Università
degli Studi
della Campania
Luigi Vanvitelli

Reti di Calcolatori e Cybersecurity

Routing Link-State

Ing. Vincenzo Abate

Algoritmi Link State

Ogni router:

- impara il suo ambito locale (linee e nodi adiacenti)
- trasmette queste informazioni a tutti gli altri router della rete tramite un **Link State Packet (LSP)**
- memorizza gli LSP trasmessi dagli altri router e costruisce una mappa della rete
- Calcola, in maniera indipendente, le sue tabelle di instradamento applicando alla mappa della rete l'algoritmo di Dijkstra, noto come **Shortest Path First (SPF)**

Questo approccio è utilizzato nello standard ISO 10589 (protocollo IS-IS) e nel protocollo OSPF (adottato in reti TCP/IP)

intermediate system to intermediate system

Update process

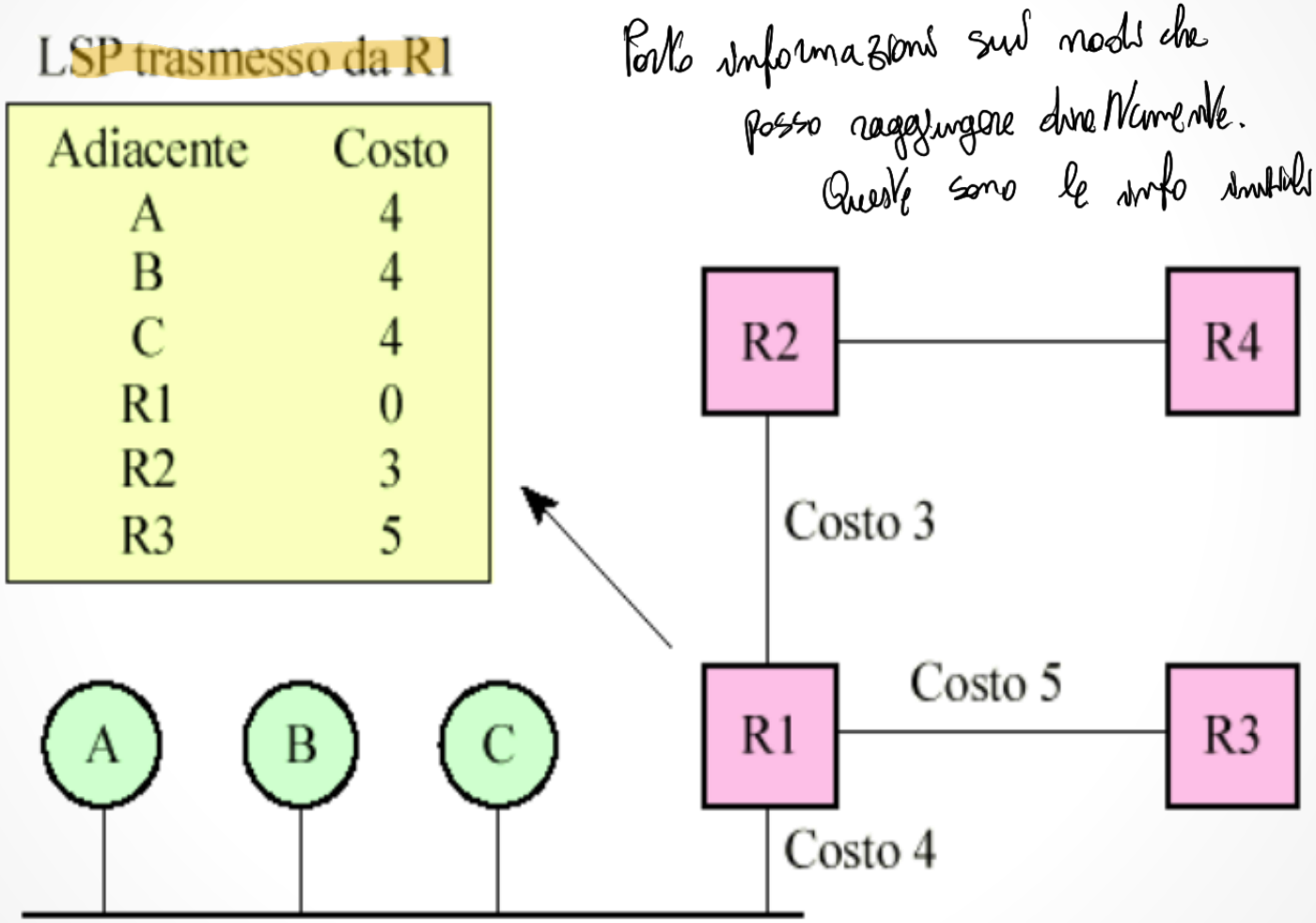
Ogni router genera un Link State Packet (LSP) contenente:

- stato di ogni link connesso al router
- identità di ogni vicino connesso all'altro estremo del link
- costo del link
- numero di sequenza per l'LSP
- checksum
- Lifetime:
  - la validità di ogni LSP è limitata nel tempo (e.g. un errore sul numero di sequenza potrebbe rendere un LSP valido per anni)

Flooding LSP

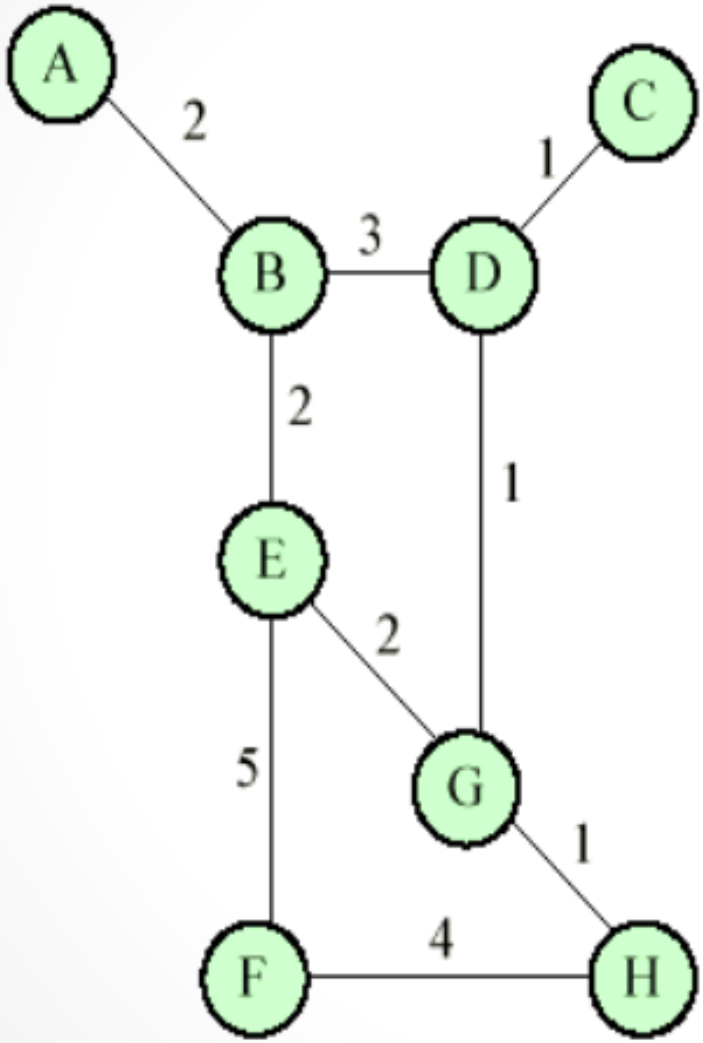
- Un LSP viene generato periodicamente, oppure quando viene rilevata una variazione nella topologia locale (adiacenze), ossia :
  - Viene riconosciuto un nuovo vicino
  - Il costo verso un vicino e' cambiato
  - Si e' persa la connettività verso un vicino precedentemente raggiungibile
- Un LSP è trasmesso in flooding su tutti i link del router
- I pacchetti LSP memorizzati nei router formano una mappa completa e aggiornata della rete:
  - **Link State Database**

Flooding LSP



Grafo rete e Database LSP

Sulla base dei pacchetti:



da A posso raggiungere

LSP Database

A	B/2		
B	A/2	D/3	E/2
C	D/1		
D	B/3	C/1	G/1
E	B/2	F/5	G/2
F	E/5	H/4	
G	D/1	E/2	H/1
H	F/4	G/1	

(replicato su ogni IS)

Database LSP

SORGENTE

DESTINAZIONE →

	A	B	C	D	E	F	G	H
A	0	2						
B	2	0		3	2			
C			0	1				
D		3	1	0			1	
E		2			0	5	2	
F					5	0		4
G				1	2		0	1
H						4	1	0

Questa rappresentazione è quella più appropriata per applicare l'agoritmo di Dijkstra

Gestione LSP

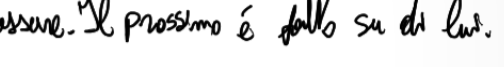
All'atto della ricezione di un LSP, il router compie le seguenti azioni:

1. se non ha mai ricevuto LSP da quel router o se l'LSP è più recente di quello precedentemente memorizzato:
   - memorizza il pacchetto
   - lo ritrasmette in flooding su tutte le linee eccetto quella da cui l'ha ricevuto
2. se l'LSP ha lo stesso numero di sequenza di quello posseduto:
   - non fa nulla
3. Se l'LSP è più vecchio di quello posseduto:
   - trasmette al mittente il pacchetto più recente

Decisioni del Router

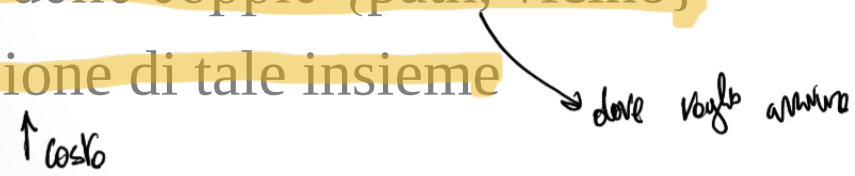
Il router elabora il Link State Database per produrre il **Forwarding Database**:

- si pone come radice dello shortest-path tree
- cerca lo shortest path per ogni nodo destinazione
- memorizza il vicino (i vicini) che sono sullo shortest path verso ogni nodo destinazione

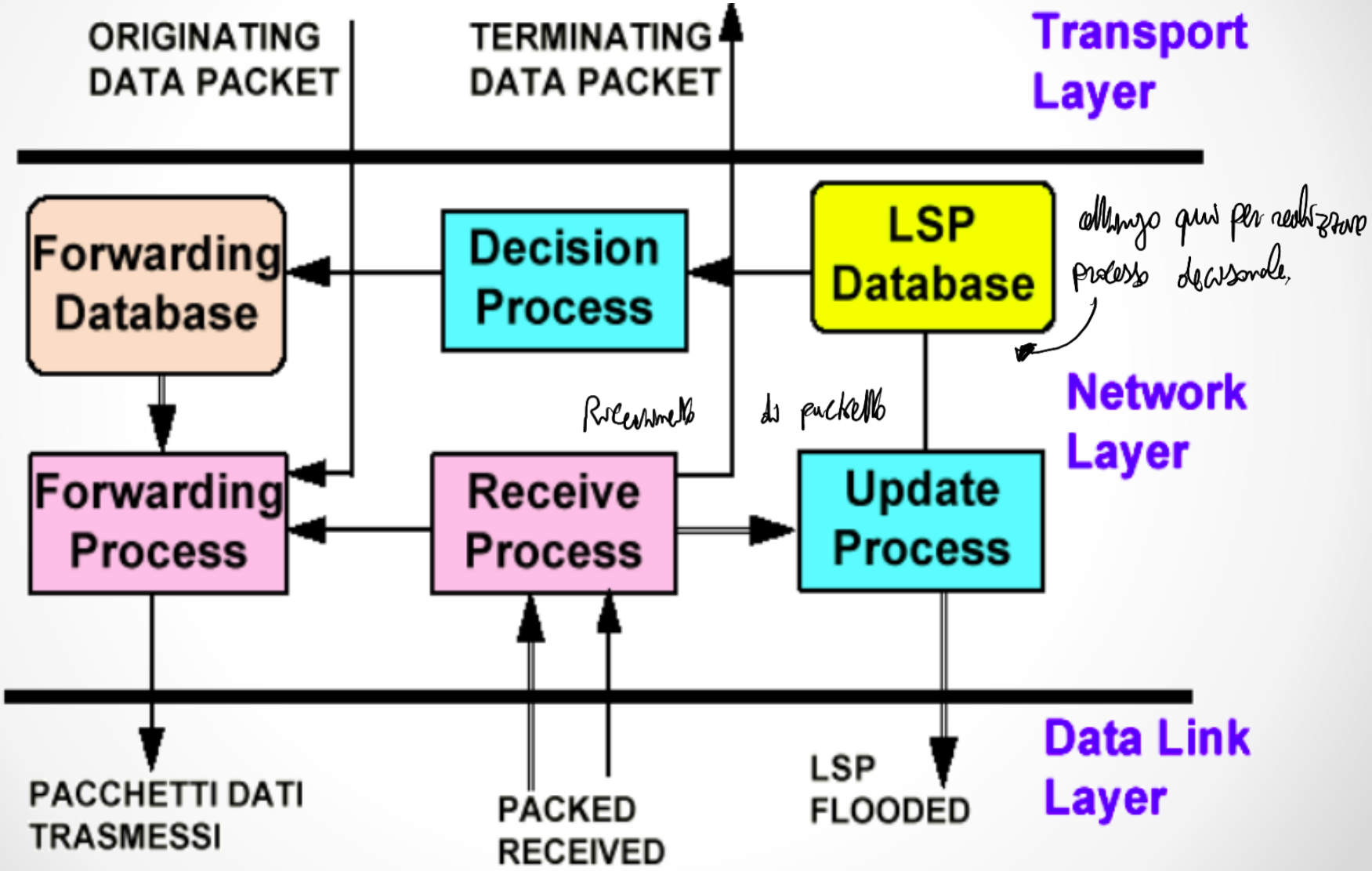


Il Forwarding Database contiene, per ogni nodo destinazione:

- l'insieme delle coppie {path, vicino}
- la dimensione di tale insieme



Architettura Router Link State



Link State: Pro e Contro

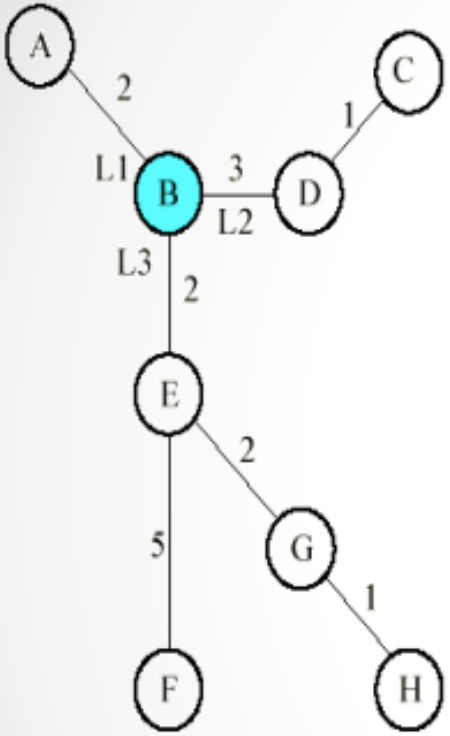
Vantaggi:

- può gestire reti di grandi dimensioni
- ha una convergenza rapida
- difficilmente genera loop, e comunque è in grado di identificarli ed interromperli facilmente
- facile da capire: ogni nodo ha la mappa della rete

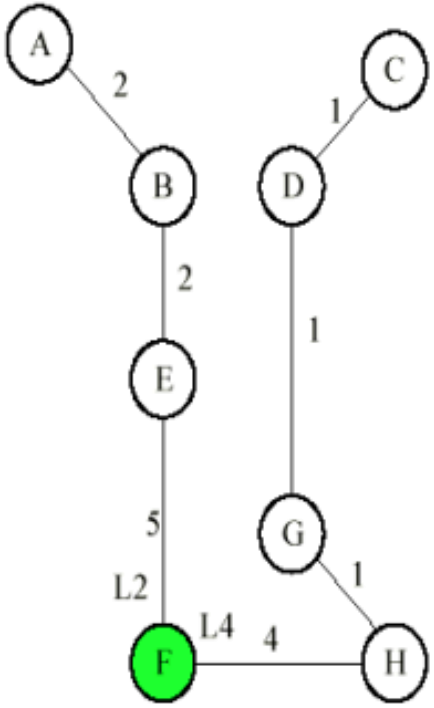
Svantaggi:

- Molto complesso da realizzare (la prima implementazione ha richiesto alla Digital 5 anni)

Link State: Pro e Contro



(a)



(b)

Tabella di B

A	L1
C	L2
D	L2
E	L3
F	L3
G	L3
H	L3

Tabella di F

A	L2
B	L2
C	L4
D	L4
E	L2
G	L4
H	L4

Algoritmo di Dijkstra

- Ogni nodo ha a disposizione il grafo della rete:
  - i nodi sono i router
  - gli archi sono le linee di collegamento tra router:
    - agli archi è associato un costo
- Ogni nodo usa l'algoritmo di Dijkstra per costruire lo Shortest Path Tree del grafo, ovvero l'albero dei cammini di costo minimo
- Ad ogni nodo si assegna un'etichetta che rappresenta il costo massimo per raggiungere quel nodo
- L'algoritmo modifica le etichette cercando di minimizzarne il valore e di renderle permanenti

Algoritmo di Dijkstra

- La Topologia della rete è nota a tutti i nodi:
  - la diffusione è realizzata via "link state broadcast"
  - tutti i nodi hanno la stessa informazione
- Si calcola il percorso minimo da un nodo a tutti gli altri:
  - l'algoritmo fornisce la **tabella di routing** per quel nodo
- **Iterativo:** un nodo, dopo k iterazioni, conosce i cammini meno costosi verso k destinazioni

Notazione:

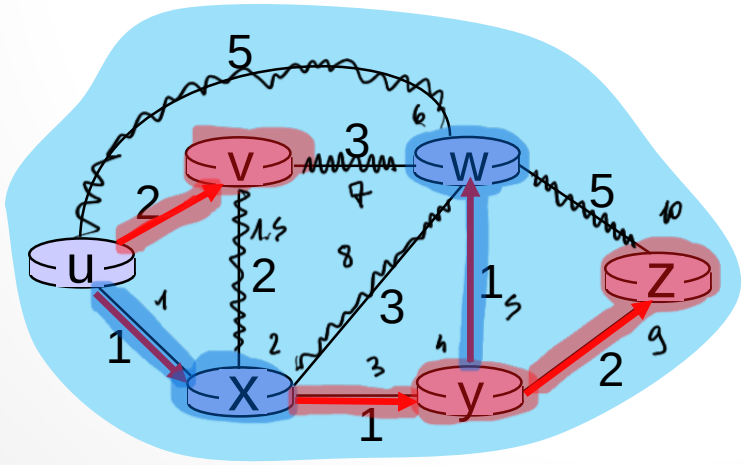
- **c(i,j)**: costo collegamento da i a j: $c(i,j) \geq 0$
  - infinito se non c'è collegamento
  - per semplicità, c(i,j) = c(j,i)
- **D(v)**: costo corrente del percorso, dalla sorgente al nodo v
- **p(v)**: predecessore (collegato a v) lungo il cammino dalla sorgente a v
- **N**: insieme di nodi per cui la distanza è stata trovata

Algoritmo di Dijkstra

```
1  Inizializzazione:
2    N = {A}
3    per tutti i nodi  v
4      if (v e'  adiacente a A)
5        then D(v) = c(A,v)
6        else D(v) = ∞
7
8  Loop
9     sia w non in N tale che  D(w) è minimo
10     aggiungi w a N
11     aggiorna D(v) per ogni v adiacente a  w e non in N:
12        D(v) = min( D(v), D(w) + c(w,v) )
13     {il nuovo costo fino a v è o il vecchio costo, oppure il costo del
           cammino piu breve fino a w più il costo da w a v }
15  fino a quando tutti i nodi sono in N
```

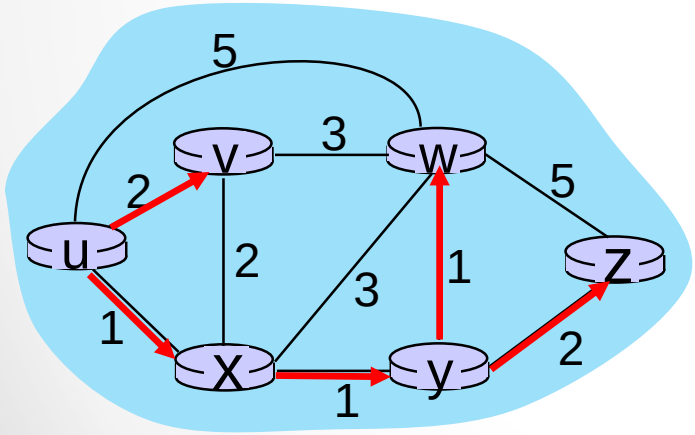
Algoritmo di Dijkstra

- L'algoritmo consiste in un passo di inizializzazione, più un ciclo di durata pari al numero di nodi della rete. Al termine avremo i percorsi più brevi dal nodo sorgente a tutti gli altri nodi
- Esempio. Calcoliamo sulla rete data i percorsi di costo minimo da U a tutte le possibili destinazioni. Ciascuna riga della tabella della slide seguente fornisce i valori delle variabili dell'algoritmo alla fine di ciascuna iterazione



Algoritmo di Dijkstra: esempio

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					

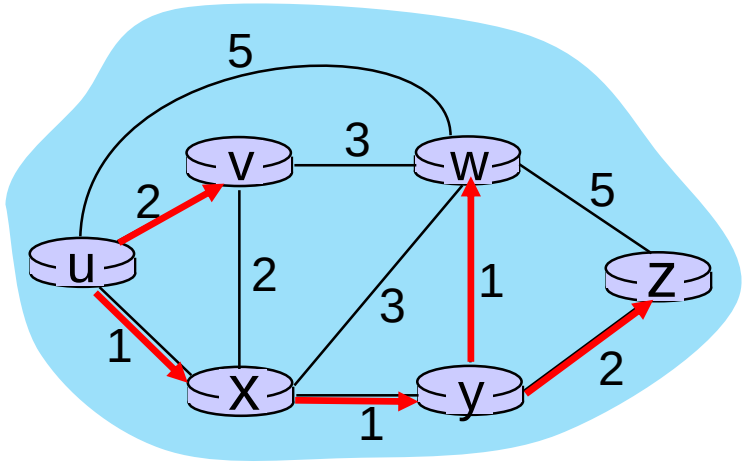


Initialization (step 0): For all a: if *a* adjacent to then $D(a) = c_{u,a}$

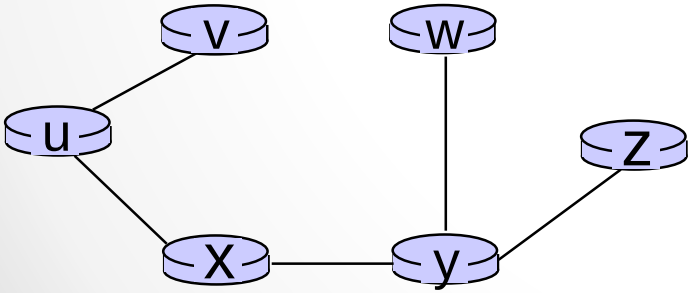
find *a* not in *N'* such that *D(a)* is a minimum
add *a* to *N'*
update *D(b)* for all *b* adjacent to *a* and not in *N'* :

$$D(b) = \min\left(D(b),\ D(a) + c_{a,b}\right)$$

Algoritmo di Dijkstra: esempio



resulting least-cost-path tree from u:



resulting forwarding table in u:

destination	outgoing link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
x	(u,x)

route from u to all other destinations via *x*

Dijkstra's algorithm: discussion

algorithm complexity: n nodes

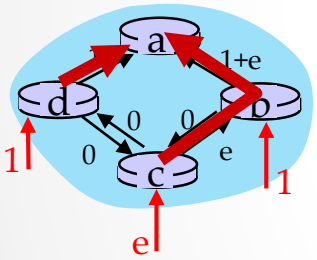
- each of n iteration: need to check all nodes, w, not in N
- $n(n+1)/2$ comparisons: $O(n^2)$ complexity
- more efficient implementations possible: $O(n\log n)$

message complexity:

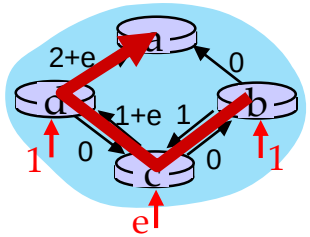
- each router must *broadcast* its link state information to other n routers
- efficient (and interesting!) broadcast algorithms: $O(n)$ link crossings to disseminate a broadcast message from one source
- each router's message crosses $O(n)$ links: overall message complexity: $O(n^2)$

Dijkstra's algorithm: oscillations possible

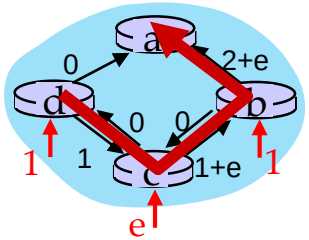
- when link costs depend on traffic volume, route oscillations possible
- sample scenario:
  - routing to destination a, traffic entering at d, c, e with rates 1, e (<1), 1
  - link costs are directional, and volume-dependent



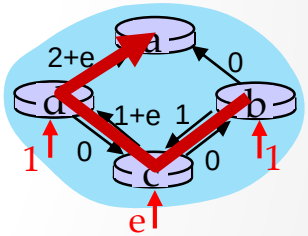
initially



given these costs, find new routing…. resulting in new costs



given these costs, find new routing…. resulting in new costs

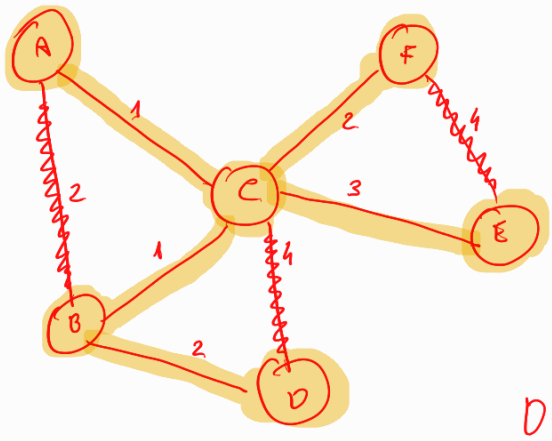


given these costs, find new routing…. resulting in new costs

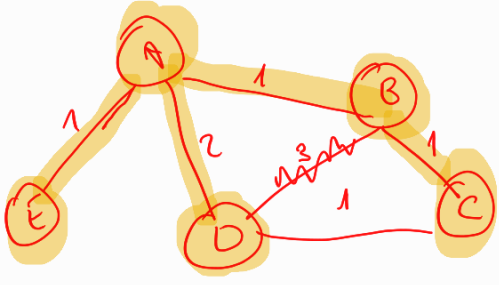
devo sincronizzare bene come si scambiano i router: se mi basa su capacità del link potrei non

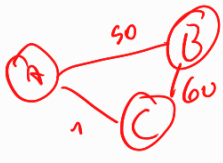
Soluzione: evitare la sincronizzazione nell'invio dei messaggi dei router

avviene mai a convergenza



D → 4





$d(B,F) = 2$

$d(B,A) = 1$ $d(B,C) = 1$

$d(B,D) = 2$

$d(B,H) = 3$