Reti di Telecomunicazioni

Routing generale

Autori

Queste slides sono state scritte da

Michele Michelotto:

michele.michelotto@pd.infn.it

che ne detiene i diritti a tutti gli effetti

Copyright Notice

Network layer

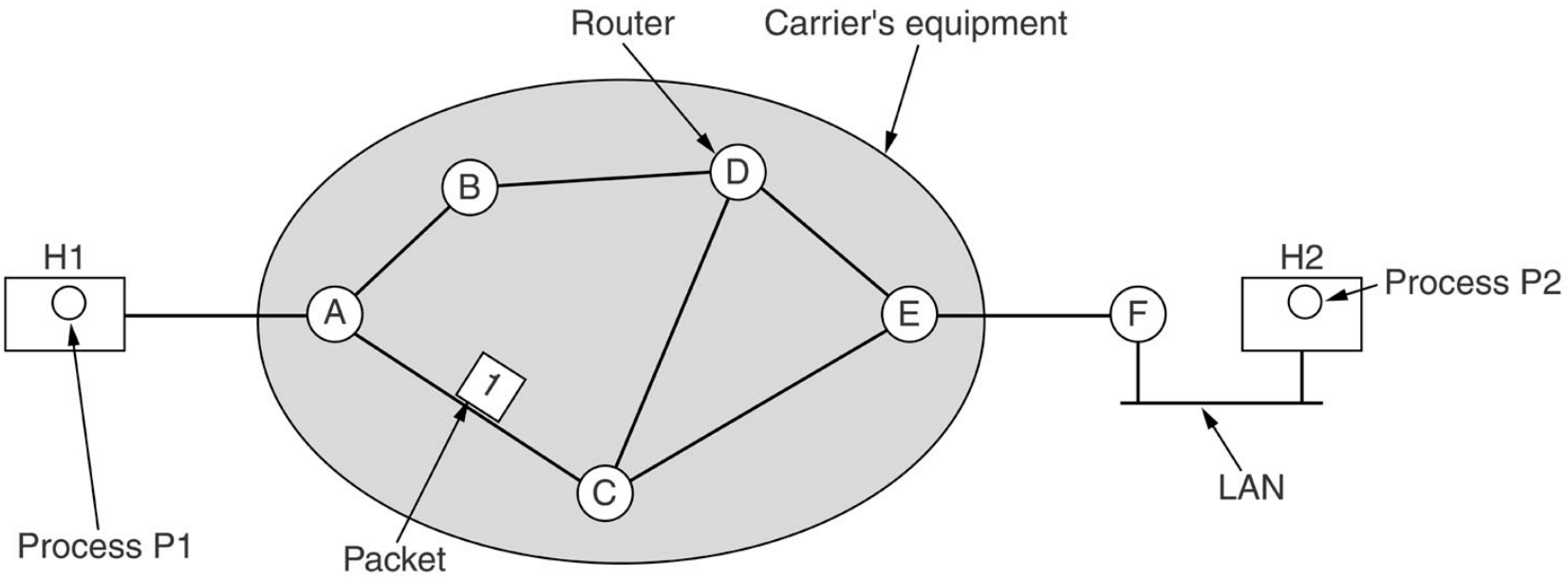
Introduzione
Layer: Modello OSI e TCP/IP
Physics Layer
Data Link Layer
MAC sublayer
Network Layer
Transport Layer
Application Layer

Network - DataLink

- **Network Layer**
  - Deve portare un pacchetto da sorgente a destinazione
  - È il layer più basso che si occupa di trasmissione end to end
  - Per questo deve conoscere la topologia della subnet di comunicazione e scegliere il cammino appropriato
  - Potrebbe anche dover scegliere tra diversi cammini per trovarne uno non troppo sovraccarico
  - Deve decidere cosa fare quando source e destination sono su reti diverse
- **DataLink Layer**
  - Deve solo portare un frame da una parte all'altra di un cavo (di un mezzo trasmissivo, eventualmente condiviso)

Contesto di utilizzo

- L'Host H2 manda pacchetti al router più vicino
- Il pacchetto viene tenuto fino a quando non è arrivato completamente, viene fatto il checksum e mandato al router successivo



Servizi forniti a L4

- Il network layer fornisce **servizi** al transport layer attraverso l'interfaccia network/transport
  - Questi servizi sono indipendenti dalla tecnologia dei router
  - Il livello di trasporto deve essere schermato dalle complicazioni relative a numero, tipo e topologia dei router
  - Gli indirizzi di rete usati dal livello di trasporto devono avere una numerazione uniforme, anche attraverso LAN e WAN

Connection(less)

- Questi obiettivi lasciano molta libertà nel dettagliare i servizi forniti.
- Questo porta ad una lotta tra due fazioni
  - Connetion Oriented Service
  - Connectionless Service

Visione IP

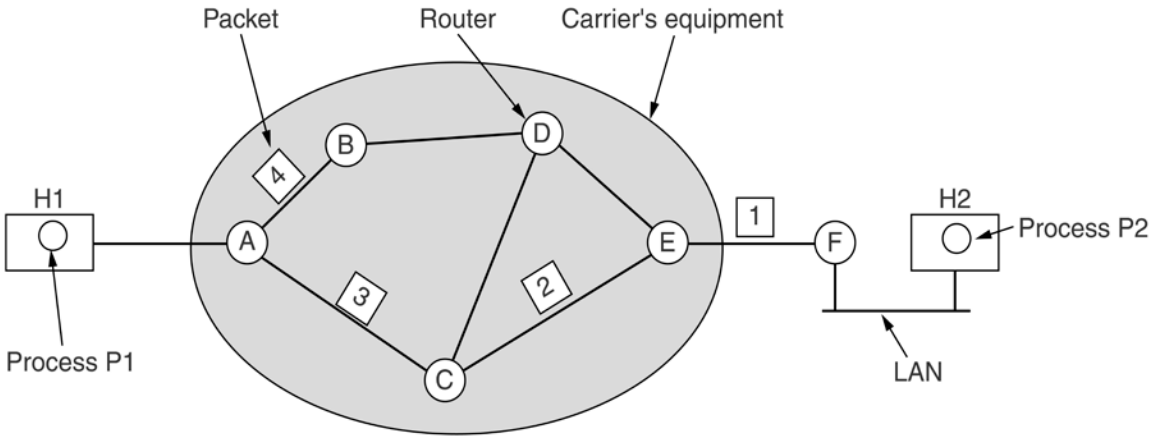
- La comunità IP (30 anni di esperienza) vorrebbe che i router facessero solo due cose: SEND PACKET, RECEIVE PACKET
  - Bisogna accettare che la rete è inaffidabile
  - Gli host devono gestire l'error control per conto loro
  - Niente flow control, niente ordinamento di pacchetti
  - Ogni pacchetto deve avere il suo indirizzo per poter viaggiare in modo indipendente
  - INTERNET è l'esempio

Visione TELECOM

- La sottorete deve fornire un servizio connection oriented affidabile
  - 100 anni di esperienza con la rete telefonica
  - Importante la qualità di servizio quasi impossibile da raggiungere senza una rete connection oriented
  - ATM è l'esempio

Servizi Connectionless

- I pacchetti sono chiamati anche **datagram** e sono instradati indipendemente
- **P1** sull'host **H1** deve mandare un messaggio lungo 4 pacchetti a **P2** sull'host **H2**
- Dopo il terzo pacchetto la tabella di routing di **A** cambia
- Il quarto pacchetto segue una strada diversa, non passa per **ACE**



A's table

	initially		later
A	–	A	–
B	B	B	B
C	C	C	C
D	B	D	B
E	C	E	B
F	C	F	B

Dest. Line

C's table

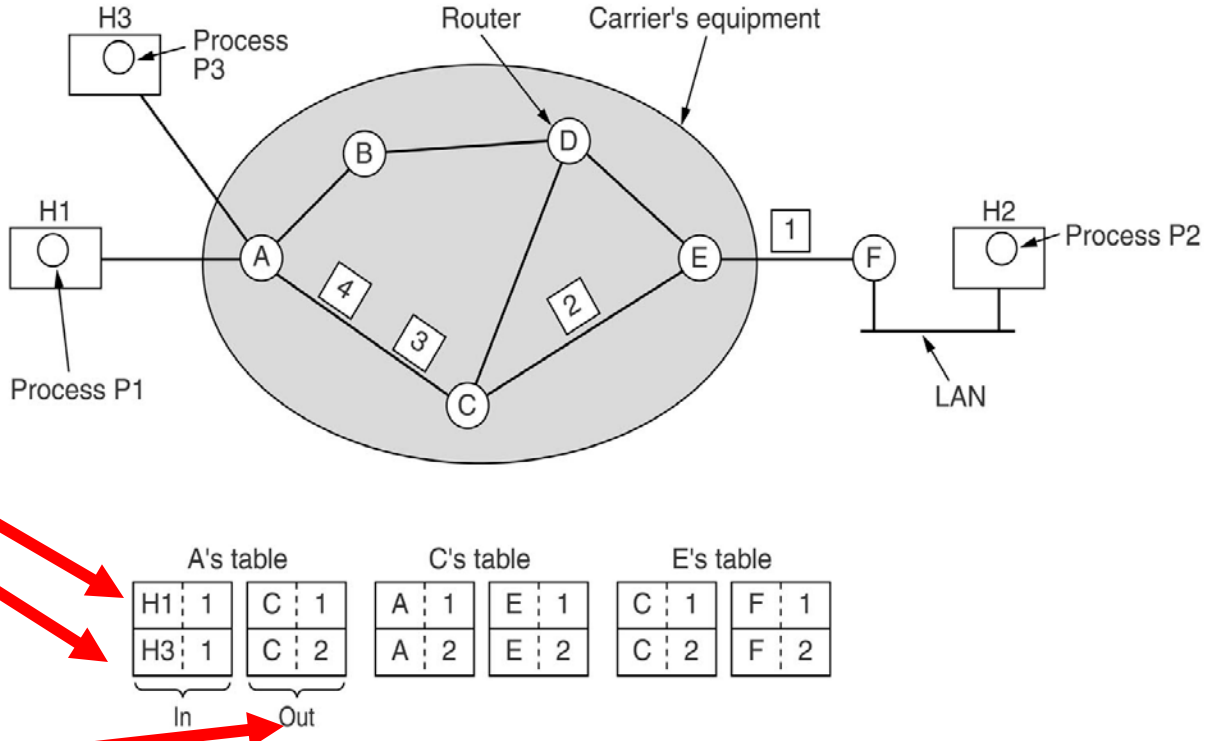
A	A
B	A
C	–
D	D
E	E
F	E

E's table

A	C
B	D
C	C
D	D
E	–
F	F

Serv. Connection Oriented

- Devo prima stabilire un path, poi posso mandare pacchetti
- Il path si chiama Virtual Circuit
- Tutti i pacchetti seguono questo VC e ogni pacchetto si porta un identificativo del proprio VC
- Il router **A** vede una connessione **VC1** da **H1** e una **VC1** da **H3**. In uscita verso router **C**, le rinomina dando diversi identificativi (**VC1** e **VC2**)



Confronto

Issue	Datagram subnet	Virtual-circuit subnet
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Routers do not hold state information about connections	Each VC requires router table space per connection
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow it
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Quality of service	Difficult	Easy if enough resources can be allocated in advance for each VC
Congestion control	Difficult	Easy if enough resources can be allocated in advance for each VC

Algoritmi di Routing

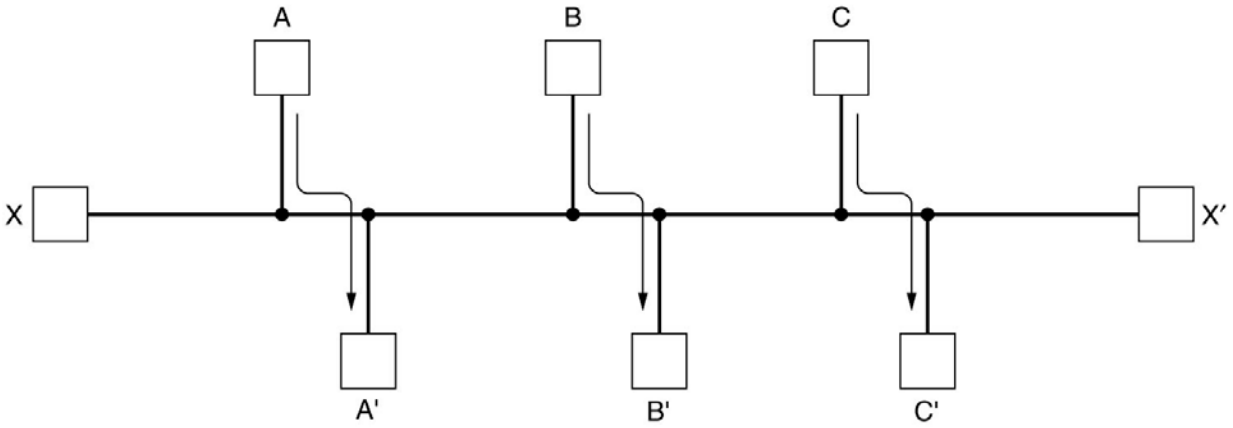
- È la parte del sw del Network Layer che decide quale linea di output usare per un pacchetto in entrata
- Se la subnet usa datagram la decisione va presa per ogni pacchetto
- Se usa virtual circuit la decisione va presa nella fase di setup del circuito virtuale
- È utile separare la fase di **forwarding** in cui un pacchetto viene forwardato sulla base della tabella di routing dalla fase in cui la **tabella di routing viene creata e aggiornata**

Criteri

- Correttezza → Ovvio
- Semplicità → Intuitivo
- Robustezza → Host e router falliscono, gli algoritmi devono sapersi adattare ai cambiamenti topologici
- Stabilità → Alcuni algoritmi non arrivano mai ad equilibrio
- Equità → Tutti devono avere un accesso equo alle risorse
- Ottimizzazione → La rete deve operare in modo ottimale.

Ottimizzazione

- Ottimizzazione → La rete deve operare in modo ottimale.
  - Questo può essere in contrasto con il criterio precedente di Equità.
  - Equità vs Ottimizzazione:Fig: Se il traffico **AA'**, **BB'** , **CC'** è sufficiente a saturare il link orizzontale io ho ottimizzato il troughput globale a spese del traffico **XX'**

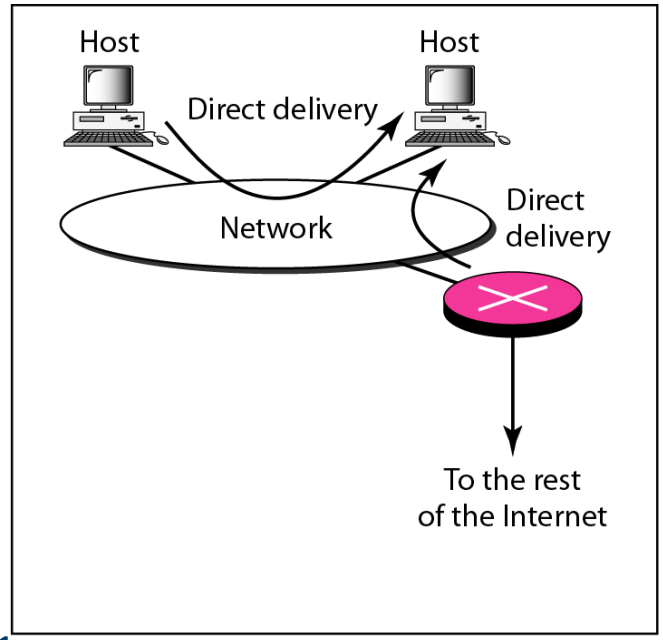


Ottimizzazione

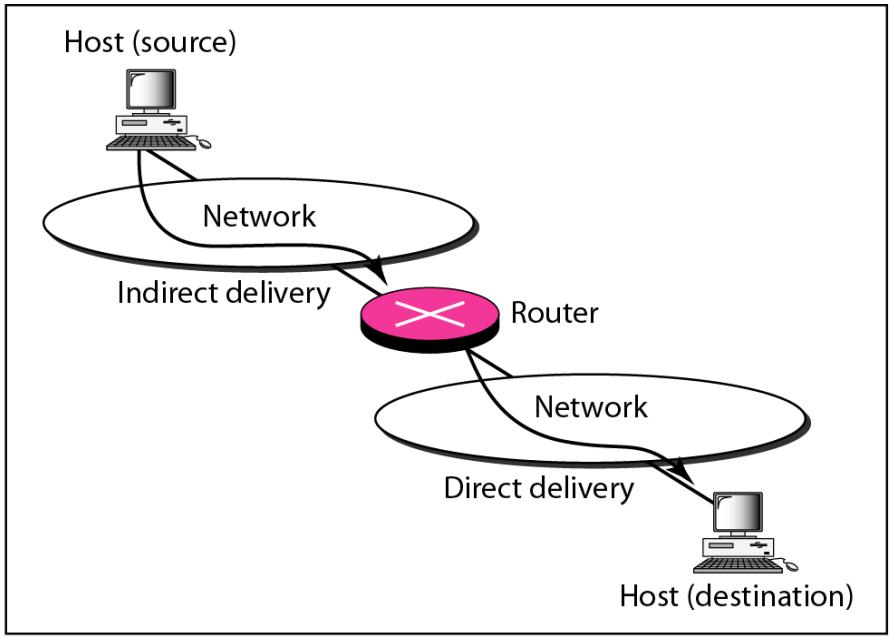
- Ottimizzazione → La rete deve operare in modo ottimale.
  - Devo anche decidere qale aspetto ottimizzare:
  - Minimizzo il packet delay?
  - Massimizzo il throughput?
  - Minimizzo il costo del servizio di trasporto? (es vado su una flat ADSL)
  - A volte si decide di minimizzare il numero di hops per minimizzare lo spreco di banda e questo aiuta a minimizzare il delay e ad usare il massimo throughput

Consegna indiretta

- Quando mittente e destinazione sono nella stessa rete, il mittente può consegnare direttamente il pacchetto
- Altrimenti, se la consegna è indiretta si deve cercare nella tabella di routing l'indirizzo del prossimo router intermedio



a. Direct delivery



b. Indirect and direct delivery

Dimensioni routing table

- Come posso mettere nella routing table le informazioni per raggiungere tutte le possibili destinazioni di internet?
- Next-Hop. Non serve mettere le informazioni relative al percorso completo. Sufficiente specificare come arrivare al Next Hop

a. Routing tables based on route

Destination	Route
Host B	R1, R2, host B

Routing table for host A

Destination	Route
Host B	R2, host B

Routing table for R1

Destination	Route
Host B	Host B

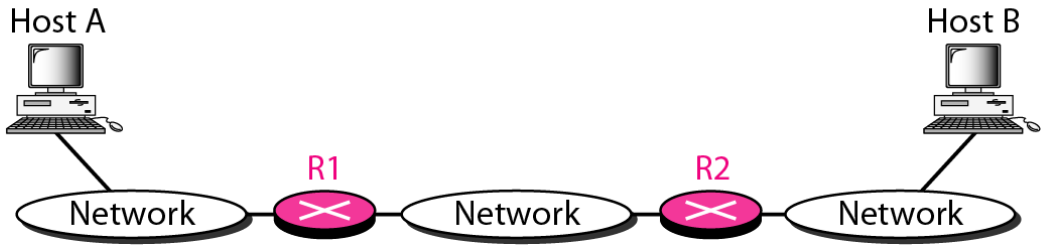
Routing table for R2

b. Routing tables based on next hop

Destination	Next hop
Host B	R1

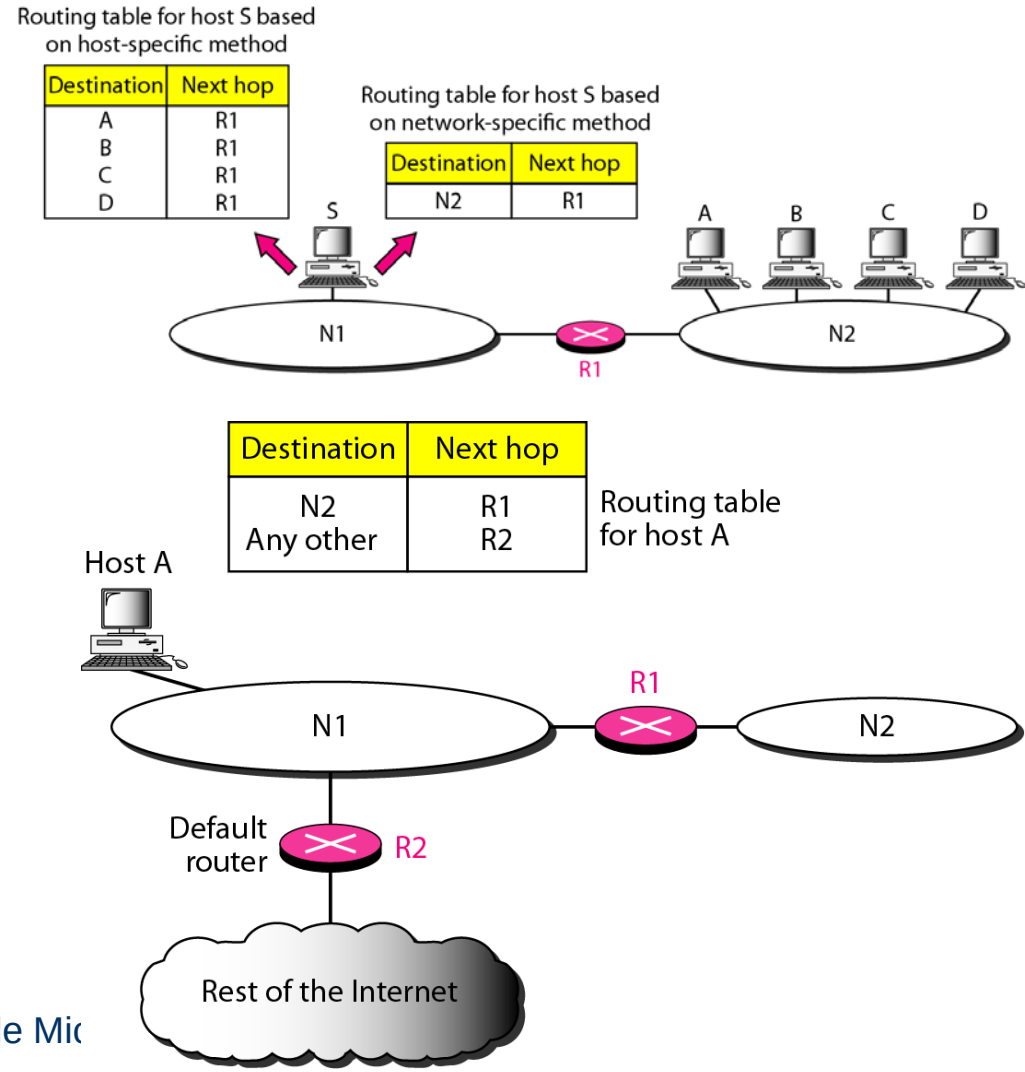
Destination	Next hop
Host B	R2

Destination	Next hop
Host B	---



Aggregazioni e default route

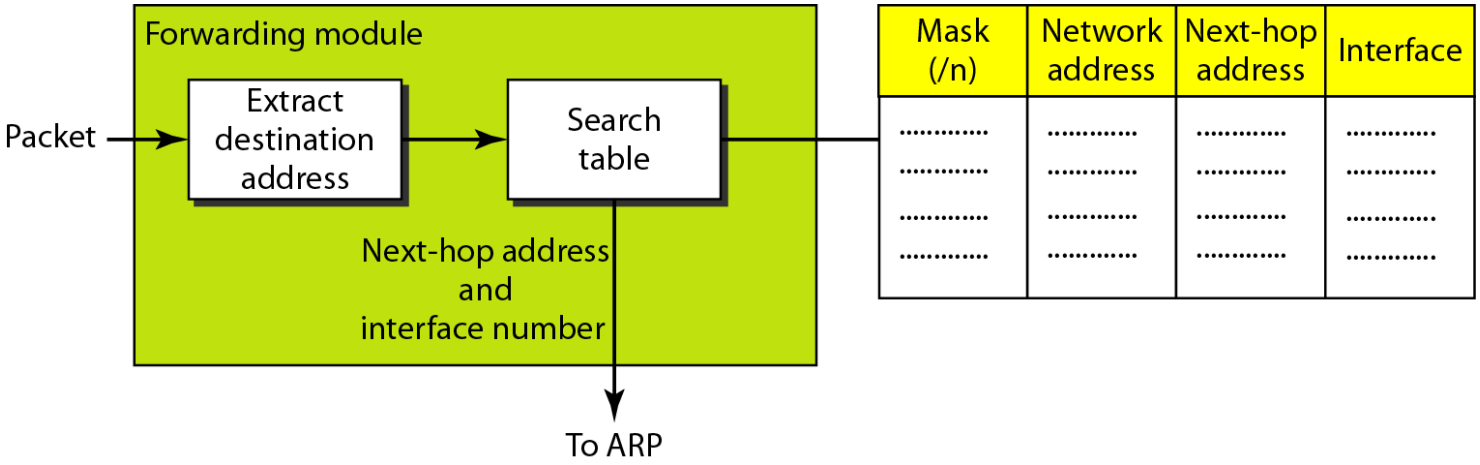
- Non occorre che io metta tutte i possibili host di destinazione nella routing table
- Basta che io inserisca le info su come raggiungere la rete per tutti gli host di quella rete
- Potrei anche specificare solo alcune reti e dare per tutte le altre reti l'indirizzo di una default route su cui saranno inoltrati tutti i pacchetti che non hanno una router



Routing Table in IP

- Se uso indirizzi Classless devo avere per ogni entry almeno 4 informazioni (4 colonne)

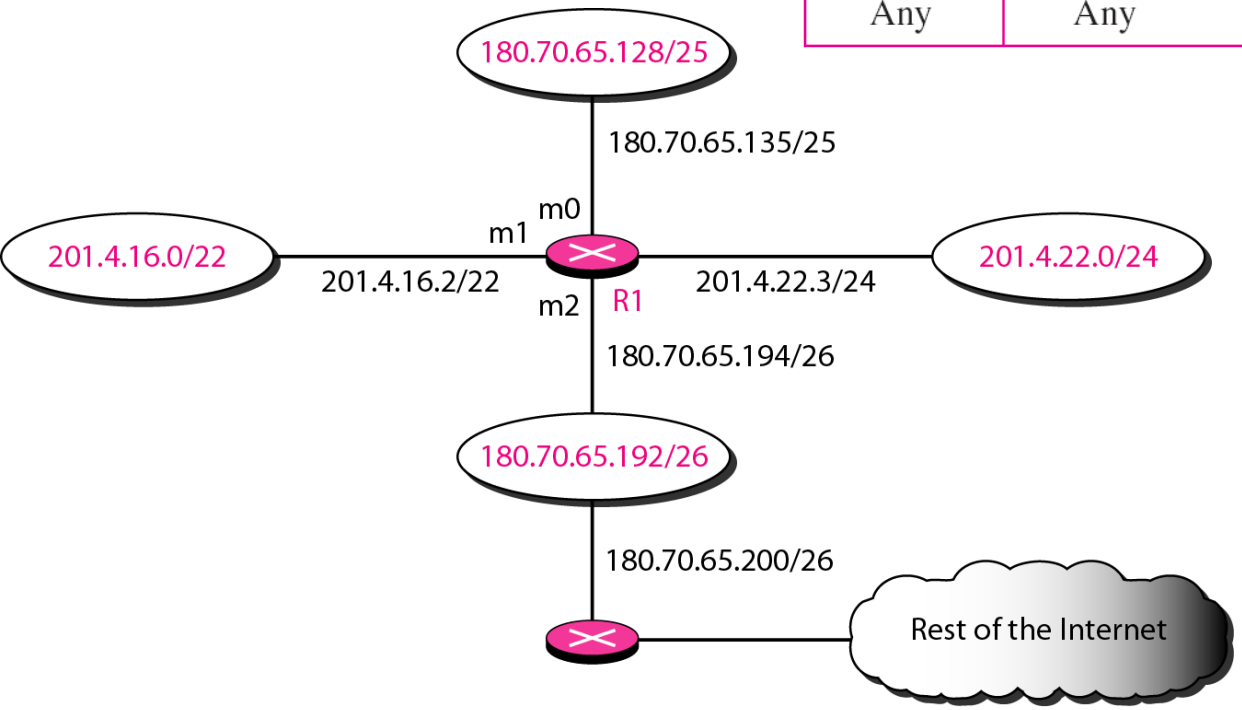
1) la maschera
2) l'indirizzo del blocco di indirizzi di destinazione
3) il next hop (l'indirizzo del prossimo router)
4) l'interfaccia da usare



Esempio

- Costruire la Routing Table per la rete in figura

Mask	*Network Address*	*Next Hop*	*Interface*
/26	180.70.65.192	—	m2
/25	180.70.65.128	—	m0
/24	201.4.22.0	—	m3
/22	201.4.16.0	....	m1
Any	Any	180.70.65.200	m2



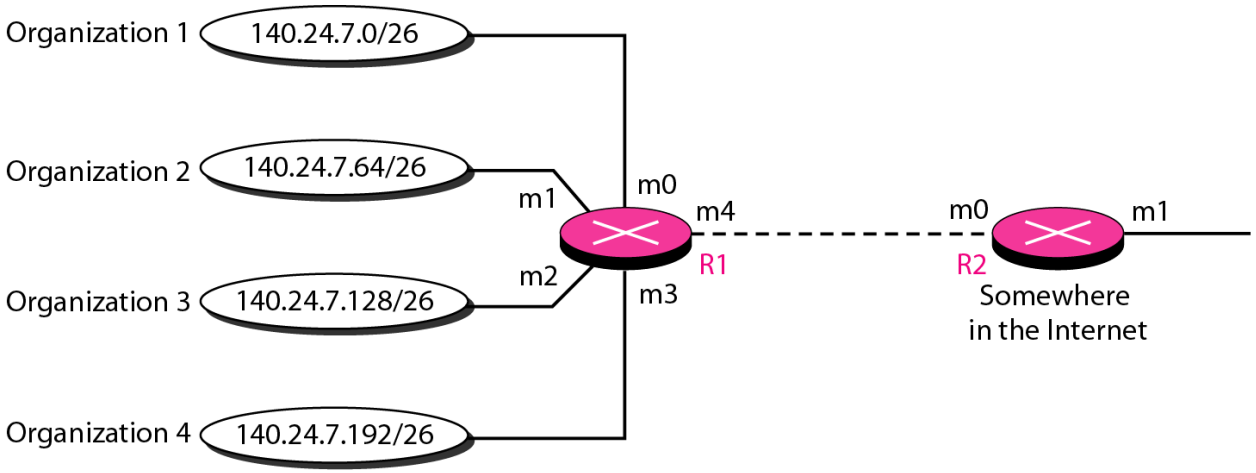
- Dove va un pacchetto che arriva a R1 con destinazione
- 180.70.65.140?
- 201.4.22.35
- 18.24.32.78

Risposta

1. Applico la mask di **26 bit** a 180.70.65.140 ottengo 180.70.65.128 che **non corrisponde** alla prima riga. Applico allora la mask a **25 bit** e ottengo **180.70.65.128 che corrisponde** alla seconda riga. Mando il pacchetto verso **m0** (con una ARP)
2. Prima riga 201.4.22.35/26 ottengo 201.4.22.0 e poi , seconda riga 201.4.22.35/25 ottengo 201.4.22.0. Nessun match. Con la terza riga **201.4.22.35/24** invece ho un **“match”.** Mando il pacchetto a **m3** (via ARP)
3. **18.24.32.78** [/26, /25, /24] non ho un “match” quindi uso **next hop di default**

Aggregazione

- L'aggregazione mi permette di indicare con l'indirizzo di rete tutte gli host in quella rete la posso usare anche per aggregare reti che hanno lo stesso next hop, come fa R2 in figura



Mask	Network address	Next-hop address	Interface
/26	140.24.7.0	----------	m0
/26	140.24.7.64	----------	m1
/26	140.24.7.128	----------	m2
/26	140.24.7.192	----------	m3
/0	0.0.0.0	Default	m4

Routing table for R1

Mask	Network address	Next-hop address	Interface
/24	140.24.7.0	----------	m0
/0	0.0.0.0	Default	m1

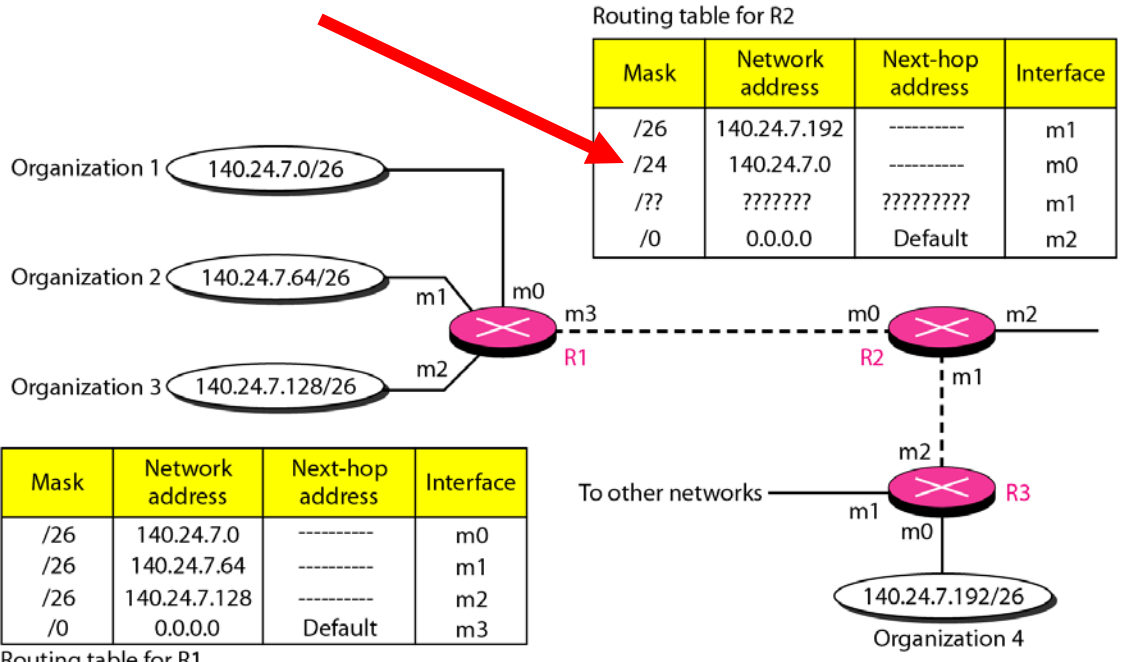
Routing table for R2

Longest mask match

- Anche se gli indirizzi aggregabili non condividono il next hop **posso comunque aggregare**
- Basta che la **tabella** di **routing** sia **ordinata mettendo prima le maschere più lunghe**
- Es per R2
- NB la riga 2 ha un match errato per la rete di “Organization 4” ma dal momento che il match viene fatto nella prima riga con la mask più lunga non ci sono problemi

m0? Ok per Org 1,2 e 3 ma Match sbagliato per Org4:

dovrebbe essere m1



Routing table for R2

Mask	Network address	Next-hop address	Interface
/26	140.24.7.192	----------	m1
/24	140.24.7.0	----------	m0
/??	???????	?????????	m1
/0	0.0.0.0	Default	m2

Mask	Network address	Next-hop address	Interface
/26	140.24.7.0	----------	m0
/26	140.24.7.64	----------	m1
/26	140.24.7.128	----------	m2
/0	0.0.0.0	Default	m3

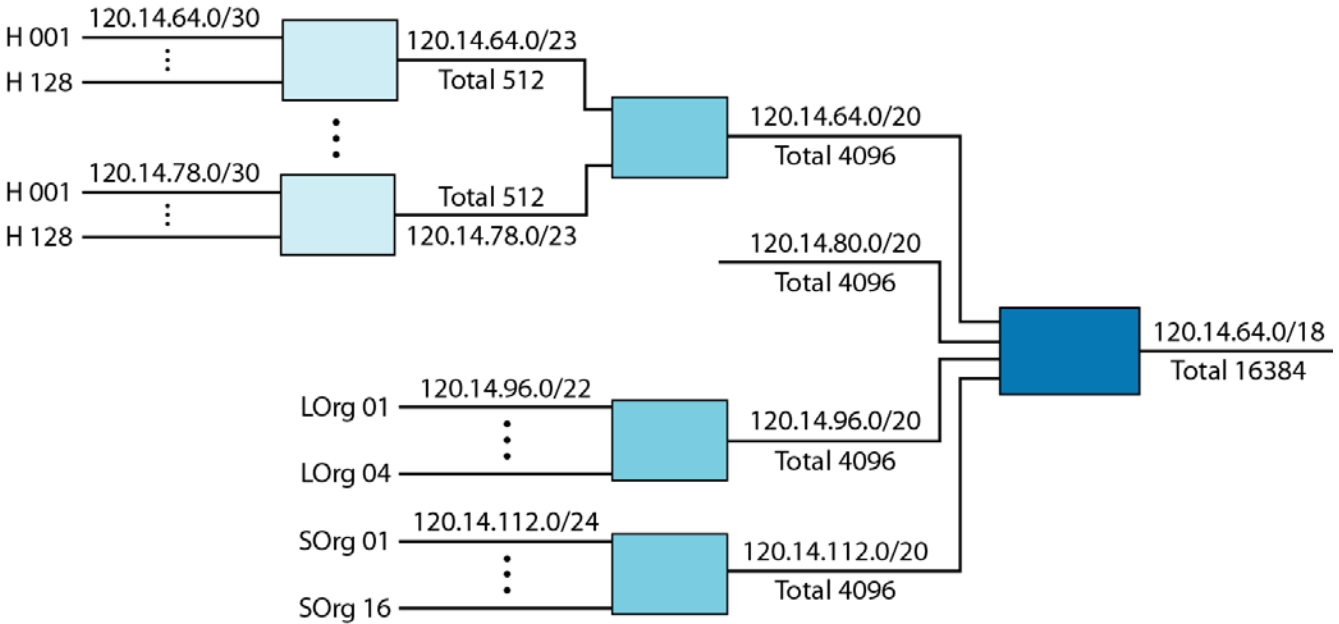
Routing table for R1

Mask	Network address	Next-hop address	Interface
/26	140.24.7.192	----------	m0
/??	???????	?????????	m1
/0	0.0.0.0	Default	m2

Routing table for R3

Routing gerarchico

- Questo permette di fare routing gerarchico
- Es questo ISP ha un blocco di indirizzi che poi spezzetta in diversi livelli. Il resto di internet non deve saperlo. Deve solo sapere come arrivare all'ISP
- Purtroppo spesso una organizzazione ha indirizzi spezzettati, altrimenti si potrebbe ridurre ulteriormente usando routing geografico
- IPv6 ha un blocco di indirizzi riservato per il routing geografico

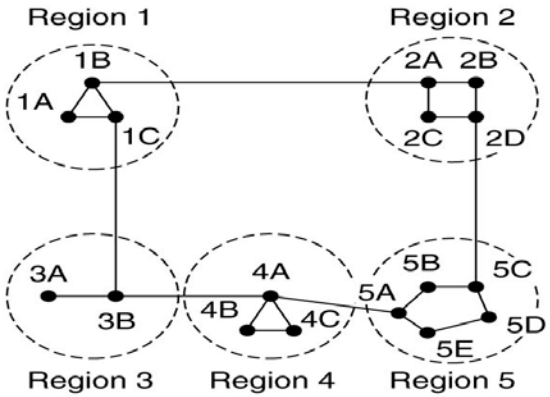


Routing gerarchico

- Al crescere della rete aumentano le tabelle di routing.
- Non solo viene usata più memoria ma anche tempo di CPU e bandwidth
- Ad un certo punto si deve passare a routing gerarchico
- I router vengono divisi in regioni. Ogni router conosce i dettagli interni della sua regione ma non sa nulla della struttura interna delle altre.
- Il router deve solo sapere come raggiungere il router che fa da tramite verso la regione che vuole raggiungere
- Per reti molto grandi una gerarchia a due livelli potrebbe non essere sufficiente.
- Al crescere del numero di regioni rispetto al numero di router per regione il risparmio in termini di spazio per le tabelle aumenta

Inefficienza

- Il risparmio non viene gratis
- Nella figura la migliore route da 1A a 5C sarebbe attraverso la regione 2 ma con il routing gerarchico tutto il traffico tra 1 e 5 passa dalla regione 3 che è quella ottimale per la maggior parte delle destinazioni nella regione 5



(a)

Full table for 1A

Dest.	Line	Hops
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

(b)

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

(c)

Formato tabella routing

- Abbiamo visto che deve avere almeno 4 colonne ma dipende dal costruttore del router che può aggiungere altre info. Vediamo un tipico esempio:
- Mask
- Indirizzo di destinazione
- Indirizzo next hop
- Interface

Mask	Network address	Next-hop address	Interface	Flags	Reference count	Use
.....	.....	.....	.....	.....	.....	.....

Formato tabella routing

- Flag
  - **U** **U**p vuol dire che il Next Hop è funzionante
  - **G** **G**ateway il Next Hop è in un'altra rete, quindi consegna indiretta
  - **H** **H**ost specific. L'indirizzo specificato è quello di un host
  - **D** A**d**ded by redirection. Riga inserita in seguito ad un messaggio ICMP di redirezione
  - **M** **m**odified by redirection. Riga modificata in seguito ad un messaggio ICMP di redirezione
- Contatore: numero di utenti che stanno usando questo link (connessioni contemporanee)
- Pacchetti spediti. Mostra quanti pacchetti sono stati spediti alle destinazioni di questa riga

Netstat

- I comandi unix o window netstat permettono di vedere le informazioni di routing
- Es netstat -rn serve per avere la tabella di routing (-r) in formato numerico (-n)
- Gateway equivale a routing e indica il next hop
- Ifconfig invece serve per avere informazioni sull'interfaccia di rete

$ netstat -rn

Kernel IP routing table

Destination	Gateway	Mask	Flags	Iface
153.18.16.0	0.0.0.0	255.255.240.0	U	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	lo
0.0.0.0	153.18.31.254	0.0.0.0	UG	eth0

$ ifconfig eth0

eth0 Link encap:Ethernet HWaddr 00:B0:D0:DF:09:5D

inet addr:153.18.17.11 Bcast:153.18.31.255 Mask:255.255.240.0

. . .

Rete corrispondente

$ netstat -rn

Kernel IP routing table

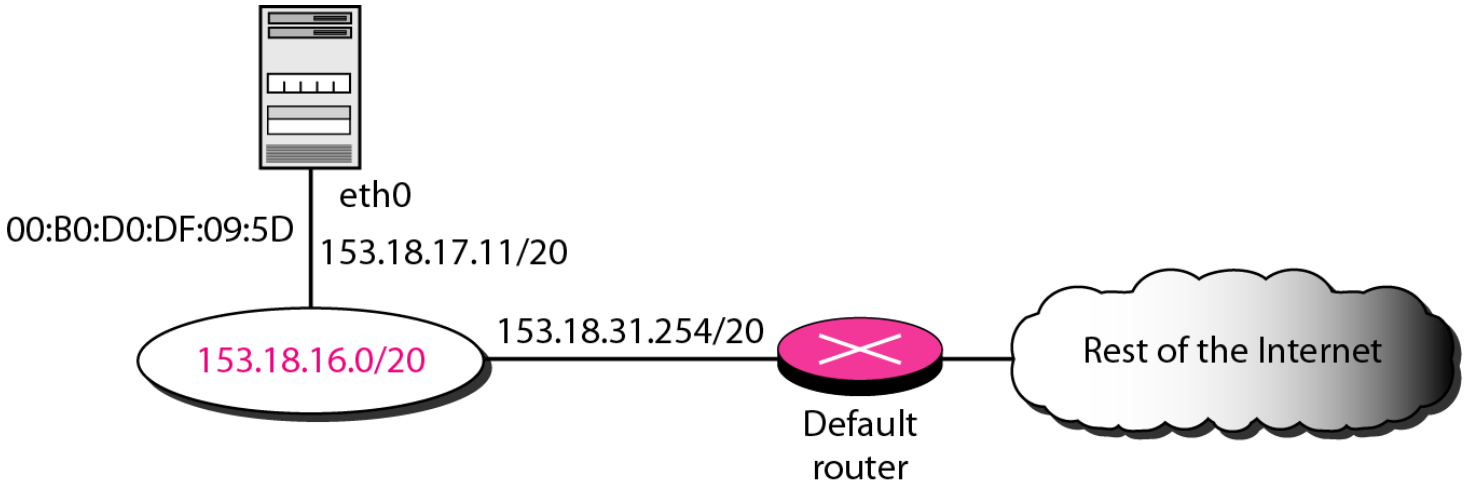
Destination	Gateway	Mask	Flags	Iface
153.18.16.0	0.0.0.0	255.255.240.0	U	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	lo
0.0.0.0	153.18.31.254	0.0.0.0	UG	eth0

$ ifconfig eth0

eth0 Link encap:Ethernet HWaddr 00:B0:D0:DF:09:5D

inet addr:153.18.17.11 Bcast:153.18.31.255 Mask:255.255.240.0

. . .



Tecniche di Routing

- Routing Statico
- Routing Dinamico
  - Algoritmi di tipo Distance Vector
  - Algortimi di tipo Link State

Non-adaptive algorithms

- Detto anche **Static Routing**:
- Non si basano su misure o stime del traffico
- Le decisioni vengono prese in anticipo, off-line e caricate dai router al bootstrap (da una flash memory)
- Sono detti algoritmi non adattivi
- Abbiamo quindi tabelle di routing statiche

Adaptive Algorithms

- Cambiano le regole di routing per riflettere cambiamente di topologia e di traffico
- Alcuni prendono informazioni solo dai router adiacenti mentre altri da tutti i router
- Usano diverse metriche
  - Distanze in km
  - Distanza in numero di hops
  - Stime del tempo di attraversamento
  - Bandwidth
  - Costi economici

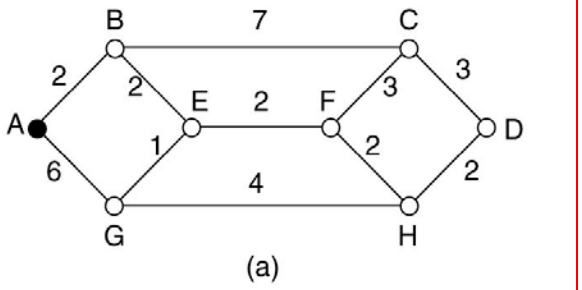
Shortest Path Routing

- Costruisco un grafo della subnet in cui ogni nodo è un router e ogni linea è il link tra i due router
- Per trovare la route migliore l'algoritmo cerca il path più corto (**shortest path**)
- La lunghezza del path la posso misurare in **hops** oppure in **lunghezza in km** dei link ma posso anche prendere **l'accodamento medio** o la **latenza** di un link, per cui il path più corto risulta essere il più veloce e non il più corto in kilometri

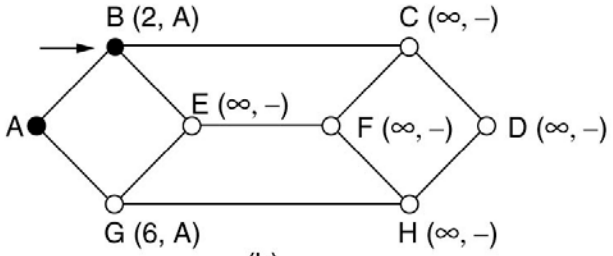
Dijkstra algorithm

- Si possono pensare diversi algoritmi per calcolare lo shortest path tra due nodi di un grafo
- Vediamo l'algoritmo di **Dijkstra** (1959) in cui ogni nodo ha come label la distanza dal nodo sorgente lungo il miglior path conosciuto
- All'inizio non so nulla per cui tutti i path sono a infinito e tutte le label sono provvisorie
- Quando scopro che una label è il path più corto possibile la label diventa permanente.

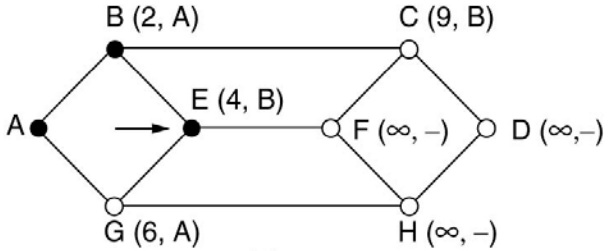
L'algoritmo in azione



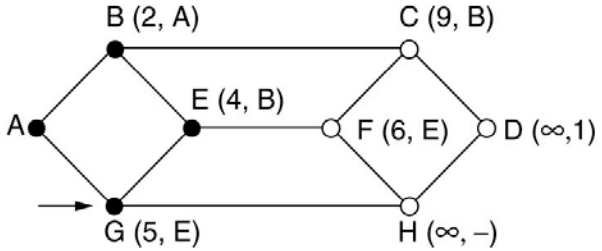
(a)



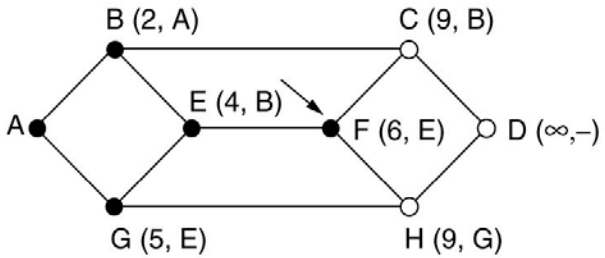
(b)



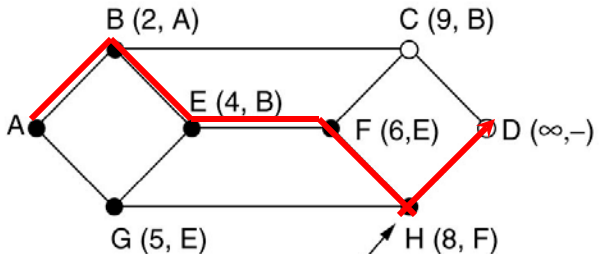
(c)



(d)



(e)



(f)

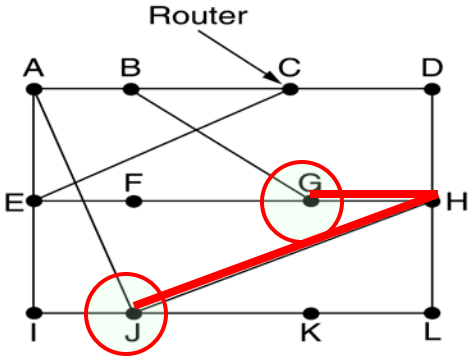
- Sto usando come metrica le distanze (figura **a**) e non gli hops
- In 5 steps trovo lo shortest path tra A e D
- La freccia indica il nodo in azione in un certo step

Distance Vector Routing

- Algoritmo di tipo dinamico
  - Noto anche come Bellman-Ford o Ford-Fulkerson, è stato usato all'inizio di Arpanet ed è ancora usato in Internet nel protocollo di routing **RIP**
  - Ogni router mantiene una tabella con le migliori distanze note per ogni destinazione e quale linea usare. La tabella viene aggiornata scambiando informazioni con i vicini
  - La tabella di ogni router è indicizzata per ogni router nella subnet ed ha una sola entry che contiene
  1. La linea preferita di uscita da usare per quella destinazione
  2. Una stima del tempo o della distanza verso quella destinazione (metriche possibili: hops, latenza, pacchetti in coda). Si assume che il router sia in grado di acquisire queste metriche

Aggiornamento di J

- J riceve input da A,I,H,K che sono i suoi primi vicini
- Questi mandano la lista di tutte le distanze a loro note nella metrica del delay (latenza)
- Come fa J a decidere come andare a **G**?
- Ci mette 18+8=26 via A, 31+10=41 via I, **6+12=18** via **H** e 31+6=37 via K
- quindi decide di mettere in **G** una entry con **[18,H]**



(a)

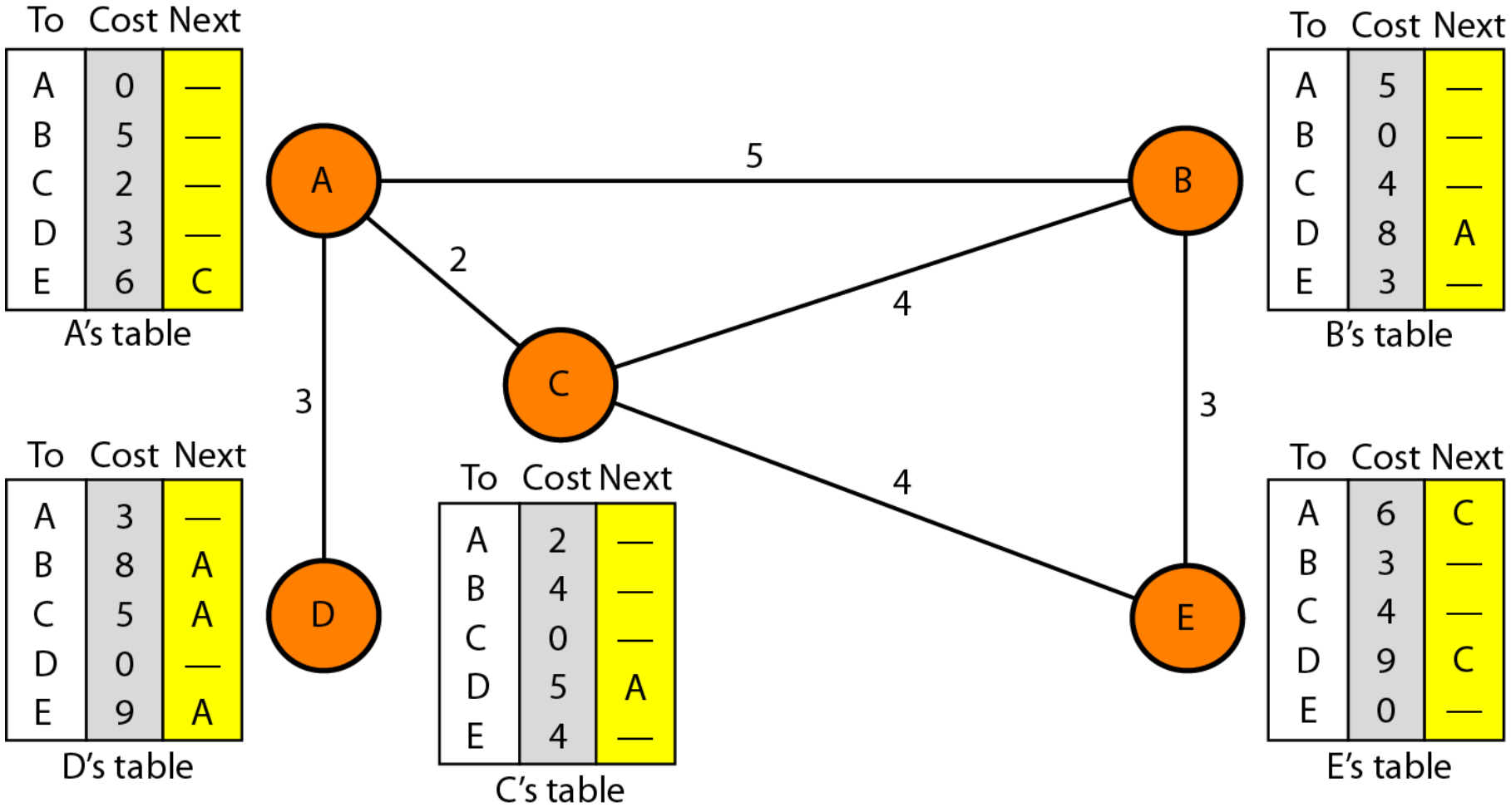
New estimated delay from J

To	A	I	H	K	Line
A	0	24	20	21	8 A
B	12	36	31	28	20 A
C	25	18	19	36	28 I
D	40	27	8	24	20 H
E	14	7	30	22	17 I
F	23	20	19	40	30 I
G	18	31	6	31	18 H
H	17	20	0	19	12 H
I	21	0	14	22	10 I
J	9	11	7	10	0 –
K	24	22	22	0	6 K
L	29	33	9	9	15 K
	JA delay is	JI delay is	JH delay is	JK delay is	New routing table for J
	8	10	12	6	

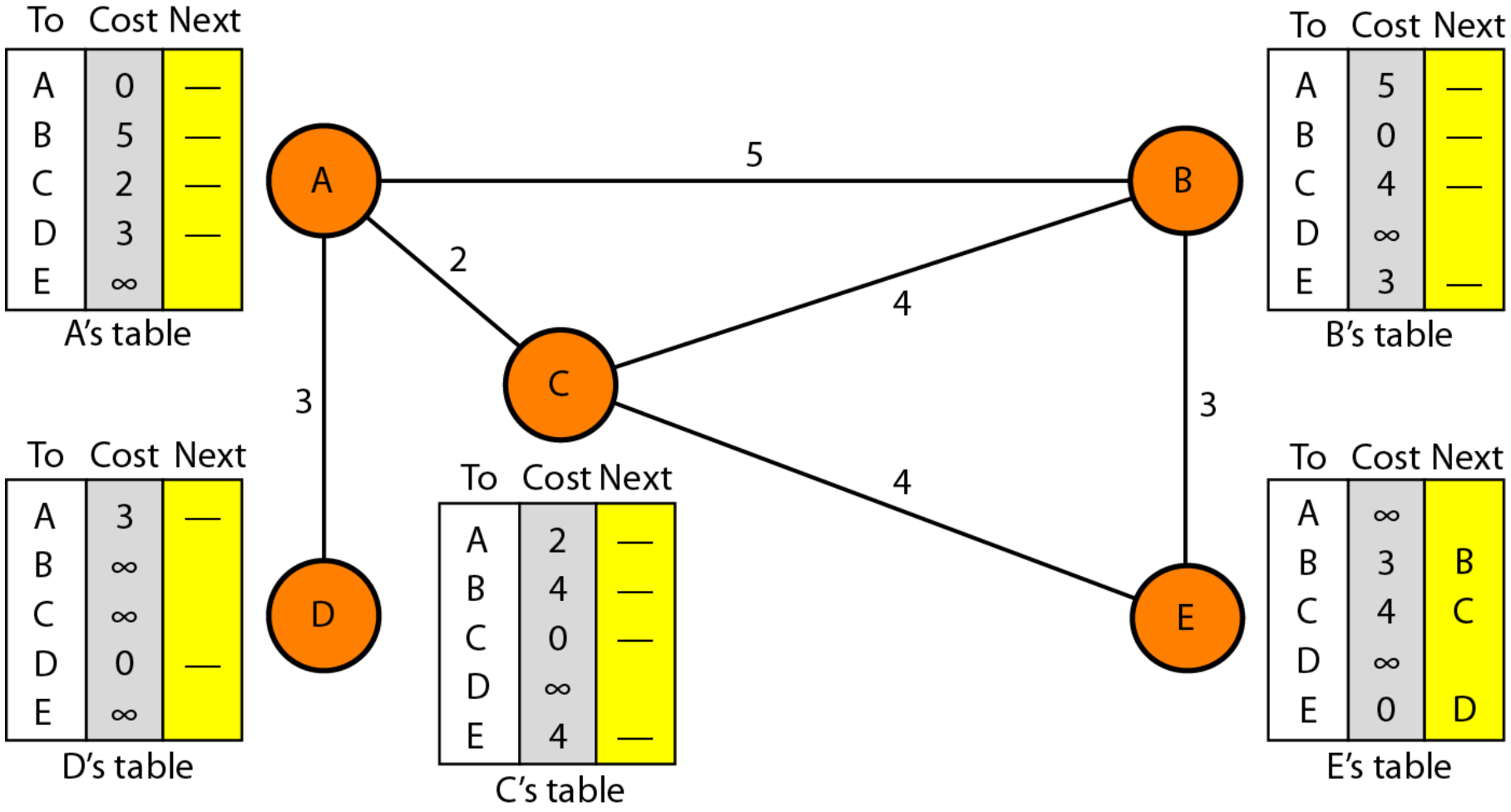
Vectors received from J's four neighbors

(b)

Esempio

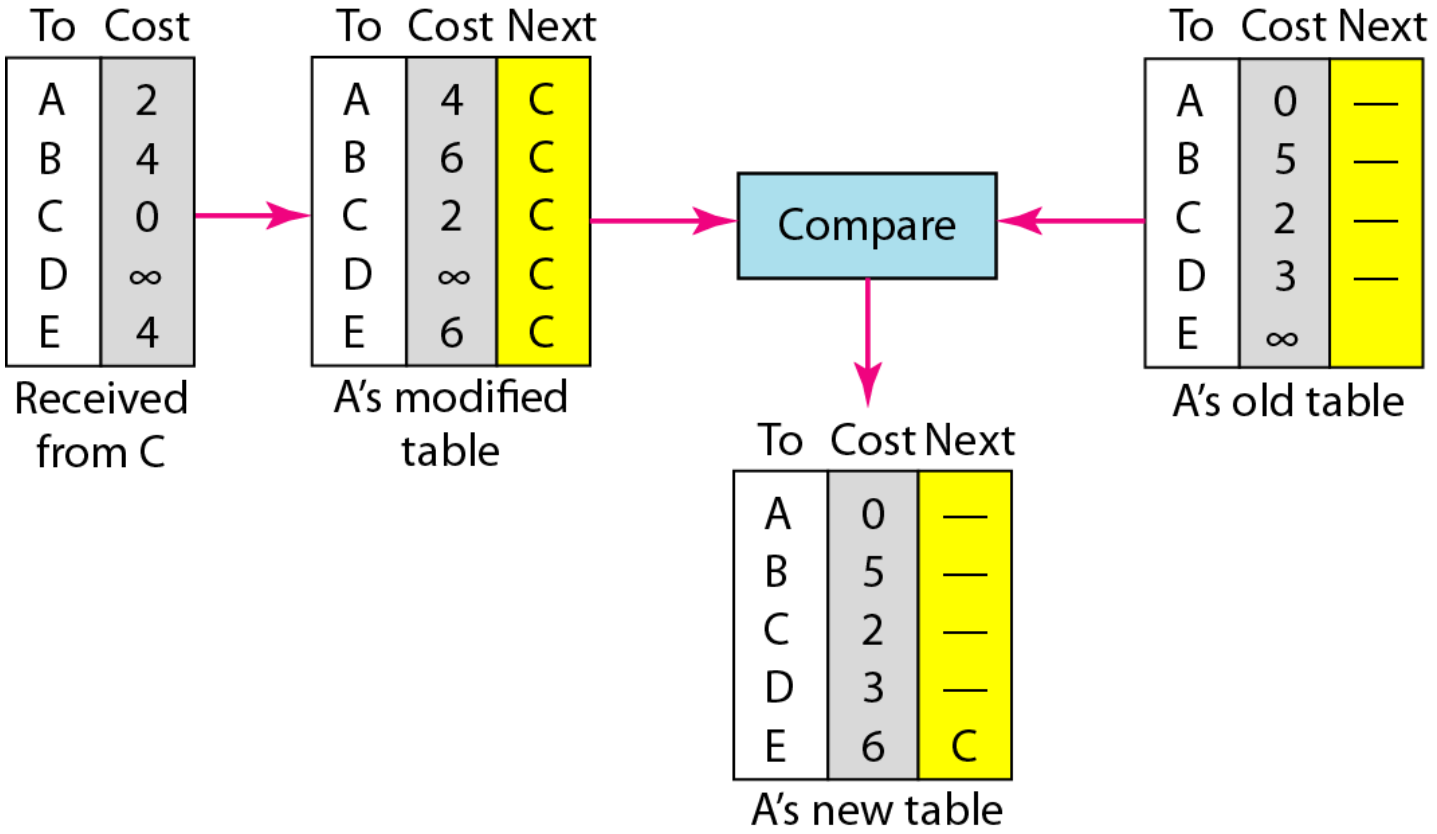

To Cost Next
A 0 —
B 5 —
C 2 —
D 3 —
E 6 C
A's table
To Cost Next
A 5 —
B 0 —
C 4 —
D 8 A
E 3 —
B's table
To Cost Next
A 3 —
B 8 A
C 5 A
D 0 —
E 9 A
D's table
To Cost Next
A 2 —
B 4 —
C 0 —
D 5 A
E 4 —
C's table
To Cost Next
A 6 C
B 3 —
C 4 —
D 9 C
E 0 —
E's table
A
B
C
D
E
5
2
4
3
3
4

Inizializzazione



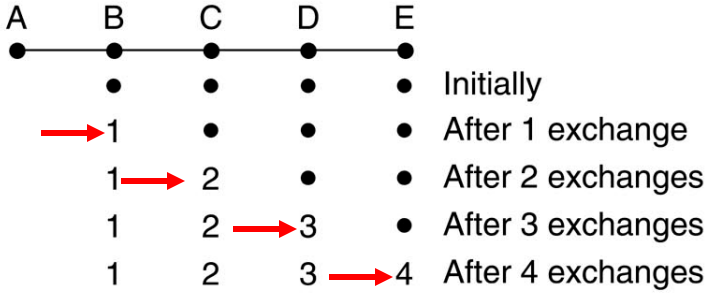
Update

- A riceve da C le nuove informazioni e le usa per aggiornare la sua tabella

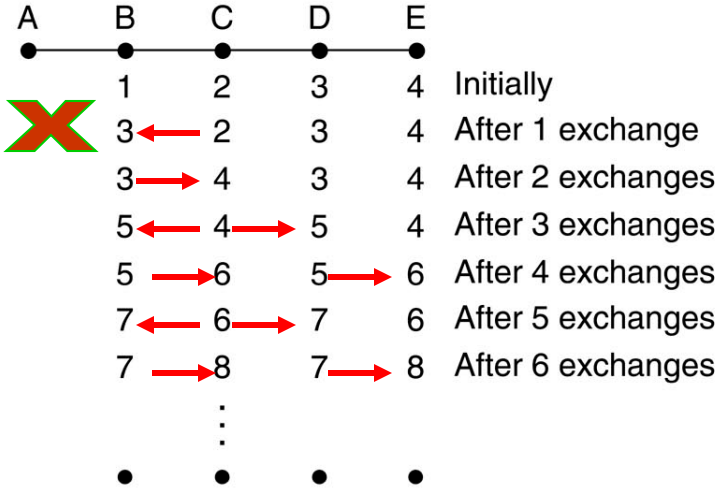


Conteggio all'infinito

- Le buone notizie viaggiano velocemente: **A torna attivo**, dopo N scambi se ci sono al massimo N hops tutti sanno che A è tornato in vita
- **A muore**: Le cattive notizie sono lentissime. Nessun router ha un valore più di uno maggiore del minimo di tutti i suoi vicini
- B non vede più A quindi prende informazioni da C che si ricorda 2 (B non sa come C vede A, magari ha diversi link verso A, non sa che lo vede proprio via B)



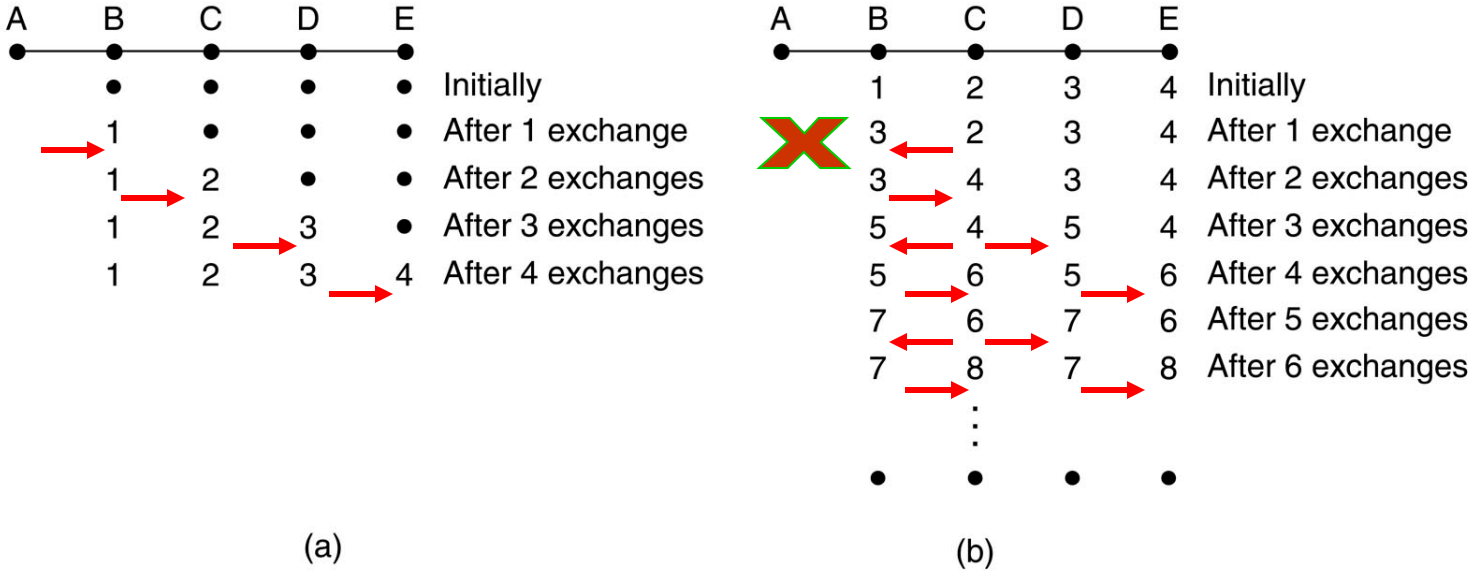
(a)



(b)

L’infinito finito

- Gradualmente tutti i router tendono verso infinito ma il numero di scambi dipende dal numero che si vuole prendere come infinito
- Per cui è meglio mettere come infinito un numero grande ma finito, come il longest path + 1 se uso gli hops oppure un tempo molto alto se uso la latenza



Split horizon

- Un nodo non spedisce **tutta** la tabella ai suoi vicini.
- Infatti se B pensa che per andare ad X si deve passare da A allora quando spedisce le sue info ad A non gli manda la route per arrivare a X
- Questo è ragionevole perché sono info ricevute da A, non ha senso rimandargliele indietro e in più risolve il problema dell'infinito che deriva proprio dal fatto che B riceve un'informazione da A, la modifica e la rispedisce ad A

Link State Routing

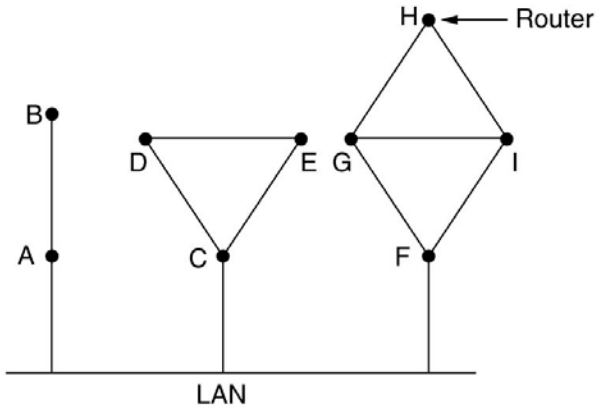
- Ha rimpiazzato Distance Vector dopo il 1979 per due problemi di quest'ultimo
  - La metrica (nelle implementazioni correnti) era la queue length e non teneva quindi conto delle bandwidth delle linee. Nessun problema quando sono tutte uguali e a bassa bandwidth ma lo divenne quando alcune linee raggiunsero i 230 kbps o 1.544 Mbps
  - L'algoritmo ci metteva troppo a convergere (conteggio all'infinito)

Un router deve:

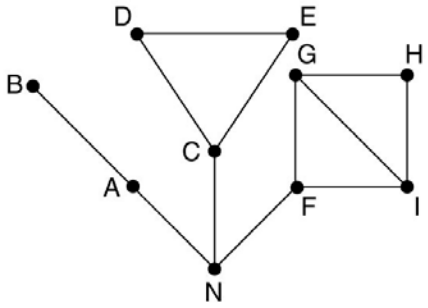
1. Scoprire i suoi vicini e imparare il loro indirizzo di rete
2. Misurare il delay o il costo verso ciascun vicino
3. Costruire un pacchetto che dice a tutti cosa ha imparato
4. Mandare questo pacchetto a tutti gli altri router
5. Calcolare il shortest path verso ogni router
6. Costruire le tabelle di routing

Conoscere i vicini

- Al boot il router vuole conoscere i suoi vicini.
- Manda loro un pacchetto HELLO su ogni linea punto-punto.
- Il router all'altro capo risponde dicendo chi è
- I nomi devono essere globalmene unici perché quando un router sente che tre router sono collegati a F deve sapere che tutti i tre si riferiscono allo stesso F
- Se due o più router sono collegati ad una LAN posso modellare la LAN come un router virtuale. Il fatto che posso andare da A a C via LAN lo modello come un path A*N*C



(a)



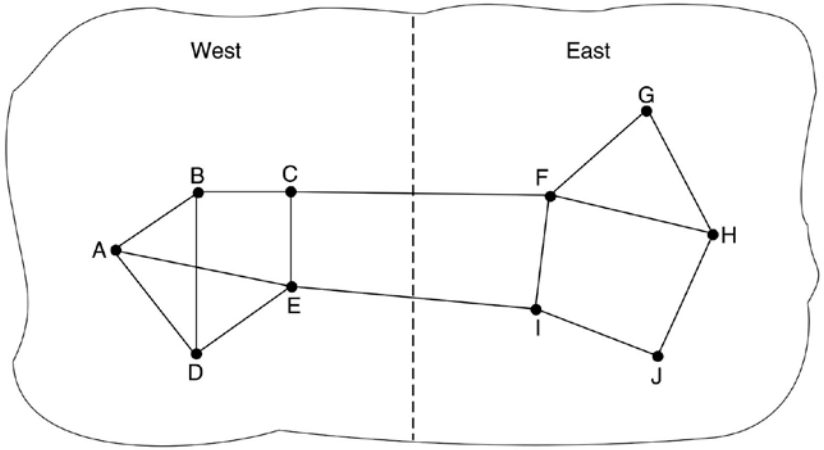
(b)

Misurare i costi

- Ogni router deve conoscere o stimare il delay verso ognuno dei suoi vicini
- Lo può fare mandando un pacchetto di ECHO che dall'altra parte viene rimandato immediatamente, misurando round trip time diviso per due ho una buona stima del delay
- Posso ripetere la misura e prendere una media
- Questo implica che i delay siano simmetrici ma questo non è sempre vero

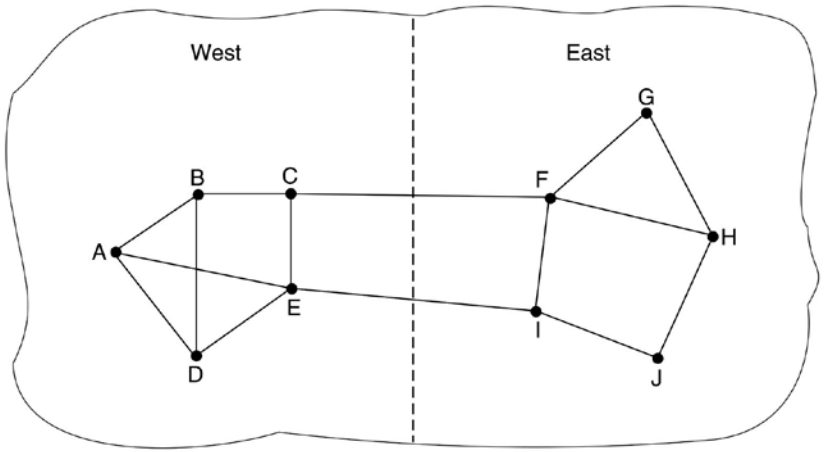
Tengo conto del carico?

- Devo tenere presente il carico? Comincio a misurare quando il pacchetto di ECHO entra in coda o quando arriva alla testa della coda?
  - Se tengo conto dei ritardi indotti dal traffico, un router quando deve scegliere tra due link con la stessa bandwidth sceglie sempre quello meno carico come shortest path → migliori performance
  - Se ho due link e uno (CF) è carico, scelgo il secondo (EI) → tutto il traffico va nel secondo e quindi il primo la prossima volta diventa quello scarico per cui un router continua ad oscillare tra un link e l'altro e quindi ad avere un routing erratico.



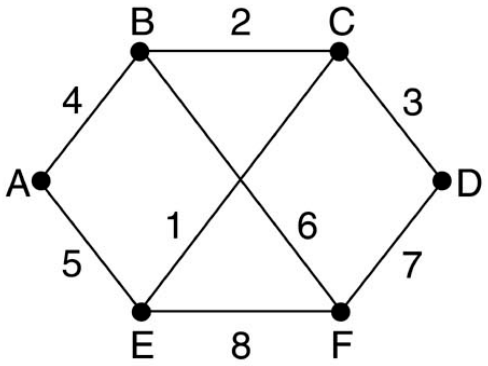
Tengo conto del carico?

- Se ignoro il carico il problema non si pone; oppure metto il carico bilanciato su entrambe le linee ma allora non ho il link migliore
- Per non avere oscillazioni nella scelta del miglior path posso distribuire il carico su diverse linee con una frazione nota su ogni linea



Creo i Link State Packets

- Ora che ho le informazioni mi costruisco i pacchetti
- Il pacchetto contiene il mittente, un numero di sequenza e anzianità e la lista dei vicini.
- Per ogni vicino inserisco il delay verso quel vicino
- Fare i pacchetti è facile. Il difficile è decidere quando. Li costruisco periodicamente o solo quando succede qualcosa di significativo? Es. Una linea (o un vicino) scompare (o ricompare) o alcune metriche cambiano significativamente?



(a)

Link State Packets

A	
Seq.	
Age	
B	4
E	5

B	
Seq.	
Age	
A	4
C	2
F	6

C	
Seq.	
Age	
B	2
D	3
E	1

D	
Seq.	
Age	
C	3
F	7

E	
Seq.	
Age	
A	5
C	1
F	8

F	
Seq.	
Age	
B	6
D	7
E	8

(b)

Distribuzione dei pacchetti

- E’ la parte più delicata perché i router che ricevono i primi pacchetti possono cambiare le loro routes, quindi abbiamo router con diverse versioni della topologia, inconsistenze, loop e macchine non raggiungibili
- Ogni pacchetto viene mandato in **flooding verso tutte le linee** esclusa quella di provenienza
- Ogni pacchetto ha un numero di sequenza che viene incrementato per ogni pacchetto mandato. Quando arriva un nuovo pacchetto Link State (LSP) se è nuovo viene forwardato, ma se è un duplicato viene scartato, se arriva con un numero più basso di quello in uso viene scartato come obsoleto.

Problemi

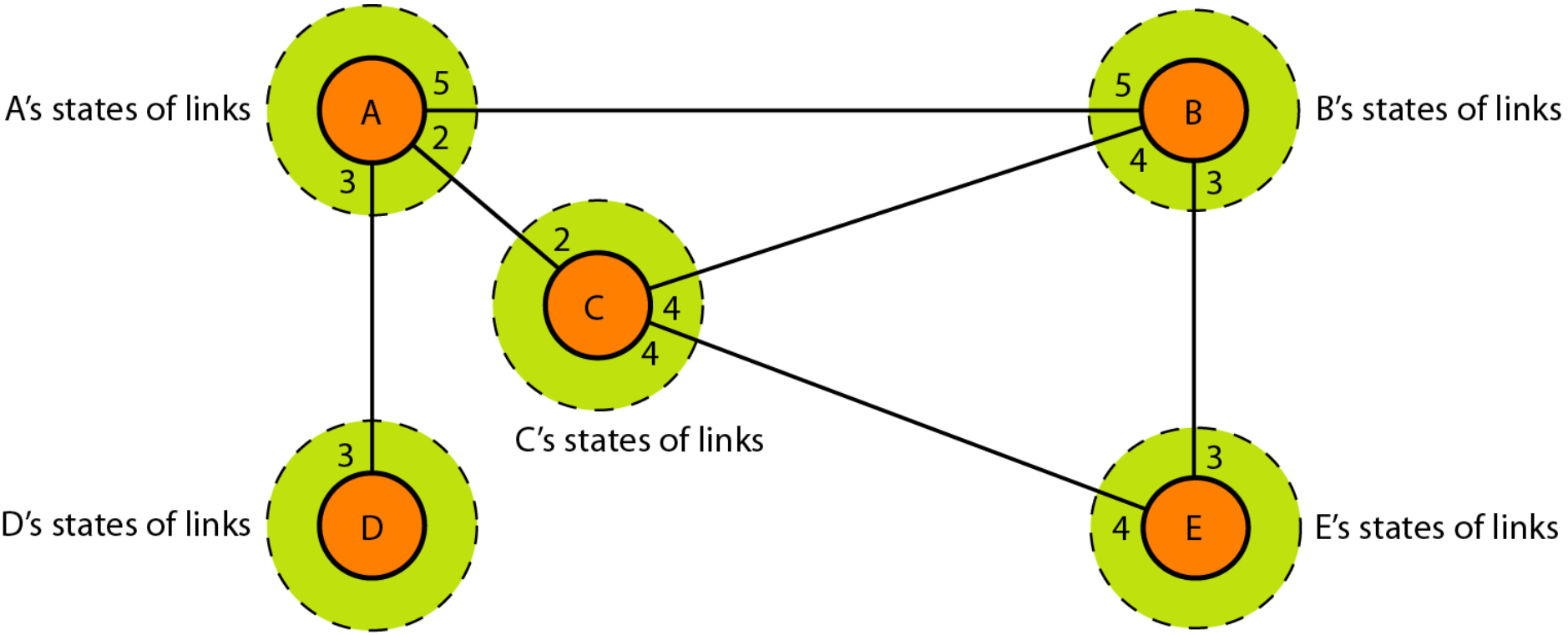
- Se il numero di sequenza va in overflow tutti si confondono. Quindi uso numeri a 32 bit che con un LSP al secondo ci mette 137 anni a wrappare
- Se un router crasha perde traccia della sequenza e ricomincia da 0. Il prossimo pacchetto viene scartato come duplicato
- Se un numero di sequenza si corrompe e per esempio ricevo 65540 (0x1004) invece che 4 (0x0004) i pacchetti da 5 a 65540 vengono buttati come obsoleti
- Soluzione. Metto un campo “Age” a 60 che viene decrementato ogni secondo. Quando arriva a zero le informazioni di quel router vengono scartate. Di solito un pacchetto arriva ogni 10 secondi per cui le informazioni vanno in timeout solo se il router è down o perde sei pacchetti di fila

Calcolo le nuove routes

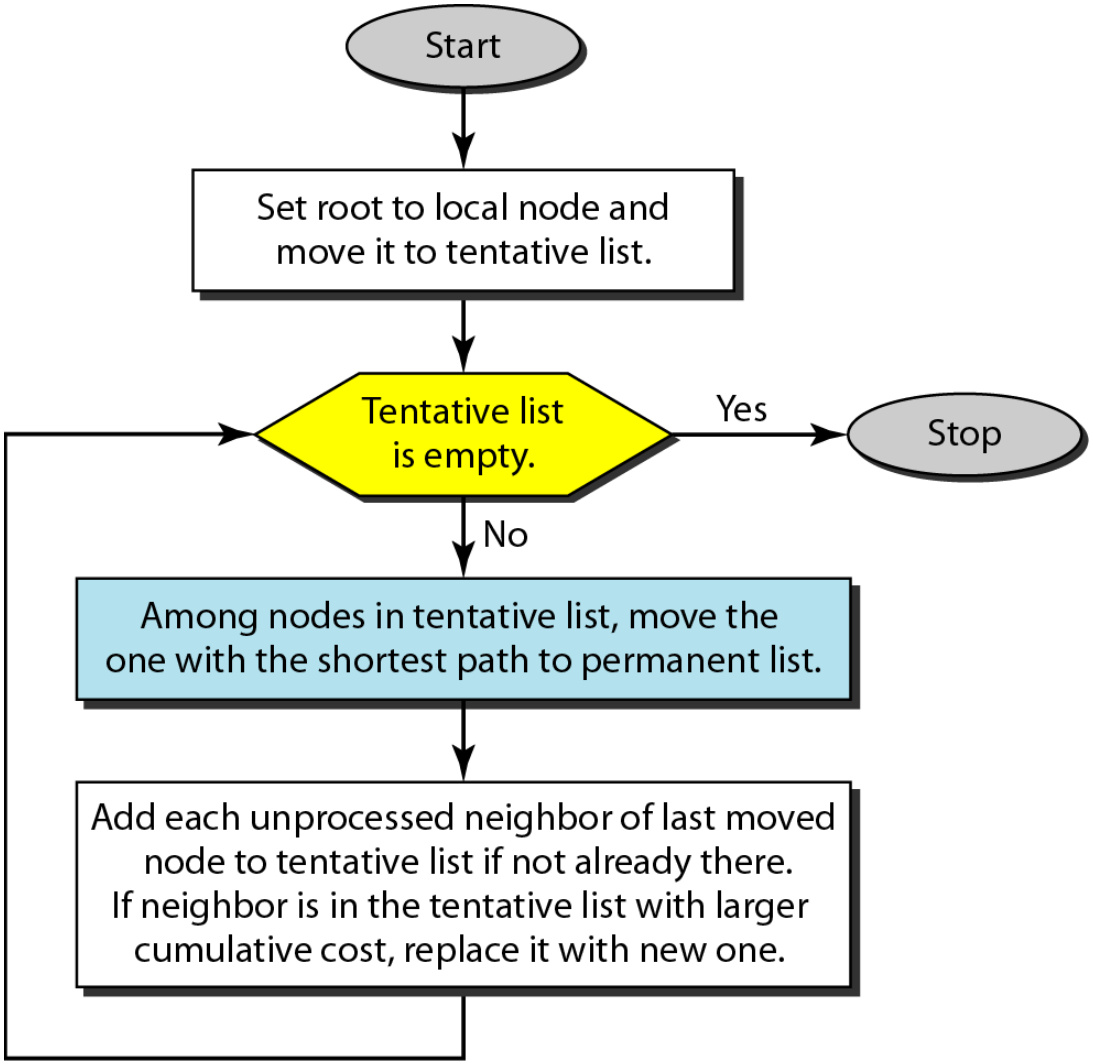
- Quando il router ha tutto un insieme di LSP può costruirsi l'intero grafo
- Ogni link è rappresentato due volte, una per ogni direzione e posso fare una media o usare i due valori separatamente
- Posso quindi usare l'algoritmo di Dijkstra per calcolarmi lo Shortest Path per ogni destinazione. Mi creo la nuova routing table e ricomincio le normali operazioni
- Il protocollo **OSPF** molto usato in Internet usa Link State Routing
- Un altro protocollo LSR è IS-IS (Intermediate System – Intermediate System) progettato per DECnet e adottato in seguito da ISO per il protocollo connectionless CLNP. Ora IS-IS viene usato anche in qualche backbone IP

Esempio

- All'inizio ho una conoscenza parziale dei singoli nodi, misurata in qualche modo, secondo qualche criterio



Dijkstra's algorithm

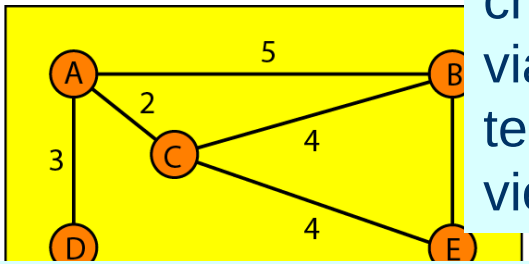


L'algoritmo in

C ha la distanza minore tra D,C, B per cui diventa permanente

Inseriamo tra i temporanei i vicini di C, A che è già finale, B che però è a distanza inferiore via A ed E quindi rimane temporaneo a distanza 5 e viene inserito solo E

Nodo A è la radice, unico tra i temporanei e a distanza 0, quindi diventa nodo finale



temporanei A

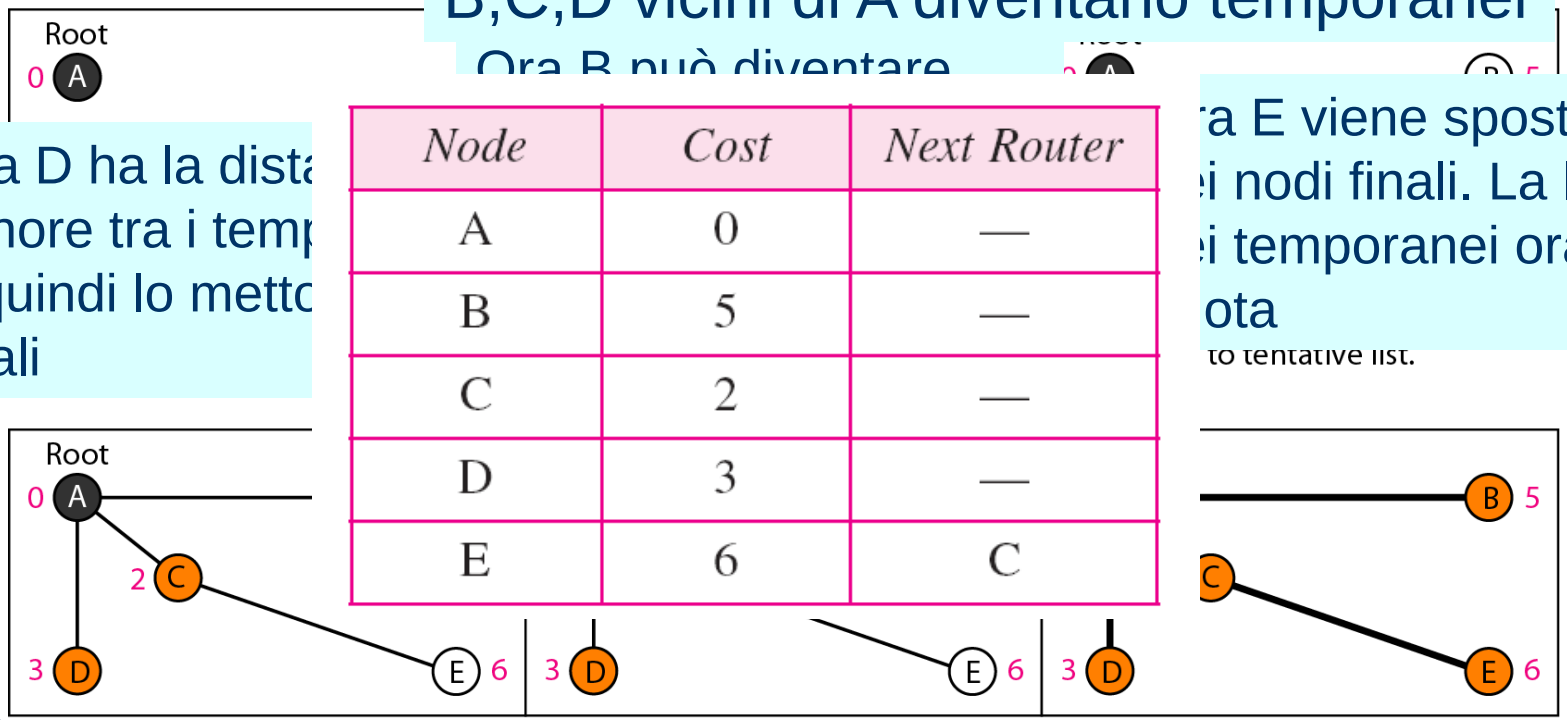
B,C,D vicini di A diventano temporanei

Ora B può diventare

Ora D ha la dista
minore tra i temp
e quindi lo metto
finali

ra E viene spostato
i nodi finali. La lista
i temporanei ora è
ota

Node	*Cost*	*Next Router*
A	0	—
B	5	—
C	2	—
D	3	—
E	6	C



Nodo A è la radice, unico tra i temporanei e a distanza 0, quindi diventa nodo finale

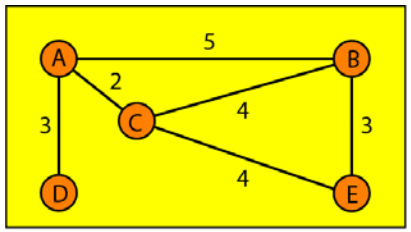
Nodi finali nessuno, temporanei A

Ora D ha la distanza minore tra i temporanei e quindi lo metto tra i finali

Ora B può diventare finale e si considerano i suoi vicini, di nuovo E che si raggiunge a 5+1 attraverso B

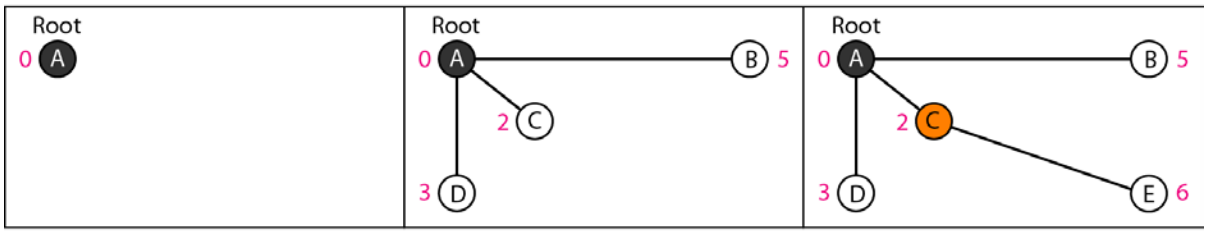
C ha la distanza minore tra D,C, B per cui diventa permanente

Inseriamo tra i temporanei i vicini di C, A che è già finale, B che però è a distanza inferiore via A ed E quindi rimane temporaneo a distanza 5 e viene inserito solo E



Topology

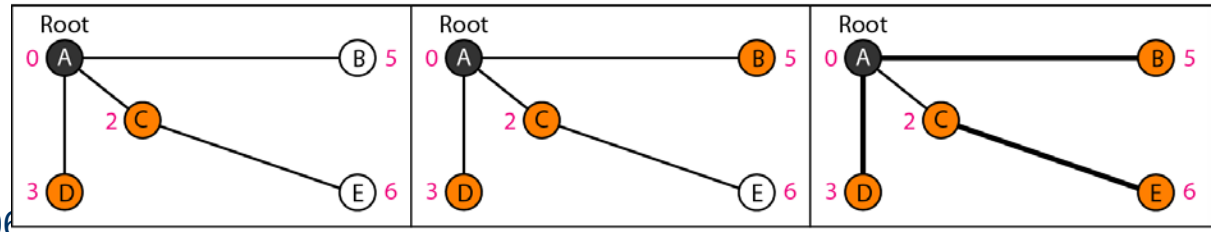
B,C,D vicini di A diventano temporanei



1. Set root to A and move A to tentative list.

2. Move A to permanent list and add B, C, and D to tentative list.

3. Move C to permanent and add E to tentative list.



4. Move D to permanent list.

5. Move B to permanent list.

6. Move E to permanent list (tentative list is empty).

Ora E viene spostato nei nodi finali. La lista dei temporanei ora è vuota