Transparencias

de

Redes de Ordenadores

Tema 11

IP Routing
1ª Parte – Routing

Uploaded by

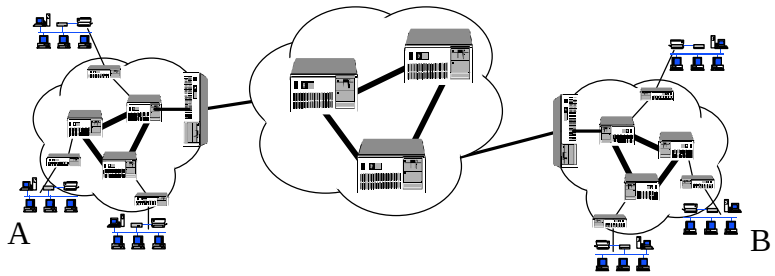
IngTeleco

http://ingteleco.iespana.es
ingtelecoweb@hotmail.com

Routing

- Objetivos y elecciones de diseño
- Routing Centralizado / Distribuido
- Protocolos Vector Distancia
- Protocolos de Estado de enlace
- Routing jerárquico
- Routing Broadcast

Objetivo del Routing



- Encontrar un camino desde el emisor hasta el receptor a través de la red
  - Se aplica a los paquetes de datos o a los mensajes de señalización
- Función de Routing:
  - Determina el siguiente salto en cada router.
  - Sub-funciones básicas de routing
    - Asignación de una métrica al enlace.
    - Distribución del estado del enlace utilizando un protocolo de routing.
    - Calculo de rutas basándose en información.

Objetivos de Diseño y Requisitos

- Obtención de rutas correctas
  - Los paquetes se envían al destino correcto
- Simplicidad
- Escalabilidad
  - Minimizar espacio en las tablas de encaminamiento
    - Acelerar las búsquedas
    - Disminuir el coste de almacenamiento
  - Reducir el número y frecuencia de los mensajes de control (menor procesamiento y overhead, ahorrando ancho de banda)
- Utilización óptima de la red
  - Uso del camin posible y/o más eficiente (mínimo coste)
  - Aprovechamiento de los enlaces
- Robustez
  - Respuesta a los cambios de topología
  - Estabilidad y convergencia al equilibrio (evitar oscilaciones)
  - Prevenir o recuperarse de bucles (decisiones inconsistentes)
  - Evitar "agujeros negros" (incapaz de alcanzar el destino)
- Proporcionar la QoS requerida por la aplicación

Elecciones generales de diseño

- ¿Dónde se calculan las rutas?
  - Centralizado (RCC o NCC) vs. Distribuido
    - El Centralizado es más simple pero no es escalable y es susceptible a fallos.
    - El Distribuido precisa la colaboración entre routers y tiene un comportamiento transitorio más complejo
- ¿Cómo se calculan las rutas?
  - Distribución de Cálculos vs. Distribución de Información a partir de las que se calculan las rutas
    - Vector Distancia: Los routers intercambian *resultados* de los cálculos (centralizado o distribuído)
    - Estado de Enlace: Los routers intercambian *información* sobre las que pueden realizarse los cálculos independientemente (distribuido)
- ¿Qué criterio se utiliza cuando se calculan las rutas?
  - Estático vs. Dinámico
  - Mediciones o estimaciones del tráfico
  - Topología de la red (cambio en los nodos y/o enlaces)

Routing Centralizado/Distribuido

- CENTRALIZADO:
  - El camino completo se calcula centralizado (RCC)
    - El estado se descarga en conmutadores
  - El protocolo de Routing distribuye
    - El estado del enlace de red al router central
  - Limitaciones
    - Ineficiente con cambios de topología frencuentes
    - Escalabilidad
- DISTRIBUIDO:
  - ¿Cómo adoptar las decisiones locales correctas?
    - Cada router debe tener información sobre el estado global.
  - Estado global
    - Inherentemente grande: Difícil de recopilar la información
    - Dinámico
  - Un protocolo de routing debe adquirir, resumir y mantenet información relevante de manera inteligente.
    - ¿Cómo descubrir otros routers y enlaces?
    - ¿Cómo utilizar la infomación para generar rutas?
    - ¿Cómo mantener rutas en presencia de cambios?

Elecciones para la Arquitectura de Routing

- ¿Dónde se toma la decisión de encaminamiento?
  - En cada salto ("hop-by-hop" routing)
  - Por el emisor ("source routing")
- ¿Con que frecuencia se toman decisiones de routing?
  - Por paquete
  - Por sesión
  - Con cada cambio de topología

Source Routing

- El emisor pre-calcula el camino
  - Utiliza información de los servidores de rutas
- Incluye el camino en los paquetes
- El protocolo de Routing distribuye información de enlace a los servidores de rutas
- Pros y Contras
  - Control de origen
  - Se evitan buces inconsistentes
  - Overhead de la cabecera o retrasaso de actualización
  - ¿Menos adaptativo?

Hop-by-Hop Routing

- En cada salto se toman una decisión de encaminamiento independiente
  - Basado en el destino
- El protocolo de Routing distribuye información de enlace
  - Rutas sin bucles cuando cada router es consistente
- Características
  - Se toman decisiones descentralizadas
  - Mejor respuesta a los cambios de topología
  - Escalable
  - Menos control de origen

Orientación del diseño

- El diseño de una sola solución que se pueda extender a toda Internet no es probable que funcione
  - Coste computacional, overhead del protocol y ancho de banda
- Entorno heterogéneo
  - Tamaños de Dominio y requisitos, capacidades de los routers
  - Diferentes restricciones en contectividad interna y externa

⇒ Enfoque básico de diseño

  - Jerarquía de Dominios (refleja la jerarquía de las direcciones)
    - Asegura la escalabilidad
  - Independencia de protocolos de routing en diferentes Dominios
    - Soporte de heterogeneidad
  - *Gateways* entre dominios para una solución end-to-end

Protocolos Vector Distancia

- Empleados en Arpanet
- Cálculo del siguiente salto distribuido
  - Adaptativo
  - Cada router mantiene un conjunto de valores:
    - **(Destino, Costo, Siguiente Salto)**
  - Cada router envía actualizaciones sólo a los routers vecinos a los que está directamente conectado
- Envío de mensajes asíncrono:
  - Periódicamente
  - Cuando se produce un cambio en la tabla de encaminamiento
    - Cambio del costo de un enlace local
    - Recepción de la notificación de un vecino
  - Unidad de intercambio de información
    - Lista de pares **(Destino, Costo)**
- Algoritmo distribuido Bellman-Ford

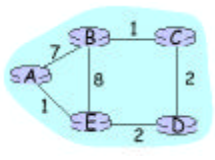
Protocolos Vector Distancia (II)

- Objetivo principal: simplicidad y reducir el overhead
  - Sólo preciso almacenar y mantener la propia tabla de routing
  - Procedimiento iterativo simple para construir y actualizar la tabla de routing basándose sólo en el conocimiento del coste del enlace con los *vecinos*
    - Capaz de calcular rutas de mínimo coste
    - Basado en la ejecución del algoritmo distribuido Bellman-Ford)
- Algunas cuestiones
  - La convergencia requiere estabilidad en el estado de la red
    - Capacidad de reaccionar ante los cambios
    - Inconsistencias en el estado de las tablas de los routers

Protocolos Vector Distancia (III)

- Estructura de la Tabla de Datos:
  - Una fila para cada destino posible
  - Una columna para cada vecino directamente conectado al nodo
- Ej.: Un nodo X para el destino a través del vecino Z
  - $D^X(Y,Z)$= distancia de X a Y a través de Z como siguiente salto
  - $D^X(Y,Z) = c(X,Z) + \min_w \{D^Z(y,w)\}$
- La convergencia depende de suposiciones mínimas:
  - Maneja actualizaciones asíncronas
  - La red y los routers no puede retrasar las actualizaciones re cálculos indefinidamente
  - Se utilizan métricas no negativas
  - No depende del punto de partida

Ejemplo de Tablas de Distancias



$D^E(C,D) = c(E,D) + \min_w\{D^D(C,w)\}$
$= 2+2 = 4$

$D^E(A,D) = c(E,D) + \min_w\{D^D(A,w)\}$
$= 2+3 = 5$ loop!

$D^E(A,B) = c(E,B) + \min_w\{D^B(A,w)\}$
$= 8+6 = 14$ loop!

$D^E()$ | cost to destination via A | B | D

destination	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

Paso de Tabla de Distancias a Tabla de encaminamiento

$D^E()$ | cost to destination via A | B | D

destination	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destination	Outgoing link to use, cost
A	A,1
B	D,5
C	D,4
D	D,4

Distance table → Routing table (or forwarding table)

Algoritmo Bellman-Ford

Condiciones de Inicio:
Cada router arranca con un vector de distancias (o) a todas las redes a las que está directamente conectado

Paso de espera
El router espera un cambio en el coste de un enlace local o una notificación de un vecino.

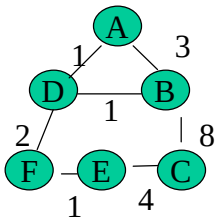
Paso de cálculo de la tabla de distancias:
Al recibir vectores de cada vecino el router calcula su propia distancia a cada vecino. A continuación, para cada red X el router encuentra el vecino que está más próximo a X. El Router actualiza su costo a X.

Paso de envío:
Si ha cambiado el camino de costo mínimo a algún destino el router notifica su vector actual a todos los vecinos.

Algoritmo Bellman-Ford

Inicialización
para todos los nodos adyacentes v:
$D^X(*,V) = \infty$ /* el operador * significa para todas las filas */
$D^X(V,V) = c(X,V)$
para todos los destinos, y
enviar $\min_w D^X(Y,W)$ a cada vecino /* W de todos los vecinos X*/
bucle
esperar (hasta encontrar un cambio en el coste del enlace al vecino V o hasta recibir una notificación del vecino V)
si (D(X,V) cambia en d)
/*cambio de coto a todos los destinos a través de V en d */
para todos los destinos y: $D^X(Y,V) = D^X(Y,V) + d$
en caso contrario si (actualización recibida de V con destino Y)
/*el camino más corto de V a algún Y ha cambiado */
/*V ha enviado un nuevo valor para su nueva minwDV(Y,w) */
/*llamar a este nuevo valor recibido "nuevoval" */
para el destino Y: $D^X(Y,V) = c(X,V) + nuevoval$
si tenemos un nuevo $\min_w D^X(Y,W)$ a cualquier destino Y
actualizar la tabla de encaminamiento y enviar el nuevo valor de $\min_w D^X(Y,W)$ a todos los vecinos
fin de bucle

Algoritmo Vector Distancia: ejemplo



D^A	B	C	D	E	F
B	3	∞	∞	∞	∞
C	∞	∞	∞	∞	∞
D	∞	∞	1	∞	∞
E	∞	∞	∞	∞	∞
F	∞	∞	∞	∞	∞

D^B	A	C	D	E	F
A	3	∞	∞	∞	∞
C	∞	8	∞	∞	∞
D	∞	∞	1	∞	∞
E	∞	∞	∞	∞	∞
F	∞	∞	∞	∞	∞

D^C	A	B	D	E	F
A	∞	∞	∞	∞	∞
B	∞	8	∞	∞	∞
D	∞	∞	∞	∞	∞
E	∞	∞	∞	4	∞
F	∞	∞	∞	∞	∞

D^D	A	B	C	E	F
A	1	∞	∞	∞	∞
B	∞	1	∞	∞	∞
C	∞	∞	∞	∞	∞
E	∞	∞	∞	∞	2
F	∞	∞	∞	∞	2

D^E	A	B	C	D	F
A	∞	∞	∞	∞	∞
B	∞	∞	∞	∞	∞
C	∞	∞	4	∞	∞
D	∞	∞	∞	∞	1
F	∞	∞	∞	∞	1

D^F	A	B	C	D	E
A	∞	∞	∞	∞	∞
B	∞	∞	∞	∞	∞
C	∞	∞	∞	∞	∞
D	∞	∞	∞	2	∞
E	∞	∞	∞	∞	1

D^A	B	C	D	E	F
B	3	∞	2	∞	∞
C	11	∞	∞	∞	∞
D	4	∞	1	∞	∞
E	∞	∞	∞	∞	∞
F	∞	∞	3	∞	∞

D^B	A	C	D	E	F
A	∞	∞	2	∞	∞
C	∞	8	∞	∞	∞
D	4	∞	1	∞	∞
E	∞	12	∞	∞	∞
F	∞	∞	3	∞	∞

D^C	A	B	D	E	F
A	∞	11	∞	∞	∞
B	∞	8	∞	∞	∞
D	∞	9	∞	∞	∞
E	∞	∞	∞	4	∞
F	∞	∞	∞	5	∞

D^D	A	B	C	E	F
A	1	4	∞	∞	∞
B	4	1	∞	∞	∞
C	∞	9	∞	∞	∞
E	∞	∞	∞	∞	3
F	∞	∞	∞	∞	2

D^E	A	B	C	D	F
A	∞	∞	∞	∞	∞
B	∞	∞	12	∞	∞
C	∞	∞	4	∞	∞
D	∞	∞	∞	∞	3
F	∞	∞	∞	∞	1

D^F	A	B	C	D	E
A	∞	∞	∞	3	∞
B	∞	∞	∞	3	∞
C	∞	∞	∞	∞	5
D	∞	∞	∞	2	∞
E	∞	∞	∞	∞	1

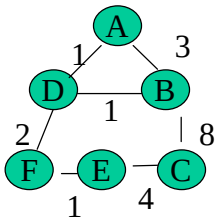
BD en

Destino	A	B	C	D	E	F
A	-	3,A	-	1,A	-	-
B	3,B	-	8,B	1,B	-	-
C	-	8,C	-	-	4,C	-
D	1,D	1,D	-	-	-	2,D
E	-	-	4,E	-	-	1,E
F	-	-	-	2,F	1,F	-

BD en

Destino	A	B	C	D	E	F
A	-	2,D	11,B	1,A	4,F	3,D
B	3,B	-	8,B	1,B	12,C	3,D
C	11,B	8,C	-	9,B	4,C	5,E
D	4,B	1,D	9,B	-	3,F	2,D
E	-	12,C	4,E	3,F	-	1,E
F	-	3,D	5,E	2,F	1,F	-

Algoritmo Vector Distancia: ejemplo(II)



D^A	B	C	D	E	F
B	3	∞	2	∞	∞
C	11	∞	∞	∞	∞
D	4	∞	1	∞	∞
E	∞	∞	∞	∞	∞
F	∞	∞	3	∞	∞

D^B	A	C	D	E	F
A	∞	∞	2	∞	∞
C	∞	8	∞	∞	∞
D	4	∞	1	∞	∞
E	∞	12	∞	∞	∞
F	∞	∞	3	∞	∞

D^C	A	B	D	E	F
A	∞	11	∞	∞	∞
B	∞	8	∞	∞	∞
D	∞	9	∞	∞	∞
E	∞	∞	∞	4	∞
F	∞	∞	∞	5	∞

D^D	A	B	C	E	F
A	1	4	∞	∞	∞
B	4	1	∞	∞	∞
C	∞	9	∞	∞	∞
E	∞	∞	∞	∞	3
F	∞	∞	∞	∞	2

D^E	A	B	C	D	F
A	∞	∞	∞	∞	∞
B	∞	∞	12	∞	∞
C	∞	∞	4	∞	∞
D	∞	∞	∞	∞	3
F	∞	∞	∞	∞	1

D^F	A	B	C	D	E
A	∞	∞	∞	3	∞
B	∞	∞	∞	3	∞
C	∞	∞	∞	∞	5
D	∞	∞	∞	2	∞
E	∞	∞	∞	∞	1

D^A	B	C	D	E	F
B	3	∞	2	∞	∞
C	11	∞	10	∞	∞
D	4	∞	1	∞	∞
E	15	∞	4	∞	∞
F	6	∞	3	∞	∞

D^B	A	C	D	E	F
A	3	19	2	∞	∞
C	14	8	10	∞	∞
D	4	17	1	∞	∞
E	∞	12	4	∞	∞
F	6	13	3	∞	∞

D^C	A	B	D	E	F
A	∞	10	∞	∞	∞
B	∞	8	∞	16	∞
D	∞	9	∞	7	∞
E	∞	20	∞	4	∞
F	∞	∞	∞	5	∞

D^D	A	B	C	E	F
A	1	3	∞	∞	5
B	3	1	∞	∞	5
C	14	9	∞	∞	7
E	∞	13	∞	∞	3
F	6	4	∞	∞	2

D^E	A	B	C	D	F
A	∞	∞	15	∞	4
B	∞	∞	12	∞	4
C	∞	∞	4	∞	6
D	∞	∞	13	∞	3
F	∞	∞	9	∞	1

D^F	A	B	C	D	E
A	∞	∞	∞	3	∞
B	∞	∞	∞	3	13
C	∞	∞	∞	11	5
D	∞	∞	∞	2	5
E	∞	∞	∞	5	1

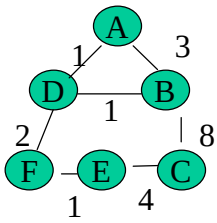
BD en

Destino	A	B	C	D	E	F
A	-	2,D	11,B	1,A	4,F	3,D
B	3,B	-	8,B	1,B	12,C	3,D
C	11,B	8,C	-	9,B	4,C	5,E
D	4,B	1,D	9,B	-	3,F	2,D
E	-	12,C	4,E	3,F	-	1,E
F	-	3,D	5,E	2,F	1,F	-

BD en

Destino	A	B	C	D	E	F
A	-	2,D	10,B	1,A	4,F	3,D
B	2,D	-	8,B	1,B	4,F	3,D
C	10,D	8,C	-	7,F	4,C	5,E
D	1,D	1,D	7,E	-	3,F	2,D
E	4,D	4,D	4,E	3,F	-	1,E
F	3,D	3,D	5,E	2,F	1,F	-

Algoritmo Vector Distancia: ejemplo (III)



D^A	B	C	D	E	F
B	3	∞	2	∞	∞
C	11	∞	10	∞	∞
D	4	∞	1	∞	∞
E	15	∞	4	∞	∞
F	6	∞	3	∞	∞

D^B	A	C	D	E	F
A	3	19	2	∞	∞
C	14	8	10	∞	∞
D	4	17	1	∞	∞
E	∞	12	4	∞	∞
F	6	13	3	∞	∞

D^C	A	B	D	E	F
A	∞	10	∞	∞	∞
B	∞	8	∞	16	∞
D	∞	9	∞	7	∞
E	∞	20	∞	4	∞
F	∞	∞	∞	5	∞

D^D	A	B	C	E	F
A	1	3	∞	∞	5
B	3	1	∞	∞	5
C	14	9	∞	∞	7
E	∞	13	∞	∞	3
F	6	4	∞	∞	2

D^E	A	B	C	D	F
A	∞	∞	15	∞	4
B	∞	∞	12	∞	4
C	∞	∞	4	∞	6
D	∞	∞	13	∞	3
F	∞	∞	9	∞	1

D^F	A	B	C	D	E
A	∞	∞	∞	3	∞
B	∞	∞	∞	3	13
C	∞	∞	∞	11	5
D	∞	∞	∞	2	5
E	∞	∞	∞	5	1

D^A	B	C	D	E	F
B	3	∞	2	∞	∞
C	11	∞	10	∞	∞
D	4	∞	1	∞	∞
E	7	∞	4	∞	∞
F	6	∞	3	∞	∞

D^B	A	C	D	E	F
A	3	18	2	∞	∞
C	13	8	8	∞	∞
D	4	15	1	∞	∞
E	7	12	4	∞	∞
F	6	13	3	∞	∞

D^C	A	B	D	E	F
A	∞	10	∞	8	∞
B	∞	8	∞	8	∞
D	∞	9	∞	7	∞
E	∞	12	∞	4	∞
F	∞	11	∞	5	∞

D^D	A	B	C	E	F
A	1	3	∞	∞	5
B	3	1	∞	∞	5
C	11	9	∞	∞	7
E	5	5	∞	∞	3
F	4	4	∞	∞	2

D^E	A	B	C	D	F
A	∞	∞	14	∞	4
B	∞	∞	12	∞	4
C	∞	∞	4	∞	6
D	∞	∞	13	∞	3
F	∞	∞	9	∞	1

D^F	A	B	C	D	E
A	∞	∞	∞	3	5
B	∞	∞	∞	3	5
C	∞	∞	∞	11	5
D	∞	∞	∞	2	4
E	∞	∞	∞	5	1

	BD en A	B	C	D	E	F
A	-	2,D	11,B	1,A	4,F	3,D
B	3,B	-	8,B	1,B	12,C	3,D
C	11,B	8,C	-	9,B	4,C	5,E
D	4,B	1,D	9,B	-	3,F	2,D
E	-	12,C	4,E	3,F	-	1,E
F	-	3,D	5,E	2,F	1,F	-

Destino

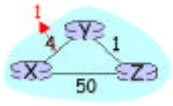
	BD en A	B	C	D	E	F
A	-	2,D	3,E	1,A	4,F	3,D
B	2,D	-	8,B	1,B	4,F	3,D
C	10,D	8,C	-	7,F	4,C	5,E
D	1,D	1,D	7,E	-	3,F	2,D
E	4,D	4,D	4,E	3,F	-	1,E
F	3,D	3,D	5,E	2,F	1,F	-

Destino

Encaminamiento en Arpanet

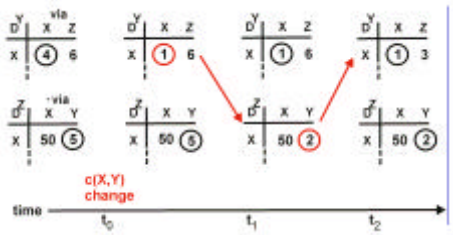
- Emplea encaminamiento Vector Distancia
- Los retrasos reales de los enlaces se utilizan como distancias
  - objetivo: obtener rutas por el camino de menor retraso
- Problemas:
  - El cálculo de rutas distorsiona el tráfico
  - Mediciones de retraso incorrectas
  - Respuesta a la congestión inesperada
  - Sincronización de las actualizaciones

Vector Distancia: cambio en los costes del enlace



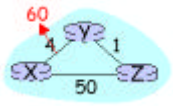
- Cambia el coste de un enlace
  - Un nodo detecta el cambio del coste de un enlace local
  - Actualiza su tabla de distancias
  - Si el coste cambia en uno de los caminos de menor costo se notifica a los vecinos

"las buenas noticias viajan rápido"



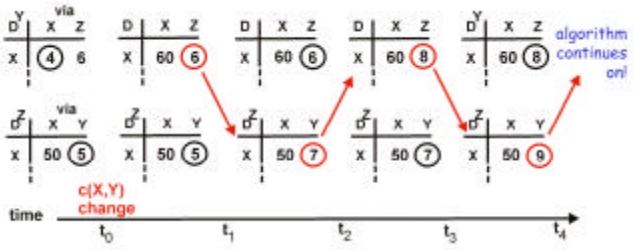
Fin del algoritmo

Vector Distancia: cambio en los costes del enlace (II)

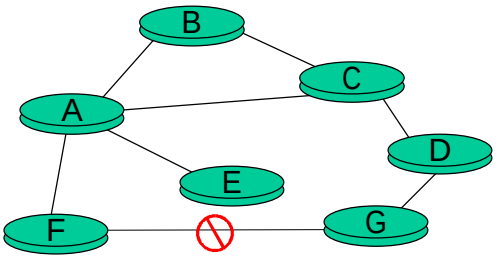


- Cambio del coste de un enlace
  - Las buenas noticias viajan rápido
  - Las malas noticias viajan más despacio
- Origen de numerosos problemas de los algoritmos de routing

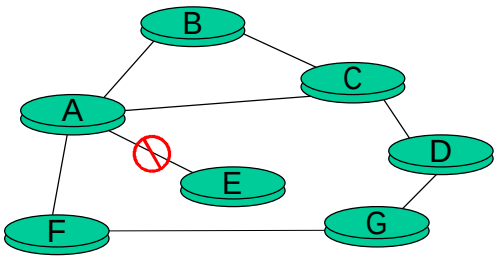
"las malas noticias viajan despacio"



Vector distancia: recuperación del fallo de un enlace



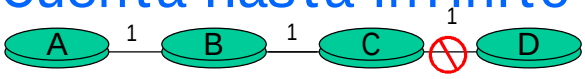
- F detecta que el enlace a G ha fallado
- F pone la distancia a G a ∞
- A pone la distancia a G a ∞ puesto que usa F para alcanzar a G
- A recibe la actualización de C con coste 2 en el camino a G
- A pone la distancia a G a 3 y envía la actualización a F (B, E y C)
- F decide que puede alcanzar G en 4 saltos a través de A



Problema: Bucles

- A notifica que la distancia a E es ∞
- B y C notifican que la distancia a E es 2
- B decide que puede alcanzar E en 3 saltos y lo notifica a A
- A decide que puede alcanzar E en 3 saltos y lo notifica a C
- C decide que puede alcanzar E en 5 saltos
- A, B y C forman un bucle

Ejemplo 1: Cuenta hasta infinito



- Costes al destino D (antes de fallar el enlace)
  - A 3, B
  - B 2, C
  - C 1, D
- El enlace C-D falla:
- Sucesivas entradas de las tablas con destino D

A	3, B	3, B	3, B	5, B	5, B	7, B	7, B ...
B	2, C	2, C	4, A	4,A	6, A	6, A	8, A ...
C	1, D	∞	3, B	5, B	5, B	7, B	7, B ...

- Eventualmente el algoritmo converge pero sólo después de "contar hasta infinito"
- Problema: el camino de B a D utiliza B, pero B no es consciente de ello

Soluciones

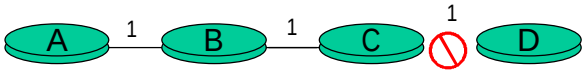
- Split Horizon
- Poisoned Reverse
- Hold Down
- Triggered Updates

- Ninguna de ellas resuelve completamente el problema, pero una combinación de ellas proporciona resultados suficientemente buenos.

Split Horizon / Poisoned Reverse

- Split Horizon:
  - Si un nodo W aprende una ruta hacia X a partir del nodo Y, W no notifica a Y dicha ruta hacia X
- Split Horizon con Poisoned Reverse
  - Se utiliza ¥ como métrica en las actualizaciones para rutas que han dejado de existir
  - Se acelera la convergencia

Ejemplo 1: Con Split Horizon



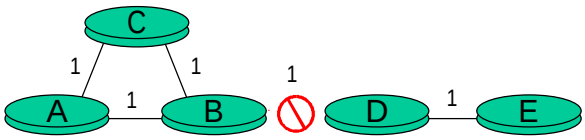
- Split Horizon: No se notifican rutas al destino al vecino desde el que aprendieron éstas
  - B no notifica a C la ruta hacia D
  - A no notifica a B la ruta hacia D
- Sucesivas entradas de las tablas con destino D

A	3, B	3, B	3, B	∞
B	2, C	2, C	∞	∞
C	1, D	∞	∞	∞

Triggered Updates

- También se acelera la convergencia "disparando" la actualización a los vecinos cuando la BD cambia
  - Acelera la cuenta (a infinito) para acabar más rapidamente con los bucles
  - Fuerza el envío de actualizaciones al detectar un cambio en la métrica de un enlace o una ruta.
  - No evita completamente bucles por la posible coincidencia de una actualización periódica con una "triggered"

Ejemplo 2: Con Split Horizon



- Sucesivas entradas de las tablas con destino E (a partir del fallo del enlace C-D)

A	3, B	3, B	4, C	5, C	6, C	...
B	2, D	∞	4, C	5, C	6, C	...
C	3, B	3, B	4, A	5, A	6, A	...

- Con Split Horizon

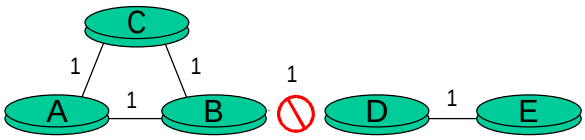
A	3, B	3, B	4, C	∞	6, C	...
B	2, D	∞	∞	5, A	6, C	...
C	3, B	3, B	4, A	5, A	6, A	...

- Split Horizon no soluciona el problema con bucles de más de dos nodos

Solución: Hold Down

- Cuando un nodo detecta que se ha perdido una ruta a un destino, no acepta una nueva ruta a dicho destino durante un cierto tiempo
  - El período depende del tamaño del bucle más grande que pueda formarse)
- Es difícil determinar el tiempo de "hold down".
  - Demasiado corto: puede conducir a aceptar (incorrectamente) nuevas rutas.
  - Demasiado largo: puede originar la pérdida de datos por incapacidad de encaminarlos hacia su destino.
- Afecta negativamente a la convergencia.

Ejemplo 2: Con Hold Down

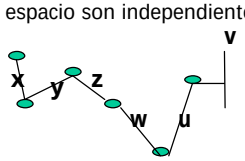


- Sucesivas entradas de las tablas con destino E (a partir del fallo del enlace C-D) con un tiempo de "Hold Down" suficiente

A	3, B	3, B	∞
B	2, D	∞	∞
C	3, B	3, B	∞

¿Cómo evitar bucles?

- Cada notificación de ruta incluye el camino completo
  - Si un router se encuentra en el camino, rechaza dicha ruta
  - Es lo que hace BGP
  - El espacio es proporcional al diámetro
- Cálculo de caminos implícitos
  - Propagar para cada red su predecesor
  - Puede calcularse recursivamente el camino
  - Los requisitos de espacio son independientes del diámetro



v	u
u	w
w	z
z	y
y	z

- ¿Elimina los bucles la técnica de cálculo implícito?"
  - No! Son posibles los bucles transitorios
  - ¿Por qué? Porque la información del camino implícito puede estar caducada
- El único modo de conseguirlo
  - Asegurar que se dispone de información actualizada preguntándolo explícitamente

Protocolos de Estado de Enlace (LS)

- Dos componentes principales en la operación de los protocolos LS:
  - Diseminación y mantenimiento de la topología
    - Inundación a través de toda la red de notificaciones de estado de enlace (LSA): disponibilidad y coste del enlace con cada vecino
    - A partir de los LSA recibidos cada router construye un *mapa* de dominio de la red (común a todos los routers)
  - Algoritmo de cálculo de rutas
    - Ejecutado en cada router independientemente
    - Métricas

Algoritmo de Estado de Enlace

Inundación:

1) **Periodicamente se distribuyen notificaciones de estado de emlace (LSA) a los vecinos**
   - **LSA contienen retrasos a cada vecino**
2) **Se instalan los LSA recibidos en la BD-LS**
3) **Se redistribuyen LSA a todos los vecinos**

Calculo del Camino

1) **Utiliza el algoritmo de Dijkstra para calcular distancias a todos los destinos**
2) **Instala pares <destino, siguiente salto> en la tabla de encaminamiento**

Notificación de Estado de Enlace (LSA)

- Paquete de los protocolos de Routing LS
  - En Internet encapsulado en datagramas UDP o segmentos TCP
- Incluye:
  - ID del router que creó el LSA
  - Costo del enlace a cada vecino directamente conectado
  - Número de secuencia (SEQNO)
  - Tiempo de vida (TTL) del LSA – contador máximo de saltos o número máximo de unidades de tiempo, según la definición del protocolo

Cálculo de rutas: Algoritmo de Dijkstra

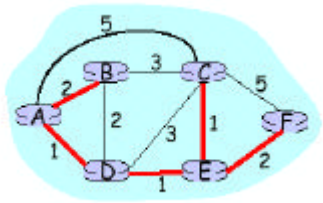
- Objetivo:
  - Calcular los caminos más cortos desde cada nodo (origen "s") a todos los demás de un grafo
- Notación:
  - C(i, j) = costo del enlace (i, j) (>= 0), vale ∞ si no son vecinos directos.
  - D(v) = Valor actual del costo del camino desde s al destino v
  - P(v) = nodo predecesor de v en el camino de mínimo costo actual desde s hasta v
  - N = conjunto de nodos cuyo camino de mínimo costo es conocido

Algoritmo de Dijkstra (II)

Inicialización
N = {A}
para todos los nodos v
si v es adyacente a A
entonces D(v) = c(A,v)
en caso contrario D(v) = ∞
bucle
encontrar w ∉ N tal que D(w) es mínimo
añadir w a N
actualizar D(v) para todo v adyacente a w y ∉N:
D(v) = min (D(v), D(w) + c(w,v)
/*nuevo costo a v es bien el costo viejo o el camino más corto conocido hasta w más el costo desde w a v */
Hasta todos los nodos Î N

Ejemplo (1) del Algoritmo de Dijkstra

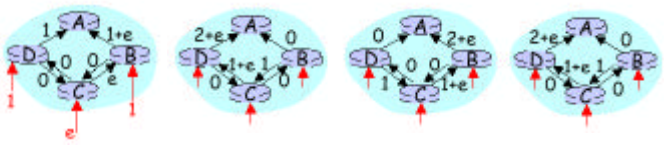
Paso	Inicio N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	∞	∞
1	AD	2,A	4,D		2,D	∞
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					E,E
5	ADEBCF					



Algoritmo de Dijkstra (III)

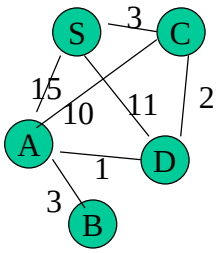
- Se mantienen dos listas:
  - *Tentativa* y *Confirmada*
- Cada lista contiene un conjunto de triplas:

  (**Destino, Costo, Siguiente Salto**)
  - El Siguiente Salto es el primer nodo en el camino desde s hasta el Destino
- Complejidad del algoritmo: n nodos
  - Cada iteración: Chequear todos los nodos, w ∉ N
  - n*(n+1)/2 comparaciones: O(n^2)
  - Implem. más eficiente posible: O(n*logn)
- Ocilaciones posibles :
  - P.e. costo del enlace = cantidad de trafico transportado

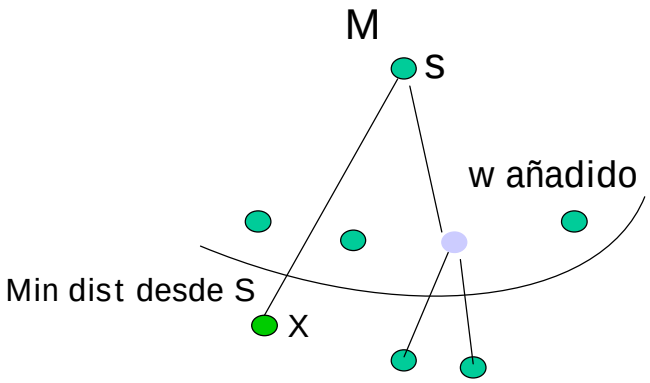


Inicial . . . recalculado . . . recalculado . . . recalculado

Ejemplo (2) del Algoritmo de Dijkstra



Paso	Confirmada	Tentativa
1	(S,0,-)	
2	(S,0,-)	(A,15,A) (C,3,C) (D,11,D)
3	(S,0,-) (C,3,C)	(A,15,A) (D,11,D)
4	(S,0,-) (C,3,C)	(A,13,C) (D,5,C)
5	(S,0,-) (C,3,C) (D,5,C)	(A,13,C)
6	(S,0,-) (C,3,C) (D,5,C)	(A,6,C)
7	(S,0,-) (C,3,C) (D,5,C) (A,6,C)	
8	(S,0,-) (C,3,C) (D,5,C) (A,6,C)	(B,9,C)
9	(S,0,-) (C,3,C) (D,5,C) (A,6,C) (B,9,C)	



- Suponer que para todos los nodos de M ya se ha encontrado la ruta de mínima desde S.
- Considerar el nodo X en N-M que está a la mímima distancia desde S de entre todos los nodos en N-M. ¿Hay algún camino más corto a X? NO
- ¿Por qué?
  - Considerar el camino más corto a X. Si el nodo antes de X no estuviera en M, entonces habría sido elegido él en lugar de X. Por lo tanto, X es el primer nodo fuera de M en este camino más corto. Pero entonces la distancia a X habría sido hya el valor más corto …

Métrica retraso del enlace: efecto sobre la estabilidad

- En la primitiva Arpanet se utilizaba la estimación del retraso
  - Longitud de la cola en cada instante como estimador de retraso
- Algoritmo más reciente
  - Tiempos de tránsito medio reales de los paquetes
  - Se notifican medias de 10 medidas, y sólo si el valor es significativamente diferente
  - Mediciones desfasadas
  - Reación más lenta a la congestión
    - Menos inestabilidades

Métricas de enlace

- Medir el retraso del enlace
  - En Arpanet
    - tiempo de partida - tiempo de llegada + tiempo de transmisión y latencia
  - Mejor métrica:
    - retraso del paquete = f(encolado, transmisión, propagación).
  - Cuando hay poco tráfico, transmisión y propagacion son buenos predictores
  - Cuando hay mucho tráfico el retraso de encolado es dominante y por lo tanto transmisión y propagacion son malos predictores

Métricas normalizadas

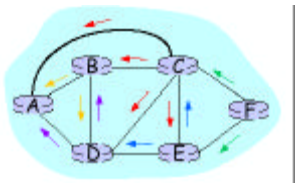
- Si un enlace cargado parace muy malo, todos lo abandonarían
- Deseamos que algunos permanezcan en él para balancear la carga y evitar oscilaciones
  - Todavía es un buen camino para algunos
- La métrica normalizada para un salto desvía rutas que tienen una alternativa que no es mucho más larga
- También limita los valores relativos y los rangos de valores notificados . . . Cambio gradual
- La medida de la utilización media limita el rango de cambio
  - 0,5 * muestreo + 0,5 * última media
- Normalización de acuerdo con el tipo de enlace (p.e. un satélite podría parecer buena opción cuando la cola en otros enlaces aumenta)
- El cambio máximo permitida depende del tipo de enlace
  - El cambio por actualización no puede ser mayor de 1/2 del valor del retraso de dicho salto (p.e. si máx es 90 y mín es 30 el peor caso es sólo 2 saltos peor que el mejor)

Inundación de LSAs y sincronización de LS BDs

- Cuando un router arranca o se activa un enlace, los routers sincronizan sus Bases de Datos (LS)
  - Los algoritmos LS dependen de la diseminación rápida fiable de los LSAs
  - Todos los routers terminan con un mapa de dominio común
  - La convergencia depende de un proceso de diseminación de LSAs apropiado (aún contando con pérdidas de paquetes).
- Los LSAs se caracterizan por un número de secuencia y su edad
  - ¿Qué es nuevo y qué es viejo?

Algoritmo de Inundación

- Los LSAs nuevos se envían por todos los enlaces excepto el de llegada
  - Cada LSA se transmite al menos dos veces por cada enlace
  - La diseminación de información es independiente del encaminamiento



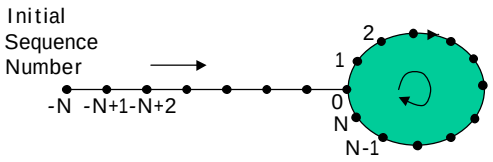
- Los LSAs nuevos reemplazan a los anteriores en la Base de Datos local
- Los LSAs se transmiten de modo fiable (reconocidos)
- La velocidad de inundación se limita mediante el intervalo mínimo entre LSAs LSAs
- Inundación periódica (30 min) para refrescar las BDs (envejecimiento de LSAs)

Números de secuencia en LSAs

- Utilizados para identificar el LSA más reciente
  - Los números de secuencia mayores son más recientes
  - LSAs más recientes sustituyen a los más antiguos
- Dos problemas
  - Uso circular del contador de números de secuencia
    - Los números de secuencia menores son en este caso más recientes
  - Elección del número en el momento del arranque
    - Necesario para poder sobreescribir anteriores LSAs
- Varios enfoques posibles:
  - Espacios de números de secuencia enormes
  - Espacio de números de secuencia "Lollipop"
  - Envejecimiento de las entradas
  - Sincronización de la BD en el instante del arranque

Número de secuencia Lollipop (no se utiliza)

- Concepto Básico
  - Forzar el uso de un único número de secuencia de *arranque único*
  - Usar la recepción de LSA con número de arranque como *disparador* para notificar al emisor él número actual
  - Una vez notificado el número de secuencia actual incrementarlo en 1
- Sa es más antiguo que Sb si:
  - Sa < 0 y Sa < Sb
  - Sa > 0, Sa < Sb, y Sb-Sa < N/2
  - Sa > 0, Sb > 0, Sa > Sb, y Sa-Sb > N/2



Envejecimiento de los LSAs

- Cada LSA tiene un campo *edad*
  - Incrementado con cada transmisión y mientras permanece almacenado
  - MaxAge determina el tiempo de vida de un LSA
    - Se inicializa cada vez que se refresca
- Dos usos de MaxAge: Inundación de LSA de MaxAge
  - Eliminación de LSAs caducados o inválidos
  - Eliminación de la entrada actual antes de que el contador vuelva al principio
- Impacto de MaxAge en el problema del arranque
  - Si el router espera a MaxAge, los LSAs viejos serán purgados
  - Pero la selección de MaxAge es difícil
    - MaxAge pequeño minimiza la latencia en el arranque
    - MaxAge pequeño imponen mayor sobrecarga de inundación y puede impedir la distribución completa de LSAs en redes grandes
- La sincronización inicial de la BD gestiona el arranque

Características de los protocolos de Estado de Enlace

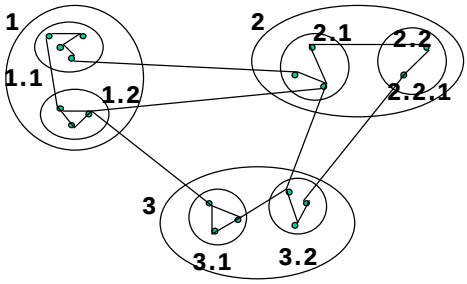
- Con LSDBs consistentes, todos los nodos calculan caminos consistentes libres de bucles
  - Más estable (convergencia más rápida)
  - Pueden ocasionar también bucles permanentes
- Soportan métricas múltiples
- Soporte de múltiples caminos: balanceo de carga
  - Implementación compleja
    - Consistencia de LS BSs
    - Minimizar coste de inundación
- Limitados por la carga computacional de Dijkstra y los requisitos de espacio de almacenamiento

Vector Distancia vs. Estado de Enlace

- Algoritmos DV
  - Implementación Simple
  - Bajo costo de almacenamiento
  - No se conoce la topología de la red
  - Sólo gestionan métricas simples
  - Puede originar bucles, con soluciones ad-hoc
  - Convergencia lenta
  - Ejemplos:
    - RIP and RIP2
    - IGRP/EIGRP
    - BGP/IDRP
- Algoritmos LS
  - Los LSA proporcionan flexibilidad
    - Soportan métricas múltiples
  - Escalabilidad
  - Mecanismo de seguridad, autenticando el origen del LSA
  - Costo de almacenamiento (tabla de encaminamiento y LSBD)
  - Más estable (conv. más rápida)
  - Soporte de múltiples caminos: balanceo de carga
  - Implementación compleja
    - Consistencia de LS BSs
    - Minimizar coste de inundación
  - Ejemplos:
    - IS-IS
    - OSPF

Routing jerárquico

- El routing plano no es escalable
  - No puede esperarse que cada nodo mantenga rutas a cada destino a nivel global en Internet
- Jerarquía de áreas: técnica para el direccionamiento jerárquico de nodos en una red
- Dividir la red en áreas
  - Las áreas pueden solaparse
  - Las áreas pueden tener sub-areas anidadas
  - Restricción: no debe existir un camino entre dos sub-areas de un área que salga de ésta
    - Las sub-areas deben estar completamente contenidas dentro del área

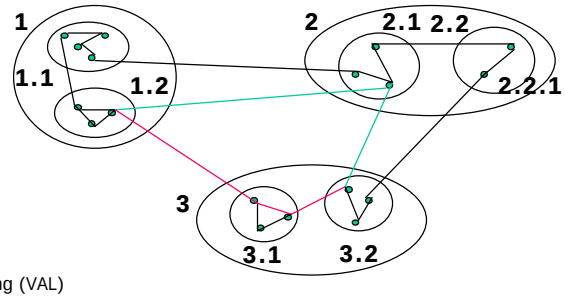


Direccionamiento

- Las direcciones de las áreas son jerárquicas
  - Las áreas del mayor nivel se numeran secuencialmente
  - Las sub-areas de un área se etiquetan en relación a aquella
  - Los nodos se numeran en relación al área más pequeña que los contiene
    - Los nodos pueden tener múltiples direcciones

Routing jerárquico

- Dentro de un área
  - Cada nodo mantiene rutas a hasta cada uno de los demás nodos del área
- Fuera del área
  - Cada nodo mantiene rutas sólo para otras áreas de máximo nivel
  - Los paquetes inter-area son encaminados al router periférico más próximo
- Pueden originarse caminos sub-óptimos

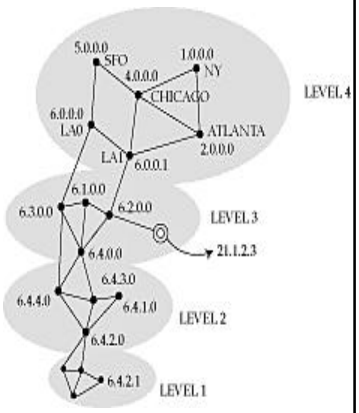


Camino rojo de tres saltos
vs
Camino verde de dos saltos

(BarNoy y Gopal describen el compromiso entre exportar información adicional y el uso de caminos sub-óptimos)

Ejemplo de Routing Jerárquico

- Hechos
  - Número limitado de routers en cada nivel
    - Necesario para la escalabilidad
  - Protocolos independentes calculan rutas en cada nivel
  - El encaminamiento "end-to-end" necesita comunicación entre niveles
    - Los gateways realizan la traducción y agregación
  - La agregación se basa en la capacidad de resumir direcciones (prefijos comunes)
    - Prefijo CIDR
    - Evitar asignación de direcciones no agregables
- Internet utiliza dos niveles básicos
  - Intra-Dominios (protocolos de routing interiores)
  - Inter-Dominios(protocolos de routing exteriores)



Jerarquía en Internet

- Jerarquía de tres niveles en las direcciones
  - Número de red
  - Número de subred
  - Número de nodo
- Jerarquía de dos niveles en la red
  - Inter-Sistema Autónomo (AS): Entre routers backbone
  - Intra-Sistemas Autónomos: Entre routers dentro de un AS
- Los "routers backbone" notifican rutas sólo a redes y no subredes
  - P.e. 135.104.* (class B), 192.20.225.* (class C)

    10000111 **110**00000
  - Incluso así: 80,000 redes en los routers del núcleo (1996)
- Los Gateways "pegan" routers inter-dominio e intra-dominio para encontrar el mejor siguiente-salto a cada otra red en la Internet

Unión de rutas Inter e Intra-Dominio Routes

- Si un AS tiene múltiples gateways
  - Registros *externos*: Aprendiendo sobre el exterior
    - Costo de la ruta (externa) desde el gateway a cualquier destino en otro AS
      - p.e. permite a 6.4.0.0 descubrir que el camino más corto a 5.* es a través de 6.0.0.0
  - Registros *resumen*: Notificando el propio dominio al exterior
    - El coste de la ruta (interna) desde el gateway a cualquier destino del propio AS
      - p.e. permite a 5.0.0.0 descubrir que el camino más corto a 6.4.0.0 es a través de 6.0.0.0
- Los gateway simples o por defecto eliminan la necesidad de registros externos
- La decisión de routing extremos a extremo se obtiene concatenando registros externos/resumen y cálculos de rutas locales (inter o intra-dominio)

Protocolos interiores (Intra-Dominio)

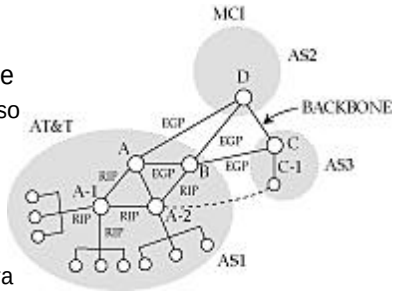
- *Cálculo* de rutas a destinos *dentro* de un AS
- *Distribuye* rutas *a* destinos externos
- Algunas propiedades generales
  - Generalmente tiene pocas restricciones administrativas
  - No interesa la escalabilidad, aunque también se utiliza una jerarquía de dos niveles *(interna)*
    - La partición de AS en *areas* conectadas por un *backbone*
    - Usa registros externos y "summary" entre áreas
- Dos familias de protocolos
  - Vector Distancia (RIP2)
  - Estado de enlace (OSPF and IS-IS)

Protocolos Exteriores (Inter-Dominio)

- Propósito
  - Permitir rutas que atraviesen dominios (AS)
  - Proporcionar suficiente información para permitir la conectividad extremo a extremo sin revelar la topología interna del dominio
    - Cada AS está operado por una entidad diferente
    - Existe cooperación pero no "confianza" entre dominios
- El *gateway periférico* es responsable de adquirir y notificar la información necesaria
  - Debe participar tanto en el routing intra como inter-dominio
  - Traduce y resume las rutas internas y externas
  - Decide qué notificar (ocultar) al exterior
- El protocolo "Border Gateway Protocol" (BGP4) es el ejemplo dominante

Consideraciones en las conexiones Inter-Dominio

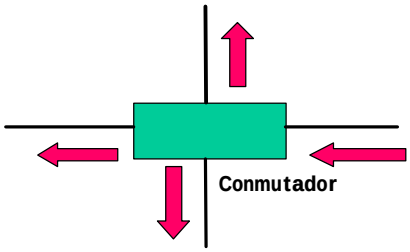
- Evitan que el tráfico interno utilice enlaces externos (y viceversa)
  - p.e., A-D-B utilizado para tráfico A-B
- Acomoda diferentes protocolos de routing en cada dominio
  - DV vs. LS
- Gestiona diferencias entre métricas de coste
  - Cuenta de saltos vs ancho de banda o retraso
- Enfoque básico
  - Vectores distancia especifican costes de routing desde los gateways a rutas internas/externas
  - El mínimo común denominador utilizado para el coste inter-AS
  - Intervención manual para "ocultar" rutas internas ("backdoor") o "mostrar" externas



ROUTING BROADCAST

- Encaminamiento de mensajes broadcast
  - Inundación
  - Spanning tree
  - Reverse-Path broadcast

Inundación



- Se reenvía el paquete por todos los enlaces menos por aquel por el que se recibió
- Su implementación es muy simple
- Ineficiente, genera multitud de duplicados
  - Para limitarlo se puede limitar el número máximo de saltos para cada paquete
  - Otra posibilidad es identificar los paquetes de manera no ambigua y que cada router mantenga una lista de los paquetes enviados, evitando reenviarlos de nuevo

Spanning Tree

- Idea Básica:
  - Sobreponer un árbol de distribución a la topología de la red
    - El árbol de distribución no tiene bucles
  - Inundar y aprender
  - Construir dinámicamente el árbol para adaptarse a los fallos
  - Los paquetes se replican en las bifurcaciones del árbol.
    - El sistema asegura que la distribución se hará generando el número mínimo de paquetes y sin duplicados.

Algoritmo Spanning Tree

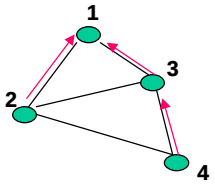
Condiciones de inicio :
Cada router tiene un ID
Cada router cree que es la raíz

1) Cada router notifica a sus vecinos
- **su propio ID de router**
- **El ID del que cree que es la raíz**
- **Su distancia a la raíz**

2) Al recibir una notificación
- **Si la raíz notificada tiene un ID menor que su propia raíz,**
  cambia su raíz a la raíz notificada
- **Ajusta su distancia a la raíz a uno más que la distancia notificada**
- **repite el paso 1**

Spanning Tree: Ejemplo

Nodo 1	Nodo 2	Nodo 3	Nodo 4
Nodo 1 Distancia 0	Nodo 2 Distancia 0	Nodo 3 Distancia 0	Nodo 4 Distancia 0
Nodo 1 Distancia 0	Nodo 1 Distancia 1	Nodo 1 Distancia 1	Nodo 2 Distancia 1
Nodo 1 Distancia 0	Nodo 1 Distancia 1	Nodo 1 Distancia 1	Node 1 Distancia 2



RPB: Reverse-Path Broadcast

- Usa la tabla de routing existente con los caminos más cortos.
- Si llega un paquete a través de una interfaz que es el camino más corto al emisor, reenvía el paquete por todos los interfaces.
- En caso contrario elimina el paquete.

RPB: Reverse-Path Broadcast

- Usa la tabla de routing existente con los caminos más cortos.
- Si llega un paquete a través de una interfaz que es el camino más corto al emisor, reenvía el paquete por todos los interfaces.
- En caso contrario elimina el paquete.