11_30_aprile

Continuo 10_23_aprile.pdf

Sistemi centralizzati e decentralizzati
Nei sistemi che usano il distance vectors, ogni nodo è indipendente e pari rispetto agli altri, si dice che è un *sistema decentralizzato*. Al contrario, un *sistema centralizzato* è più semplice da gestire perché c'è un singolo nodo che comanda/gestisce tutti gli altri ma se questo nodo "capo" viene a mancare per un guasto o un attacco, la rete non funzionerà più bene. Nei sistemi decentralizzati (come quelli che implementano il distance vectors) ogni nodo conosce solo i suoi vicini diretti e conosce le informazioni che questi gli passano (cioè il vettore delle distanze)

Problema del Distance Vectors
Uno dei problemi più importanti del distance vectors è che le false notizie si propagano dato che il nodo conosce le informazioni che i suoi vicini gli passano, ma se quelle informazioni sono false verranno prese per vere

RIP e problema della distanza
Il protocollo di distance vectors è stato implementato nel protocollo RIP, in cui la distanza veniva misurata attraverso i salti (hop), 1 distanza = 1 hop, senza tener conto della distanza fisica nella realtà o della larghezza di banda di quel collegamento
Esempio: potrebbero esserci 2 router che hanno stessa distanza per il nodo (1 hop) ma uno si trova a 1 km l'altro a 10 metri.

Protocollo LINK STATE ROUTING
LSR ha sostituito il protocollo RIP.
Questo protocollo usa un idea diversa (nel distance vectors si usava Bellman-Ford), qui si usa l'algoritmo di Dijkstra.
L'obbietti di questo protocollo è avere una conoscenza completa della rete da parte di ogni nodo. Infatti qui ogni router investiga sui suoi vicini e poi manda le informazioni in multicast (quindi solo ai router) in modo che tutti possano conoscersi e determinare la via con minor costo. Il calcolo del percorso ottimale è basato sull'algoritmo di Dijkstra

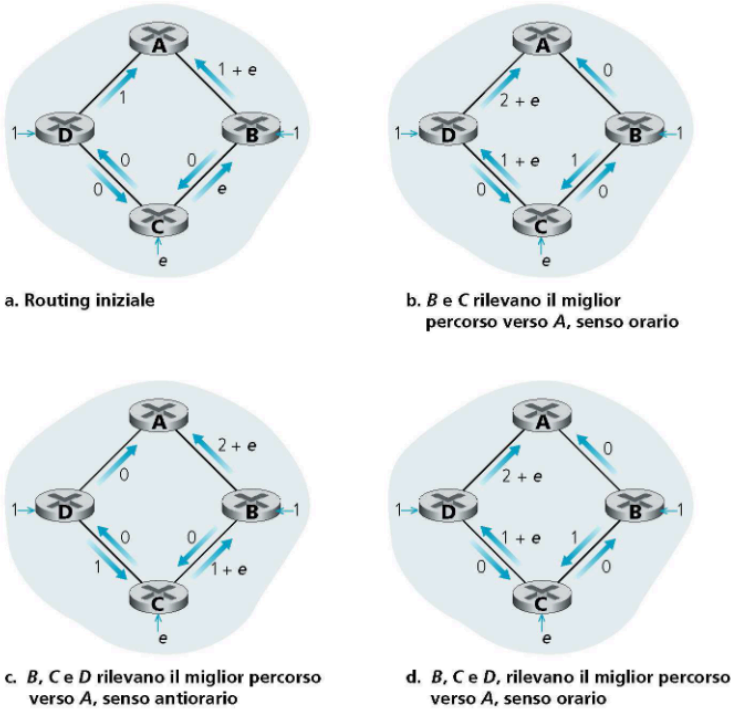
LSR periodicamente manda dei pacchetti di aggiornamento sullo stato del link, questo pacchetto è chiamato Link State Packet (LSP) e contiene informazioni del tipo: mittente, interfacce attive, costo del collegamento ecc...
Queste informazioni vengono inoltrate a tutti i nodi della rete attraverso l'algoritmo di flooding, quindi il primo router manda l' LSP ai suoi vicini e questi ultimi li inoltrano ai loro vicini (tranne a quello da cui l'hanno ricevuto) si fermeranno solo quando tutti i nodi hanno ricevuto il pacchetto e tutti hanno quindi ricevuto l'aggiornamento sullo stato della rete.

Questo è utile quando ad esempio nella rete si verificano dei guasti a dei nodi o ne viene cambiata la configurazione, scambiandosi LSP i router possono aggiornarsi sulla nuova disposizione della rete

LSR è migliore di RIP anche perché considera nel calcolo della distanza e del costo anche la larghezza di banda. Ad esempio permettendo di preferire un percorso con 3 salti a 10 Gbps rispetto a uno con 1 salto a 10 Mbps.

Problemi di LSR
LSR non è perfetto anzi esistono diversi problemi come:

- Il maggior uso di CPU e memoria: utilizzando Dijkstra anche su reti con diversi nodi questo influisce negativamente sulle performance dei router, dato che questo algoritmo non ha una bassissima complessità
- Oscillazione delle rotte



Il calcolo del costo viene fatto anche in base alla larghezza di banda disponibile in quel collegamento, quindi immaginiamo questa situazione: i 4 router sono collegati da due collegamenti che offrono le stesse prestazioni: uno ha un flusso di traffico supponiamo ad intensità 1 mentre l'altro 1+e, i router quindi calcolano che il percorso con minor costo sia quello con intensità 1 quindi cominciano a mandare pacchetti in quel collegamento, nel frattempo il collegamento che aveva intensità 1+e ora si è svuotato, i router ricalcolano i costi e cambiano di nuovo collegamento. Questo effetto si chiama *Oscillazione delle rotte*, i router tenderanno ad alternare i 2 collegamenti anziché usarli entrambi dividendo il traffico, questo porta ad un calo di prestazioni sia della rete che dei router perché questi ultimi dovendo sempre ricalcolare dijkstra avranno sempre CPU e memoria occupate e potrebbero anche perdere dei pacchetti nel frattempo.

Comparison between DV and LSR

DV	LSR
Decentralized	Global
Messages for neighbors only	Broadcast Messages
Information regarding all destinations	Information about your links only
Infinity Counting	Synchronisation problems (oscillations)
Messages only for link changes	Increased message traffic (periodic sends)
Not very robust against malicious nodes	Robust to attacks

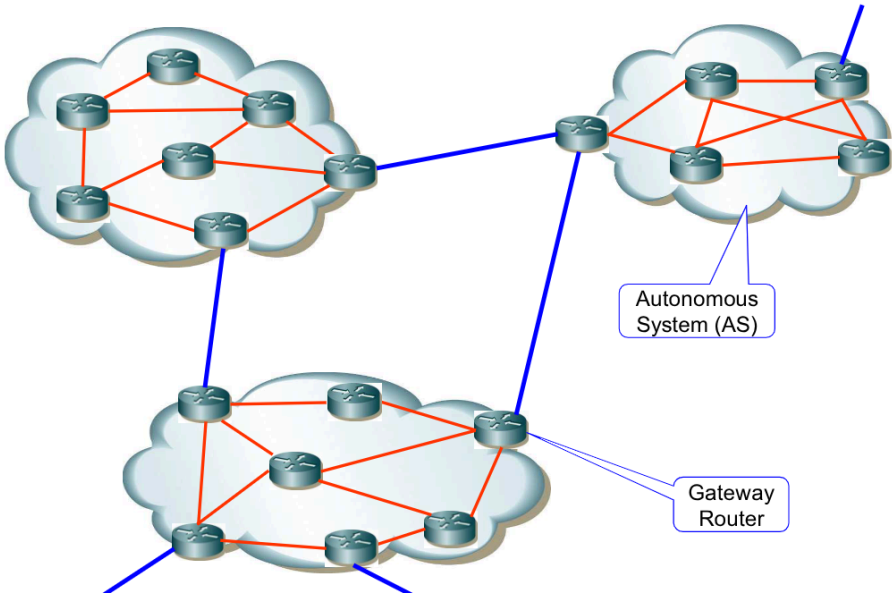
LSR può lavorare su reti piccole e medio-grandi ma non conviene più su reti grandissime, perché la complessità e la matrice di adiacenza di dijkstra cresce sempre più quando aumenta il numero di nodi.

AS: Autonomous System

Per questo motivo internet o comunque reti giganti sono divise in AS → Autonomous system, gestite e amministrate da enti come ISP (Internet Services Provider) come TIM Vodafone e altri, ma anche grandi aziende come Google hanno il loro AS. Un AS può essere grande come un intera nazione.

Protocols:

- Interior Gateway Protocols (IGPs) within AS (RIP, OSPF, HELLO, IS-IS)
- Exterior Gateway Protocols (EGPs) among AS (EGP, BGP-4)



Siamo passati al **DATA LINK LAYER** 5_DLL_N.pdf

Il DLL si occupa di trasformare il segnale in bit e i bit in segnale, si occupa anche degli errori, raggruppa i bit per creare i frame

In ogni singolo host (device e router) di una rete, è implementato il Data Link Layer. Ciò avviene tramite una combinazione di hardware, software e firmware presente all'interno dei NIC (Network Interface Controller) dei dispositivi di rete. Buona parte del DLL è implementato in hardware, come il protocollo Ethernet nell'adattatore Intel 700, o il protocollo Wi-Fi nel chipset Atheros AR5006. Tuttavia, è il software a farsi ponte tra il livello di rete, e il livello fisico

Comunicazione tra 2 Network Interface Controller
Lato mittente:

1. Il datagramma da trasmettere viene incapsulato in un frame del DLL.
2. I campi d'intestazione del DLL vengono riempiti.
3. (Se presenti) vengono stabiliti i valori dei bit nei campi per la rilevazione e correzione degli errori.
4. Il pacchetto viene trasferito.

Dal lato del ricevente:

1. Il Network Interface Controller riceve l'intero frame.
2. (Se presenti) effettua le verifiche relative ai campi di rilevazione e correzione errori.
3. Estrae il datagramma.
4. Manda il datagramma al livello superiore.

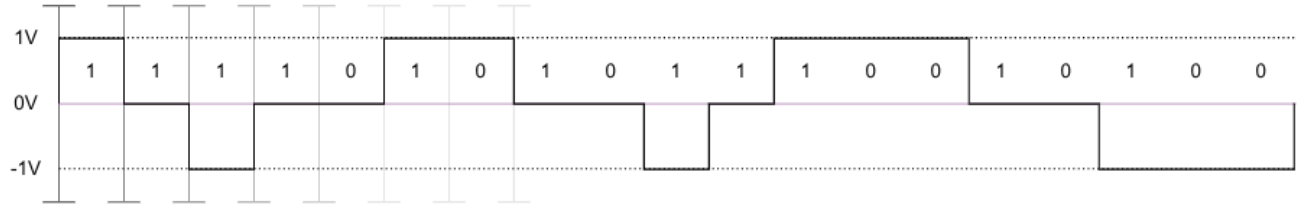
Data Framing

Si potrebbe pensare che usare 2 segnali per la comunicazione possa essere sufficiente, ma non è così, infatti in questo caso non è possibile stabilire se una

sequenza di 0 sia una parte della comunicazione o se ci sia assenza di comunicazione, per questo motivo si utilizzano codifiche a più livelli:

Tree level encoding
Prevede 3 livelli di segnale: alto zero e basso.
Ci sono anche delle regole:

- prossimo bit = 0 → nessuna transizione del segnale
- Prossimo bit = 1 e livello del segnale $\neq 0$ → Il segnale passa a 0.
- Prossimo bit = 1 e livello di segnale = 0 → Il segnale passa ad un livello opposto a quello del più recente segnale diverso da 0.



Alcuni mezzi fisici come la gigabit Ethernet usano una codifica a 5 livelli che permette di inviare 2 bit per volta, *è quindi intuitivo pensare che più livelli aggiungo più la velocità della comunicazione aumenta, ed è così, però nella realtà devo tener conto del rumore intrinseco che si trova in ogni mezzo fisico reale*, ad esempio se ho un range di 5 volt (0-5 volt) è facile distinguere tra 0 e 5 (2 livelli) ma anche tra 0,1,2,3,4,5 (5 livelli) ma se volessi dividere in 1000 livelli la situazione si complica perché il la differenza tra un livello e l'altro è pochissima quindi basta anche un minimo rumore (che nella realtà è sempre presente) per generare degli errori, quindi dopo un tot di livelli smette di essere conveniente perché il canale è sempre più soggetto a errori man mano che i livelli aumentano

Problemi del tree level encoding
Il tree level encoding è la base per la comunicazione ma presenta 2 problematiche:

1. **Il problema del DC-Balance (Componente Continua)**

In un mondo ideale, vorremmo che la media dei voltaggi inviati su un cavo fosse **0V**. Se la media è diversa da zero, si crea quella che chiamiamo "Componente Continua" (DC Offset).

- **Saturazione e Calore:** Se invii molti segnali positivi senza "compensarli" con quelli negativi, la linea accumula energia. Questo può surriscaldare i componenti o mandare in saturazione i trasformatori di accoppiamento che si trovano nelle schede di rete.
- **Spostamento della soglia:** Il ricevitore deve capire se un segnale è "alto" o "basso" confrontandolo con una media. Se la media elettrica del cavo si sposta verso l'alto a causa del mancato bilanciamento, il ricevitore potrebbe fare fatica a distinguere

uno zero da un uno, perché la sua "linea di terra" di riferimento è diventata instabile.

2. **Perdita di Sincronizzazione (Clock Recovery)**

- **Il silenzio degli zeri:** Se invii una lunghissima sequenza di zeri e la tua codifica prevede che lo zero sia $0V$, il cavo rimane "piatto" per molto tempo. Senza cambiamenti di tensione (transizioni), il ricevitore non ha più riferimenti per capire dove finisce un bit e dove inizia il prossimo.

Per ovviare a questi problemi entra in gioco la codifica a blocchi:
Codifica a blocchi
sostituisce blocchi di bit di una determinata dimensione, con blocchi leggermente più grandi (4 → 5, 8 → 10), traducendo le sequenze in maniera tale da garantire un numero minimo di transizioni.

4B5B
Associa a blocchi di 4 bit, blocchi di 5 bit. Richiede una bandwitdh del 25% più capiente. E' utilizzata per fast Ethernet.

Nome	4B	5B	Descrizione
0	0000	11110	hex data 0
1	0001	01001	hex data 1
2	0010	10100	hex data 2
3	0011	10101	hex data 3
4	0100	01010	hex data 4
5	0101	01011	hex data 5
6	0110	01110	hex data 6
7	0111	01111	hex data 7
8	1000	10010	hex data 8
9	1001	10011	hex data 9
A	1010	10110	hex data A
B	1011	10111	hex data B
C	1100	11010	hex data C
D	1101	11011	hex data D
E	1110	11100	hex data E
F	1111	11101	hex data F
I	-NONE-	11111	Idle
J	-NONE-	11000	SSD #1
K	-NONE-	10001	SSD #2
T	-NONE-	01101	ESD #1
R	-NONE-	00111	ESD #2
H	-NONE-	00100	Halt

Esiste anche la codifica 8B10B che è molto usata in vari standard:
• PCI Express (< 3.0)
• IEEE 1394b (Firewire)
• Serial ATA
• Fibre Channel
• Gigabit Ethernet (alcune versioni)
• DisplayPort Main Link
• DVI e HDMI
• USB 3.0