Corso di Laurea in Ingegneria Informatica

Corso di Reti di Calcolatori
(a.a. 2011/12)

Roberto Canonico (roberto.canonico@unina.it)
Giorgio Ventre (giorgio.ventre@unina.it)

Routing Distance Vector

7 novembre 2011

I lucidi presentati al corso sono uno strumento didattico che NON sostituisce i testi indicati nel programma del corso

Nota di copyright per le slide COMICS

Nota: alcune delle slide di questa lezione sono direttamente prese dal materiale didattico preparato dagli autori del libro di testo Kurose e Ross

Algoritmo di routing Distance Vector

- Ogni router mantiene una tabella di tutti gli instradamenti a lui noti
  - inizialmente, solo le reti a cui è connesso direttamente
- Ogni entry della tabella indica:
  - una rete raggiungibile
  - il *next hop*
  - il numero di hop necessari per raggiungere la destinazione
- Periodicamente, ogni router invia a tutti i vicini (due router sono vicini se sono collegati alla stessa rete fisica):
  - un messaggio di aggiornamento contenente tutte le informazioni della propria tabella (***vettore delle distanze – distance vector***)
- I router che ricevono tale messaggio aggiornano la tabella nel seguente modo:
  - eventuale modifica di informazioni relative a cammini già noti
  - eventuale aggiunta di nuovi cammini
  - eventuale eliminazione di cammini non più disponibili

Distance Vector: un esempio

Destin.	Dist.	Route
net 1	0	direct
net 2	0	direct
net 4	8	router L
net 17	5	router M
net 24	6	router A
net 30	2	router Q
net 42	2	router A

Tabella del router B

→

Destin.	Dist.
net 1	2
net 4	3
net 17	6
net 21	4
net 24	5
net 30	10
net 42	3

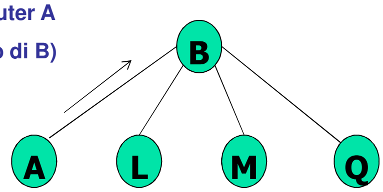
Messaggio di aggiornamento del router A

(vicino di B)

→

Destin.	Dist.	Route
net 1	0	direct
net 2	0	direct
net 4	4	router A
net 17	5	router M
net 24	6	router A
net 30	2	router Q
net 42	4	router A
net 21	5	router A

Tabella aggiornata del router B



Bellman-Ford Equation

Bellman-Ford Equation (dynamic programming)

Define

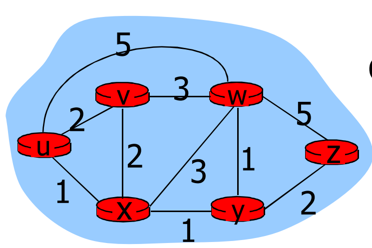
$d_x(y)$:= cost of least-cost path from x to y

Then

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

where min is taken over all neighbors v of x

Bellman-Ford example



Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$d_u(z) = \min \{ c(u,v) + d_v(z),\ c(u,x) + d_x(z),\ c(u,w) + d_w(z) \}$$
$$= \min \{2 + 5,\ 1 + 3,\ 5 + 3\} = 4$$

Node that achieves minimum is next hop in shortest path ➜ forwarding table

Distance Vector Algorithm

- Define
  - $D_x(y)$ = estimate of least cost from x to y
  - Distance vector: $\mathbf{D}_x = [D_x(y): y \in N]$

- Node x knows cost to each neighbor v: $c(x,v)$
- Node x maintains $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x also maintains its neighbors' distance vectors
  - For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y): y \in N]$

Distance Vector Algorithm

Basic idea:

- Each node periodically sends its own distance vector estimate to neighbors
- When a node x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- Under "natural conditions" the estimate $D_x(y)$ **converges to the actual least cost** $d_x(y)$

Distance Vector Algorithm

Each node:

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

↓

recompute estimates

↓

if DV to any dest has changed, *notify* neighbors

- Iterative
  - Continues until no more info is exchanged
  - Each iteration caused by:
    - local link cost change
    - DV update message from neighbor
- Asynchronous
  - Nodes do not operate in lockstep
- Distributed
  - Each node receives info only from its directly attached neighbors
  - NO Global info

Distance Vector: esempio

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

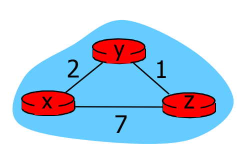
node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

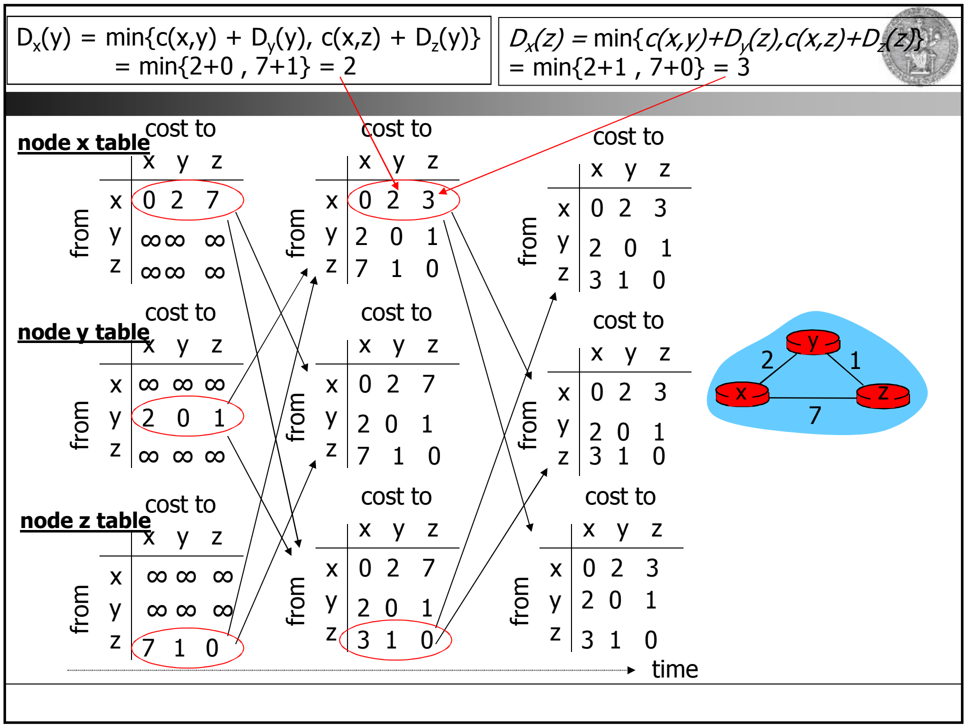
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$



time



Distance Vector: analisi

- Vantaggi:
  - facile da implementare
- Svantaggi
  - ogni messaggio contiene un'intera tabella di routing
  - lenta propagazione delle informazioni sui cammini:
    - converge alla velocità del router più lento
  - se lo stato della rete cambia velocemente, le rotte possono risultare inconsistenti
    - possono innescarsi dei loop a causa di particolari variazioni della topologia
  - difficile capirne e prevederne il comportamento su reti grandi
    - nessun nodo ha una mappa della rete!

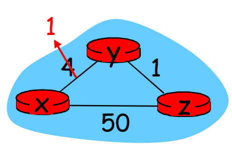
Convergence Speed

- How fast the routers learn about link-status change in the network
- With distance vector routing
  - Good news travels fast
  - Bad news travels slow

Distance Vector: link cost changes (1/2)

Link cost decreased:

- ❑ node detects local link cost change
- ❑ updates routing info, recalculates distance vector
- ❑ if DV changes, notify neighbors



"good news travels fast"

At time t_0, y detects the link-cost change, updates its DV, and informs its neighbors.

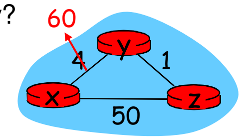
At time t_1, z receives the update from y and updates its table. It computes a new least cost to x and sends its neighbors its DV.

At time t_2, y receives z's update and updates its distance table. y's least costs do not change and hence y does *not* send any message to z.

Distance Vector: link cost changes (2/2)

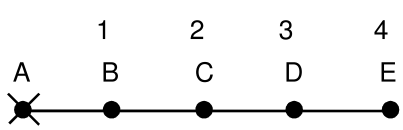
Link cost increased:

- t_0: y detects change, updates its cost to x to be **6**. Why?
  - ❖ Because z previously told y that "I can reach x with cost of 5"
  - ❖ **6 = min {60+0, 1+5}**
- Now we have a **routing loop**!
  - ❖ Pkts destined to x from y go back and forth between y and z forever (or until loop is broken)
- t_1: z gets the update from y. z updates its cost to x to be??
  - ❖ **7 = min {50+0, 1+6}**
- Algorithm will take several iterations to stabilize
  - ❖ This is called "**count to infinity**" problem!
- **Solutions?**



"Bad news travels slow"

Convergence speed

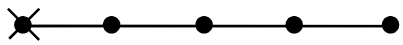


Initially

16

Count-to-Infinity

	A	B	C	D	E	
		1	2	3	4	Initially
		3	2	3	4	After 1 exchange
		3	4	3	4	After 2 exchanges
		5	4	5	4	After 3 exchanges
		5	6	5	6	After 4 exchanges
		7	6	7	6	After 5 exchanges



etc… to infinity

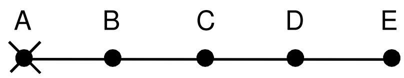
17

Poisoned Reverse

- If Z routes through Y to get to X :
  - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)

18

Poisoned Reverse



inf.	2	3	4	B learns A is dead
inf. —inf.→	2	3	4	B reports to C that A's metric is inf.
inf.	inf.	3	4	After 1 exchange
inf.	inf.	inf.	4	After 2 exchanges
inf.	inf.	inf.	inf.	After 3 exchanges

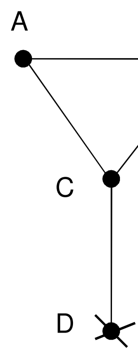
19

Poisoned Reverse

- If Z routes through Y to get to X :
  - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- will this completely solve count to infinity problem?

20

Poisoned Reverse Failure



If D goes down, A and B will still count to infinity.

Distance Vector (DV) Routing Protocols

- Routers broadcast routing control messages to their neighbors containing lowest cost-to-destination information.
- A router using a DV protocol propagates a routing control message by adding its own cost to the received information and broadcasting the sums to its neighbors.
- If there are no rules to stop propagation of routing control messages, then the routers keep adding their own cost and broadcasting, collectively counting to infinity.

Beating the count-to-infinity problem

- Include a sequence number in the routing control messages.
- Sequence numbers are monotonically increasing. Larger sequence numbers -> later control message.
- A neighbor receiving a message with a higher sequence number always propagates the routing information and remembers the sequence number.
- Routing information that reduces the receiving node's lowest cost-to-destination should be propagated.
- Routing information that increases the receiving node's lowest cost-to-destination should be propagated.

23

Beating the count-to-infinity problem

- Routing information that neither increases or reduces the receiving node's lowest cost-to-destination should NOT be propagated.
- Routing information that is propagated keeps the same sequence number as the received information.

24