Programmazione in Rete

a.a. 2005/2006

http://www.di.uniba.it/~lisi/courses/prog-rete/prog-rete0506.htm

dott.ssa Francesca A. Lisi
lisi@di.uniba.it

Orario di ricevimento: mercoledì ore 10-12

dib

Sommario della lezione di oggi: Lo strato di rete (3bis/3)

- Servizi e protocolli dello strato di rete
- Reti a circuito virtuale vs reti a datagramma
- Struttura di un router
- Inoltro e indirizzamento in Internet: il protocollo IP
- Instradamento in Internet
- Approfondimento sugli algoritmi di instradamento

Algoritmi di instradamento: una classificazione

Globale o decentralizzato?

Globale:

- tutti i router hanno completa info su topologia e costi dei link
- algoritmi "link state"

Decentralizzato:

- ogni router conosce i vicini connessi fisicamente e costi di link ai vicini
- processo iterativo di calcolo, scambio di info con i vicini
- algoritmi "distance vector"

Statico o dinamico?

Statico:

- le rotte cambiano lentamente nel tempo

Dinamico:

- le rotte cambiano più rapidamente
  - aggiornamento periodico
  - in risposta ai cambiamenti nei costi dei link

Approccio "link-state": Algoritmo di Dijkstra

- topologia di rete e costi dei link noti a tutti i nodi
  - eseguito mediante "link state broadcast"
  - tutti i nodi hanno stessa info
- calcola percorsi a costo minimo da un nodo ("sorgente") a tutti i nodi
  - fornisce tabella di instradamento per quel nodo
- iterativo: dopo *k* iterazioni, conosce cammini a costo minimo verso *k* destinazioni

Notazione:

- c(i,j): costo del link da *i* a *j*; costo infinito se *i* e *j* non sono adiacenti
- D(v): costo del percorso attualmente a costo minimo da sorgente a *v*
- p(v): nodo predecessore di *v* nel percorso attualmente a costo minimo da sorgente a *v*
- N: insieme di nodi il cui percorso a costo minimo è definitivamente noto

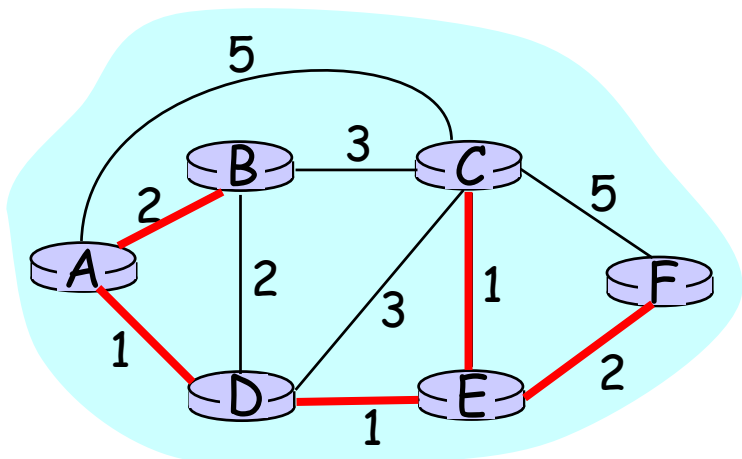
Algoritmo di Dijkstra: pseudo-codice

```
Initialization:
  N = {A}
  for all nodes v
    if v adjacent to A
      then D(v) = c(A,v)
      else D(v) = infty

Loop
   find w not in N such that D(w) is a minimum
   add w to N
   update D(v) for all v adjacent to w and not in N:
       D(v) = min( D(v), D(w) + c(w,v) )
   /* new cost to v is either old cost to v or known
    shortest path cost to w plus cost from w to v */
until all nodes in N
```

Algoritmo di Dijkstra: un esempio

Step	start 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	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					

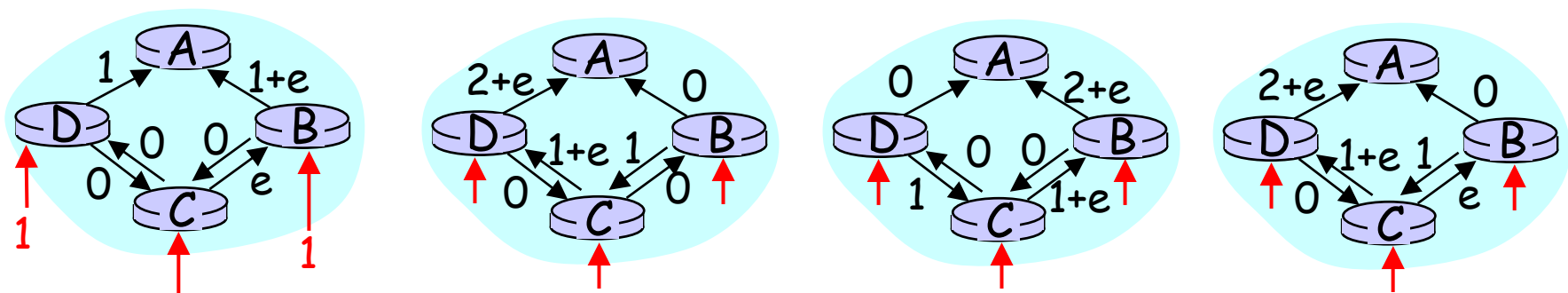


Algoritmo di Dijkstra: discussione

Complessità dell'algoritmo: n nodi

- ad ogni iterazione, bisogna controllare tutti i nodi, w, non contenuti in N -> n*(n+1)/2 confronti: $O(n^2)$
- implementazioni + efficienti: O(nlogn)

Oscillazioni possibili, p.es. costo del link = carico di traffico smaltito



inizialmente

... ricalcola instradamento

... ricalcola

... ricalcola

Approccio "distance vector": Algoritmo di Bellman-Ford

iterativo:

- continua finchè nodi scambiano info.
- *auto-terminante*: nessun "segnale" di fermata

asincrono:

- i nodi non devono scambiare info/iterare in passo di lock!

distribuito:

- ogni nodo comunica *solo* con vicini direttamente collegati

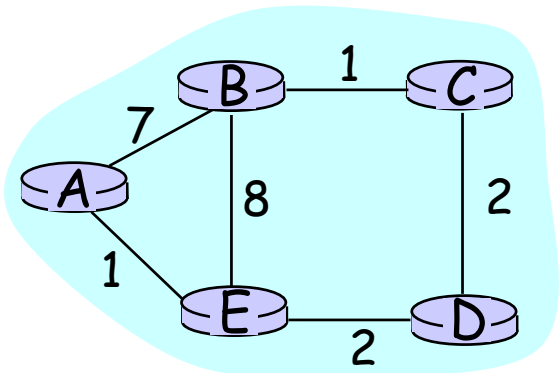
Tabella delle distanze

- ogni nodo ha sua propria tabella
- riga per ogni possibile destinazione
- colonna per ogni vicino direttamente collegato al nodo
- es.: nel nodo X, per la destinazione Y via vicino Z:

$$D^X(Y,Z) = \text{distance } \textit{from } X \textit{ to } Y, \textit{ via } Z \text{ as next hop}$$

$$= c(X,Z) + \min_w \{D^Z(Y,w)\}$$

Algoritmo di Bellman-Ford: esempio di tabella delle distanze



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

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

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

costo alla destinazione via

$D^E()$	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

destinazione

Algoritmo di Bellman-Ford: dalle distanze all'instradamento

Costo per destinazione via

$D^E()$ (destinazione)	A	B	D
A	1	14	5
B	7	8	5
C	6	9	4
D	4	11	2

Link in uscita da usare, costo associato

destinazione	
A	A,1
B	D,5
C	D,4
D	D,4

Tabella delle distanze → Tabella di instradamento

Algoritmo di Bellman-Ford: panoramica

Iterativo, asincrono: ogni iterazione locale causata da:

- modifica di costo di un link
- messaggio da un vicino riguardo modifica di suo percorso a costo minimo

Distribuito:

- ogni nodo notifica ai vicini *solo* se suo percorso a costo minimo cambia
- i vicini a loro volta notificano ai propri vicini se necessario, e così via

Cosa accade in ogni nodo?

wait for (change in local link cost or msg from neighbor)

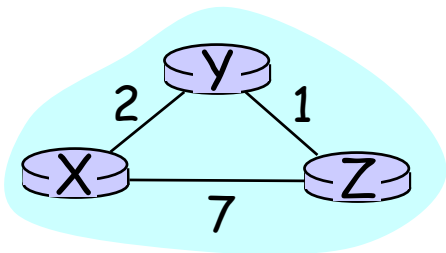
recompute distance table

if least cost path to any dest has changed, *notify* neighbors

Algoritmo di Bellman-Ford: pseudo-codice per ogni nodo X

```
for all adjacent nodes v:
   D^X(*,v) = infty        /* the * operator means "for all rows" */
   D^X(v,v) = c(X,v)
for all destinations y:
   send  min_w D^X(y,w) to each neighbor  /* w over all X's neighbors */
loop
  wait (until I see a link cost change to neighbor V
        or until I receive update from neighbor V)
  if (c(X,V) changes by d)
     /* change cost to all dest's via neighbor v by d (pos or neg value) */
     for all destinations y: D^X(y,V) = D^X(y,V) + d
  else if (update received from V wrt destination Y)
     /* shortest path from V to some Y has changed  */
     /* V has sent a new value "newval" for its min_w DV(Y,w) */
     for the single destination y: D^X(Y,V) = c(X,V) + newval
  if we have a new min_w D^X(Y,w) for any destination Y
     then send new value of min_w D^X(Y,w) to all neighbors
forever
```

Algoritmo di Bellman-Ford: un esempio di esecuzione



X	cost via Y	cost via Z
dest Y	2	∞
dest Z	∞	7

Y	cost via X	cost via Z
dest X	2	∞
dest Z	∞	1

Z	cost via X	cost via Y
dest X	7	∞
dest Y	∞	1

X	cost via Y	cost via Z
dest Y	2	8
dest Z	3	7

Y	cost via X	cost via Z
dest X	2	8
dest Z	9	1

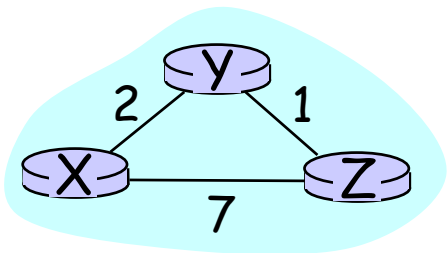
Z	cost via X	cost via Y
dest X	7	3
dest Y	9	1

X	cost via Y	cost via Z
dest Y		
dest Z		

Y	cost via X	cost via Z
dest X		
dest Z		

Z	cost via X	cost via Y
dest X		
dest Y		

Algoritmo di Bellman-Ford: un esempio di esecuzione (cont.)



D^X

dest	cost via Y	cost via Z
Y	2	∞
Z	∞	7

D^Y

dest	cost via X	cost via Z
X	2	∞
Z	∞	1

D^Z

dest	cost via X	cost via Y
X	7	∞
Y	∞	1

D^X

dest	cost via Y	cost via Z
Y	2	8
Z	3	7

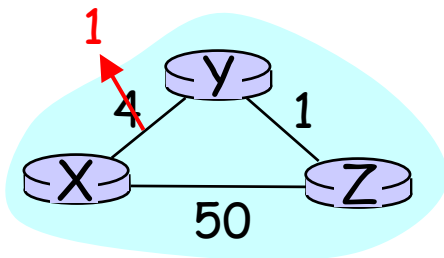
$$D^X(Y,Z) = c(X,Z) + \min_w\{D^Z(Y,w)\}$$
$$= 7+1 = 8$$

$$D^X(Z,Y) = c(X,Y) + \min_w\{D^Y(Z,w)\}$$
$$= 2+1 = 3$$

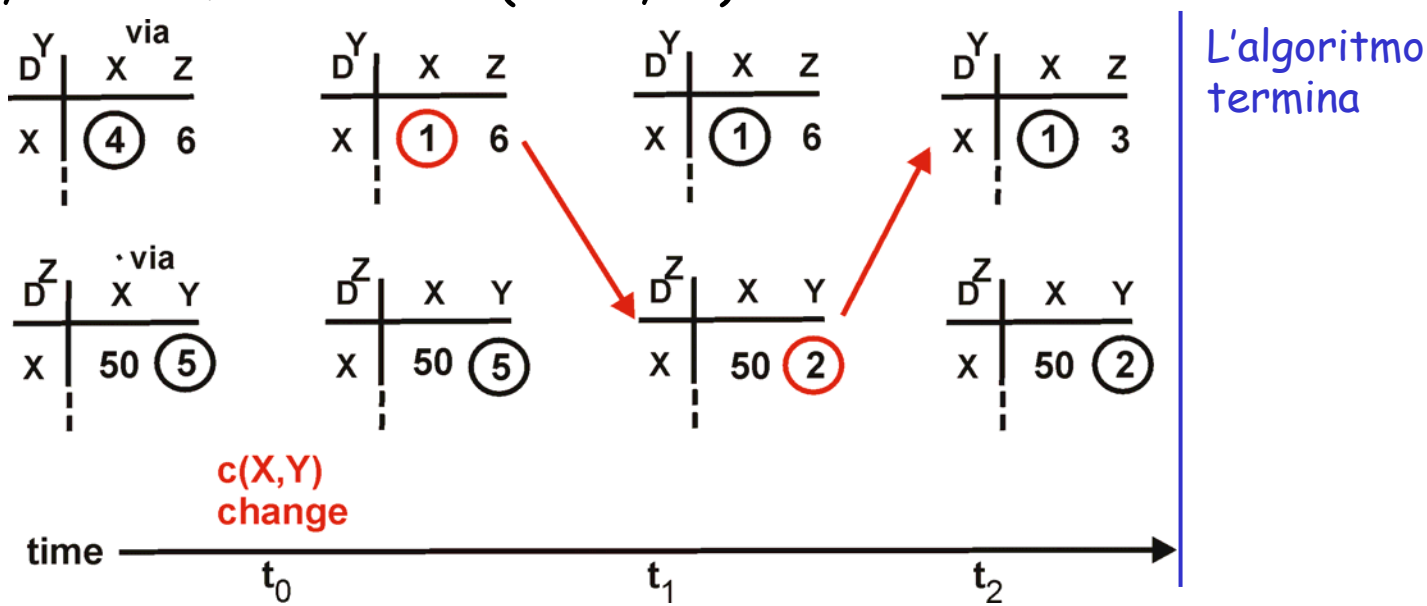
Algoritmo di Bellman-Ford: variazioni dei costi dei link

Abbassamento costi:

- un nodo rileva localmente variazione di costo di link
- aggiorna tabella delle distanze (r.15)
- se costo cambia nel cammino a costo minimo, lo notiifca ai vicini (r. 23,24)



"le buone notizie vaggiano in fretta"

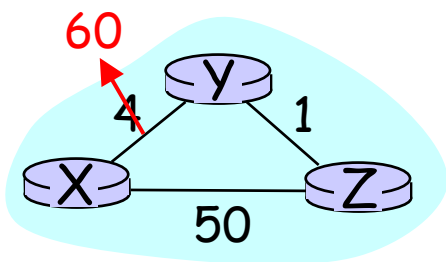


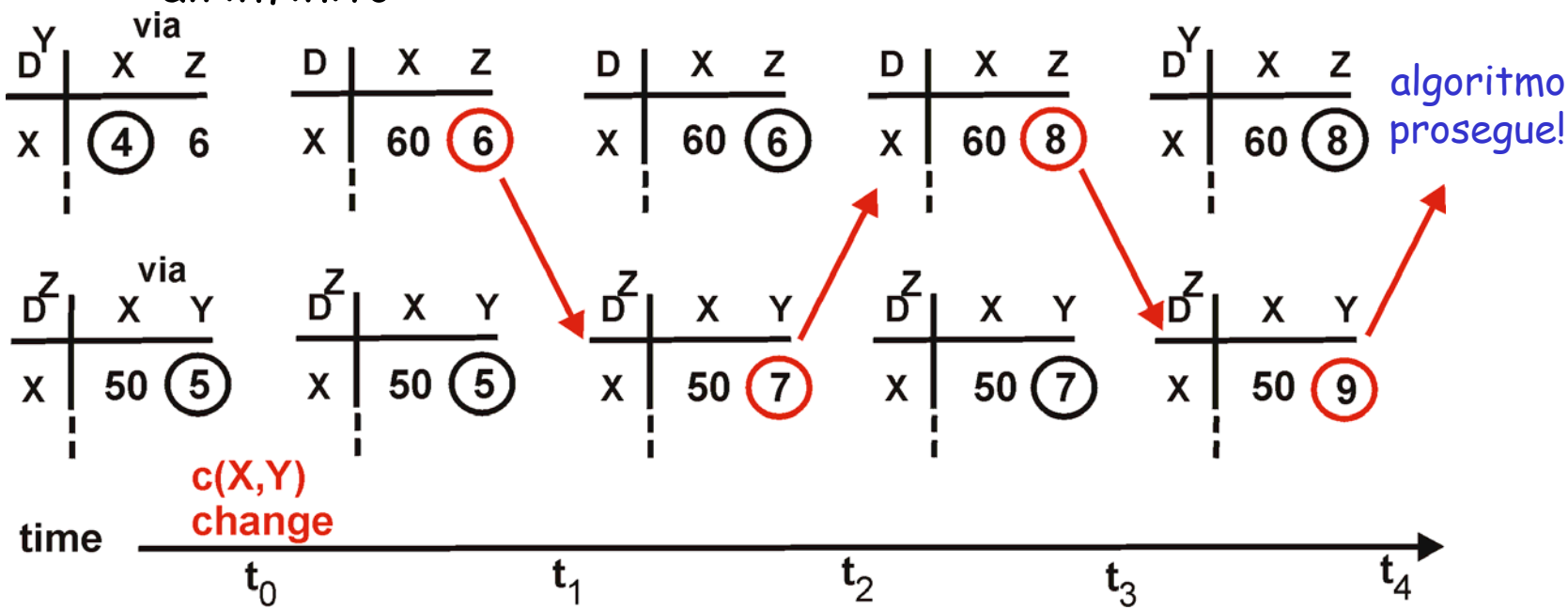
L'algoritmo termina

Algoritmo di Bellman-Ford: variazioni dei costi dei link (cont.)

Innalzamento costi:

- le cattive notizie viaggiano lentamente
- problema di "conteggio all'infinito"

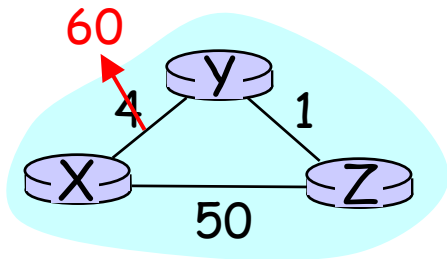


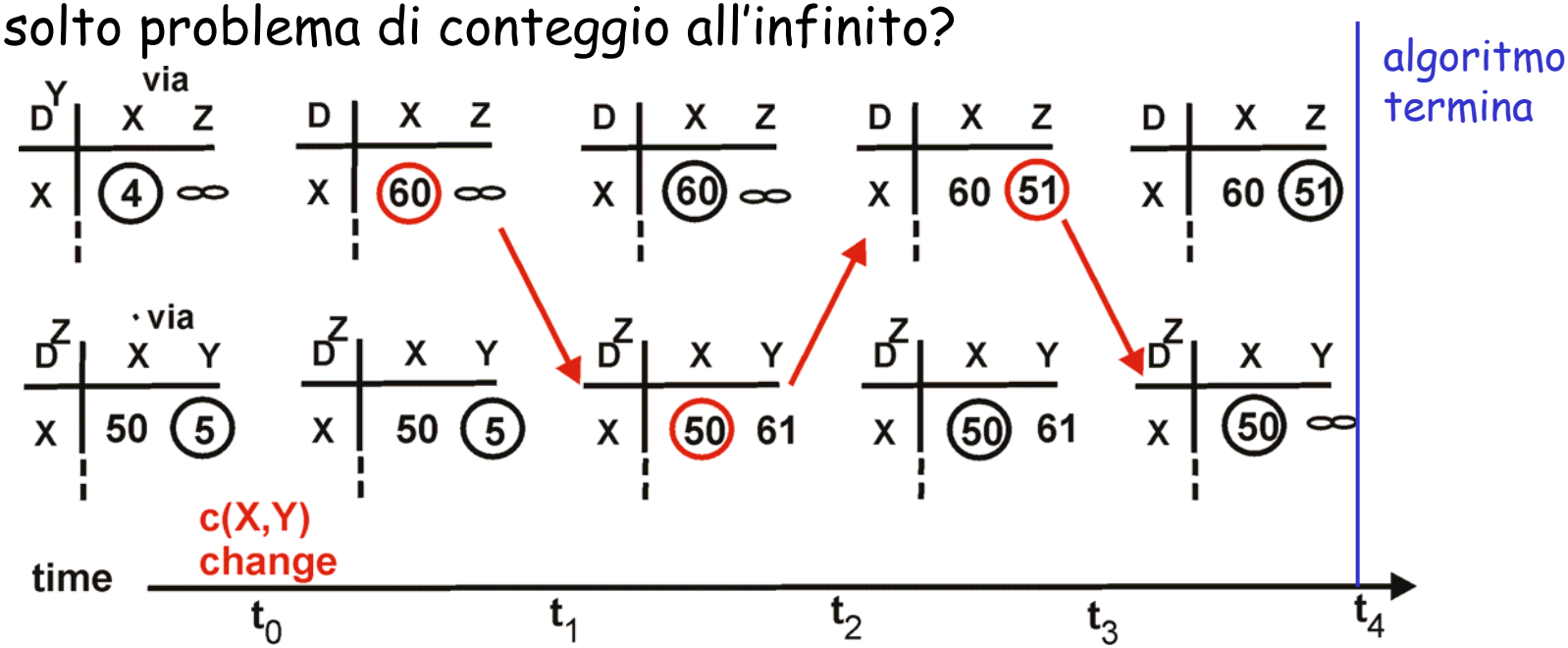


Algoritmo di Bellman-Ford: aggiunta del "poisoned reverse"

Se Z instrada attraverso Y per raggiungere X :

- Z dice a Y che la propria distanza da X è infinita (in modo che Y non instraderà ad X via Z)
- Risolto problema di conteggio all'infinito?





Confronto fra algoritmi LS e DV

Complessità del messaggio

- LS: con *n* nodi, *E* link, ciascun nodo invia *O(nE)* msg
- DV: scambio msg solo fra vicini

Velocità di convergenza

- LS: algoritmo di complessità $O(n^2)$ richiede O(nE) msg
  - può avere oscillazioni
- DV: tempo variabile di convergenza
  - possibili percorsi ciclici
  - problema di "conteggio all'infinito"

Robustezza: che accade se un router funziona male?

- LS:
  - il nodo può comunicare costo errato di *link*
  - ogni nodo calcola solo la *propria* tabella
- DV:
  - nodo può comunicare costo errato di *percorso*
  - tabella di ogni nodo usata dagli altri
    - errore si propaga attraverso la rete