



Algoritmi e protocolli di Routing

A.A. 2003/2004

Walter Cerroni

Tabelle di routing

- **Routing** : scelta del percorso su cui inviare i dati
 - i router formano struttura interconnessa e cooperante: i datagrammi passano dall'uno all'altro finché raggiungono quello che può consegnarli direttamente al destinatario
- La funzione di instradamento viene eseguita dai router tramite l'utilizzo di apposite **tabelle di routing**
 - presenti in ogni nodo
 - per ogni indirizzo di destinazione è indicata la porta di uscita da usare per raggiungere quella destinazione
 - possono essere presenti direzioni alternative corrispondenti a costi diversi

2

Il trasferimento dei datagrammi

Ha due componenti funzionali distinte in ciascun nodo

Instradamento (routing)

- Creazione di tabelle di instradamento tramite:
 - scambio di informazioni tra router
 - elaborazione locale

Inoltro (forwarding)

- Algoritmo di forwarding
 - lettura dell'intestazione IP ed estrazione dell'indirizzo di destinazione
 - consultazione delle tabelle di instradamento
 - Identificazione dell'interfaccia di uscita su cui inviare il datagramma
- Switching
 - trasferimento fisico dei datagrammi da ingresso a uscita

3

Implementazione

- L'implementazione della funzione di forwarding dipende dal costruttore del router
- La funzione di routing deve essere standard al fine di avere coerenza nel comportamento dei router
- La funzione di routing si avvale di:
 - **algoritmi di routing**
 - usati per il calcolo delle tabelle di instradamento note le informazioni sulla topologia della rete
 - **protocolli di routing**
 - usati per lo scambio delle informazioni sulla topologia della rete necessarie per applicare l'algoritmo

4

Caratteristiche del routing

- I vari algoritmi e protocolli di routing si differenziano per le modalità con cui la tabella di instradamento viene creata ed eventualmente aggiornata nel tempo
- Un routing ideale dovrebbe essere:
 - corretto
 - semplice e veloce
 - robusto (rispetto a cambiamenti di topologia e di traffico)
 - stabile
 - equo
 - ottimale
 - scalabile
- Tipologie di routing
 - Statico
 - Dinamico
 - Centralizzato
 - Distribuito

5

Routing statico

- Le tabelle di instradamento sono:
 - invarianti nel tempo
 - indipendenti dalle condizioni di traffico della rete
 - adottano algoritmi non adattativi
- Le tabelle di instradamento vengono create in fase di configurazione del router
 - grande lavoro di configurazione
- modificate **con l'intervento di un operatore** solo in caso di variazioni strutturali o topologiche della rete (inserimento o caduta di nodi, collegamenti)

6

Routing dinamico

- Le tabelle di instradamento vengono create e periodicamente aggiornate **in modo automatico**
 - adottano algoritmi adattativi
- Consentono di adattare le decisioni di instradamento a
 - variazioni topologiche della rete
 - inserimento di nuovi collegamenti
 - caduta di un collegamento per guasto
 - condizioni di traffico
 - si evita la scelta di percorsi che comprendono collegamenti congestionati

7

Routing centralizzato

- Un unico nodo centrale:
 - raccoglie tutte le informazioni sullo stato e la topologia della rete
 - calcola le tabelle di instradamento per ogni nodo
 - le trasmette a tutti i nodi
- Pro
 - garantisce massima consistenza delle informazioni
- Contro
 - dipende dal corretto funzionamento di un solo apparato di rete
 - il nodo centrale è soggetto ad un grande traffico di overhead

8

Routing distribuito

- Ogni nodo calcola in modo autonomo le sue tabelle di instradamento
- Il calcolo può essere basato su informazioni:
 - locali
 - riguardanti il solo nodo in cui sta avvenendo il calcolo,
 - senza scambio di informazioni tra i nodi
 - distribuite
 - si utilizzano informazioni relative agli altri nodi e collegamenti della rete
- Nel caso di routing basato su informazioni distribuite deve essere previsto un meccanismo di scambio delle informazioni (protocollo)

9

Routing: piano di utente e di controllo

- L'instradamento è una delle funzioni dello strato di rete
 - dal punto di vista dell'utilizzo della rete (piano di utente) i router implementano le funzionalità fino allo strato di rete
- I protocolli di routing si basano su uno scambio di informazioni tra applicazioni
 - dal punto di vista del controllo della rete (piano di controllo) i router implementano anche funzionalità di livello superiore (sono calcolatori specializzati)

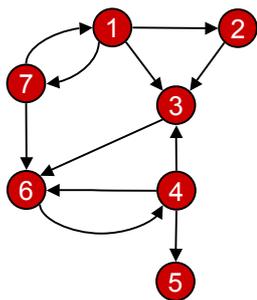
10

Routing: applicazione della teoria dei grafi

- Una rete è un insieme di nodi di commutazione interconnessi da collegamenti
- Per rappresentarla si possono usare i modelli matematici della teoria dei grafi
 - Sia V un insieme finito di **nodi**
 - Un **arco** è definito come una coppia di nodi (i,j) , $i,j \in V$
 - Sia E un insieme di archi
 - Un **grafo** G è definito come la coppia (V,E) e può essere
 - **orientato** se E consiste di coppie ordinate, cioè se $(i,j) \neq (j,i)$
 - **non orientato** se E consiste di coppie non ordinate, cioè se $(i,j) = (j,i)$
 - Se $(i,j) \in E$, il nodo j è **adiacente** al nodo i

11

Rappresentazione di grafi

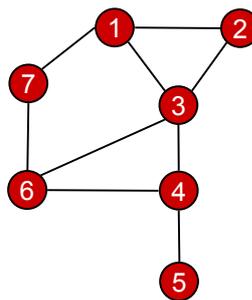


Grafo orientato

$$V = \{1,2,3,4,5,6,7\}$$

$$E = \{(1,2), (1,3), (1,7), (2,3), (3,6), (4,3), (4,5), (4,6), (6,4), (7,1), (7,6)\}$$

$$\text{Dimensioni: } |V|=7, |E|=11$$



Grafo non orientato

$$V = \{1,2,3,4,5,6,7\}$$

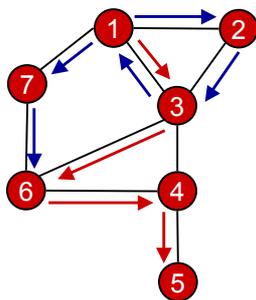
$$E = \{(1,2), (1,3), (1,7), (2,3), (3,4), (3,6), (4,5), (4,6), (6,7)\}$$

$$\text{Dimensioni: } |V|=7, |E|=9$$

12

Cammini e cammini semplici

- In un grafo $G=(V,E)$, un **cammino** (path) di lunghezza k dal nodo u al nodo v è una sequenza di $k+1$ nodi $p = (v_0, v_1, v_2, \dots, v_k)$ tali che $v_0=u$, $v_k=v$, e $(v_{i-1}, v_i) \in E$ per ogni $i=1, 2, \dots, k$
- Un cammino $p = (v_0, v_1, v_2, \dots, v_k)$ è **semplice** se contiene solo nodi distinti, cioè $v_i \neq v_j$ per ogni $i \neq j$



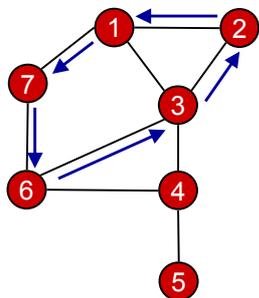
$p_1 = (1, 2, 3, 1, 7, 6)$ è un cammino di lunghezza 5

$p_2 = (1, 3, 6, 4, 5)$ è un cammino semplice di lunghezza 4

13

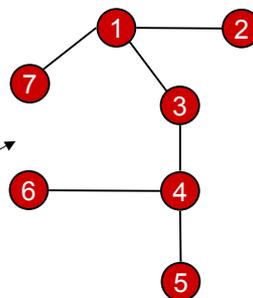
Cicli

- In un grafo $G=(V,E)$, un **ciclo** è un cammino di lunghezza k $p = (v_0, v_1, v_2, \dots, v_k)$ tale che $k \geq 3$ e $v_0=v_k$
- Un grafo è detto **aciclico** se non contiene cicli



$p = (1, 7, 6, 3, 2, 1)$ è un ciclo di lunghezza 5

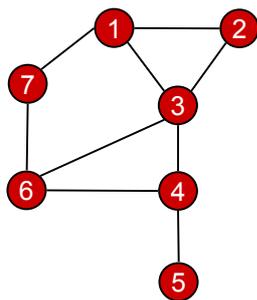
Grafo aciclico



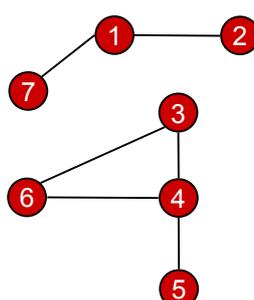
14

Connettività

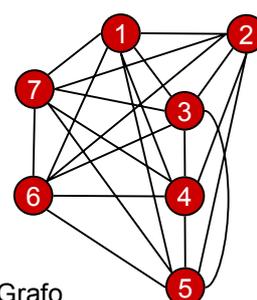
- Un grafo $G=(V,E)$ è **connesso** se per ogni coppia di nodi $u,v \in V$ esiste sempre un cammino da u a v
- Un grafo è detto **completamente connesso** se per ogni coppia di nodi $u,v \in V$, $(u,v) \in E$, cioè se tutti i nodi sono adiacenti tra loro (in tal caso $|E| = |V| \cdot (|V| - 1) / 2$)



Grafo connesso



Grafo non connesso

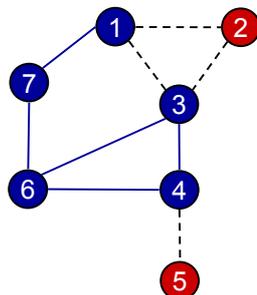


Grafo completamente connesso

15

Sottografi

- Un grafo $G'=(V',E')$ è un **sottografo** di $G=(V,E)$ se $V' \subseteq V$ e $E' \subseteq E$



$$V = \{1,2,3,4,5,6,7\}$$

$$V' = \{1,3,4,6,7\}$$

$$E = \{(1,2),(1,3),(1,7),(2,3),(3,4), \\ (3,6),(4,5),(4,6),(6,7)\}$$

$$E' = \{(1,7),(3,4),(3,6),(4,6),(6,7)\}$$

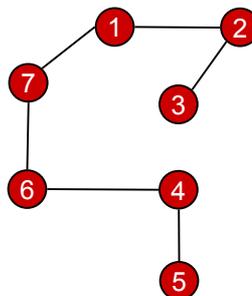
16

Alberi

- Un **albero** (tree) è un grafo connesso e aciclico

Sia $G=(V,E)$ un grafo non orientato;
le seguenti affermazioni sono equivalenti:

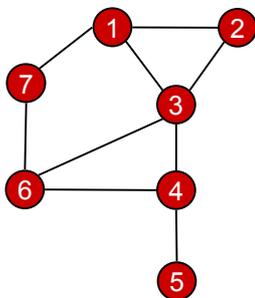
1. G è un albero
2. ogni coppia di nodi in V è connessa da un unico cammino semplice
3. G è connesso, ma se si rimuove un qualunque arco da E il grafo risultante è non connesso
4. G è connesso e $|E| = |V| - 1$
5. G è aciclico e $|E| = |V| - 1$
6. G è aciclico, ma se si aggiunge un qualunque arco ad E il grafo risultante contiene un ciclo



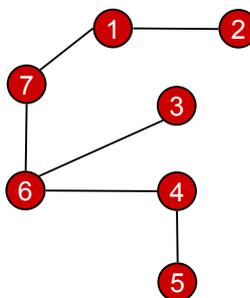
17

Spanning Tree

- Dato un grafo connesso non orientato $G=(V,E)$, uno **spanning tree** (albero di ricoprimento) di G è un sottografo $G'=(V',T)$ tale che:
 - G' è un albero
 - $V' = V$



Grafo G

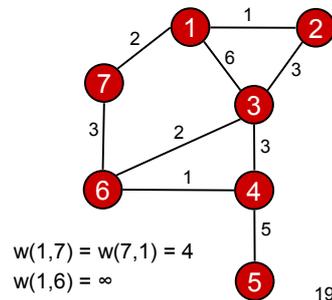
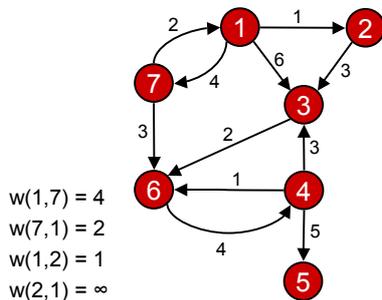


Spanning tree di G

18

Grafo pesato

- Un **grafo pesato** è un grafo $G=(V,E)$ tale che ad ogni arco $(i,j) \in E$ è associato un numero reale $w(i,j)$ chiamato **peso** (o costo, o distanza)
 - In un grafo non orientato vale sempre $w(i,j) = w(j,i)$
 - In un grafo orientato vale in generale $w(i,j) \neq w(j,i)$
 - Se $(i,j) \notin E$, allora $w(i,j) = \infty$
 - Per semplicità si assume $w(i,j) > 0$ per ogni arco $(i,j) \in E$

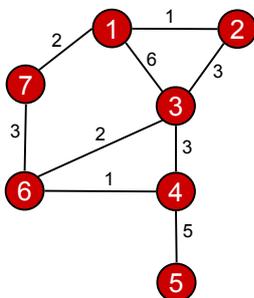


19

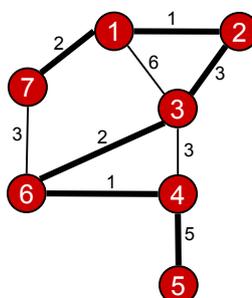
Minimum Spanning Tree (MST)

- Dato un grafo pesato connesso non orientato $G=(V,E)$, un **minimum spanning tree** (albero di ricoprimento minimo) di G è uno spanning tree $G'=(V,T)$ di peso minimo, cioè tale che

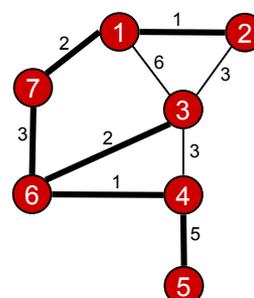
$$w(T) = \sum_{(i,j) \in T} w(i,j) \text{ è minimo}$$



Grafo pesato G



Un MST di G
 $w(T) = 14$



Un altro MST di G
 $w(T) = 14$

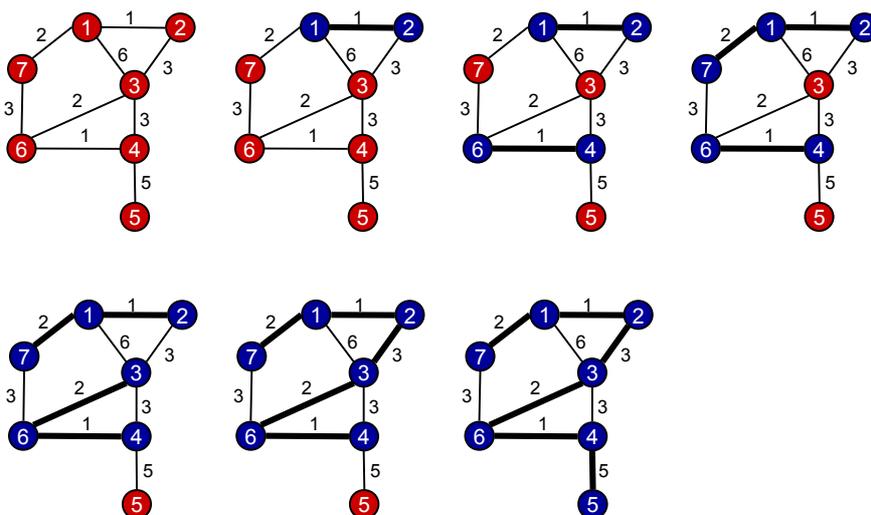
20

Algoritmi per il calcolo del MST

- Dato un grafo pesato connesso non orientato $G=(V,E)$, per trovare un MST di G esistono algoritmi di tipo “greedy” che ad ogni passo aggiungono un arco di E ad un sottografo A di G che è anche un sottografo di un MST, fino al ricoprimento di tutti i nodi
 - Algoritmo di **Kruskal** $\rightarrow O(E \log V)$
 - ordina gli archi di E secondo il peso crescente
 - parte da un sottografo A vuoto
 - aggiunge ad A l'arco di peso minore possibile, senza creare cicli
 - A può essere non connesso durante i passi intermedi
 - Algoritmo di **Prim** $\rightarrow O(E + V \log V)$
 - parte da un nodo radice come unico elemento di A
 - aggiunge l'arco connesso ad A di peso minore, senza creare cicli
 - A deve sempre essere un albero connesso durante i passi intermedi

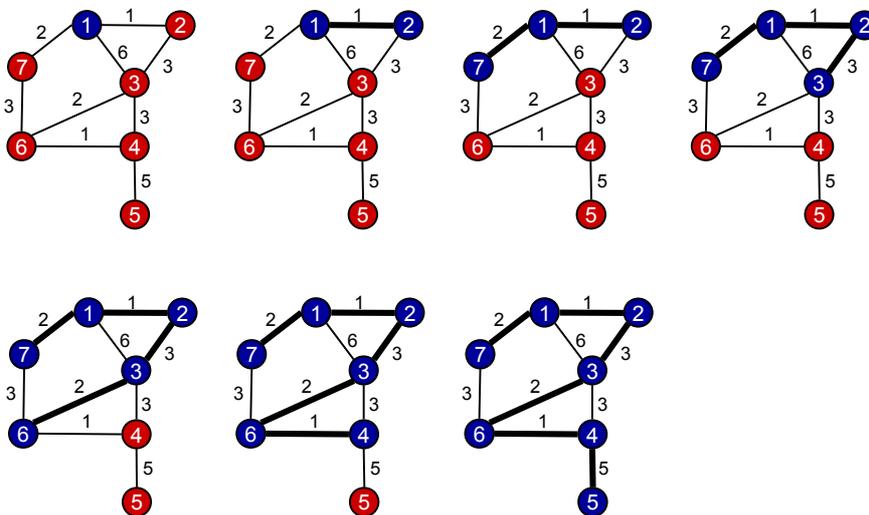
21

Algoritmo di Kruskal



22

Algoritmo di Prim



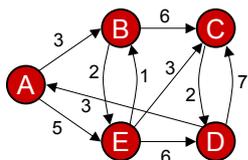
23

Shortest Path (SP)

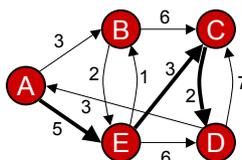
- Dato un grafo pesato orientato $G=(V,E)$, il peso del cammino $p = (v_0, v_1, v_2, \dots, v_k)$ è la somma dei pesi degli archi che lo costituiscono, cioè

$$w(p) = \sum_{i=1}^k w(v_{i-1}, v_i)$$

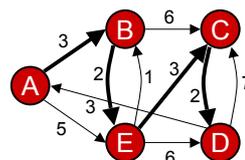
- Uno **shortest path** (cammino minimo) dal nodo u al nodo v di V è un cammino $p = (u, v_1, v_2, \dots, v)$ tale che $w(p)$ è minimo



Grafo pesato orientato



Uno SP da A a D
 $w(p) = 10$

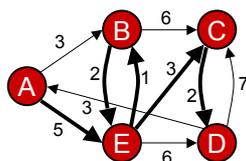


Un altro SP da A a D
 $w(p) = 10$

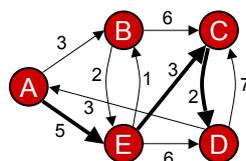
24

Cicli negli SP

- Uno shortest path non contiene cicli
 - $p = (v_0, v_1, \dots, v_k)$ è un cammino con peso $w(p)$
 - $c = (v_i, v_{i+1}, \dots, v_j)$ è un ciclo contenuto in p con peso $w(c)$
 - se si estrae c da p si ottiene $p' = (v_0, v_1, \dots, v_i, v_{j+1}, \dots, v_k)$ che è un cammino con peso minore di p , poiché $w(p') = w(p) - w(c)$



$p = (A, E, B, E, C, D)$
 $w(p) = 13$
 $c = (E, B, E)$
 $w(c) = 3$



$p' = (A, E, C, D)$
 $w(p') = 10$

25

Principio di ottimalità

- Dato un grafo pesato orientato $G=(V,E)$ e uno shortest path $p = (v_0, v_1, \dots, v_k)$ da v_0 a v_k , qualsiasi sottocammino $p_{ij} = (v_i, v_{i+1}, \dots, v_j)$ contenuto in p è anch'esso uno shortest path tra v_i e v_j
 - $p_{1i} = (v_0, v_1, \dots, v_i)$
 - $p_{ij} = (v_i, v_{i+1}, \dots, v_j)$
 - $p_{jk} = (v_j, v_{j+1}, \dots, v_k)$
 - $w(p) = w(p_{1i}) + w(p_{ij}) + w(p_{jk})$
 - se ci fosse un altro cammino p'_{ij} tra v_i e v_j tale che $w(p'_{ij}) < w(p_{ij})$, il cammino p' composto da p_{1i} , p'_{ij} e p_{jk} sarebbe un cammino tra v_0 e v_k tale che $w(p') < w(p)$, il che è assurdo

26

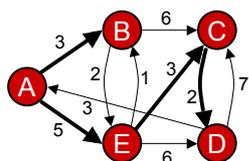
Algoritmi per il calcolo dello SP

- Dato un grafo pesato connesso orientato $G=(V,E)$ e un nodo sorgente $s \in V$, esistono algoritmi per trovare uno SP da s verso ogni altro nodo di V (single-source shortest path problem)
- Dall'esecuzione si ottiene come risultato, per ogni nodo $v \in V$
 - $d[v]$ = distanza del nodo v dal nodo sorgente s lungo uno SP
 - $\pi[v]$ = predecessore del nodo v lungo uno SP da s a v
- Inizializzazione: per ogni nodo $v \in V$
 - $d[v] = \infty$ se $v \neq s$, altrimenti $d[s] = 0$
 - $\pi[v] = \emptyset$
- Durante l'esecuzione si usa la tecnica del **rilassamento** (relaxation) di un generico arco (u,v)
 - verifica se si può migliorare il valore di $d[v]$ scegliendo u come predecessore
 - se $d[v] > d[u] + w(u,v)$, allora $\pi[v] = u$ e $d[v] = d[u] + w(u,v)$
- Gli algoritmi si differenziano sulla modalità di eseguire il rilassamento
 - Algoritmo di **Bellman-Ford** $\rightarrow O(E V)$
 - Algoritmo di **Dijkstra** $\rightarrow O(E + V \log V)$

27

Algoritmo di Bellman-Ford

- Dopo l'inizializzazione, l'algoritmo ripete $|V| - 1$ volte il rilassamento di tutti gli archi del grafo:
 - per i da 1 a $|V| - 1$
 - per ogni $(u,v) \in E$
 - relax(u,v);
- Si può dimostrare che alla fine del ciclo $d[v]$ e $\pi[v]$ sono quelli di uno SP, anche se l'algoritmo potrebbe convergere con meno iterazioni
- L'ordine con cui si rilassano gli archi è arbitrario



	d[v], $\pi[v]$				
	A	B	C	D	E
start	0, \emptyset	∞ , \emptyset	∞ , \emptyset	∞ , \emptyset	∞ , \emptyset
i = 1	0, \emptyset	3, A	9, B	11, C	5, A
i = 2	0, \emptyset	3, A	8, E	10, C	5, A
i = 3	0, \emptyset	3, A	8, E	10, C	5, A
i = 4	0, \emptyset	3, A	8, E	10, C	5, A

Ordine di rilassamento archi: (D,A),(A,B),(E,B),(B,C),(D,C),(E,C),(C,D),(E,D),(A,E),(B,E)

28

Algoritmo di Dijkstra

- Dopo l'inizializzazione, l'algoritmo crea un insieme S vuoto e ad ogni passo vi inserisce un nodo di $S' = V - S$ che ha distanza minima e rilassa tutti gli archi uscenti da tale nodo, interrompendosi quando $S = V$ e $S' = \emptyset$:

$S = \emptyset$;

$S' = V$;

finché $S' \neq \emptyset$

trova $u \in S'$ tale che $d[u] = \min \{d[v], v \in S'\}$;

$S = S \cup \{u\}$;

$S' = S' - \{u\}$;

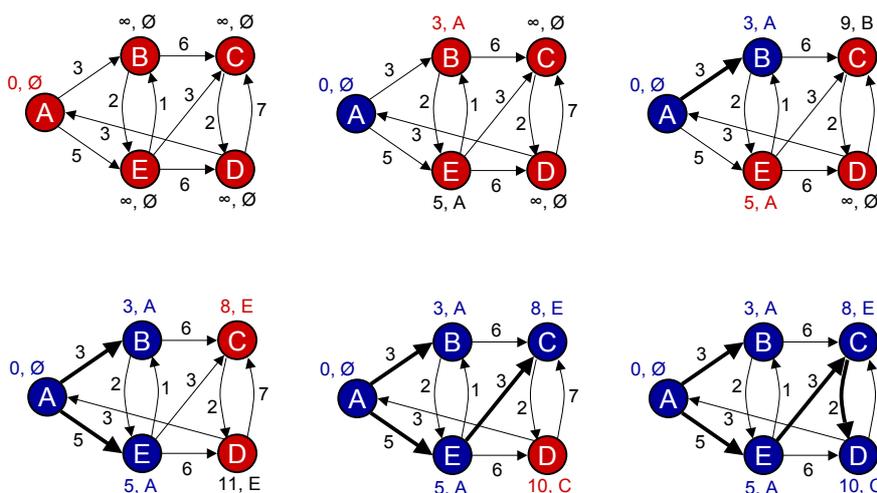
per ogni $v \in S'$ tale che $(u,v) \in E$

relax(u,v);

- Si può dimostrare che, alla fine del ciclo, per ogni nodo $v \in V$, $d[v]$ e $\pi[v]$ sono quelli di uno SP

29

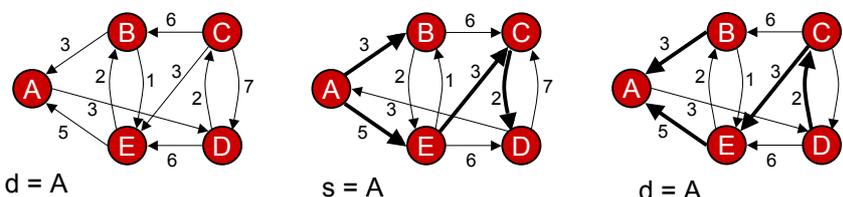
Algoritmo di Dijkstra



30

Single-destination shortest path problem

- Dato un grafo pesato connesso orientato $G=(V,E)$ e un nodo destinazione $d \in V$, esistono algoritmi per trovare uno SP da ogni altro nodo di V verso d
- Dall'esecuzione si ottiene come risultato, per ogni nodo $v \in V$
 - $d[v]$ = distanza del nodo v dal nodo destinazione d lungo uno SP
 - $\pi[v]$ = successore del nodo v lungo uno SP da v a d (next hop)
- Riconducibile al single-source shortest path problem sostituendo d con s e invertendo l'orientazione degli archi



31

Applicazione alle reti

- Ad una generica rete di telecomunicazioni si può facilmente associare un grafo orientato:
 - i nodi rappresentano i terminali ed i commutatori
 - gli archi rappresentano i collegamenti
 - l'orientamento degli archi rappresenta la direzione di trasmissione
 - il peso degli archi rappresenta il costo dei collegamenti, che può essere espresso in termini di
 - numero di nodi attraversati (ogni arco ha peso unitario)
 - distanza geografica
 - ritardo introdotto dal collegamento
 - inverso della capacità del collegamento
 - costo di un certo instradamento
 - una combinazione dei precedenti

32

Un semplice algoritmo di routing: il flooding

- **Flooding**: ogni nodo ritrasmette su tutte le porte di uscita ogni pacchetto ricevuto
- Un generico pacchetto verrà sicuramente ricevuto da tutti i nodi della rete e quindi anche da quello a cui è effettivamente destinato
- Dal momento che tutte le strade possibili sono percorse, il primo pacchetto che arriva a un nodo è quello che ha fatto la strada più breve possibile
- L'elaborazione associata a questo algoritmo è pressoché nulla
- Molto adatto quando si voglia inviare una certa informazione a tutti i nodi della rete (**broadcasting**)

33

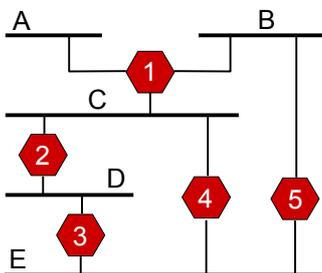
Miglioramenti del flooding

- E' necessario introdurre ulteriori regole per evitare la proliferazione dei pacchetti broadcast
- un nodo non ritrasmette il pacchetto nella direzione dalla quale è giunto
- un nodo ritrasmette un pacchetto una sola volta
 - ad ogni pacchetto viene associato un identificatore unico (l'indirizzo della sorgente e un numero di sequenza) e ciascun nodo mantiene in memoria una lista con gli identificativi dei pacchetti già trasmessi
 - bisogna evitare che la lista cresca all'infinito → contatore
 - quando un pacchetto già trasmesso giunge al nodo, viene ignorato
- Un instradamento più efficiente per il broadcasting si basa sull'uso di uno spanning tree
 - si trasmette solo lungo lo spanning tree
 - miglioramento per l'assenza di cicli

34

Transparent bridge

- Interconnessione di più LAN (anche di tipo diverso)
- Estensione del dominio di broadcast
- Confinamento dei domini di collisione
- Autoapprendimento degli indirizzi MAC e costruzione delle tabelle di instradamento
 - presenza di cicli deleteria (proliferazione di trame)
 - designazione di un bridge radice e costruzione di uno spanning tree



35

Deflection routing (hot potato)

- Quando un nodo riceve un pacchetto lo ritrasmette sulla linea d'uscita avente il minor numero di pacchetti in attesa di essere trasmessi
- E' adatto a reti in cui
 - i nodi di commutazione dispongano di spazio di memorizzazione molto limitato
 - si voglia minimizzare il tempo di permanenza dei pacchetti nei nodi
- I pacchetti possono essere ricevuti fuori sequenza
- Alcuni pacchetti potrebbero percorrere all'infinito un certo ciclo in seno alla rete, semplicemente perché le sue linee sono poco utilizzate
 - Si deve prevedere un meccanismo per limitare il tempo di vita dei pacchetti
- Non tiene conto della destinazione finale del pacchetto

36

Scelta ibrida (load sharing)

- Si legge su di una tabella la linea d'uscita preferenziale nella direzione della destinazione finale del pacchetto
- Il pacchetto viene posto nella coda di trasmissione per tale linea
 - se la coda non supera una soglia
 - e se non vi sono altri pacchetti richiedenti contemporaneamente tale linea d'uscita
- Altrimenti viene inviato sulla linea d'uscita avente coda di trasmissione più breve
- Vantaggio: in condizioni di basso carico della rete, l'instradamento non viene fatto a caso, ma sulla base della effettiva destinazione finale

37

Shortest path routing

- Si assume che ad ogni collegamento della rete possa essere attribuita una lunghezza
- La lunghezza
 - è un numero che serve a caratterizzare il peso di quel collegamento nel determinare una certa funzione di costo del percorso totale di trasmissione
- L'algoritmo cerca la strada di lunghezza minima fra ogni mittente e ogni destinatario
- Si applicano gli algoritmi di calcolo dello shortest path (Bellman-Ford e Dijkstra) in modalità
 - centralizzata
 - distribuita
 - **Distance Vector**
 - **Link State**

38

Tabelle di routing con alternative

- Dall'applicazione degli algoritmi di calcolo dello shortest path si ottengono anche percorsi alternativi
 - di peso minimo
 - di peso superiore
- Uso **probabilistico** della tabella di routing
 - Se si instrada sempre sulla linea più breve si creano nella rete dei cammini ad alto traffico e si lasciano alcune linee scariche
 - Si potrebbero instradare i pacchetti su tutte le uscite disponibili con probabilità inversamente proporzionale al peso del cammino corrispondente
 - Questa tecnica rende la distribuzione del traffico più uniforme sui vari link della rete

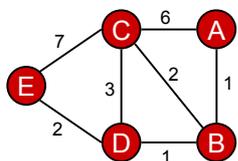
39

Protocolli Distance Vector

- Si basano sull'algoritmo di Bellman-Ford, in una versione dinamica e distribuita proposta da Ford-Fulkerson
- Ogni nodo scopre i suoi vicini e ne calcola la **distanza**
- Ad ogni passo, ogni nodo invia ai propri vicini un vettore contenente la stima della sua distanza da tutti gli altri nodi della rete (quelli di cui è a conoscenza)
- Ogni nodo può così eseguire un'operazione di "rilassamento" verso ogni altro nodo ed eventualmente aggiornare la stima della distanza e il next-hop
- E' un protocollo semplice e richiede poche risorse
- Problemi:
 - convergenza lenta, partenza lenta (cold start)
 - problemi di stabilità: conto all'infinito

40

Esempio: calcolo delle tabelle di routing



Distance Vector iniziali: $DV(i) = \{(i,0)\}$, per $i = A,B,C,D,E$

Distance Vector dopo la scoperta dei vicini:

$DV(A) = \{(A,0), (B,1), (C,6)\}$

$DV(B) = \{(A,1), (B,0), (C,2), (D,1)\}$

$DV(C) = \{(A,6), (B,2), (C,0), (D,3), (E,7)\}$

$DV(D) = \{(B,1), (C,3), (D,0), (E,2)\}$

$DV(E) = \{(C,7), (D,2), (E,0)\}$

Evoluzione delle tabelle di routing

1. A riceve DV(B)

dest	Costo, next hop
A	0
B	1, B
C	3, B
D	2, B

2. A riceve DV(C)

dest	Costo, next hop
A	0
B	1, B
C	3, B
D	2, B
E	10, B

3. B riceve DV(D)

dest	Costo, next hop
A	1, A
B	0
C	2, C
D	1, D
E	3, D

4. A riceve DV(B)

dest	Costo, next hop
A	0
B	1, B
C	3, B
D	2, B
E	4, B

41

Cold start e tempo di convergenza

- Allo start up le tabelle dei singoli nodi contengono solo l'indicazione del nodo stesso a distanza 0
 - i distance vector scambiati al primo passo contengono solo queste informazioni
- Da qui in poi lo scambio dei distance vector permette la creazione di tabelle sempre più complete
- L'algoritmo converge al più dopo un numero di passi pari al numero di nodi della rete
- Se la rete è molto grande il tempo di convergenza può essere lungo.
- Cosa succede se lo stato della rete cambia in un tempo inferiore a quello di convergenza dell'algoritmo?
 - Risultato imprevedibile → si ritarda la convergenza

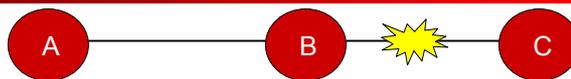
42

Bouncing effect

- Il link fra due nodi A e B cade
- A e B si accorgono che il collegamento non funziona e immediatamente pongono ad infinito la sua lunghezza
- Se altri nodi hanno nel frattempo inviato anche i loro vettori delle distanze, si possono creare delle incongruenze temporanee, di durata dipendente dalla complessità della rete
 - ad esempio A crede di poter raggiungere B tramite un altro nodo C che a sua volta passa attraverso A
- Queste incongruenze possono dare luogo a cicli, per cui due o più nodi si scambiano datagrammi fino a che non si esaurisce il TTL o finché non si converge nuovamente

43

Count to infinity



- Situazione iniziale: $D_{AB} = 1$, $D_{AC} = 2$ e $D_{BC} = 1$
 - Link BC va fuori servizio
 - B riceve il DV di A che contiene l'informazione $D_{AC} = 2$, per cui esso computa una nuova $D'_{BC} = D_{BA} + D_{AC} = 3$
 - B comunica ad A la sua nuova distanza da C
 - A calcola la nuova distanza $D_{AC} = D_{AB} + D'_{BC} = 4$
 - ...
- La cosa può andare avanti all'infinito
 - Si può interrompere imponendo che quando una distanza assume un valore $D_{ij} > D_{max}$ allora si suppone che il nodo destinazione J non sia più raggiungibile
- Inoltre si possono introdurre meccanismi migliorativi
 - **Split horizon**
 - **Triggered update**

44

Split horizon

- Split horizon è una tecnica molto semplice per risolvere in parte i problemi suddetti
 - se A instrada i pacchetti verso una destinazione X tramite B, non ha senso per B cercare di raggiungere X tramite A
 - di conseguenza non ha senso che A renda nota a B la sua distanza da X
- Un algoritmo modificato di questo tipo richiede che un router invii informazioni diverse ai diversi vicini
- Split horizon in forma semplice:
 - A omette la sua distanza da X nel DV che invia a B
- Split horizon with poisonous reverse:
 - A inserisce tutte le destinazioni nel DV diretto a B, ma pone la distanza da X uguale ad infinito

45

Triggered update

- Una ulteriore modifica per migliorare i tempi di convergenza è relativa alla tempistica con cui inviare i DV ai vicini
 - i protocolli basati su questi algoritmi richiedono di inviare periodicamente le informazioni delle distanze ai vicini
 - è possibile che un DV legato ad un cambiamento della topologia parta in ritardo e venga sopravanzato da informazioni vecchie inviate da altri nodi
- Triggered update: un nodo deve inviare immediatamente le informazioni a tutti i vicini qualora si verifichi una modifica della propria tabella di instradamento

46

Protocolli Path Vector

- Vari rimedi proposti in realtà non sono davvero risolutivi
 - Sono ancora presenti situazioni patologiche in cui i protocolli Distance Vector convergono troppo lentamente o non convergono affatto
- Evoluzione: protocolli **Path Vector**
 - Il vettore che ogni router manda ai vicini contiene, oltre alle distanze dagli altri nodi, anche l'intero cammino che il pacchetto deve seguire
 - Il router ignora tutti i cammini dove compare lui stesso
 - Maggiori informazioni da scambiare, ma si evitano i cicli

47

Protocolli Link State

- Ogni nodo della rete si procura un'immagine della topologia della rete
- Sulla base di tale immagine calcola le tabelle di routing utilizzando un determinato algoritmo
- Il protocollo di routing ha come scopo fondamentale quello di permettere ad ogni nodo di crearsi l'immagine della rete
 - scoperta dei nodi vicini
 - raccolta di informazioni dai vicini
 - diffusione delle informazioni raccolte a tutti gli altri nodi della rete

48

Raccolta delle informazioni

- Ogni router deve comunicare con i propri vicini ed “imparare” i loro indirizzi
 - **Hello Packet**
- Deve poi misurare la distanza dai vicini
 - **Echo Packet**
- In seguito ogni router costruisce un pacchetto con lo stato delle linee (**Link State Packet** o LSP) che contiene
 - la lista dei suoi vicini
 - le lunghezze dei collegamenti per raggiungerli

49

Diffusione ed elaborazione delle informazioni

- I pacchetti LSP devono essere trasmessi da tutti i router a tutti gli altri router della rete
 - Si usa il protocollo Flooding
 - A tal fine nel pacchetto LSP occorre aggiungere
 - L'indirizzo del mittente
 - Un numero di sequenza
 - Una indicazione dell'età del pacchetto
- Avendo ricevuto i LSP da tutti i router, ogni router è in grado di costruirsi un'immagine della rete
 - Tipicamente si usa l'algoritmo di Dijkstra per calcolare i cammini minimi verso ogni altro router

50

Distance Vector - Link State: confronto

- Distance Vector
 - Semplici da implementare
 - Richiedono in genere una quantità di memoria inferiore, in particolare se la connettività della rete è bassa, e minori risorse di calcolo
 - Convergenza lenta
- Link State
 - Offrono maggiori funzionalità in termini di gestione di rete
 - La topologia generale della rete e i cammini al suo interno possono essere ricavati da qualunque router
 - La velocità di convergenza è solitamente maggiore
 - Maggiore velocità di adattamento ai cambi di topologia
 - Il flooding di pacchetti LSP può provocare un aumento significativo di traffico

51

Routing gerarchico

- Nel caso di reti di grandi dimensioni non è possibile gestire le tabelle di routing per l'intera rete in tutti i router, in questo caso il routing deve essere **gerarchico**:
 - la rete viene ripartita in porzioni, chiamate per ora aree di routing
 - i router all'interno di un'area sono in grado di effettuare l'instradamento relativamente alla sola area
 - per destinazioni al di fuori dell'area si limitano ad inviare i pacchetti a dei router "di bordo" che sono a conoscenza della topologia esterna dell'area
 - i router "di bordo" si occupano solamente dell'instradamento dei pacchetti fra aree
- In linea di principio la ripartizione può essere effettuata tante volte quante si vuole creando più livelli nella gerarchia di routing

52