenemos para cada nodo su predecesor junto con el camino de menor coste desde el nodo fuente. Para cada predecesor también tenemos a su predecesor y así de esta manera podemos construir el camino entero desde la fuente a todos los nodos. La tabla de envíos en un nodo por ejemplo el nodo u puede ser construido por medio de la información guardando para cada destinación el salto al próximo nodo en el camino de coste con menor coste desde u hasta su destino.

¿Cuál es la complejidad computacional de este algoritmo? En otras palabras dados n nodos (sin contar el nodo fuente) ¿qué tantos cálculos se deben hacer en el peor de los casos para encontrar el camino con menor coste desde un nodo fuente hasta todas las destinaciones? En la primera iteración debemos buscar en todos los n nodos para determinar el nodo w que no este en N' que tenga el coste mínimo. En la segunda iteración, necesitamos verificar $n-1$ nodos para determinar el costo mínimo; en la tercera iteración $n-2$ nodos y así consecutivamente. El total de nodos que debemos

buscar en todas las iteraciones es $n(n+1)/2$, y de ahí se puede decir que la implementación precedente del algoritmo estado enlace tiene una complejidad en el peor de los casos de orden n cuadrada: $O(n^2)$.

2.6 Algoritmo de *Bellman-Ford*

Mientras que el algoritmo de estado enlace utiliza información global el algoritmo de vector distancia es iterativo, asíncrono y distribuido. Se dice que es distribuido porque cada nodo recibe información de uno o más de sus nodos vecinos directamente enlazados a él, realiza el cálculo y luego distribuye de regreso el resultado de los cálculos a sus nodos vecinos. Se dice que es iterativo en el sentido de que éste proceso continua hasta que no se intercambia información entre los vecinos. Es interesante notar que este algoritmo se termina por sí mismo; no existe una señal que le indique al algoritmo cuando parar, simplemente se para. Por último se dice que el algoritmo es asíncrono, pues no requiere que los nodos operen de manera conjunta entre ellos [KUR05].

Analicemos la relación que existe entre los costos y el camino con menor coste. Hagamos que $d_x(y)$ sea el costo del camino con menor coste del nodo x al nodo y. Entonces, los costos mínimos están relacionados por la ecuación *Bellman-Ford*

$$d_x(y) = \min_v[c(x,v) + d_v(y)]$$

(2.1)

Donde el $\min_v$ en la ecuación es obtenido de todos los vecinos x. La ecuación de *Bellman-Ford* es un tanto intuitiva. De hecho, después de haber viajado desde x a v, y si después tomamos el camino con menor coste de v a y el coste del camino será $c(x,v) + d_v(y)$ dado que debemos comenzar viajando a un vecino v el menor coste desde x hasta y es el mínimo de $c(x,v) + d_v(y)$ obtenidos de todos los vecinos v.

Como ejemplo, refiriéndonos al ejemplo de la Figura 2.3, verifiquemos para el nodo fuente u y como destino el nodo z. El nodo fuente u tiene tres vecinos: nodos v, x y w. Si recorremos varios caminos en la gráfica es fácil observar que $d_v(z) = 5, d_x(z) = 3$ y $d_w(z) = 3$. Insertando estos valores en la ecuación de *Bellman-Ford* junto con los costos $c(u,v) = 2, c(u,x) = 5$ y $c(u,w) = 1$ da como resultado

$d_u(z) = \min[2+5, 5+3, 1+3] = 4$ lo cual es correcto y es exactamente igual por lo obtenido por el algoritmo de *Dijkstra* para la misma red.

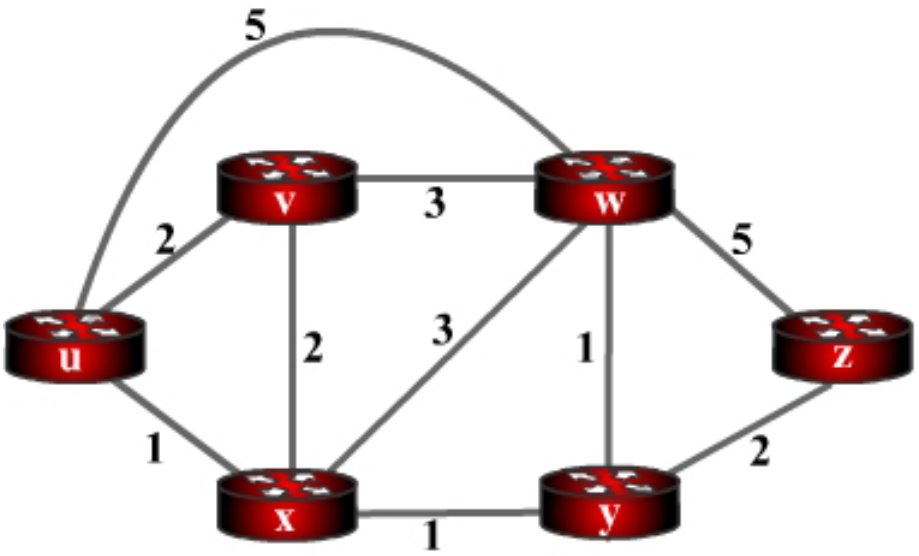


Figura 2.3 Modelo de una red de comunicaciones para ejemplo Bellman-Ford

La ecuación de *Bellman-Ford* provee las entradas en la tabla de envíos del nodo x. Para ejemplificar esto, digamos que v^* es cualquier nodo vecino que es el menor en la ecuación de *Bellman-Ford.* Entonces, si el nodo *x* quiere mandar un paquete al nodo *y* a lo largo del camino con menor coste, entonces este debería enviar el paquete hacia el nodo v^*. Así, la tabla de envíos del nodo *x* debe especificar como nodo siguiente (o brinco inmediato) al nodo v^* para el destino ultimo del nodo *y*.

La idea básica es la siguiente: cada nodo *x* comienza con un estimado del camino con menor coste hacia el nodo *y*: $D_x(y)$. Para todos los nodos *N* hagamos que $D_x = [D_x(y) : y \ \ en \ \ N]$ sea el vector distancia del nodo *x*. Este vector contiene los costes estimados desde el nodo *x* hacia todos los demás nodos *y* en *N.* En el algoritmo de vector distancia cada nodo *x* mantiene los siguientes datos de ruteo.

Para cada vecino *v*, el coste $c(x, y)$ desde *x* hacia sus vecinos directamente ligados *v.* El vector distancia del nodo *x*. Este es: $D_x = [D_x(y) : y \ \ en \ \ N]$ que contiene los estimados del nodo *x* hacia todas las destinaciones *y* en *N*. Los vectores distancia para cada vecino $D_v = [D_v(y) : y \ \ en \ \ N]$ para cada vecino *v* de *x*.

En el algoritmo de *Bellman-Ford*, cada nodo manda una copia de su vector distancia a cada uno de sus vecinos. Cuando un nodo *x* recibe un nuevo vector distancia de cualquiera de sus vecinos *v*, este guarda el vector distancia de *v* y luego usa la ecuación de *Bellman-Ford* para actualizar su propio vector distancia mediante la siguiente ecuación:

$$D_x(y) = \min_v [c(x,v) + D_v(y)] \text{ Para cada nodo } y \text{ en } N.$$

Si el vector distancia de *x* cambia como resultado de este paso de actualización, el nodo *x* entonces manda su vector distancia actualizado a cada uno de sus vecinos, los cuales actualizan su propio vector distancia. Mientras todos los nodos continúen intercambiando sus vectores distancia en un modo asíncrono cada costo estimado $D_x(y)$ converge a $d_x(y)$ que es coste del camino de menor coste desde el nodo *x* hacia el nodo *y*.

Algoritmo Bellman-Ford [KUR05].

1: **Inicialización**
2: Para todos los destinos en *N*
3: $D_x(y) = c(x,y)$ /* si *y* no es un vecino que $c(x,y) = \infty$ /*
4: Para cada vecino *w*
5: $D_x(y) = \infty$ para todos los destinos *y* en *N*
6: Para cada vecino *w*
7: mandar vector distancia $D_x = [D_x(y) : y \ en \ N]$ a *w*
8:
9: **Lazo**
10: esperar (hasta que vea el cambio del coste de enlace en un vecino *w* o
11: hasta que reciba un vector distancia de algún vecino *w*)
12:
13: Para cada y en N:
14: $D_x(y) = \min_v [c(x,v) + D_v(y)]$
15:

16: SI $D_x(y)$ cambia para cualquier destino y

17: mandar vector distancia $D_x = [D_x(y) : y \ \ en \ \ N]$ a todos

18: los vecinos

19: **para siempre**

El algoritmo de vector distancia muestra como un nodo x actualiza su vector distancia y estima cuando ya sea que vea un cambio en el coste de alguno de sus vecinos que estén directamente conectados o cuando reciba la actualización del vector distancia de alguno de sus vecinos que estén directamente conectados a el. Pero para actualizar su propia tabla de envíos para un destino dado *y*, lo que el nodo *x* realmente necesita saber no es el camino con menor distancia a *y* sino el nodo vecino $v*$ que es el *router* de siguiente salto a lo largo del camino mas corto a *y*. Como se puede esperar el *router* de siguiente salgo $v*$ es el vecino *v* que obtuvo el mínimo en la línea 14 del algoritmo. Entonces en las líneas 13 y 14, para cada destino *y*, el nodo *x* también determina a $v*$ y actualiza su tabla de envíos para el destino *y*.

El algoritmo de vector distancia, como se mencionó anteriormente, es descentralizado y no usa información global. De hecho, la única información un nodo tendrá es el coste de los enlaces de sus vecinos directamente enlazados y la información que reciba de ellos. Cada nodo espera a ser actualizado por algún vecino (Líneas 10.11), calcula su nuevo vector distancia cuando recibe una nueva actualización (Línea 14), y distribuye su nuevo vector distancia a sus vecinos (Líneas 16.17). El algoritmo de vector distancia es utilizado en muchos protocolos de ruteo que se utilizan hoy en día, incluyendo los protocolos *RIP* y *BGP*.

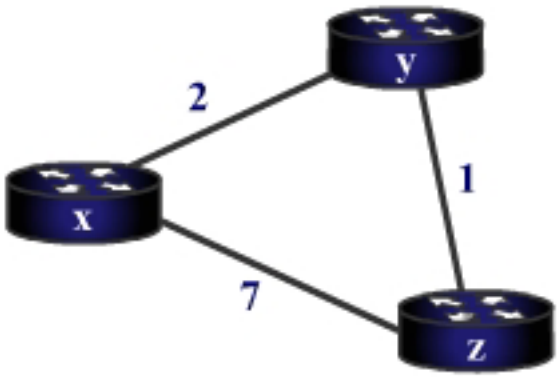


Figura 2.4 Red de comunicaciones de tres nodos.

Veamos un ejemplo de una simple red de tres nodos como se muestra en la Figura 2.4 Las tablas correspondientes se encuentran en la Tabla 2.7. Esta tabla contiene tres tablas de ruteo para cada uno de los tres nodos. Por ejemplo, la tabla que se encuentra en la esquina superior izquierda es la tabla inicial de ruteo del nodo *x*. Dentro de una tabla de ruteo específica, cada fila es el vector distancia. Cada tabla de ruteo incluye su propio vector distancia y el vector distancia de cada uno de sus vecinos. Así, la primera fila en la tabla inicial de ruteo del nodo *x* es $D_x = [D_x(x), D_x(y), D_x(z)] = [0,2,7]$. La segunda y tercera fila en esta tabla representan los vectores distancia recientemente recibidos desde los nodos *y* ó *z*, las entradas en la segunda y tercera fila son inicializados en infinito.

Después de su inicialización, cada nodo manda su propio vector distancia a cada uno de sus dos nodos vecinos como se muestra en la Tabla 2.7 por las flechas desde la primera columna de las tablas hasta la segunda columna de las tablas. Por ejemplo, el nodo *x* manda su vector distancia $D_x = [0,2,7]$ a ambos nodos: *y* y *z*. Una vez recibidas las actualizaciones, cada nodo calcula nuevamente su propio vector distancia. Por ejemplo el nodo *x* calcula:

$$D_x(x) = 0$$

$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,y) + D_z(y)\} = \min\{2+0,7+1\} = 2$$

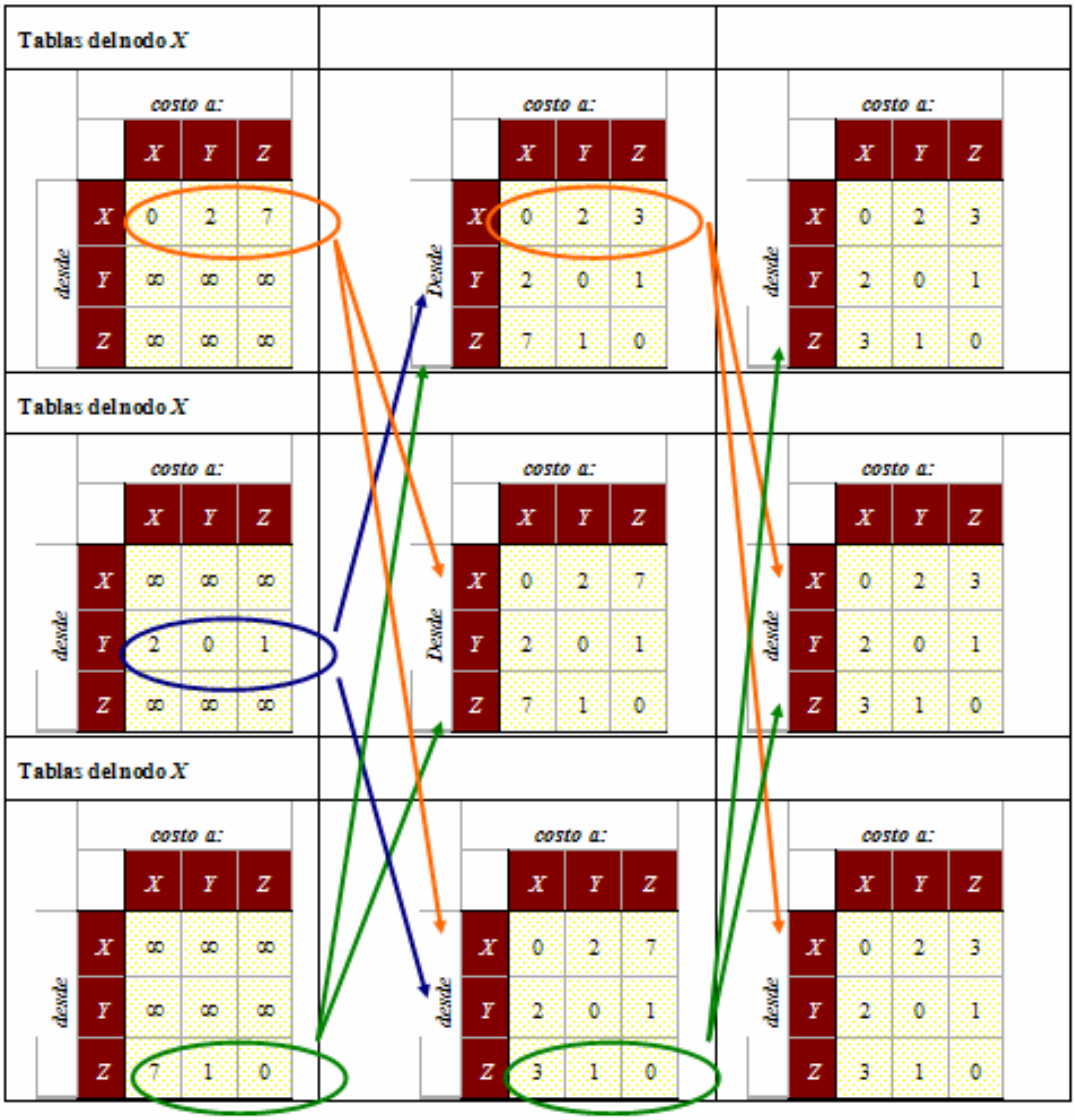
$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\} = \min\{2+1,7+0\} = 3$$

La segunda columna de esta manera ahora muestra, para cada nodo, el nuevo vector distancia del nodo junto con lo vectores distancia recién recibidos de sus vecinos. Note, por ejemplo, que el menor coste estimado del nodo *x* al nodo *z*: $D_x(z)$, cambió de 7 a 3. También note que para ambos nodos *y* y *z*, el nodo *y* adquiere el mínimo. Así que en esta etapa, el *router* de siguiente salto $v^*(y) = y$ y el $v^*(z) = y$.

Después de que los nodos recalculan sus vectores distancia, estos mandan sus vectores distancia actualizados a sus vecinos. Esto es ilustrado en la Tabla 4.1 por las flechas de la segunda columna de las tablas hacia la tercera columna de las tablas. Note que solamente los nodos *x* y *z* mandan sus actualizaciones, pues el vector distancia del

nodo *y* no cambio. Así el nodo *y* no manda su vector distancia pues no recibió cambio alguno. Después de recibir esta actualización, los nodos recalculan sus vectores distancia y actualizan sus tablas de ruteo, las cuales se muestran en la tercera columna.

Tabla 2.7 Tablas obtenidas para el ejemplo de Bellman-Ford [KUR05].



El proceso de recibir los vectores distancia actualizados de los vecinos, recalcular las tablas de ruteo, e informar a los vecinos de los cambios del coste del camino con menor coste a un destino, continúa hasta que no se envíen más actualizaciones. Cuando el algoritmo llega a este punto se dice que converge y se termina. Queda en espera de nuevas actualizaciones como se muestra en la Línea 10-11 del algoritmo.