MODELO OSI - NIVEL DE RED

Edwin Delgado Huaynalaya
Universidad Nacional Jorge Basadre Grohmann
Tacna, Perú
e-mail edychrist@gmail.com

ABSTRACT

This article of investigation is about the Third Level of OSI Model – Network Layer which is oriented to how function exactly this network layer and the implementation in Delphi of Dijkstra algorithm. The paper structure is like that:
Datagrams and Virtual circuits, Routing control algorithms and Congestion Control algorithms.
Keywords: Network layer, Routing, Dijkstra

RESUMEN

El artículo de investigación trata sobre el Tercer Nivel del Modelo OSI – Nivel de Red el cual está orientado a cómo funciona exactamente el nivel de red, y la Implementación en Delphi del algoritmo de Dijkstra. La estructura del artículo es como sigue:
Datagramas y Circuitos Virtuales, Algoritmos de Control de Routing y Algoritmos de Control de Congestión.
Palabras Claves: Nivel de Red, Encaminamiento, Dijkstra

I. INTRODUCCION

Teniendo en cuenta las necesidades y los avances producidos en una sociedad sumamente compleja, resulta de gran importancia destacar tanto la transmisión de información, como la necesidad de que ésta llegue a destino en el momento preciso mediante el uso de las redes.
Es a través de la Internet que queda probado, y todos los días se muestra con mejor y mayor detalle, que ésta ha sido y será revolucionaria en las áreas de los servicios financieros, de entretenimiento, salud, educación y gobierno.
Los principales cambios estructurales de la sociedad se producen ahora entorno del tratamiento y de la transmisión de la información.
La capa de Red, dentro de una arquitectura de red de datos, es la que se encarga de llevar los paquetes de datos desde el origen (estación transmisora) hasta el destino (estación receptora). Llegar a destino, en tiempo y forma, puede requerir que el algoritmo de ruteo, que es el encargado de escoger las rutas y las estructuras de datos, cumpla con ciertas propiedades que aseguren la eficiencia de su trabajo, estas propiedades son: corrección, estabilidad, robustez, equitatividad, sencillez y optimalidad. [1]

II. MARCO TEORICO

2.1 DATAGRAMAS Y CIRCUITOS VIRTUALES

Hay dos formas en las que el nivel de red puede funcionar internamente, mediante datagramas o por circuitos virtuales. En una red de datagramas cada paquete se encamina independientemente, sin que el origen y el destino tengan que pasar por un establecimiento de comunicación previo. [2]
También llamada no orientada a conexión, cada paquete de información se encamina de forma independiente entre conmutadores hasta que se recibe en el destino. [4]

Ventajas

- No existe establecimiento de llamada.
- Mayor flexibilidad en situaciones de congestión local.
- Más tolerante a fallos. Si un nodo falla se pueden usar caminos alternativos.[12]

En una red de circuitos virtuales dos equipos que quieran comunicarse tienen que empezar por establecer una conexión, durante este estableciemiento de conexión, todos los encaminadores (o routers) que haya por el camino elegido reservarán recursos para ese circuito virtual. [2]
Llamada también orientada a conexión, implica el establecimiento de una conexión a través de la red antes de llevar a cabo la transmisión de la información. El procedimiento de establecimiento conlleva generalmente el intercambio de mensajes de señalización y la reserva de recursos a lo largo del camino desde el origen hasta el destino durante la conexión. [4]

Ventajas

- La red puede ofrecer servicios de orden secuencial: paquetes son recibidos en orden al usar el mismo circuito virtual.
- Servicios de control de errores entre nodos que intercambian paquetes.
- Los paquetes viajan más rápidamente al no necesitarse decisiones sobre el routing, [12]

2.2 ALGORITMOS DE CONTROL DE ROUTING

2.2.1 El principio de optimalidad

Este postulado establece que, si el enrutador J está en la trayectoria óptima del enrutador I al enrutador K, entonces la trayectoria óptima de J a K también está en la misma ruta. Haciendo referencia a la Figura 1, llamemos r1 a la parte de la ruta de I a J, y r2 al resto de la ruta. Si existiera una ruta mejor que r2 entre J y K, podría concatenarse con r1 para mejorar la ruta entre I y K, contradiciendo nuestra aseveración de que r1 y r2 es óptima.
Como consecuencia directa del principio de optimalidad, podemos ver que el grupo de trayectorias óptimas de todas las de orígenes a un destino dado forma un árbol con raíz en el destino. Ese árbol que se forma, se llama árbol de descenso, donde la métrica de distancia es el número de escalas. El árbol de descenso puede no ser único, pueden existir otros árboles con las mismas longitudes de trayectoria. La meta de todos los algoritmos de enrutamiento es descubrir y usar los árboles de descenso para todos los enrutadores.

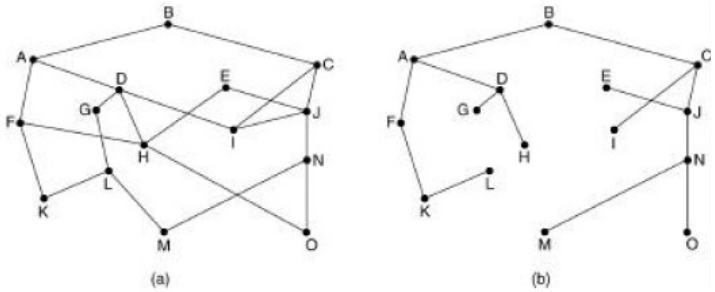


Fig. 1. (a) Subred. (b) Árbol descendente para el ruteador B. [10]

Dado que un árbol de descenso ciertamente es un árbol, no contiene ciclos, por lo que cada paquete será entregado con un número de escalas finito y limitado. En la práctica, no siempre sucede esto, los enlaces y los enrutadores pueden caerse y reactivarse durante la operación, por lo que diferentes enrutadores pueden tener ideas distintas sobre la topología actual de la subred. El fin último de los algoritmos de ruteo es descubrir y usar los árboles de descenso de todos los enrutadores. [1]

2.2.2 Algoritmos Estáticos

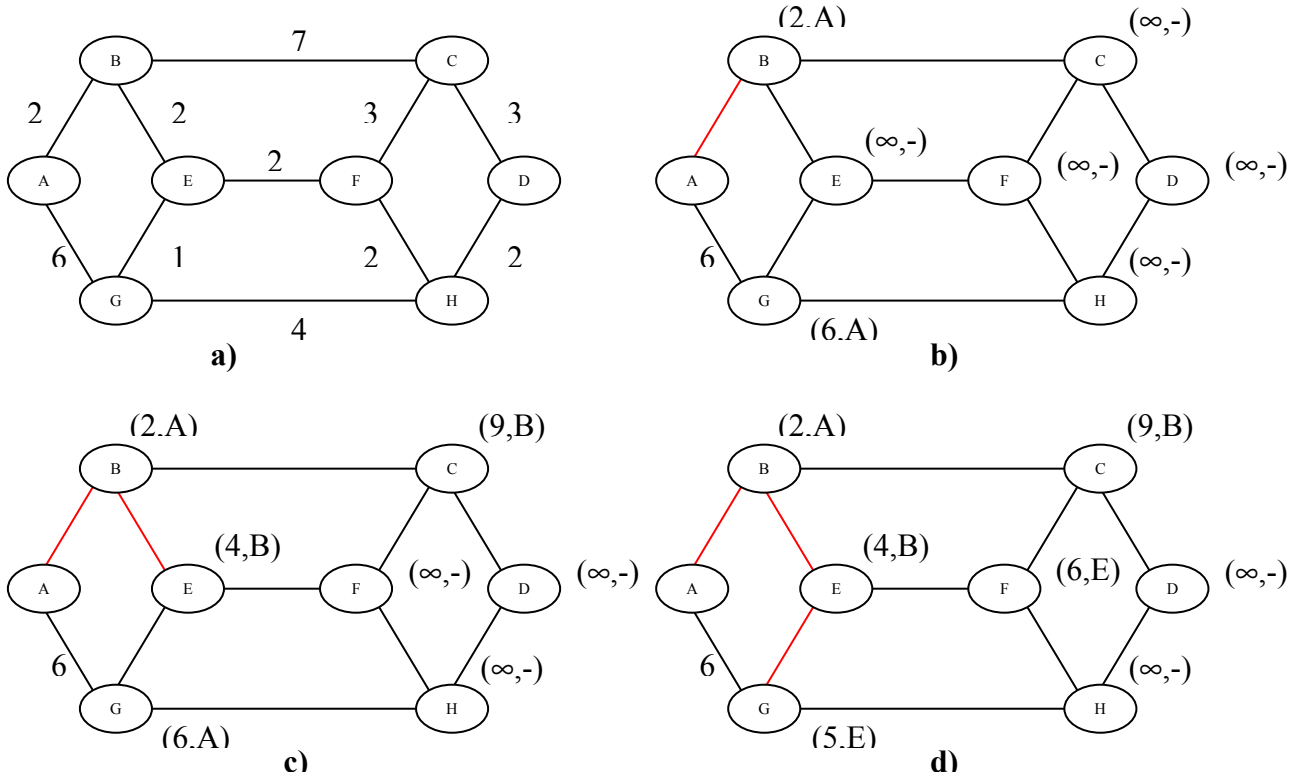
No basan sus decisiones de enrutamiento en mediciones o estimaciones del tráfico ni en la topología. La decisión de qué ruta tomar de I a J se calcula por adelantado, fuera de línea y se cargan en los routers al iniciar la red. Éste procedimiento se llama enrutamiento estáticos. La desventaja de este tipo de algoritmos es que no es posible responder a situaciones cambiantes como por ejemplo saturación, exceso de tráfico o fallo en una línea. [1]

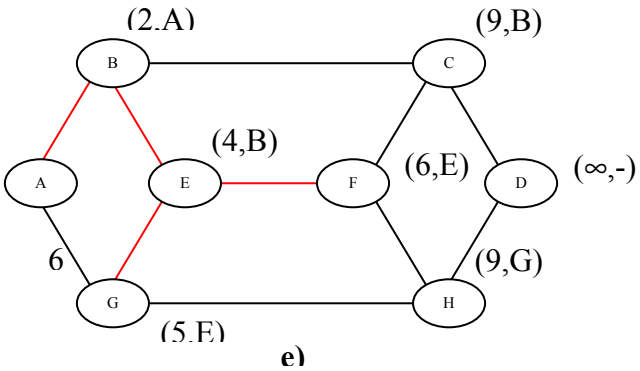
Encaminamiento por la trayectoria más corta

Esta es una técnica de amplio uso en muchas formas, ya que es sencilla y fácil de entender. La idea es armar un grafo de la subred en el que cada nodo representa un enrutador y cada arco del grafo una línea de comunicación (enlace). Para seleccionar la ruta entre un par dado de enrutadores, el algoritmo simplemente encuentra en el grafo la trayectoria más corta entre ellos.

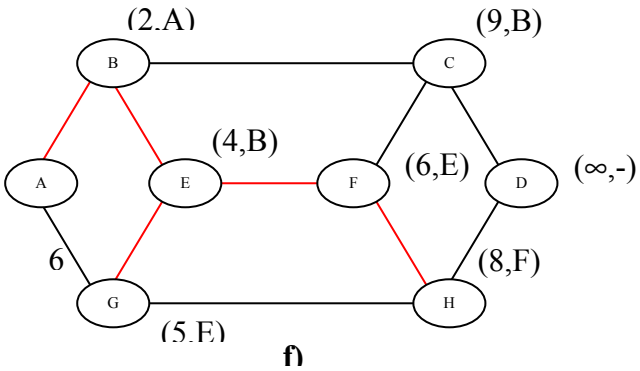
Se conocen varios algoritmos de cálculo de la trayectoria más corta entre dos nodos de un grafo (uno de ellos Dijkstra). Cada nodo se etiqueta (entre paréntesis) con su distancia al nodo de origen a través de la mejor trayectoria conocida. Inicialmente no se conocen trayectorias, por lo que todos los nodos tienen la etiqueta infinito. A medida que avanza el algoritmo y se encuentran trayectorias, pueden cambiar las etiquetas, reflejando mejores trayectorias. Una etiqueta puede ser tentativa o permanente.

Inicialmente todas las etiquetas son tentativas. Al descubrirse que una etiqueta representa la trayectoria más corta posible del origen a ese nodo, se vuelve permanente y no cambia más. [1]

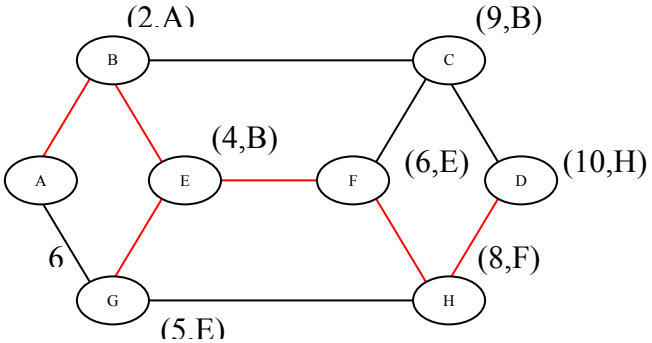




e)



f)



g)

El Algoritmo de Dijkstra:

Edsger Wybe Dijkstra nació en Rotterdam, (Holanda) en 1930. Sus padres eran ambos intelectuales y él recibió una excelente educación. Su padre era químico y su madre matemática. En 1942, cuando Dijkstra tenía 12 años, entró en Gymnasium Erasminium, una escuela para estudiantes especialmente brillantes, donde dio clases, fundamentalmente, de Griego, Latín, Francés, Alemán, Inglés, biología, matemáticas y química. En 1956, Dijkstra anunció su algoritmo de caminos mínimos. [3]

Los cálculos del algoritmo avanzan de un nodo i a un nodo inmediatamente siguiente j, utilizando un procedimiento especial de clasificación. Digamos que u_i es la distancia más corta del nodo 1 del punto de origen al nodo i y defina $d_{ij}(>=0)$ como la longitud del arco (i,j). Entonces la clasificación para el nodo j se define como:
$[u_j , i] = [u_i+d_{ij} , i] , d_{ij} >= 0$ (1)

Las clasificaciones de nodos en el algoritmo de Dijkstra son de dos tipos: temporales y permanentes. Una clasificación temporal puede ser reemplazada con otra clasificación si se puede encontrar una ruta más corta al mismo nodo. En el punto en el cual es evidente que no se puede encontrar una ruta mejor, el estado de la clasificación temporal cambia a permanente.

Los pasos del algoritmo se resumen como sigue:
PASO 0.

- Clasifique el nodo del punto de origen (nodo 1) con la clasificación permanentemente [0,--]. Determine i = 1

PASO i.

- Calcule las clasificaciones temporales $[u_i+d_{ij} , i]$ para cada nodo j al que se puede llegar desde el nodo i, siempre y cuando j no esté clasificado permanentemente. Si el nodo j ya está clasificado con [u , k] a través de otro nodo k y si $u_i+d_{ij} < u_j$, reemplace $[u_j , k]$ con $[u_i+d_{ij} , i]$.
- Si todos los nodos tienen clasificaciones permanentes, deténgase. De lo contrario, seleccione la clasificación $[u_r , s]$ con la distancia más corta ($= u_r$) entre todas las clasificaciones temporales (rompa los empates arbitrariamente). Sea i = r y repita el paso i. [9]

Anexo en Archivo ZIP: Implementación del Algoritmo de Dijkstra en Delphi 7.

Inundación
Otro algoritmo estático es la inundación, en la que cada paquete de entrada se envía por cada una de las líneas de salida, excepto aquella por la que llegó. La inundación evidentemente genera grandes cantidades de paquetes duplicados, de hecho, una cantidad infinita a menos que se tomen algunas medidas para limitar ese proceso. Una de tales medidas puede ser un contador de escalas contenido en la cabecera de cada paquete, el cual disminuye en cada escala, descartándose al llegar el contador a cero.
Idealmente el contador debe inicializarse a la longitud de la trayectoria; puede inicializar el contador en el peor de los casos, es decir, el diámetro de la subred. Una variación de la inundación, un poco más práctica es la inundación selectiva.
En este algoritmo, los enrutadores no envían cada paquete de entrada por todas las líneas, sino sólo por aquellas que van aproximadamente en la dirección correcta. La inundación no es práctica en la mayoría de las aplicaciones, pero tiene algunos usos. Por ejemplo, en aplicaciones militares y en las aplicaciones de bases de datos distribuidas a veces es necesario actualizar concurrentemente todas las bases de datos, en cuyo caso puede ser útil la inundación. [1]

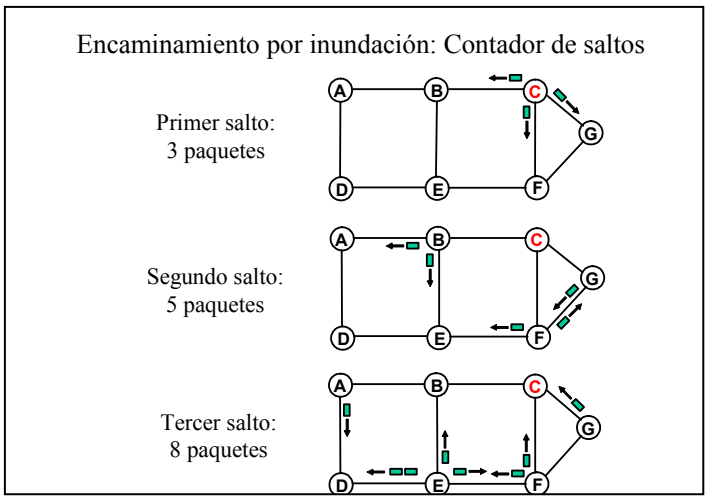


Fig. 2. Inundación. [6]

Enrutamiento basado en el flujo
Los algoritmos vistos hasta ahora sólo toman en cuenta la topología; no consideran la carga. Si por ejemplo, siempre hay una gran cantidad de tráfico entre un nodo A y un nodo B, ambos adyacentes, podría ser mejor enrutar el tráfico de ambos por caminos alternativos un poco más largos tal vez. Seguidamente veremos un algoritmo estático; el enrutamiento basado en flujo usa tanto la topología como la carga para el enrutamiento.
La idea en que se basa el análisis es que, para una línea dada, si se conocen la capacidad y el flujo promedio, es posible calcular el retardo promedio de los paquetes en esa línea a partir de la teoría de colas. De los retardos promedio de todas las líneas, es directo el cálculo de un promedio ponderado por el flujo para obtener el retardo de paquete medio de la subred completa. El problema de enrutamiento se reduce entonces a encontrar el algoritmo de enrutamiento que produzca el retardo promedio mínimo para la subred.
Para usar esta técnica, debe conocerse por adelantado cierta información: primero, la topología de la subred, segundo debe estar dada la matriz de tráfico y tercero debe estar disponible la matriz de capacidad, donde se especifica la capacidad de cada línea en bps. Por último, debe escogerse algún algoritmo tentativo de enrutamiento. [1]

2.2.3 Algoritmos Dinámicos
En contraste con los algoritmos no adaptables, éstos cambian sus decisiones de enrutamiento para reflejar los cambios de topología y de tráfico. Difieren de los algoritmos estáticos en el lugar de obtención de su información (ej. localmente, en los routers adyacentes o de todos), el momento del cambio de sus rutas (ej. cada Dt seg., o cuando cambia la carga) y la métrica usada para la optimalidad (ej. distancia, nº de escalas, tiempo estimado del tránsito). Este tipo de algoritmos no pueden ser demasiado complejos ya que son implementados en los routers y deben ejecutarse en tiempo real con recursos de CPU y la memoria con que el router dispone. [1]

Enrutamiento por vector de distancia
Los algoritmos de enrutamiento por vector de distancia operan haciendo que cada enrutador mantenga una tabla (por ejemplo, un vector) que da la mejor distancia conocida a cada destino y la línea a usar para llegar ahí. Estas tablas se actualizan intercambiando información con vecinos.
Este algoritmo recibe otros nombres como: algoritmo de enrutamiento Bellman-Ford distribuido y el algoritmo Ford-Fullkerson, en reconocimiento a los investigadores que lo desarrollaron.
Se supone que cada enrutador conoce la "distancia" a cada uno de sus vecinos. Si la métrica es de escalas, la distancia simplemente es una escala. Si la métrica es la longitud de la cola, el enrutador simplemente examina cada cola. Si la métrica

es el retardo, el enrutador puede medirlo directamente con paquetes especiales de ECO que el receptor simplemente marca con la hora y envía de regreso tan rápido como puede.
Supóngase que se usa como métrica el retardo y que el enrutador conoce el retardo a cada uno de sus vecinos. Cada T mseg, cada enrutador envía a todos sus vecinos una lista de los retardos estimados a cada uno de los destinos. También recibe una lista parecida de cada vecino. Imagine que una de estas tablas acaba de llegar del vecino X, siendo Xi la estimación de X respecto al tiempo que le toma llegar al enrutador i a través de X en Xi + m mseg vía X . Efectuando este cálculo para cada vecino, un enrutador puede encontrar la estimación que parezca ser la mejor y usar esa estimación y la línea correspondiente en su nueva tabla de enrutamiento.
Este proceso de actualización se ilustra en la figura 3. En la parte (a) se muestra una subred. En las primeras cuatro columnas de la parte (b) aparecen los vectores de retardo recibidos de los vecinos del enrutador J. A indica tener un retardo de 12 mseg a B, un retardo de 25 mseg a C, un retado de 40 mseg a D, etc. Suponiendo que J ha medido o estimado el retardo de sus miembros, A, I, H y K en 8, 10, 12 y 16 mseg, respectivamente.

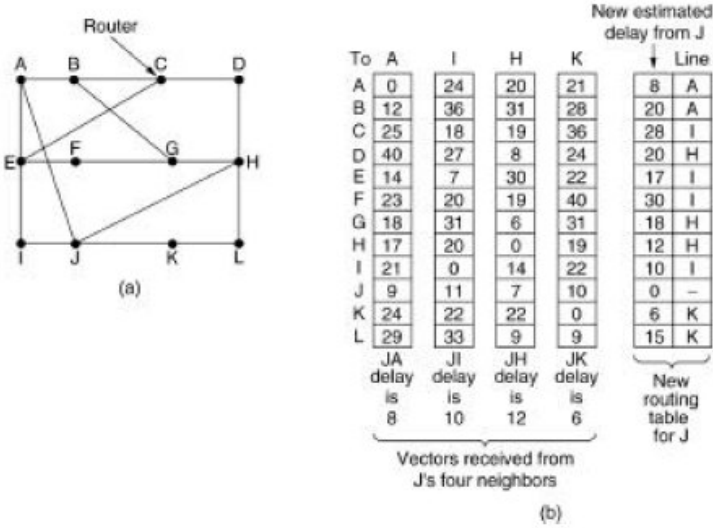


Fig. 3. (a) Subred. (b) Entrada de A, I, H, K, y la nueva tabla de enrutamiento de J. [10]

Considere la manera en que J calcula su nueva ruta al enrutador G. Sabe que puede llegar a A en 8 mseg, y A indica ser capaz de llegar a G en 18 mseg, por lo que J sabe que puede contar con un retador de 26 mseg a G si enviaría a A los paquetes destinados a G. Del mismo modo, J calcula el retardo a G a través de I, H y K en 41 (31+10), 18 (6+12) y 37 (31+6) mseg, respectivamente. El mejor de estos valores es 18, por lo que escribe una entrada en su tabla de enrutamiento indicando que el retardo a G es de 18 mseg, y que la ruta a usar es vía H. Se lleva a cabo el mismo cálculo para los demás destinos, y la nueva tabla de enrutamiento se muestra en la última columna de la figura 3. [1]

Para resumir podemos explicarlo en 3 puntos como sigue:
1. Conocimiento de toda la red. Cada encaminador comparte su conocimiento sobre la red entera. Envía todo el conocimiento que tiene sobre la red a sus vecinos. Al comienzo, el conocimiento que tiene un encaminador de la red puede ser muy escaso. Cuánto conoce, sin embargo, no es importante: el encaminador envía lo que tenga.
2. Encaminamiento sólo a los vecinos. Cada encaminador envía periódicamente su conocimiento sobre la red sólo a los encaminadores con los que tiene enlaces directos. Envía el conocimiento que tenga sobre la red completa a través de todos sus puertos. Esta información es recibida y almacenada en cada encaminador vecino, y utilizada para actualizar la propia información que tiene el encaminador sobre la red.
3. Se comparte información a intervalos regulares. Por ejemplo, cada 30 segundos, cada encaminador envía su información sobre la red completa a sus vecinos. Este envío ocurre haya cambiado o no la red desde el último intercambio de información. [5]

Enrutamiento por estado del enlace
El concepto de este algoritmo es sencillo y puede describirse en cinco partes. Cada enrutador debe:
1. Descubrir a sus vecinos y conocer sus direcciones de red.
Al ponerse en operación un enrutador, su primera tarea es averiguar quiénes son sus vecinos; esto se logra enviando un paquete especial de HOLA (HELLO) por cada línea punto a punto. Se espera que el enrutador del otro extremo envíe de regreso su dirección única.

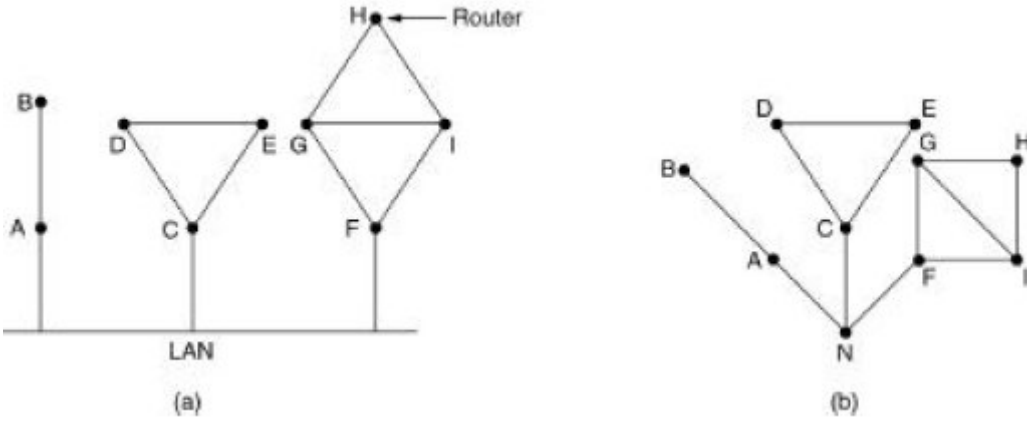


Fig. 4. (a) Nueve enrutadores y una LAN. (b) Modelo de grafo de (a).

Al conectarse dos o más enrutadores mediante una LAN, la situación es ligeramente más complicada. En la figura 4 se ilustra una LAN a la que están conectados directamente tres enrutadores, A, C, y F. Cada uno de estos enrutadores está conectado a uno o más enrutadores adicionales. Una manera de modelar la LAN es considerarla como otro nodo, como se muestra en la figura 4 parte (b). Aquí se ha introducido un nodo artificial nuevo, N, al que están conectados A, C y F. El hecho de que sea posible ir de A a C a través de la LAN se representa aquí mediante la trayectoria ANC.

2. Medición del costo de la línea.

El algoritmo de enrutamiento por estado de enlace requiere que cada enrutador sepa, o cuanto menos tenga una idea razonable del estado de cada uno de sus vecinos.

La manera más directa de determinar este retardo es enviar un paquete especial ECO (ECHO) a través de la línea, el cual debe enviar de regreso inmediatamente el otro lado. Si mide el tiempo de ida y vuelta y lo divide entre dos, el enrutador transmisor puede tener una idea razonable del retardo. Para obtener mejores resultados aún la prueba puede llevarse a cabo varias veces y usarse el promedio.

3. Construcción de los paquetes de estado de enlace.

Una vez que se ha recabado la información necesaria para el intercambio, el siguiente paso es que cada enrutador construya un paquete con todos los datos. Este paquete comienza con la identidad del transmisor, seguida de un número de secuencia, una edad y una lista de vecinos. Para cada vecino, se coloca el retardo a ese vecino. En la figura 5 (a) se muestra un ejemplo de una subred, con los retardos en las líneas. Los paquetes de estado de enlace de los seis enrutadores se muestran en la figura 5 (b).

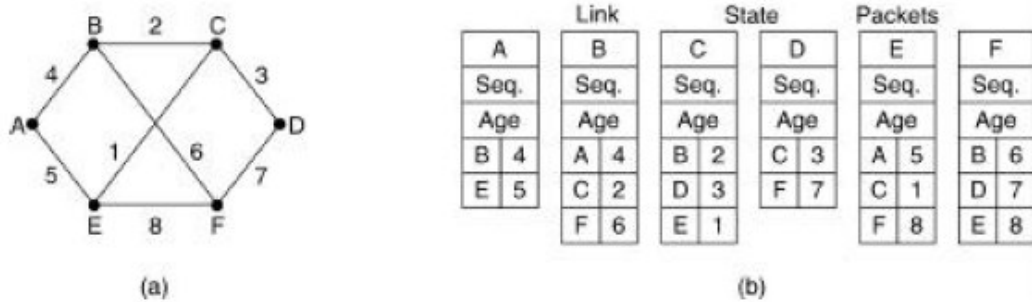


Fig. 5. (a) Subred. (b) Paquetes de estado de enlace para esta subred.

Es fácil construir los paquetes de estado de enlace. La parte difícil es determinar cuándo construirlos. Una posibilidad es construirlos periódicamente, es decir, a intervalos regulares. Otra posibilidad es al ocurrir un evento significativo, como la caída o reactivación de una línea o de un vecino, o el cambio apreciable de sus propiedades.

4. Distribución de los paquetes de estado de enlace.

La parte más complicada del algoritmo es la distribución confiable de los paquetes de estado de enlace. A medida que se distribuyen e instalan los paquetes los enrutadores que reciban los primeros cambiarán sus rutas. En consecuencia, los distintos enrutadores podrían estar usando versiones diferentes de la topología, lo que puede conducir a inconsistencias, ciclos, máquinas inalcanzables, y otros problemas.

El algoritmo que se utiliza para la distribución de los paquetes de estado de enlace sería inundación.

5. Cálculo de nuevas rutas.

Una vez que un enrutador ha acumulado un grupo completo de paquetes, puede construir el grafo de la subred completa porque todos los enlaces están representados. De hecho, cada enlace se representa dos veces, para cada dirección. Los dos valores pueden promediarse o usarse por separado. Ahora puede ejecutarse localmente el algoritmo de la trayectoria más

corta posible a todos los destinos. Los resultados de este algoritmo pueden instalarse en las tablas de enrutamiento, y reiniciarse la operación normal.
Para una subred con n enrutadores, cada uno de los cuales tiene k vecinos, la memoria requerida para almacenar los datos de entrada es proporcional a nk. En las subredes grandes este puede ser un problema. También puede serlo el tiempo de cómputo. Sin embargo en muchas situaciones prácticas, el enrutamiento por estado de enlace funciona bien. Se usa ampliamente en redes actuales, algunos protocolos que lo usan son: el protocolo OSPF, que se emplea cada vez con mayor frecuencia en Internet, el IS-IS (sistema intermedio - sistema intermedio), diseñado por DECnet y el NetWare de Novell usa una variante menor del IS-IS (NLSP) para el enrutamiento de paquetes IPX. [1]

Las claves para comprender el encaminamiento basado en el estado del enlace son diferentes de las del encaminamiento basado en vector distancia. En el encaminamiento basado en el estado del enlace, cada encaminador comparte el conocimiento que tiene de sus vecinos con el resto de encaminadores de la red.

1. Conocimiento sobre sus vecinos. En lugar de enviar su tabla de encaminamiento entera, un encaminador sólo envía información sobre su vecindad.
2. A todos los encaminadores. Cada encaminador envía esta información a todos los encaminadores de la red, no sólo a sus vecinos. Esto se hace mediante un proceso denominado inundación. La inundación significa que un encaminador envía su información a todos sus vecinos (a través de todos sus puertos de salida). Cada vecino envía el paquete a todos sus vecinos y así sucesivamente. Cada encaminador que recibe el paquete envía copias a todos sus vecinos. Finalmente, cada encaminador (sin excepción) recibe una copia de la misma información.
3. Compartir información cuando hay cambios. Cada encaminador envía la información sobre sus vecinos cuando hay algún cambio. [5]

Enrutamiento jerárquico
A medida que crece el tamaño de las redes, crecen proporcionalmente las tablas de enrutamiento del enrutador. Las tablas que siempre crecen no solo consumen memoria del enrutador, sino que también necesitan más tiempo CPU para examinarlas y más ancho de banda para enviar informes de estado entre enrutadores. En cierto momento, la red puede crecer hasta el punto en que ya no es factible que cada enrutador tenga una entrada para cada uno de los demás enrutadores, por lo que el enrutamiento tendrá que hacerse jerárquicamente, como ocurre en la red telefónica.
Al usarse el enrutamiento jerárquico, los enrutadores se dividen en lo que llamamos regiones, en donde cada enrutador conoce todos los detalles de la manera de enrutar paquetes a destinos dentro de su propia región, pero no sabe nada de la estructura interna de las otras regiones. Al interconectar diferentes redes, es natural considerar cada una como región independiente, a fin de liberar a los enrutadores de una red de la necesidad de conocer la estructura topológica de las demás. [1]

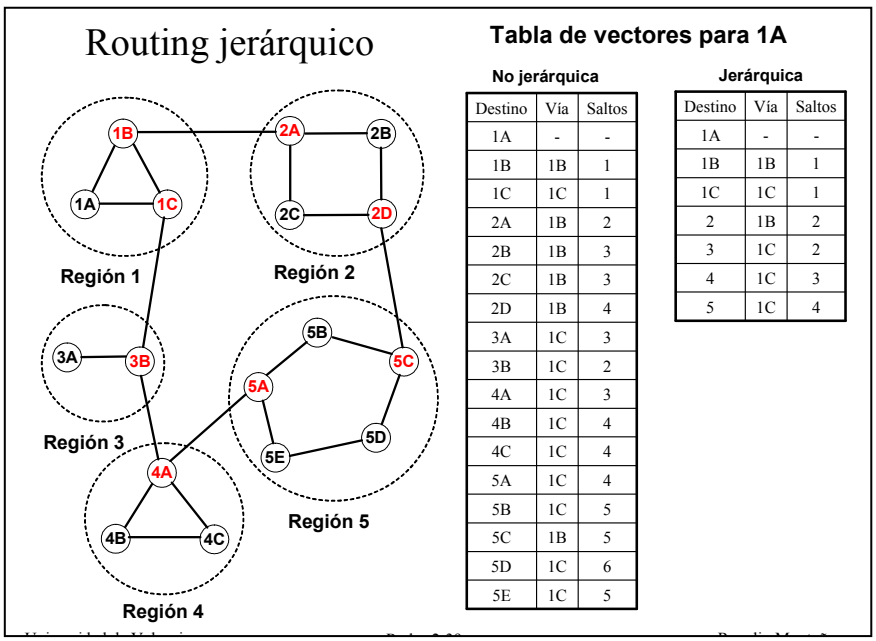


Tabla de vectores para 1A

No jerárquica

Destino	Vía	Saltos
1A	-	-
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Jerárquica

Destino	Vía	Saltos
1A	-	-
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Fig. 6. Enrutamiento Jerárquico. [6]

Enrutamiento por difusión (broadcast)
En algunas aplicaciones, los host necesitan enviar mensajes a varios otros host o a todos los demás. Por ejemplo, el servicio de distribución de informes ambientales, la actualización de los precios de la bolsa o los programas de radio en vivo podrían funcionar mejor difundiéndolos a todas las máquinas y dejando que aquellas interesadas lean los datos. El envío simultáneo de un paquete a todos los destinos se llama difusión.
Hay varios métodos para llevarlo a cabo.
Un método de difusión que no requiere características especiales de la red es que el origen simplemente envíe copias del paquete a todos los destinos. El método no sólo desperdicia ancho de banda, sino que también requiere que el origen tenga una lista completa de todos los destinos. En la práctica, este es el método menos deseable.
La inundación es otro candidato pero el problema de éste como técnica de difusión es el mismo que tiene como algoritmo de enrutamiento punto a punto: genera demasiados paquetes y consume demasiado ancho de banda.
Un tercer algoritmo es el enrutamiento multidestino. Con este método cada paquete contiene una lista de destinos que indican los destinos deseados. El enrutador genera una copia nueva del paquete para que cada línea de salida a usar, e incluye en cada paquete sólo aquellos destinos que usan la línea. En efecto, el grupo de destinos se divide entre las líneas de salida. Este enrutamiento es idéntico al de los paquetes con direccionamiento individual, excepto que, cuando varios paquetes deben seguir la misma ruta, uno de ellos paga la tarifa completa y los demás viajan gratis. El último algoritmo de difusión es un intento de aproximar el comportamiento del algoritmo el árbol de extensión, aún cuando los enrutadores no saben nada en lo absoluto sobre árboles de extensión de los demás enrutadores. La idea es excepcionalmente sencilla una vez planteada. Cuando llega un paquete difundido a un enrutador, éste lo revisa para ver si llegó por la línea normalmente usada para enviar paquetes al origen de la difusión. De ser así, hay excelentes posibilidades de que el paquete difundido haya seguido la mejor ruta desde el enrutador y, por lo tanto, sea la primera copia en llegar al mismo. Siendo este el caso reenvía copias del paquete por todas las líneas, excepto por aquella por la que llegó. Sin embargo, si el paquete difundido llegó por otra línea diferente de la preferida, se descarta el paquete como probable duplicado. [1]

Resumen de Factores a tomar en cuenta:
- **Criterios de Funcionamiento**
  - Numero de Saltos, Coste, retardo, eficiencia
- **Instante de decisión**
  - Paquete (datagrama), Sesión (circuitos virtuales)
- **Lugar de decisión**
  - Cada Nodo, Nodo Central, Nodo Origen
- **Fuente de información de la red**
  - Ninguna, local, nodo adyacente, nodo a lo largo de la ruta, todos los nodos.
- **Tiempo de actualización de la información de la red.**
  - Continuo, periódico, cambio importante en la carga, cambio en la topología. [11]

2.3 ALGORITMOS DE CONTROL DE CONGESTION

Cuando en una red un nodo recibe más tráfico del que puede cursar se puede dar una congestión. El problema es que una vez que se da congestión en un nodo el problema tiende a extenderse por el resto de la red. Por ello hay técnicas de prevención y control que se pueden y deben aplicar en el nivel de red. [2]
Cuando hay demasiados paquetes en alguna parte de la subred, el rendimiento baja. Esta situación se llama congestión [7]

2.3.1 Factores que pueden influir en la creación de situaciones de congestión
- Contienda para las líneas.
- Memoria insuficiente en los ruteadores. Más memoria puede ayudar hasta un punto, pero aún cuando los ruteadores tienen memoria infinita, la congestión empeora, ya que los paquetes expiran antes de llegar a la cabeza de la cola.
- Procesadores lentos. Si las CPUs no son suficiente rápidas, las colas pueden aumentar aun cuando hay capacidad en las líneas. [7]

- El Nivel de llegada de paquetes excede a la capacidad máxima de salida. En este caso la red empezará a borrar paquetes.
- Memoria insuficiente para almacenar la llegada de paquetes. El buffer del nodo receptor no es suficiente para almacenar la llegada de paquetes de datos.
- Cuando el tráfico llega en ráfaga puede también originar una congestión.
- Un Procesador bajo. [8]

El control de la congestión no es el mismo que el control de flujo. En el último el problema es evitar que el emisor mande más datos que el receptor puede procesar, mientras que en el control de congestión el problema es evitar sobrecargar la capacidad de la red.

2.3.2 Perfil de tráfico y vigilancia

Perfil de tráfico o conformado de tráfico (traffic shaping):
Condiciones máximas de uso de la red que el usuario se compromete a cumplir con el proveedor del servicio.

Vigilancia de tráfico (traffic policing):
Labor de monitorización que el proveedor realiza para asegurarse que el usuario cumple su palabra.
Si el usuario incumple el proveedor puede:

- Descartar el tráfico no conforme
- Marcarlo como de 'segunda clase' y pasarlo a la red, o
- Pasarlo a la red sin más (no es habitual) [6]

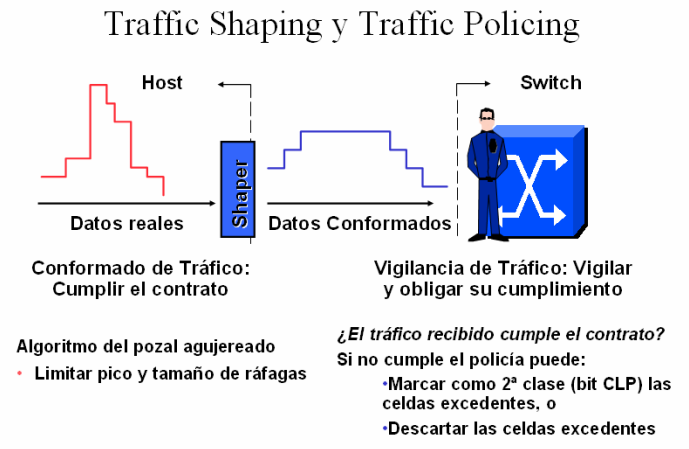


Fig. 7. Perfil de Tráfico y Vigilancia. [6]

2.3.3 Técnicas de Control de Congestión

Muchas técnicas pueden ser empleadas. Estas incluyen:

2.3.3.1 Traffic shaping

Un método de control de congestión es la "forma" del tráfico antes de entrar a la red.
Traffic shaping controla el ratio el cuál los paquetes son enviados. Esta es usada en ATM y Redes de Servicios Integrados.
Hay dos algoritmos de traffic shaping.

Pozal agujereado ('Leaky Bucket')
El pozal agujereado se utiliza para:

- Suavizar las ráfagas que el usuario produce (conformado de tráfico o traffic shaping)
- Asegurar que el tráfico introducido se corresponde con lo acordado (vigilancia de tráfico o traffic policing)

Se define con dos parámetros:

- ρ : caudal constante máximo que se puede inyectar en la red (el agujero del pozal). Se expresa en Mb/s
- C: buffer (la capacidad del pozal) que absorberá las ráfagas que produzca. Se expresa en Mb

Si el buffer se llena el tráfico excedente se considera no conforme. Normalmente se descarta o se pasa como tráfico de 'segunda' clase (candidato a descartar).

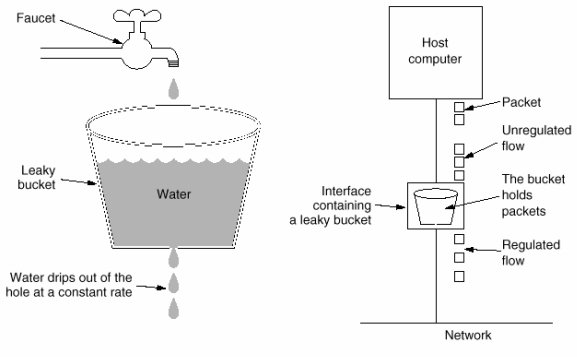


Fig. 8. (a) Un pozal agujerado. (b) Un pozal agujerado con paquetes [6]

Imagine un conmutador con una única cola de entrada y otra de salida. Si la interfaz de salida tiene una velocidad de 1.544Mbps y la entrada puede recibir ráfagas de 40Mbps durante 100 milisegundos (y no recibe nada hasta el siguiente segundo), ¿cuál debería ser el tamaño de la cola?

40Mbps x 100ms = 4 Megabits

La interfaz de salida debería tener una cola (buffer) de 4 Megabits o, lo que es lo mismo, ½ millon de bytes.

¿Cómo se puede controlar la velocidad de salida de modo que sea inferior a la velocidad fija (por ejemplo de 1.544 Mbps) en una red de conmutación de paquetes donde el tamaño de cada paquete puede ser diferente?
Se puede utilizar un contador y un reloj. En cada pulso de reloj (el comienzo de un segundo, por ejemplo), el contador se fija a la cantidad de datos que pueden ser sacados en cada pulso de reloj (normalmente en bytes). El algoritmo comprueba el tamaño de la trama situada en el frente de la cola. Si el tamaño es menor o igual que el valor de contador, se envía el paquete; si el tamaño es mayor que el valor del contador, el paquete se deja en la cola y espera al próximo pulso de reloj. La figura 9 muestra el diagrama de flujo del algoritmo del pozal agujerado.

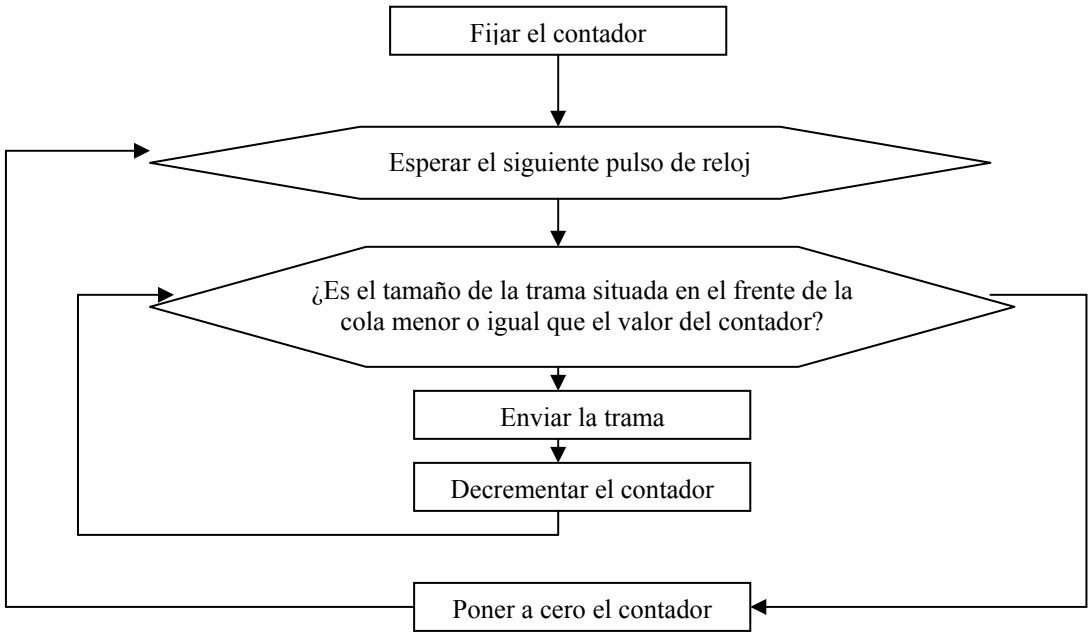
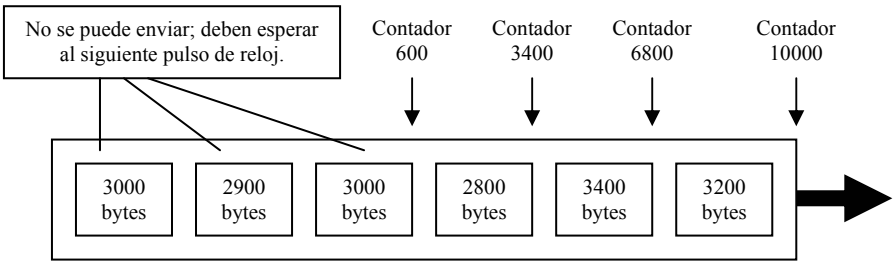
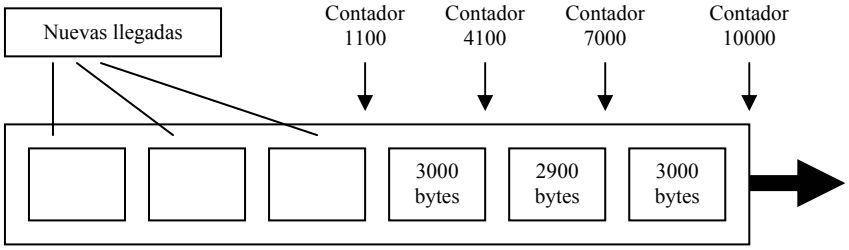


Fig. 9. Diagrama de Flujo de algoritmo Pozal Agujerado [5]



a. Primer Pulso



b. Segundo Pulso

Fig. 10. Ejemplo del algoritmo del pozal agujerado [5]

Observe que para que éste algoritmo funcione, el tamaño de la trama debería ser más pequeño que el valor del contador máximo. La figura 10 muestra un ejemplo. Considere que la velocidad de salida es de 80Kbps. Esto significa 80000 bits por segundo o 10000 bytes por segundo. El contador se fija inicialmente a 10000; después de enviar tres tramas, el valor del contador es 600, que es menor que el tamaño de las siguientes tramas. Las tres siguientes tramas no pueden ser enviadas. Tienen que esperar el siguiente pulso de reloj. [5]

Pozal Token ('Token Bucket')
En contraste al Pozal Agujerado, este algoritmo permite la salida de un ratio variable, dependiendo del tamaño de la ráfaga. Para transmitir un paquete, el host debe capturar y destruir un token.
Los Tokens son generados por un reloj en el ratio de un token cada dT seg.
Los hosts pueden capturar y guardar tokens (arriba del máximo tamaño del pozal) en orden para después enviar ráfagas más largas.

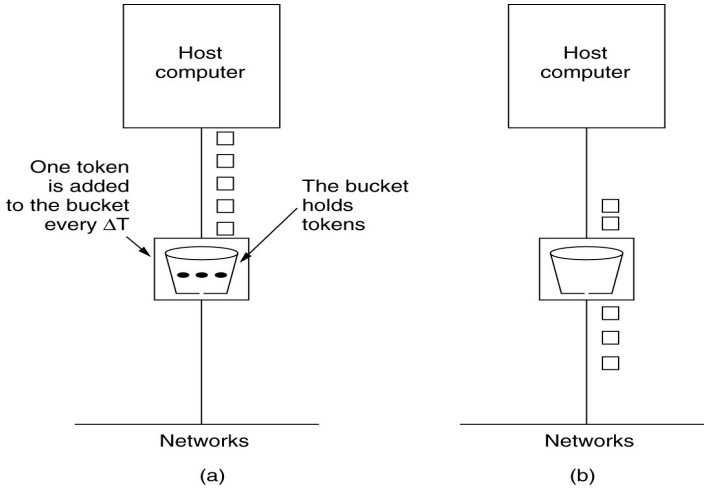


Fig. 11. (a) Un pozal token con paquetes de entrada. (b) Un pozal token con paquetes de salida [8]

TABLA I
Comparación entre Pozal Agujerado y Pozal Token [8]

Pozal Agujerado	Pozal Token
Descarta paquetes.	Descarta tokens.
Un paquete puede ser transmitido si el pozal no esta lleno.	Un paquete solo puede ser transmitido si hay suficientes tokens para cubrir su tamaño en bytes.
Envía los paquetes a un ratio promedio.	Para largas ráfagas permite enviarlas más rápido por el exceso de velocidad de salida.
No permite guardar; un ratio constante es mantenido.	Permite guardar tokens (permisos) para enviar largas ráfagas.

2.3.3.2 Warning bit
Un bit especial en la cabeza del paquete es establecido por el router para advertir a la fuente cuando una congestión es detectada.
El bit es copiado y a cuestas de la respuesta (ACK) es enviado al emisor.
El emisor monitorea el número de paquetes ACK, este recibe el bit de aviso establecido y ajusta la transmisión acorde al ratio. [8]

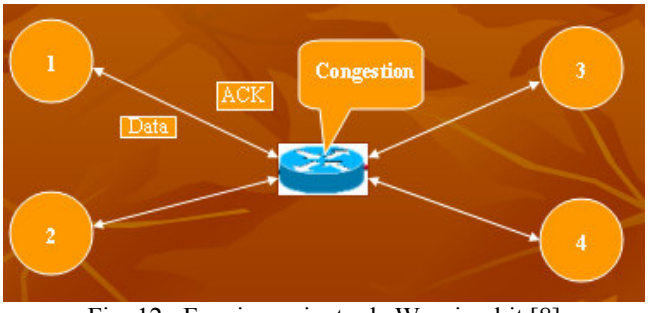


Fig. 12 Funcionamiento de Warning bit [8]

2.3.3.3 Choke packets
Una forma más directa de decir a la fuente para bajar.
Un choke packet es un paquete de control generado en un nodo congestionado y transmitido al flujo del tráfico restringido
La fuente, al recibir el choke packet debe reducir su ratio de transmisión por un cierto porcentaje. [8]

2.3.3.4 Load shedding
Cuando el buffer llega a estar lleno, los routers simplemente descartan los paquetes.
Para una transferencia de archivo, por ejemplo, nosotros no podemos descartar desde los paquetes más viejos, esto causará una brecha en recibir los datos
Para voz y video en tiempo real es probablemente mejor arrojar datos viejos y guardar los nuevos paquetes.[8]

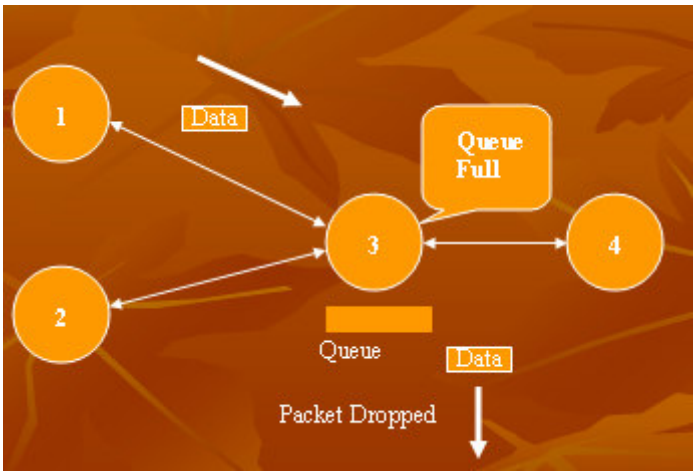


Fig. 13. Funcionamiento de Load shedding [8]

2.3.3.5 Random early discard
Es adecuado enfocar a cuál el router descartará uno o más paquetes antes de que el buffer llegue ha estar completamente lleno. Cada tiempo un paquete llega, el algoritmo de red computa el promedio del tamaño de cola, avg.

Si avg es más pequeña que algunos pequeños umbrales, la congestión es asumida como mínima o no existente y el paquete es colocado en cola.
Si avg es más grande que algunos grandes umbrales, la congestión es asumida como seria y el paquete es descartado.
Si avg esta entre dos umbrales, esta debería indicar el principio de una congestión. La probabilidad de congestión es luego calculada.
De esta forma, el buffer es prevenido de que llegue a estar lleno descartando paquetes y la congestión es prevenida. [8]

III. CONCLUSIONES

Para comprender el funcionamiento exacto de la capa de red, es necesario analizar y comprender los diferentes algoritmos tanto estáticos como dinámicos para poder tomar decisiones según los casos que se presenten. Esto implica muchos factores como estabilidad, robustez, topología, disponibilidad, tiempo, tamaño del paquete, etc. a tener en cuenta para elegir el algoritmo que se adapte mejor a los requerimientos.

Después de estudiar los distintos algoritmos de encaminamiento se puede ver que no existe aquél que ante cualquier circunstancia o evento sea el que mejor resuelva el problema de enrutamiento. Sólo dependerá de qué recurso o criterio se elija como prioritario para el envío de paquetes de datos.

Los protocolos y algoritmos del nivel de red son un tema de gran interés debido a que se encuentran en permanente cambio debido a la continua evolución de las redes de comunicaciones de datos, cada vez con más requerimientos y prestaciones que involucran el desempeño de la red.

IV. REFERENCIAS

[1] María Julieta Goitia. (2005, Octubre). "Protocolos de Enrutamiento Para la Capa de Red en Arquitecturas de Redes de Datos". Dpto. Informática. Universidad Nacional del Nordeste. Corrientes. Argentina. Disponible: http://exa.unne.edu.ar/depar/areas/informatica/SistemasOperativos/ProtocolosRed.PDF

[2] WikiPedia - La enciclopedia Libre. (2005, Octubre). Disponible: http://es.wikipedia.org/wiki/Nivel_de_red

[3] Publicación de alumnos UNICAN (2005, Octubre). Disponible: http://www.alumnos.unican.es/uc900/Algoritmo.htm

[4] Alberto León García, Indra Widjaja. Redes de Comunicación – Conceptos Fundamentales y Arquitectura Básica. Universidad de Toronto. Primera Edición. McGRAW-HILL / INTERAMERICANA DE ESPAÑA, S.A.U. © 2002 pp. 413

[5] Behrouz A. Forouzan, Transmisión de Datos y Redes de Comunicaciones. Segunda Edición. McGRAW-HILL / INTERAMERICANA DE ESPAÑA, S.A.U. © 2002 pp 521-522, 609, 614,

[6] Rogelio Montañana. (2005, Octubre). "El Nivel de Red: Generalidades ". Departamento de Informática. Universidad de Valencia. Disponible: http://www.uv.es/~montanan/

[7] Fritz Knabe (2005, Octubre). Computer Science at the University of Virginia. Disponible: http://www.cs.virginia.edu/~knabe/iic3512/apuntes_7.html

[8] Ashish Babbar. (2005, Octubre). Disponible: http://www2.hawaii.edu/~babbar/Congestion%20Control.ppt

[9] Hamdy A. Taha. Investigación de Operaciones – 6ª Edición. Prentice Hall. (1998). pp 227-228

[10] Tanenbaum, A. S. Redes de Computadoras – 3ª Edición. Prentice Hall. (1997). pp 337-359

[11] Stallings, W. Comunicaciones y Redes de Computadores – 6ª Edición. Prentice Hall. (1997). pp 297

[12] Dpto de Arquitectura de Computadoras. Tema 4: Redes de Comunicación (2005, Octubre). Disponible Redes.PDF