PAS 2013/14
SISTEMI E RETI DI CALCOLATORI PER L'INSEGNAMENTO
Lezione n.3
IL LIVELLO NETWORK

04/06/2014

Laura Ricci

LIVELLO IP: CARATTERISTICHE GENERALI

- Livello Network = livello IP
- Modello di servizio di IP: invio di datagrams secondo la filosofia best effort.
- datagram (pacchetto IP) : contiene informazioni sufficienti affinchè la rete (i routers) possa inoltrarlo verso la destinazione
- best effort: IP fornisce un servizio minimale. Motivazione: possibilità di eseguire il protocollo su reti che utilizzano diverse tecnologie

IP attualmente può essere eseguito su qualsiasi rete, anche su reti con tecnologie non esistenti al momento della sua definizione

metafora: "IP può operare su una rete che trasporta i messaggi mediante piccioni viaggiatori"

LIVELLO IP: FORMATO DEL PACCHETTO

LIVELLO IP: FORMATO DEL PACCHETTO

IP Version: IPV4 / IPV6

TOS (Type of Service) Consente un trattamento differenziato dei pacchetti.
Esempio: un particolare valore di TOS indica che il pacchetto ha una priorità maggiore rispetto agli altri.
Utile per distinguere per distinguere tipi diversi di traffico (traffico real time, messaggi per la gestione della rete,..)

TTL – Time to Live: consente di limitare la diffusione del pacchetto sulla rete

- valore iniziale impostato dal mittente
- quando il pacchetto attraversa un router, il valore viene decrementato
- quando il valore diventa 0, il pacchetto viene scartato

Introdotto inizialmente per evitare percorsi circolari infiniti del pacchetto.

Utilizzato anche per limitare la diffusione del pacchetto nel multicast

PACCHETTI IP: FRAMMENTAZIONE

Trasmissione di pacchetti IP su reti fisiche diverse ed eterogenee:

- quando un pacchetto P viene spedito su una rete fisica R, P deve essere incapsulato in un pacchetto P_{link}.

- la dimensione di P_{link} dipende dal livello link della rete fisica R, in particolare dall' MTU (Maximum Transfer Unit) di quella rete

- Reti fisiche diverse $\Rightarrow$ MTU diversi

  Ethernet: MTU = 1500 bytes

  FDDI : MTU = 45500 bytes

- la dimensione di P_{link} in genere è diversa da quella di P

PACCHETTI IP: FRAMMENTAZIONE

Approcci per la gestione di links eterogenei

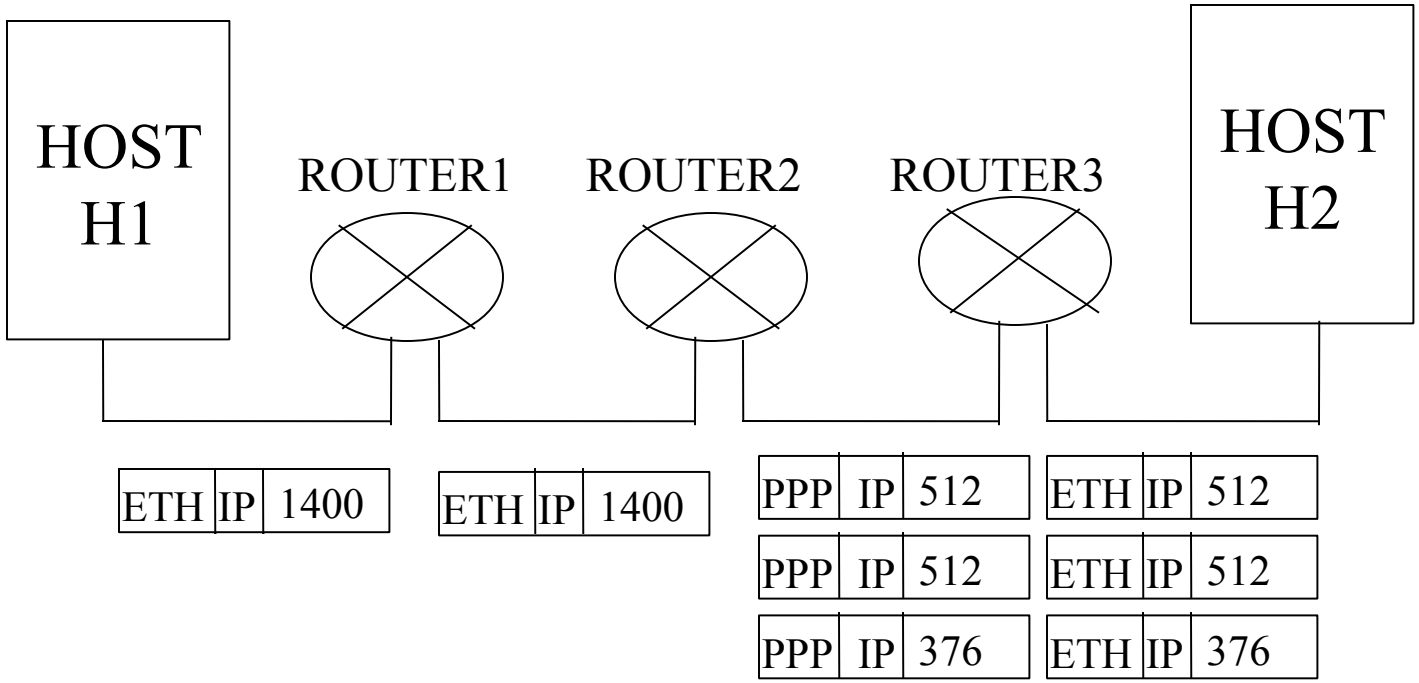
- definire la dimensione massima di un pacchetto IP come la minima dimensione dei pacchetti supportata a livello link dalle rete fisiche

svantaggi:

- difficoltà relativa alla definizione a priori di un limite inferiore alla dimensione dei pacchetti a livello link (specie per reti in espansione come Internet)
- spreco di risorse

- fragmentazione: dividere un datagram IP in una o più parti (fragmenti) nel caso in cui l'MTU di una rete fisica sia inferiore alla dimensione del datagram.

PACCHETTI IP: FRAMMENTAZIONE



PACCHETTI IP: FRAMMENTAZIONE

- la frammentazione del pacchetto viene effettuata dai routers
- la ricostruzione del pacchetto avviene nell'host finale (un pacchetto una volta frammentato viene ricostruito solamente quando arriva a destinazione)
- ricostruzione del pacchetto frammentato:
  - utilizza i campi identificatore, flag, offset contenuti nel pacchetto IP
  - identificatore: identifica in modo univoco i pacchetti generati da un host. Quando un router fragmenta un pacchetto assegna a tutti i segmenti generati l'identificatore del pacchetto da cui provengono
  - offset: identifica la posizione del fragmento all'interno del datagram originale
  - flag M (more): indica se questo è l'ultimo fragmento del pacchetto IP

PACCHETTI IP: FRAMMENTAZIONE

IP	1400

Pacchetto IP, identificatore 233

PPP IP 512

Fragmento 1, identificatore 233, offset 0, M=1

PPP IP 512

Fragmento 2, identificatore 233, offset 512, M=1

PPP IP 376

Fragmento 3, identificatore 233, offset 1024, M=0

CLASSFULL ADDRESSING

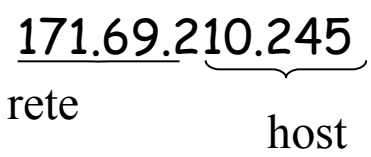
Schemi di indirizzamento IP

- Classful Addressing
- Subnetting
- Classless Addressing CIDR

Classfull Addressing:

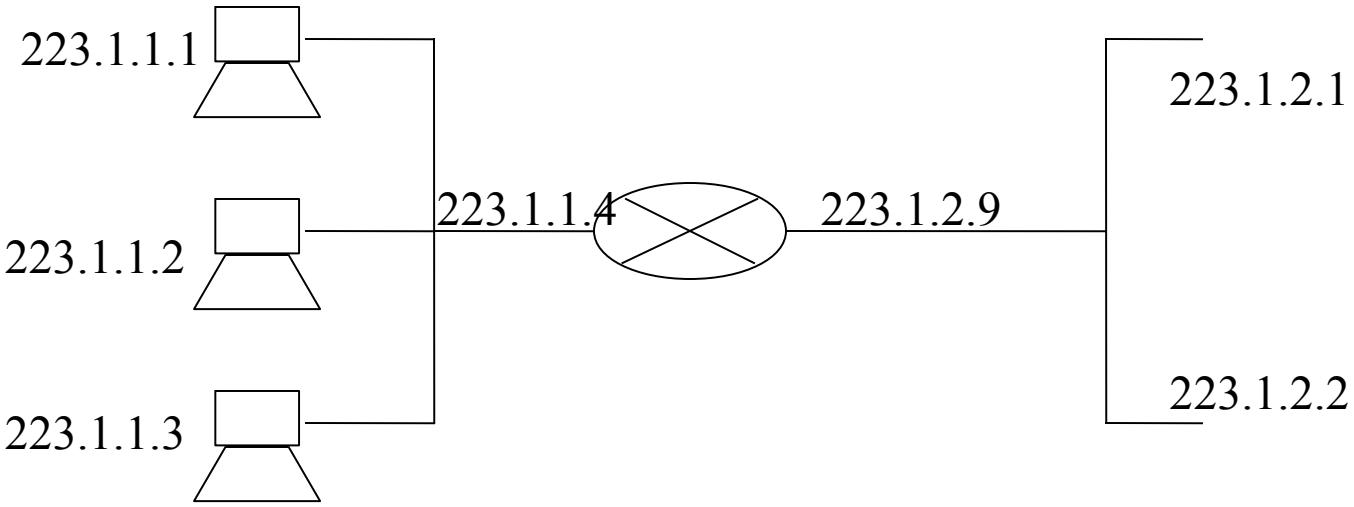
ogni indirizzo IP rappresenta una gerarchia a due livelli

- la prima parte dell'indirizzo identifica la rete fisica a cui appartiene l'host individuato da quell'indirizzzo
- la seconda parte individua l'host



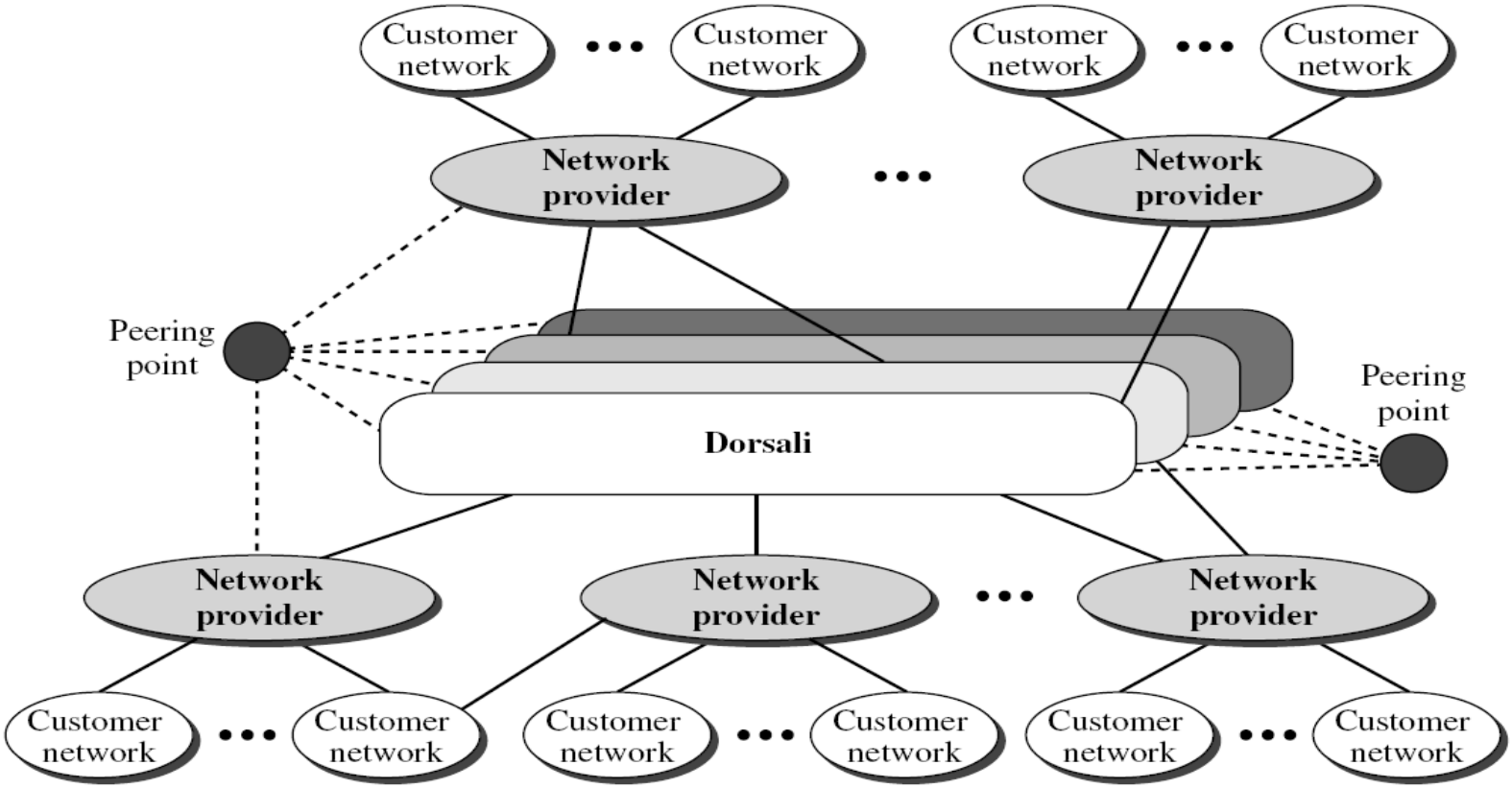
CLASSFULL ADDRESSING

- Tutti gli hosts ed i routers che condividono lo stesso indirizzo di rete sono connessi alla stessa rete fisica



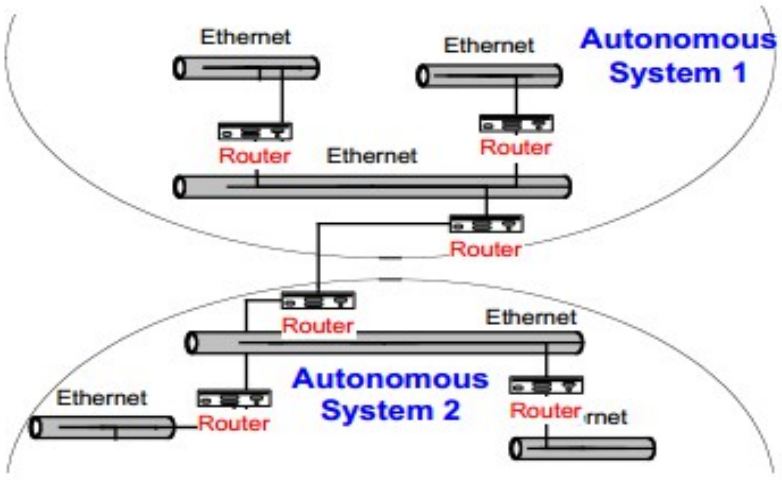
- ogni rete fisica connessa ad Internet ha almeno un router che è connesso ad un'altra rete fisica
- un router possiede un insieme di interfacce di rete

LA STRUTTURA DI INTERNET



LA STRUTTURA DI INTERNET

- Autonomous System (AS) o routing domain è una regione di Internet amministrata da una singola entità
  - i prefissi di tutti gli host sono caratterizzati da un prefisso comune
  - le politiche di routing appartengono ad un dominio amministrativo unico
- Esempi:
  - IBM corporate network
  - in generale gestite da grosse compagnie, università, enti



LA STRUTTURA DI INTERNET

- AS diversi diversi possono utilizzare algoritmi di routing diversi

- Routing all'interno di un AS Intra-AS Routing:
  - focus sulla performance
  - intra-domain or internal gateway (IGP) routing
  - l'amministratore decide quale protocollo di roouting adottare (interior gateway protocol)
  - decisioni locali, non visibili all'esterno

- Routing tra AS diversi
  - focus sulla politica di instradamento
  - inter-domain or external gateway (EGP) routing

INOLTRO DI PACCHETTI

- routing: meccanismo di inotro utilizzato da routers ed hosts per decidere su quale porta di uscita inviare un pacchetto ricevuto su una porta di ingresso

- il meccanismo di inoltro sfrutta un insieme di tabelle di inoltro che contengono un insieme di coppie

  (numero di rete, interfaccia di uscita, prossimo router)

- indirizzamento gerarchico
  - nelle tabelle dei routers vengono memorizzati solo gli indirizzi delle reti, piuttosto che gli indirizzi dei singoli hosts della rete.
  - il router associato alla rete destinataria invia il messaggio all'host selezionato.

- indirizzamento gerarchico= aumento della scalabilità del sistema

INTERNETWORKS (INTERNETS)

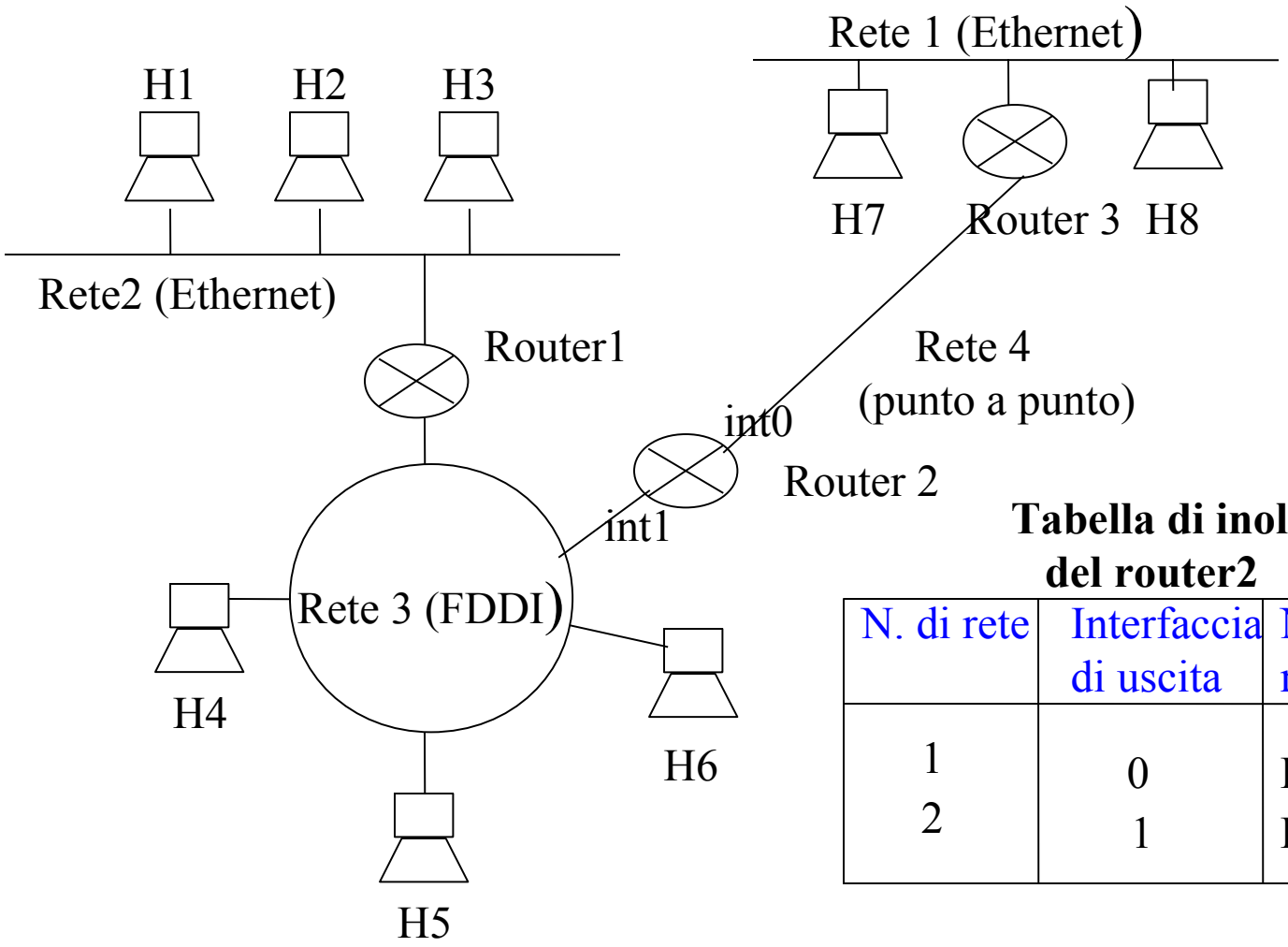


Tabella di inoltro del router2

N. di rete	Interfaccia di uscita	Next router
1	0	Router3
2	1	Router1

INOLTRO DEI PACCHETTI

Algoritmo di inoltro eseguito dal nodo N (host o router)

NetworkAdd = indirizzo di rete contenuto nel pacchetto da inoltrare

- Networkadd è uguale all' indirizzo di rete di una delle interfacce di N

  in questo caso il destinatario si trova sulla stessa rete del mittente, il pacchetto può essere consegnato direttamente dal livello link

- Networkadd è contenuto nella tabella di inoltro di N

  in questo caso il pacchetto va consegnato al router next router

- altrimenti

  il pacchetto va consegnato ad un default router

  (ad esempio un host può avere un default router a cui consegnare tutti i pacchetti non destinati alla rete loacle su cui tale host è connesso)

INOLTRO DEI PACCHETTI

- la trasmissione di un pacchetto IP su una rete fisica avviene mediante il livello link

- ogni nodo connesso ad una rete fisica possiede un indirizzo detto indirizzo fisico o MAC (media acces control)

- quando un pacchetto IP viene spedito su una rete fisica va utilizzato il MAC del nodo destinazione

- traduzione indirizzo IP–MAC address avviene mediante

  ARP (Address Resolution Protocol)

- Pacchetto IP incapsulato in un frame a livello link che contiene il MAC address del prossimo router o dell'host destinatario

INSTRADAMENTO (ROUTING)

- Autonomous System Architecture
  - la rete viene vista come un insieme di gruppi indipendenti
  - ogni gruppo corrisponde ad un sistema autonomo(AS)
  - ogni AS comprende un gruppo di routers gestiti in modo indipendente da una organizzazione
  - il tipo di routing utilizzato all'interno di un sistema autonomo è diverso da quello utilizzato tra sistemi autonomi diversi

- All'interno di un sistema autonomo, si distinguono due tipi di routers
  - routers interni: sono connessi unicamente ad altri routers dello stesso sitema autonomo. Eseguono solo interior routing protocols
  - border routers: sono connessi sia a routers interni che a routers di altri sistemi autonomi. Eseguono interior ed exterior routing protocols

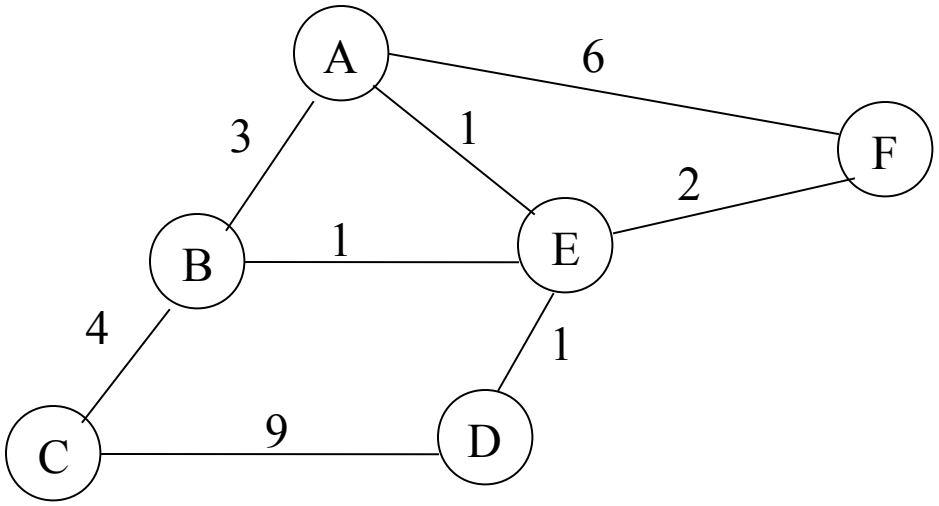
INTERIOR ROUTING PROTOCOLS

- assunzione base: la rete è rappresentata come un grafo, in cui i nodi rappresentano routers, gli archi linee fisiche di collegamento

- Ad ogni arco è associato un valore che indica il costo dell'invio di un pacchetto su quel link. Nel caso più semplice i costi sono unitari ed il percorso di costo minimo è quello che attraversa il minor numero di routers

- Assegnazione dei costi ai link: possono essere assegnati in base
  - traffico sulla linea
  - alla latenza delle linea
  - alla banda delle linea

ROUTING PROTOCOLS



Problema: dato un nodo mittente N1 ed un nodo destinatario N1, trovare il cammino di costo minimo tra N1 ed N2 (costo di un cammino= somma dei costi associati agli archi che definiscono il cammino)

INTERIOR ROUTING PROTOCOLS

Algoritmi di routing intradominio: tutti gli algoritmi vengono eseguiti in modo distribuito = ogni nodo esegue lo stesso algoritmo, i nodi cooperano scambiandosi informazioni sulla topologia della rete e sui costi associai ai links

- Algoritmi di routing con conoscenza globale: ogni nodo deve avere conoscenza del grafo dell'intera rete e dei costi associati ai links.

  Algoritmi basati sullo Stato delle Linee (OSPF)

- Algoritmi di routing decentralizzati: ogni nodo ha solo conoscenza dei costi dei costi relativi ai links che lo collegano ai nodi vicini

  Algoritmi basati su vettori distanza (RIP)

CLASSIFICAZIONE PROTOCOLLI DI ROUTING

Classificazione dei protocolli di routing:

- interior routing protocols: utilizzati all'interno di un unico sistema autonomo
  - RIP (Routing Information Protocol)
  - OSPF (Open shortestest Path First)
  - IS-IS
  - EIGRP
  - Protocolli proprietari

- Exterior gateway protocols : utilizzati tra sistemi autonomi diversi
  - BGP

CLASSIFICAZIONE ALGORITMI DI ROUTING

- Distance-Vector
  - vettori (destinazione-distanza) inviati ai vicini

    "Tell your neighbors about the rest of the network"
  - distanza definita in termini di diverse metriche: hop count, delay, banda
  - utilizza Distributed Bellman-Ford path selection algorithm
  - protocolli: Routing Information Protocol (RIP)

- Link-State
  - flooding della descrizione dei propri links (link state)

    "Tell the rest of the network about your neighbors"
  - usa l'algoritmo di selezione dei path di Dijkstra
  - protocol: Open Shortest Path First (OSPF)

OVERVIEW

Bellman-Ford shortest path algorithm

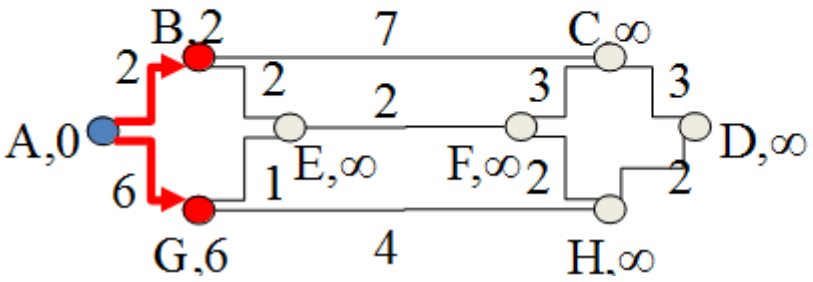
- Distributed Bellman-Ford (DBF) e Routing Information Protocol (RIP)
- Uso di algoritmi shortest-path algorithms in reti reali
  - La debolezza principale del DBF: counting-to-infinity

Dijkstra algorithm

- Dijkstra e OSPF

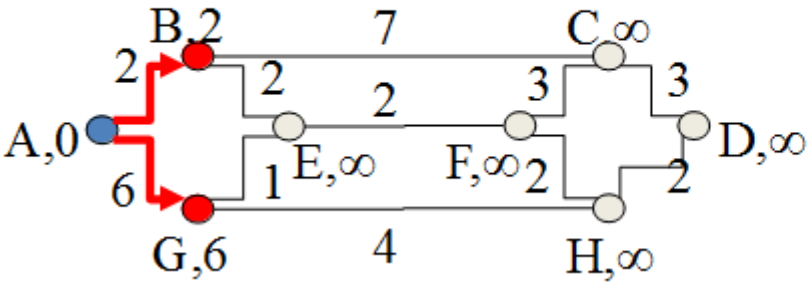
L'ALGORITMO DI BELLMAN-FORD

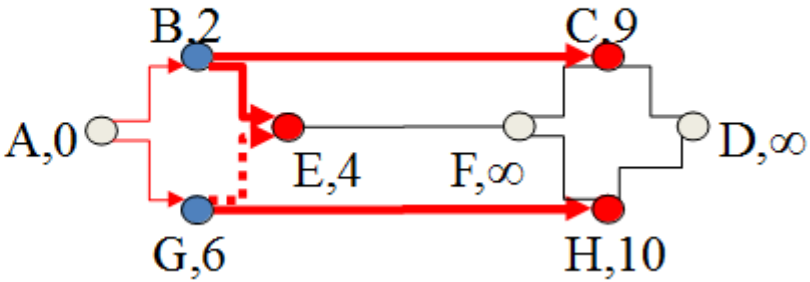
Bold = this iteration, Dashed = rejected



L'ALGORITMO DI BELLMAN-FORD

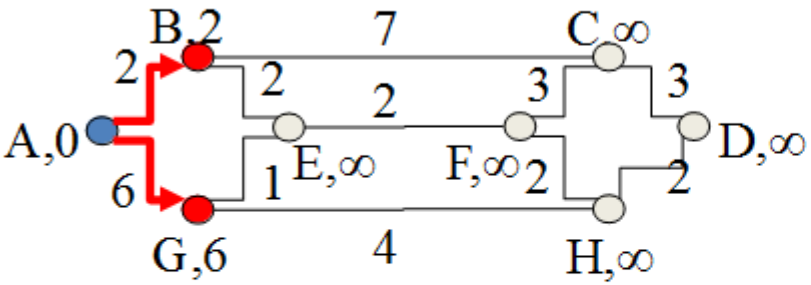
Bold = this iteration, Dashed = rejected

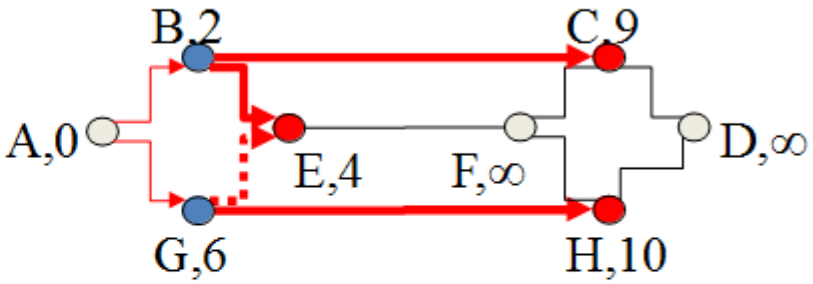


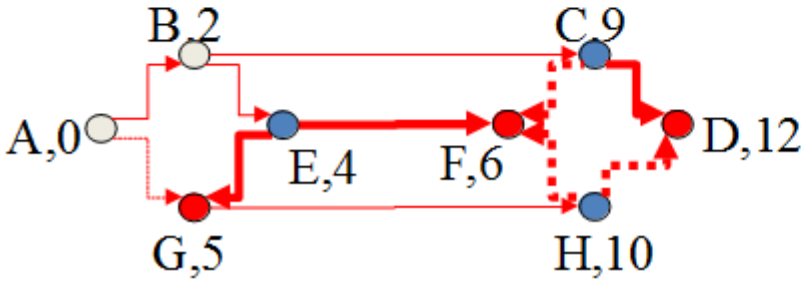


L'ALGORITMO DI BELLMAN-FORD

Bold = this iteration, Dashed = rejected

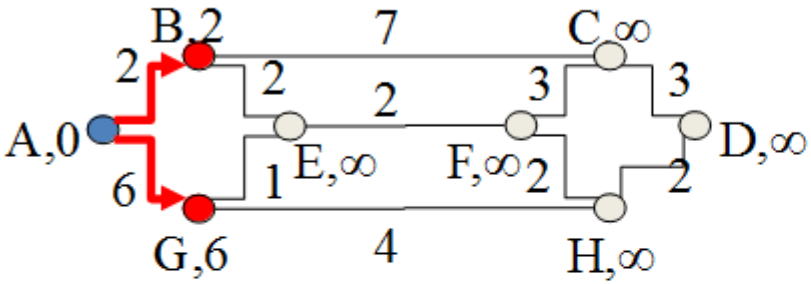


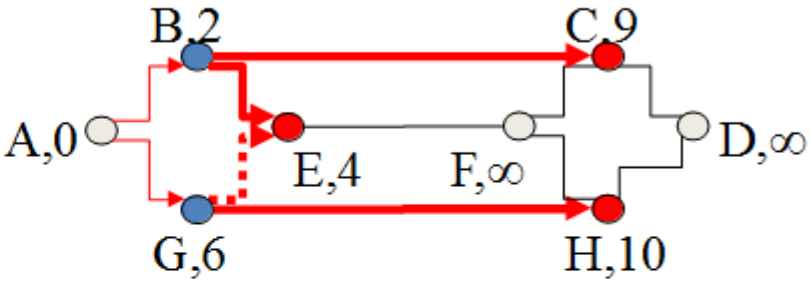


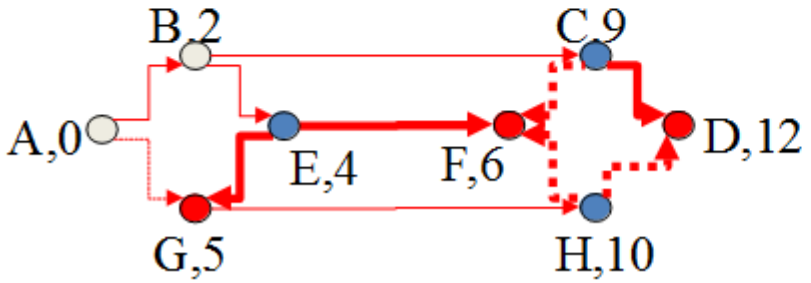


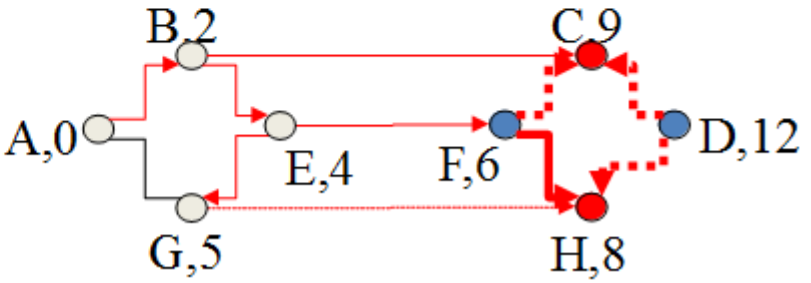
L'ALGORITMO DI BELLMAN-FORD

Bold = this iteration, Dashed = rejected



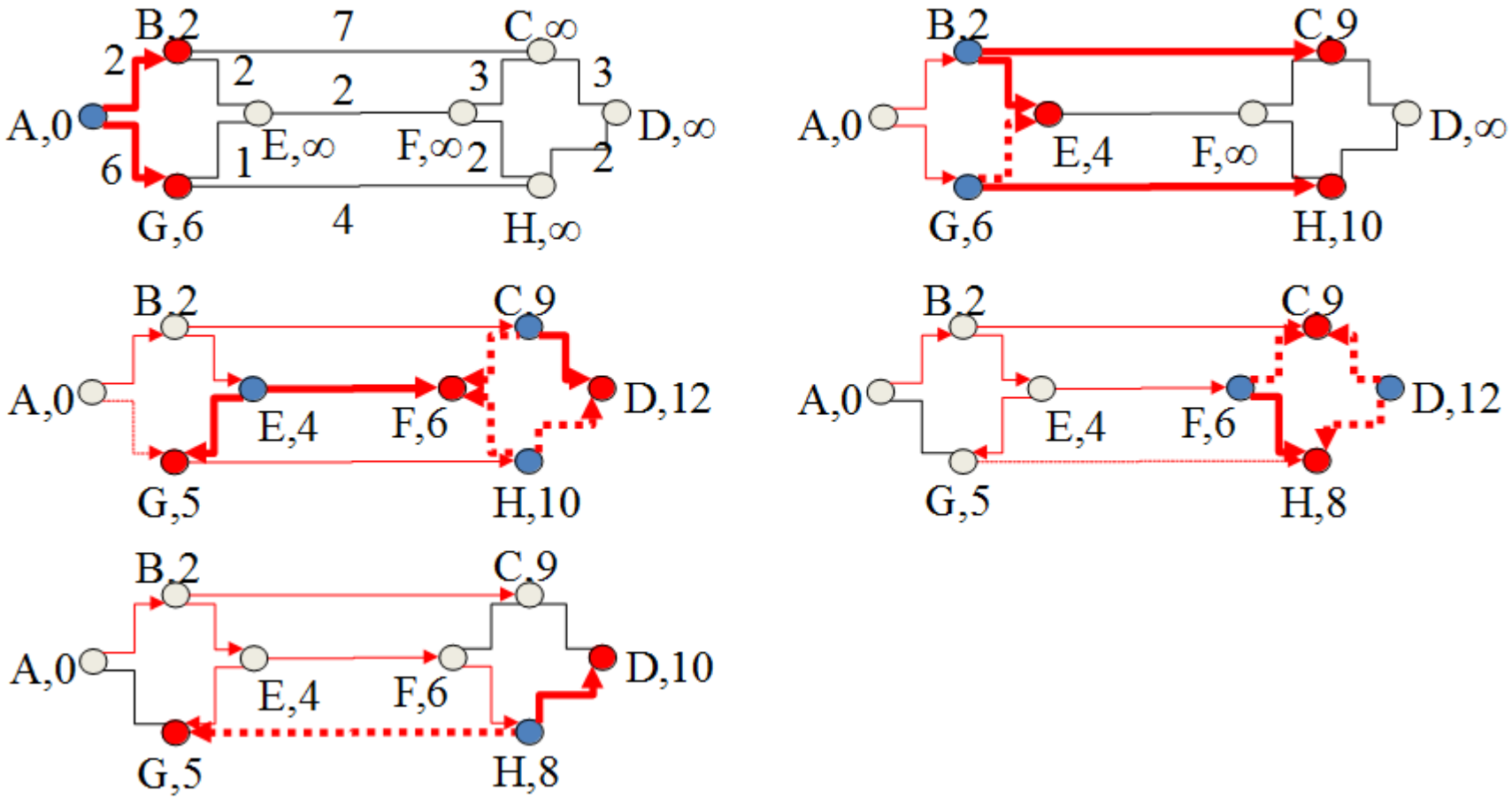




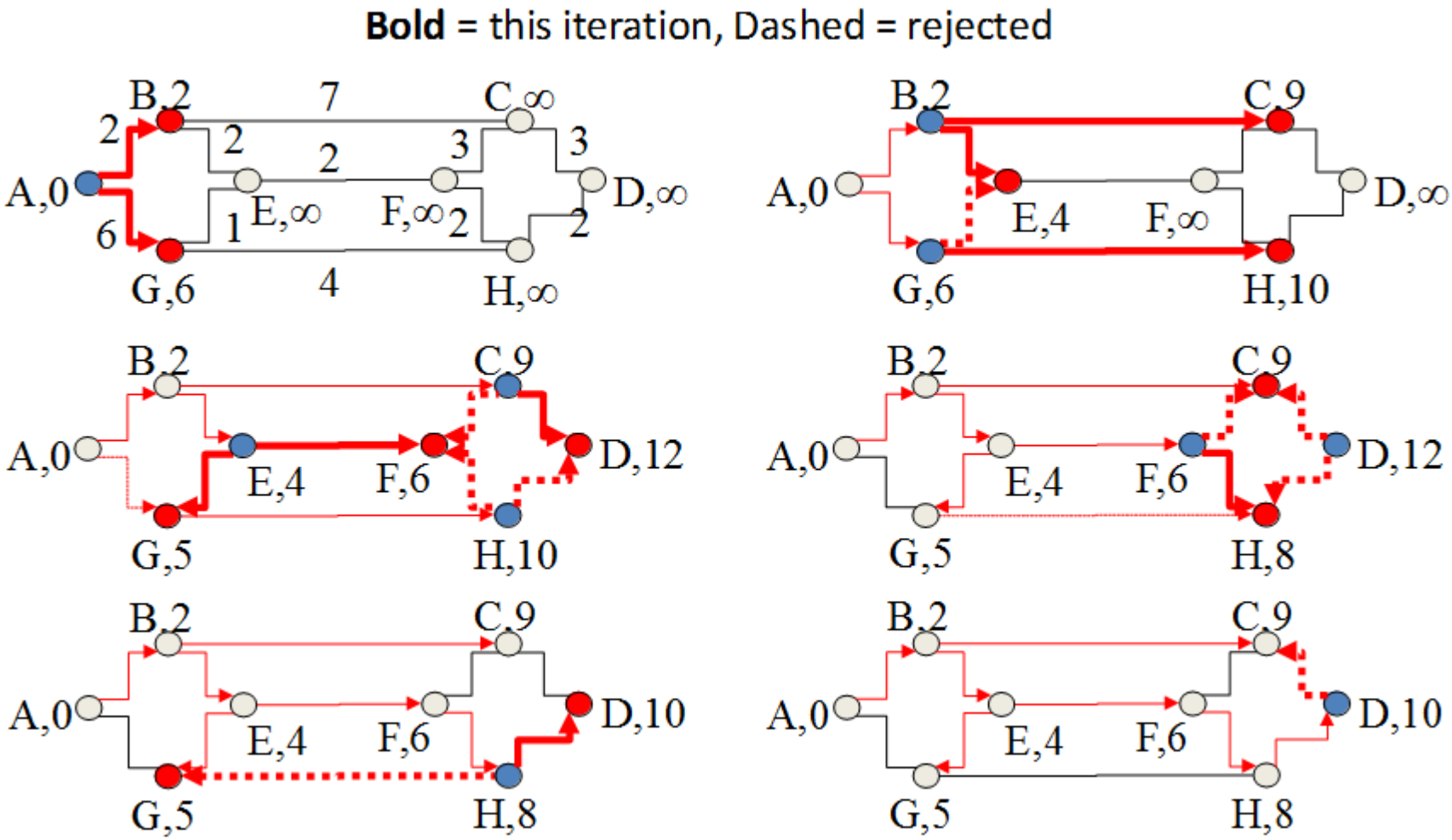


L'ALGORITMO DI BELLMAN-FORD

Bold = this iteration, Dashed = rejected



L'ALGORITMO DI BELLMAN-FORD



L'ALGORITMO DI BELLMAN-FORD

- Idea generale:
  - ricerca breadth-first dei cammini, incremento del numero di hops, ricerca dei cammini migliori dal nodo sorgente a tutte le destinazioni
  - termina quando non è possibile trovare un miglioramento
- L'algoritmo in breve:
  - R: insieme dei nodi per cui è stato trovato un cammino
  - passo iterativo:
    - analizza tutti i nodi vicini di R, sia questo insieme R'
    - se si individua un nodo n in R' per cui un cammino tramite uno dei nodi in R è migliore del cammino migliore attuale associato al nodo n, si aggiorna il cammino di n in R
  - ripeti il passo iterativo viene ripetuto fino a che nessun cammino dei nodi in R può essere migliorato
- alla fine da R si ricava la routing table

L'ALGORITMO DI BELLMAN-FORD

algorithm CBF **run at node** i
  begin
1 $d_i \leftarrow 0; p_i \leftarrow i;$
2 **for each** $\{x \in V - \{i\}\};$
3 $d_x \leftarrow \infty; p_x \leftarrow \emptyset;$
4 $R \leftarrow \{i\};$
  do
5 $R' \leftarrow \{j \mid x \in R, (x,j) \in E, d_j > d_x + \omega_{xj}\};$
6 **for each** $\{j \in R'\}$
    begin
7 $d_j \leftarrow Min\{d_x + \omega_{xj} \mid x \in R', (x,j) \in E\};$
8 $p_j \leftarrow \{x \mid (x,j) \in E, d_x + \omega_{xj} = d_j\};$
    end
9 $R \leftarrow R \cup R';$
10 **until** $(R' = \emptyset)$
  end

FIGURE 1. Centralized Bellman-Ford Algorithm.

V insieme di vertici

E insieme di archi

w_{ij} peso del link tra i nodi i e j.

R insieme dei nodi per cui si è trovato un cammino

R' insieme dei nodi vicini dei nodi in R

d_j distanza dal nodo i al nodo j

p_j predecessore del nodo j nel cammino che *parte da* i

BELLMAN-FORD DISTRIBUITO

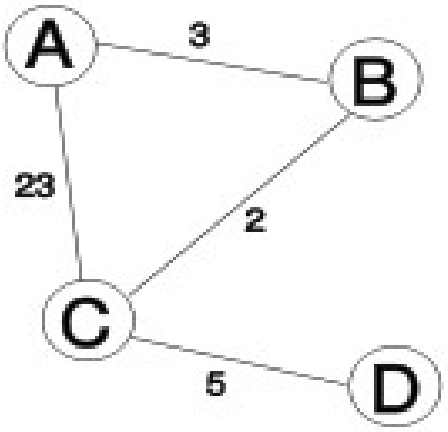
- Paradigma message-passing
- Richiede conoscenza delle neighbours tables
- Invio updates
  - non appena c'è un cambiamento
  - periodicamente
- Gestione cambiamenti pesi link

```
protocol DBF1 run at node i
  Initialize:
    begin
1   d_i ← 0; n_i ← i;
2   for each {x ∈ V − {i}};
3     d_x ← ∞; n_x ← ∅;
4   for each {x ∈ N};
      begin
5     d_x ← ω_ix; n_x ← x;
6     SendUpdate(d_x);
      end
    end

  ReceiveUpdate: u_j^k
    begin
7   d_j^k ← u_j^k;
8   t_j ← Min{d_j^x + ω_ix | x ∈ N};
9   n_j ← {x ∈ N | d_j^x + ω_ix = d};
10  if (t_j ≠ d_j)
11    then begin // No split-horizon, poison-reverse
12      d_j = t_j;
13      SendUpdate(d_j);
        end
    end

  LinkCostChange: l_ik
    begin
13  if (l_ik < ω_ik)
14    then T = V; // ω_ik improves, recompute all routes.
15    else T = {j | n_j = k}; // ω_ik worse, recompute routes using k only.
16  ω_ik ← l_ik;
17  for each {j ∈ T}
18    begin
19    d ← Min{d_j^x + ω_ix | x ∈ N};
20    if (d < d_j)
21      then begin
22        d_j ← d; n_j ← {x ∈ N | d_j^x + ω_ix = d};
23        SendUpdate(d_j);
          end
      end
    end
```

Run at node *i*

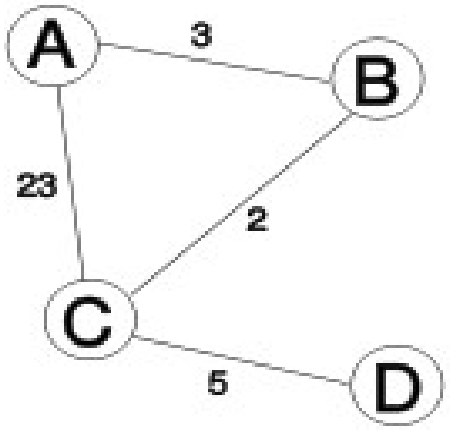
- ***nj*** – next hop to ***j***
- ***dj*** – distance to ***j***
- ***dkj*** – nbr table for ***j*** nbr ***k***
- ***ukj*** – update for ***j*** nbr ***k***
- ***lik*** – update for link ***i-k***

BELLMANN-FORD: UN PRIMO ESEMPIO



- considerimo un grafo generale
- ogni nodo mantiene una struttura dati (distance vector) che indica la sua distanza dagli altri nodi
- il contenuto del vettore è costruito mediante iterazioni successive
- mostreremo il comportamento dell'algoritmo alle diverse iterazioni

BELLMANN-FORD: UN PRIMO ESEMPIO



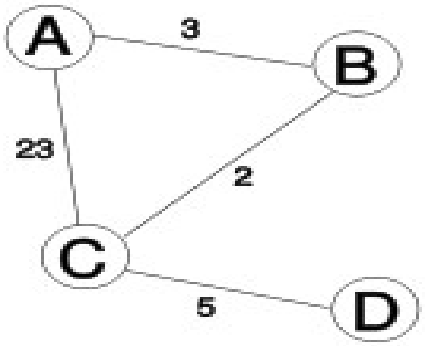
Distance Vector di *A*

T=0

from A	via A	via B	via C	via D
to A				
to B		3		
to C			23	
to D				

- il distance vector di *A* contiene la distanza minima di *A* dagli ogni altro nodo X
- per raggiungere X da *A* si considerano tutti i cammini che passano per uno qualsiasi degli altri nodi
- alla prima iterazione (T=0) il distance vector è inizializzato considerando solo i vicini di *A*

BELLMANN-FORD: UN PRIMO ESEMPIO



Distance Vector di B

from B	via A	via B	via C	via D
to A	3			
to B				
to C			2	
to D				

T=0

Distance Vector di C

T=0

from C	via A	via B	via C	via D
to A	23			
to B		2		
to C				
to D				5

Distance Vector di D

from D	via A	via B	via C	via D
to A				
to B				
to C			5	
to D				

T=0

BELLMANN-FORD: UN PRIMO ESEMPIO

- A questo punto, ogni router invia il proprio DV ai suoi vicini
- Ogni router ricalcola il proprio distance vector mediante le informazioni ricevute

DV(B)

T=0

from B	via A	via B	via C	via D
to A	3			
to B				
to C			2	
to D				

DV(A)

T=0

from A	via A	via B	via C	via D
to A				
to B		3		
to C			23	
to D				

DV(C)

T=0

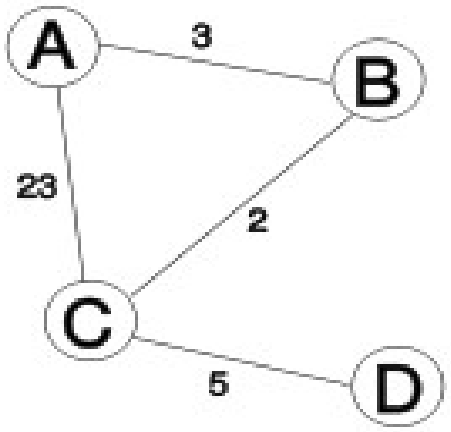
from C	via A	via B	via C	via D
to A	23			
to B		2		
to C				
to D				5

DV(A)

T=1

from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D			28	

BELLMANN-FORD: UN PRIMO ESEMPIO



T=1

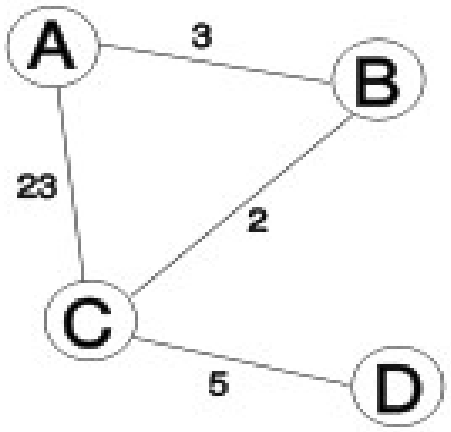
from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D			28	

from B	via A	via B	via C	via D
to A	3		25	
to B				
to C	26		2	
to D			7	

from C	via A	via B	via C	via D
to A	23	5		
to B	26	2		
to C				
to D				5

from D	via A	via B	via C	via D
to A			28	
to B			7	
to C			5	
to D				

BELLMANN-FORD: UN PRIMO ESEMPIO



T=2

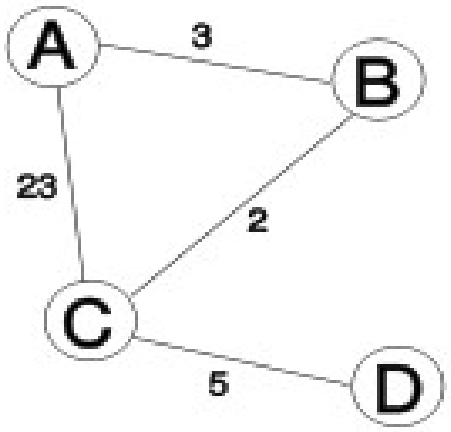
from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D		10	28	

from B	via A	via B	via C	via D
to A	3		7	
to B				
to C	8		2	
to D	31		7	

from C	via A	via B	via C	via D
to A	23	5		33
to B	26	2		12
to C				
to D	51	9		5

from D	via A	via B	via C	via D
to A			10	
to B			7	
to C			5	
to D				

BELLMANN-FORD: UN PRIMO ESEMPIO



T=3

from A	via A	via B	via C	via D
to A				
to B		3	25	
to C		5	23	
to D		10	28	

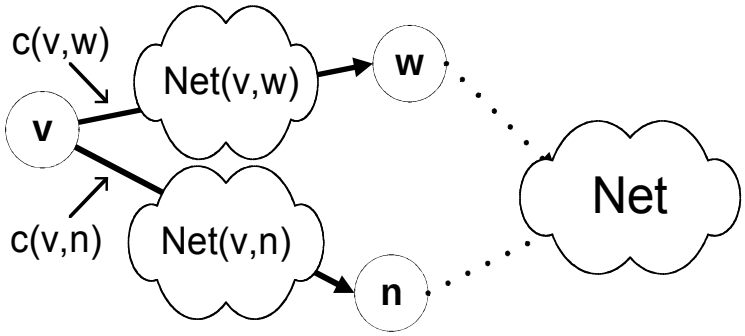
from B	via A	via B	via C	via D
to A	3		7	
to B				
to C	8		2	
to D	13		7	

from C	via A	via B	via C	via D
to A	23	5		15
to B	26	2		12
to C				
to D	33	9		5

from D	via A	via B	via C	via D
to A			10	
to B			7	
to C			5	
to D				

L'ALGORITMO DI BELLMAN-FORD

- Le reti sono tipicamente rappresentate come un grafo
- Nodi connessi mediante reti
- Una rete può essere un link o una LAN
- Una interfaccia di rete ha costo c(v,w)
- destinazioni: rete
- `Net(v,w):` indirizzo IP di una rete



Per semplicità di notazione, sostituiremo le "nuvole" con dei link semplici

DISTANCE VECTORS

RoutingTable of node v

Dest	via (next hop)	cost
Net	n	D(v,Net)

Distance Vector (routing table): un array monodimensionale per ogni nodo n

- un elemento del vettore per ogni possibile destinazione
- valore della cella: costo minimo dal nodo n verso ogni possibile destinazione + vicino attraverso cui passare per seguire il cammino minimo

DISTANCE VECTORS

RoutingTable of node v

Dest	via (next hop)	cost
Net	n	D(v,Net)

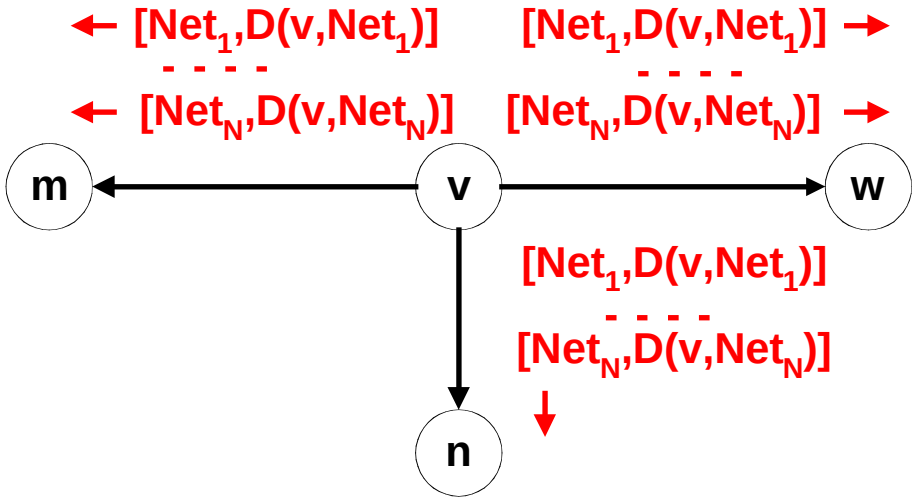
- i nodi inviano messaggi che contengono entrate del proprio distance vector(tabella di routing) ai loro nodi vicini
- un Mesaggio dela forma **[Net , D(v,Net)]** significa ***"Il costo per andare da me alla rete Net è D(v,Net)"***

DISTANCE VECTORS: INVIO UPDATES

RoutingTable of node v

Dest	via (next hop)	cost
Net_1	m	$D(v,Net_1)$
Net_2	n	$D(v,Net_2)$
- - -	- - -	- - -
Net_N	w	$D(v,Net_N)$

Periodicamente, ogni nodo invia il contenuto della sua Routing table ai suoi vicini



DISTANCE VECTORS: INVIO UPDATES

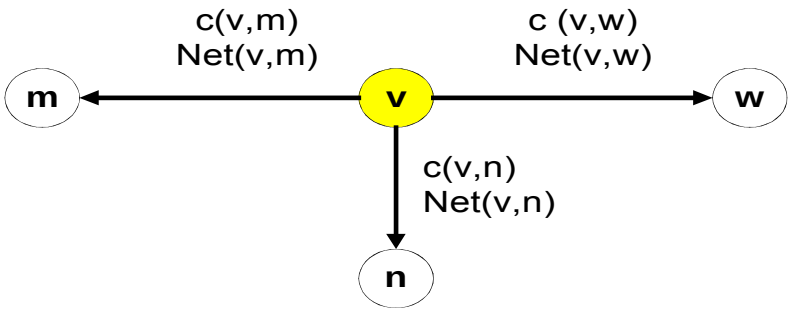
Supponiamo che un nuovo nodo v diventi attivo.

Inizializzazione: in termini di hops il costo di accesso alla reti che sono direttamente connesse è 0:

$$D(v,Net(v,m)) = 0$$

$$D(v,Net(v,w)) = 0$$

$$D(v,Net(v,n)) = 0$$



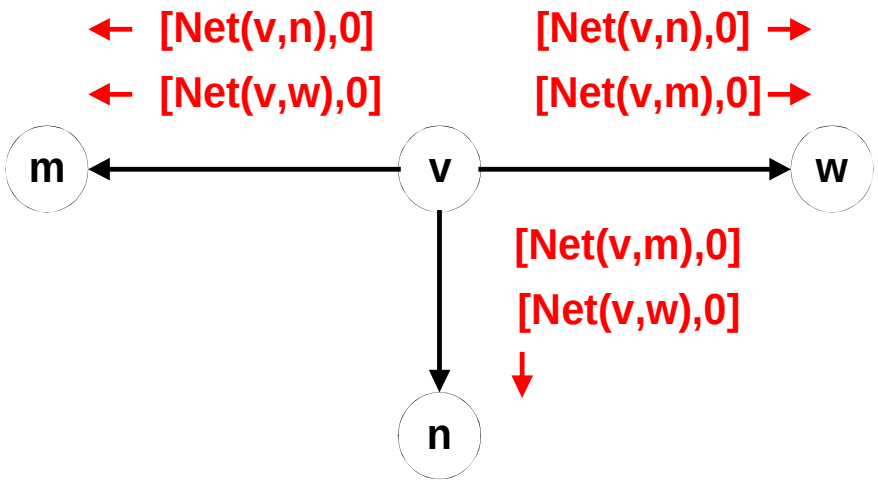
RoutingTable

Dest	via (next hop)	cost
Net(v,m)	m	0
Net(v,w)	w	0
Net(v,n)	n	0

DISTANCE VECTORS: INVIO UPDATES

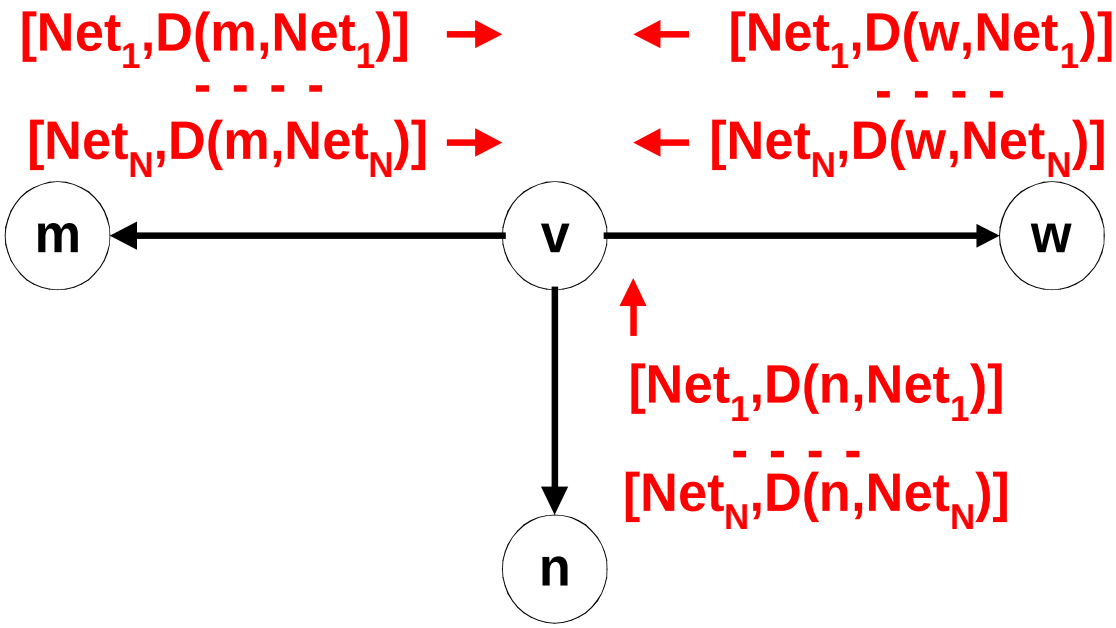
RoutingTable

Dest	via (next hop)	cost
Net(v,m)	m	0
Net(v,w)	w	0
Net(v,n)	n	0

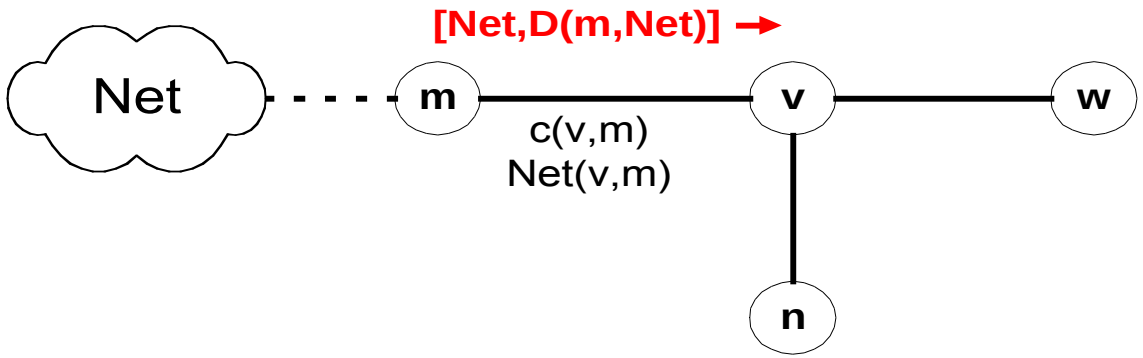


DISTANCE VECTORS: INVIO UPDATES

Il nodo v riceve i distance vectors dei vicini e costruisce la propria routing table



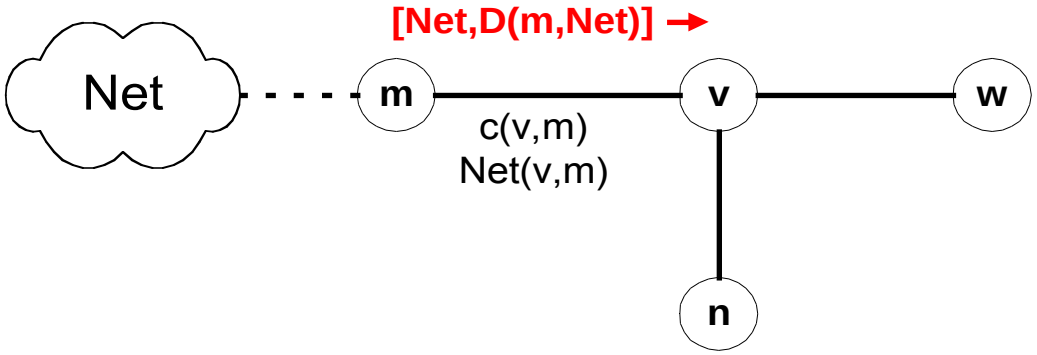
DISTANCE VECTORS: AGGIORNAMENTO ROUTING TABLES



Il nodo v aggiorna la sua routing table, se il messaggio ricevuto riduce il costo di un path ed invia l'aggiornamento a tutti i vicini

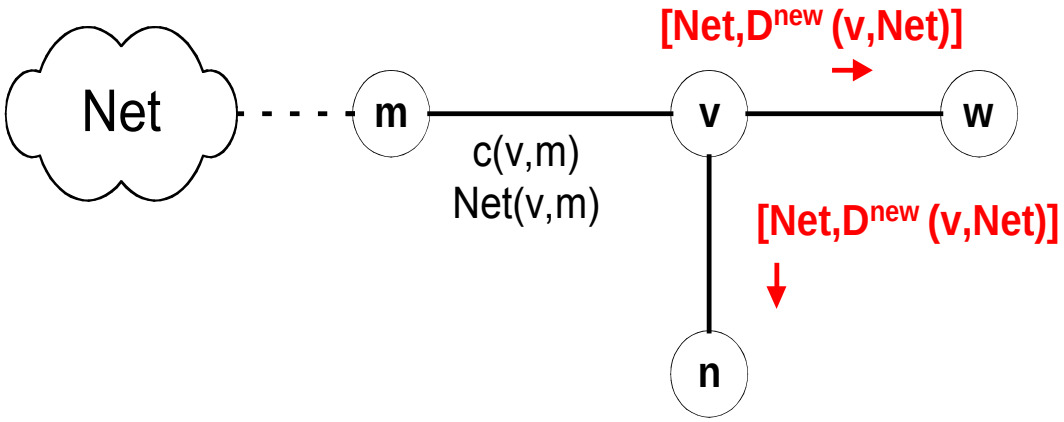
```
if   (D(m,Net) + c(v,m) < D(v,Net)) {
    Dnew(v,Net) = D(m,Net) + c(v,m);
    Update routing table;
    send message [Net, Dnew(v,Net)] to all neighbors
}
```

DISTANCE VECTORS: AGGIORNAMENTO ROUTING TABLES



RoutingTable

Dest	via (next hop)	cost
Net	??	D(v,Net)



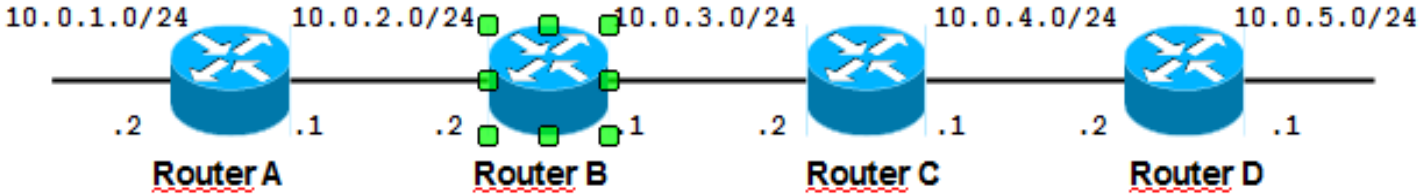
RoutingTable

Dest	via (next hop)	cost
Net	m	$D^{new}(v,Net)$

DISTANCE VECTORS: UN ESEMPIO CONCRETO

Assunzioni:

- link cost = 1, i.e., c(v,w) = 1
- tutti gli aggiormanenti avvengono simultaneamente
- inizialmente, ogni router conosce solo il costo delle reti connesse direttamente



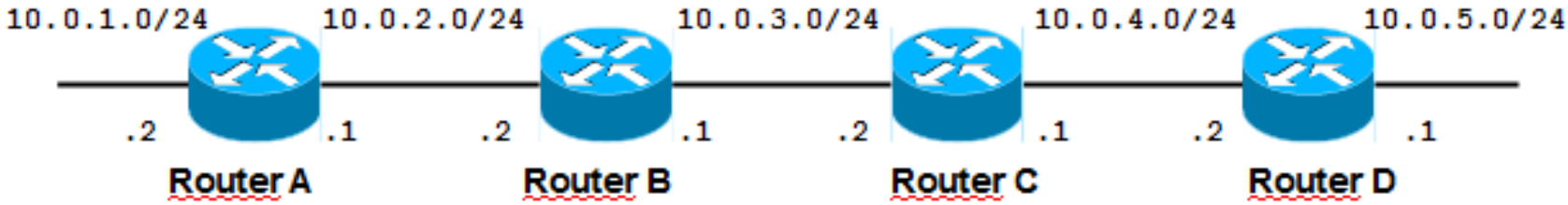
Net	via	cost
t=0:		
10.0.1.0	-	0
10.0.2.0	-	0
t=1:		
10.0.1.0	-	0
10.0.2.0	-	0
10.0.3.0	10.0.2.2	1
t=2:		
10.0.1.0	-	0
10.0.2.0	-	0
10.0.3.0	10.0.2.2	1
10.0.4.0	10.0.2.2	2

Net	via	cost
t=0:		
10.0.2.0	-	0
10.0.3.0	-	0
t=1:		
10.0.1.0	10.0.2.1	1
10.0.2.0	-	0
10.0.3.0	-	0
10.0.4.0	10.0.3.2	1
t=2:		
10.0.1.0	10.0.2.1	1
10.0.2.0	-	0
10.0.3.0	-	0
10.0.4.0	10.0.3.2	1

Net	via	cost
t=0:		
10.0.3.0	-	0
10.0.4.0	-	0
t=1:		
10.0.2.0	10.0.3.1	1
10.0.3.0	-	0
10.0.4.0	-	0
10.0.5.0	10.0.4.2	1
t=2:		
10.0.1.0	10.0.3.1	2
10.0.2.0	10.0.3.1	1
10.0.3.0	-	0
10.0.4.0	-	0

Net	via	cost
t=0:		
10.0.4.0	-	0
10.0.5.0	-	0
t=1:		
10.0.3.0	10.0.4.1	1
10.0.4.0	-	0
10.0.5.0	-	0
t=2:		
10.0.2.0	10.0.4.1	2
10.0.3.0	10.0.4.1	1
10.0.4.0	-	0
10.0.5.0	-	0

DISTANCE VECTORS: UN ESEMPIO CONCRETO



Net	via	cost
t=2:		
10.0.1.0	–	0
10.0.2.0	–	0
10.0.3.0	10.0.2.2	1
10.0.4.0	10.0.2.2	2
t=3:		
10.0.1.0	–	0
10.0.2.0	–	0
10.0.3.0	10.0.2.2	1
10.0.4.0	10.0.2.2	2
10.0.5.0	10.0.2.2	3

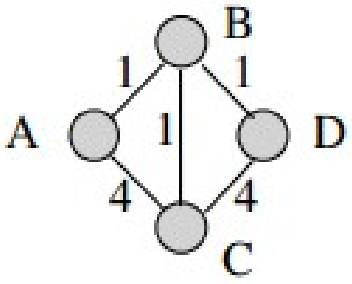
Net	via	cost
t=2:		
10.0.1.0	10.0.2.1	1
10.0.2.0	–	0
10.0.3.0	–	0
10.0.4.0	10.0.3.2	1
10.0.5.0	10.0.3.2	2
t=3:		
10.0.1.0	10.0.2.1	1
10.0.2.0	–	0
10.0.3.0	–	0
10.0.4.0	10.0.3.2	1
10.0.5.0	10.0.3.2	2

Net	via	cost
t=2:		
10.0.1.0	10.0.3.1	2
10.0.2.0	10.0.3.1	1
10.0.3.0	–	0
10.0.4.0	–	0
10.0.5.0	10.0.4.2	1
t=3:		
10.0.1.0	10.0.3.1	2
10.0.2.0	10.0.3.1	1
10.0.3.0	–	0
10.0.4.0	–	0
10.0.5.0	10.0.4.2	1

Net	via	cost
t=2:		
10.0.2.0	10.0.4.1	2
10.0.3.0	10.0.4.1	1
10.0.4.0	–	0
10.0.5.0	–	0
t=3:		
10.0.1.0	10.0.4.1	3
10.0.2.0	10.0.4.1	2
10.0.3.0	10.0.4.1	1
10.0.4.0	–	0
10.0.5.0	–	0

Al passo successivo, convergenza!

DISTANCE VECTOR: ANCORA ESEMPI

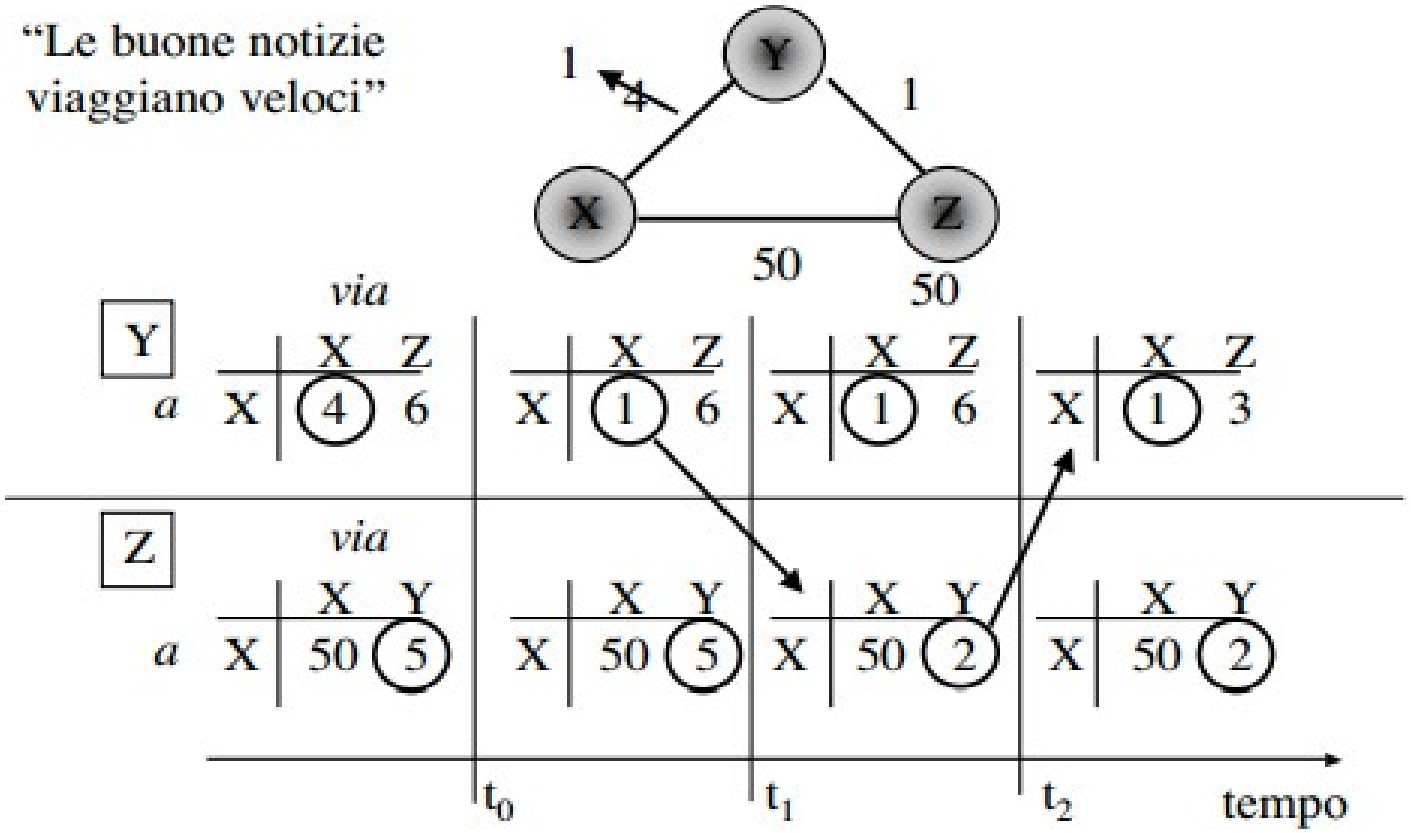


Situazione Iniziale

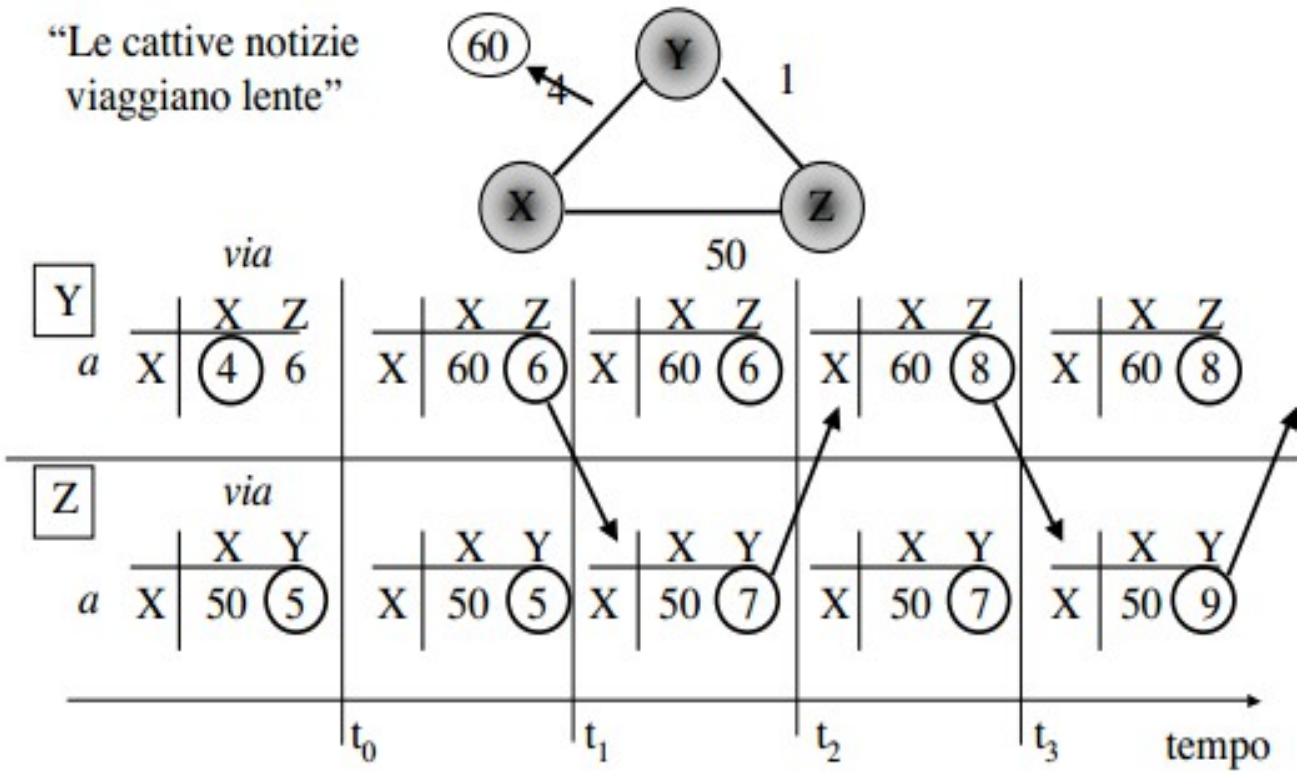
	A	B	C	D
A	0	1	4	∞
B	1	0	1	1
C	4	1	0	4
D	∞	1	4	0

1	Costo per raggiungere B da A
+	
1 0 1 1	Costi da B agli altri
=	
2 1 2 2	Costi passando per B
0 1 4 ∞	Costi attuali in A
min	
0 1 2 2	Nuovo DV
- - B B	*Next hop*

LE BUONE NOTIZIE VIAGGIANO VELOCI....



LE CATTIVE NOTIZIE VIAGGIANO LENTE....



COUNTING TO INFINITY

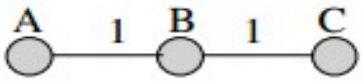
- Problemi del routing basato sull'algoritmo di Bellman-Ford
  - Routing loops
  - Count-to-infinity

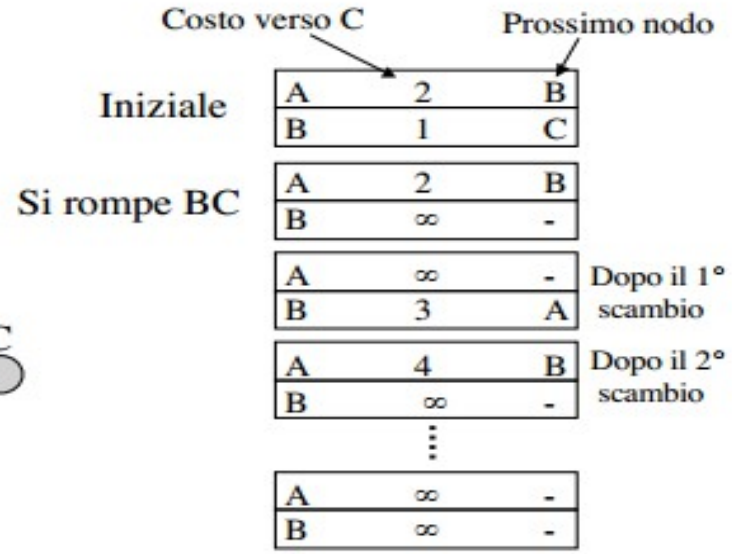
- Count to infinity:
  - se A dice a B che esso possiede un path verso un altro nodo X, non c'è alcun modo con cui B possa capire se B è parte di questo cammino

- Questo può portare problemi nel caso di fallimenti dei link.

COUNTING TO INFINITY

- il collegamento tra B e C si rompe
- costo link BC diventa infinito
- anche A imposta ad infinito il costo per andare a C (non ha altre alternative)
- tuttavia B pensa che passando per A si possa raggiungere C in due hops, non rendendosi conto che il cammino da A a C passa per B stesso
- scambio iterativo di messaggi fino a che tutti impostano ad infinito la distanz





LINK STATE ROUTING

- Principi generali: ogni router gestisce un database in cui è memorizzata la topologia dell'intera rete
- Flooding dello stato dei link
- Ogni router calcola la propria routing table sulla base della topologia
- Basato sull'algoritmo "shortest path" di Dijkstra's
- Implementazione maggiormente conosciuta: OSPF (Open Shortest Path First)

OSPF: OPEN SHORTEST PATH FIRST

- basato sull'algoritmo di Dijkstra
- utilizza reliable flooding (inondazione affidabile) : ogni nodo conosce lo stato delle linee che lo collegano ai nodi vicini
- reliable flooding: tutti i nodi che partecipano al protocollo ricevono da tutti gli altri nodi una copia delle informazioni relative allo stato delle linee.
- ogni nodo riceve dai propri vicini le informazioni e le invia a sua volta ai propri vicini (inondazione)

L'ALGORITMO DI DIJKSTRA

- l'algoritmo di Dijkstra richiede che ogni nodo della rete abbia una conoscenza completa della topologia dell'intera rete
- a differenza di Dijkstra, Bellmann-Ford usa solo informazione sui vicini di un nodo e sul costo dei loro link, per aggiornare i cammini ed i costi
- ogni nodo deve conoscere i costi di tutti i link della rete
- utilizza l'equazione *di* Bellmann-Ford in modo diverso
- itera sul "prossimo cammino più corto"
  - ad ogni iterazione cerca il cammino minimo su tutti I nodi della rete raggiungibile a qualla iterazione

L'ALGORITMO DI DIJKSTRA

- Dijkstra richiede una implementazione "centralizzata"
- Mantiene un grafo dell'intera rete,
- Ri-calcola i path ogni volta che il grafo cambia
- Flooding dei cambi attraverso tutti link
- Protocollo "brute force"
- Facile da capire, ma... inefficiente, molto overhead

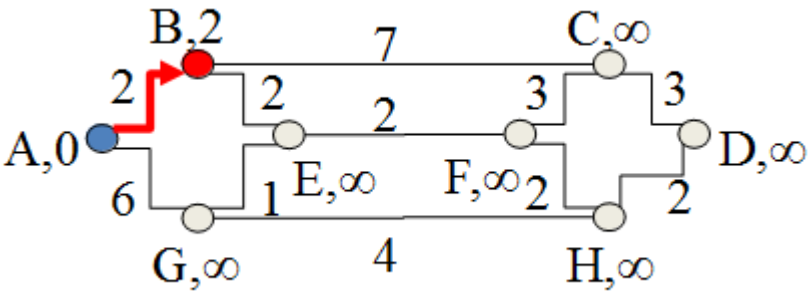
  Straight-forward, easy to understand

L'ALGORITMO DI DIJKSTRA

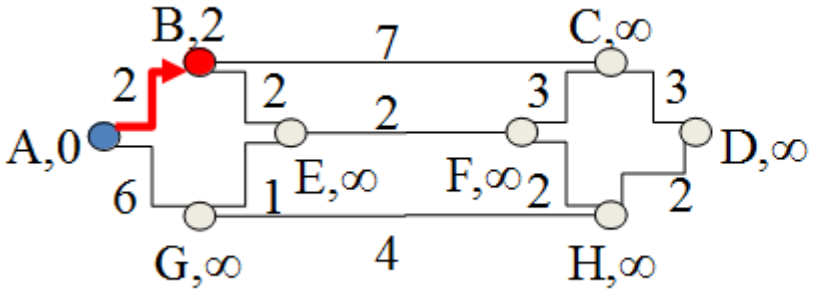
- siano i nodi numerati 0..N; l'algoritmo calcola i cammini migliori a partire dal nodo 0.
- `c(i,j)` costo dell'arco `(i,j)`
- `pred(i)` predecessore di i nell'albero che viene costruito
- `m(j)` distanza dal nodo 0 al nodo j.

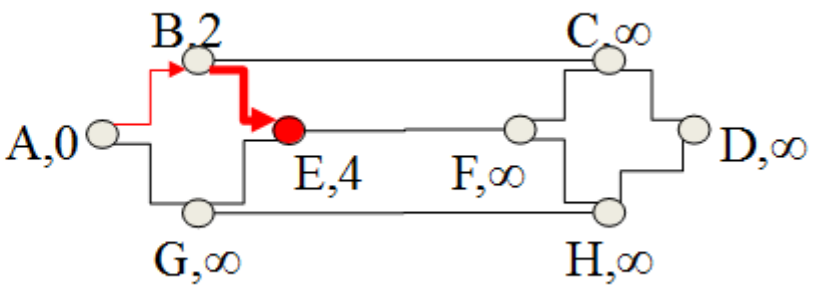
```
m(0) = 0; M = {0};
for k=1 to N {
   find (i0, j0) that minimizes m(i) + c(i,j),
                        with i in M, j not in M
   m(j0) = m(i0) + c(i0, j0)
   pred(j0) = i0
   M = M ∪ {j0}
}
```

L'ALGORITMO DI DIJKSTRA

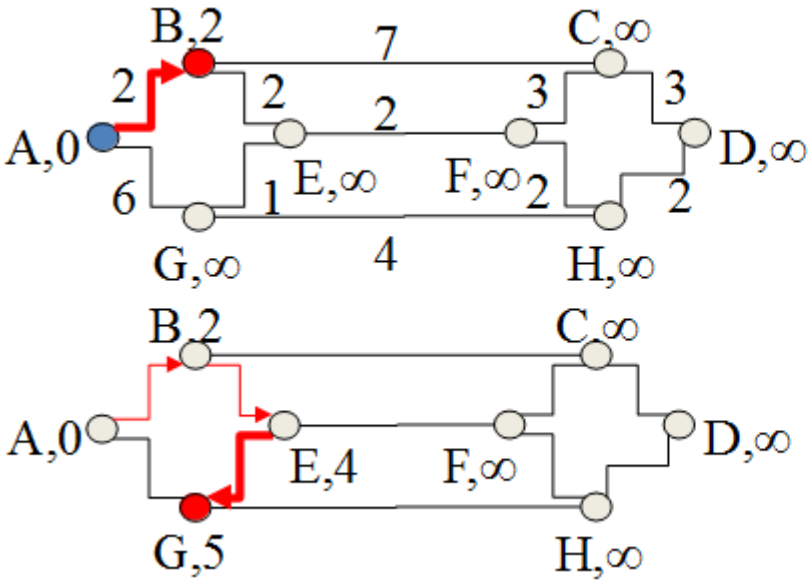


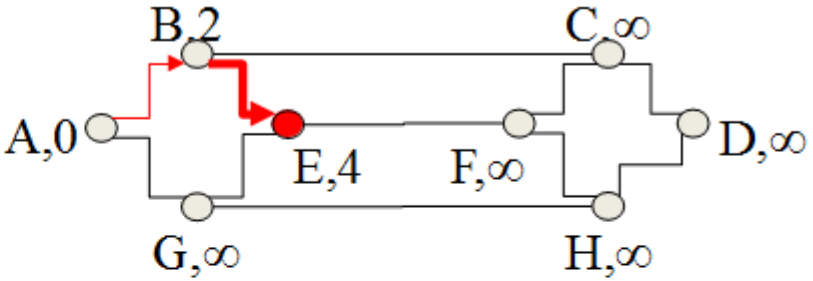
L'ALGORITMO DI DIJKSTRA



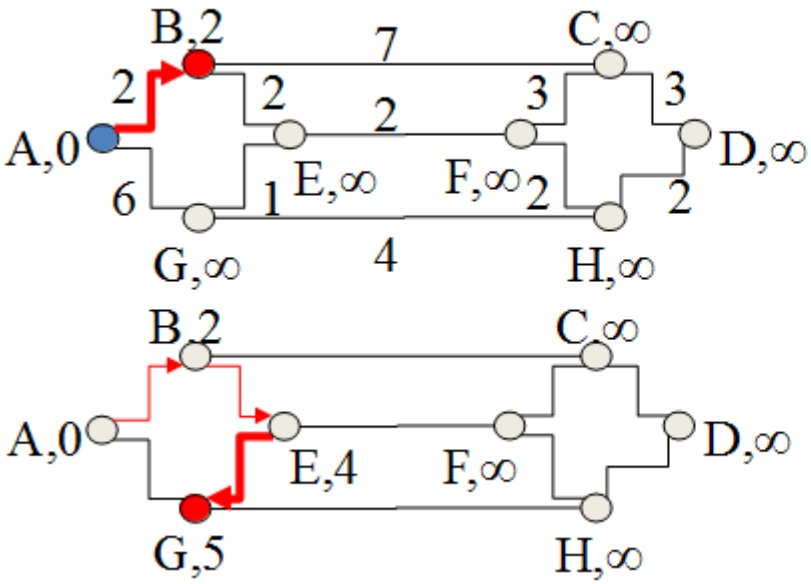


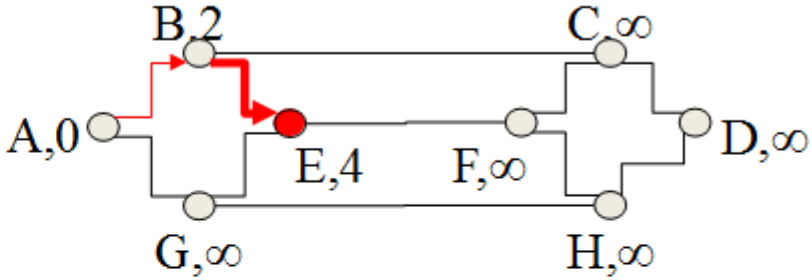
L'ALGORITMO DI DIJKSTRA

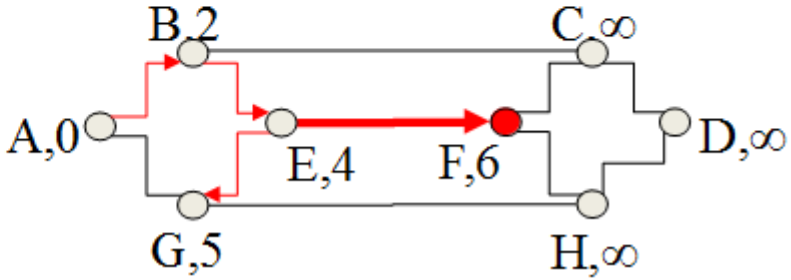




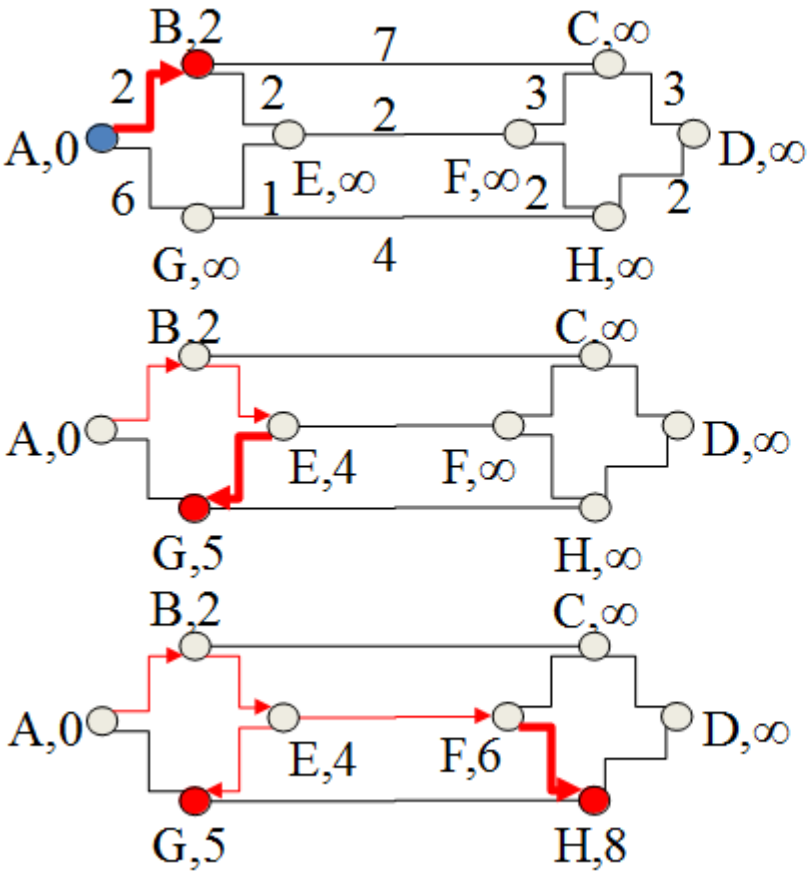
L'ALGORITMO DI DIJKSTRA

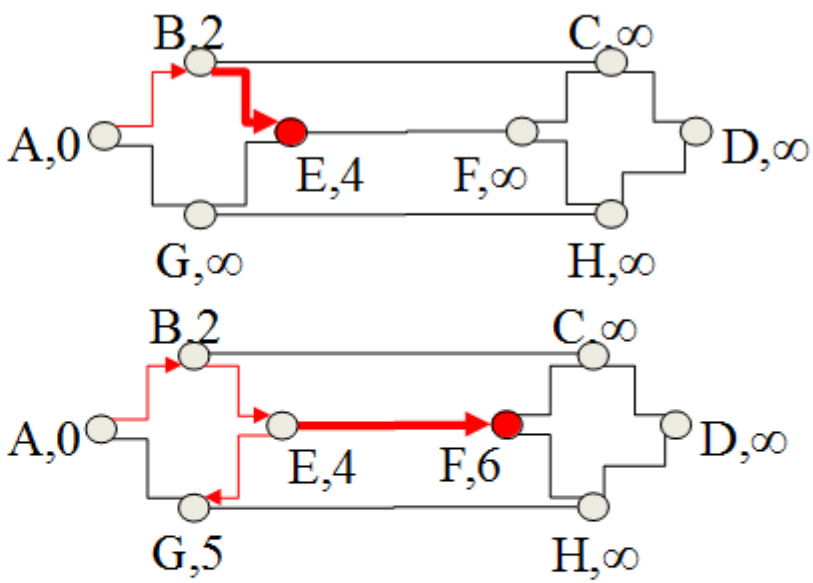




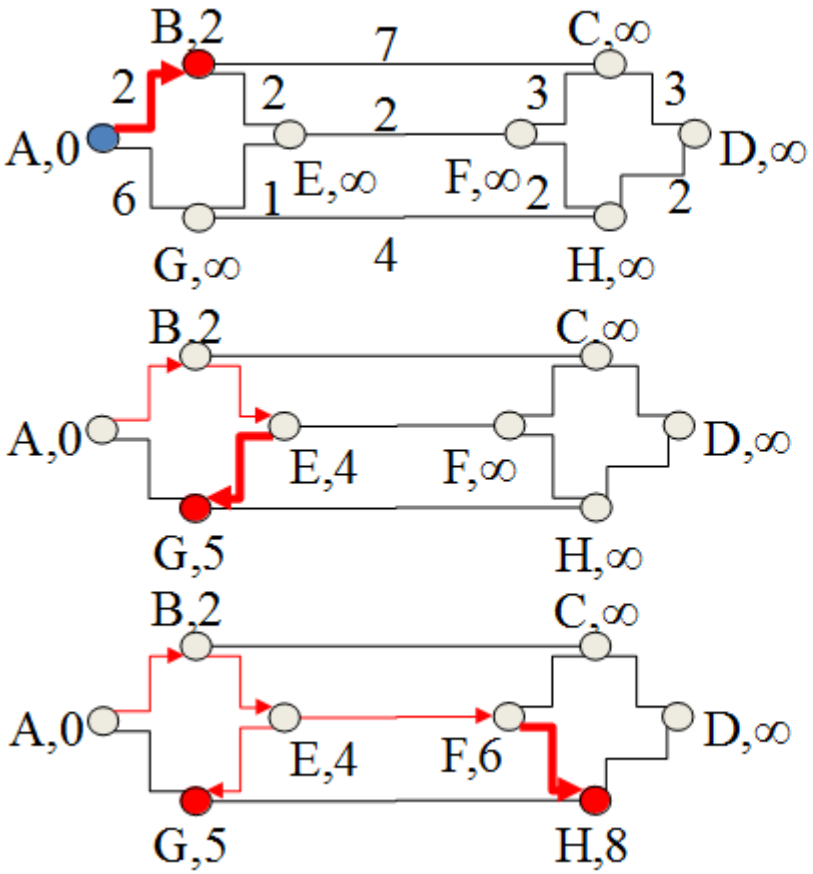


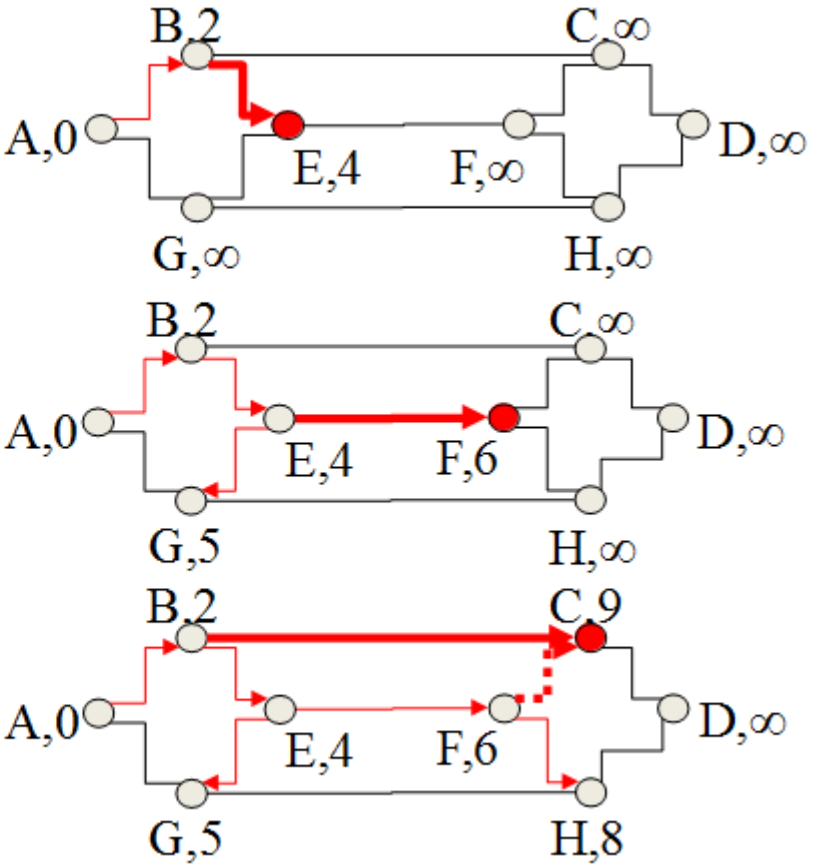
L'ALGORITMO DI DIJKSTRA



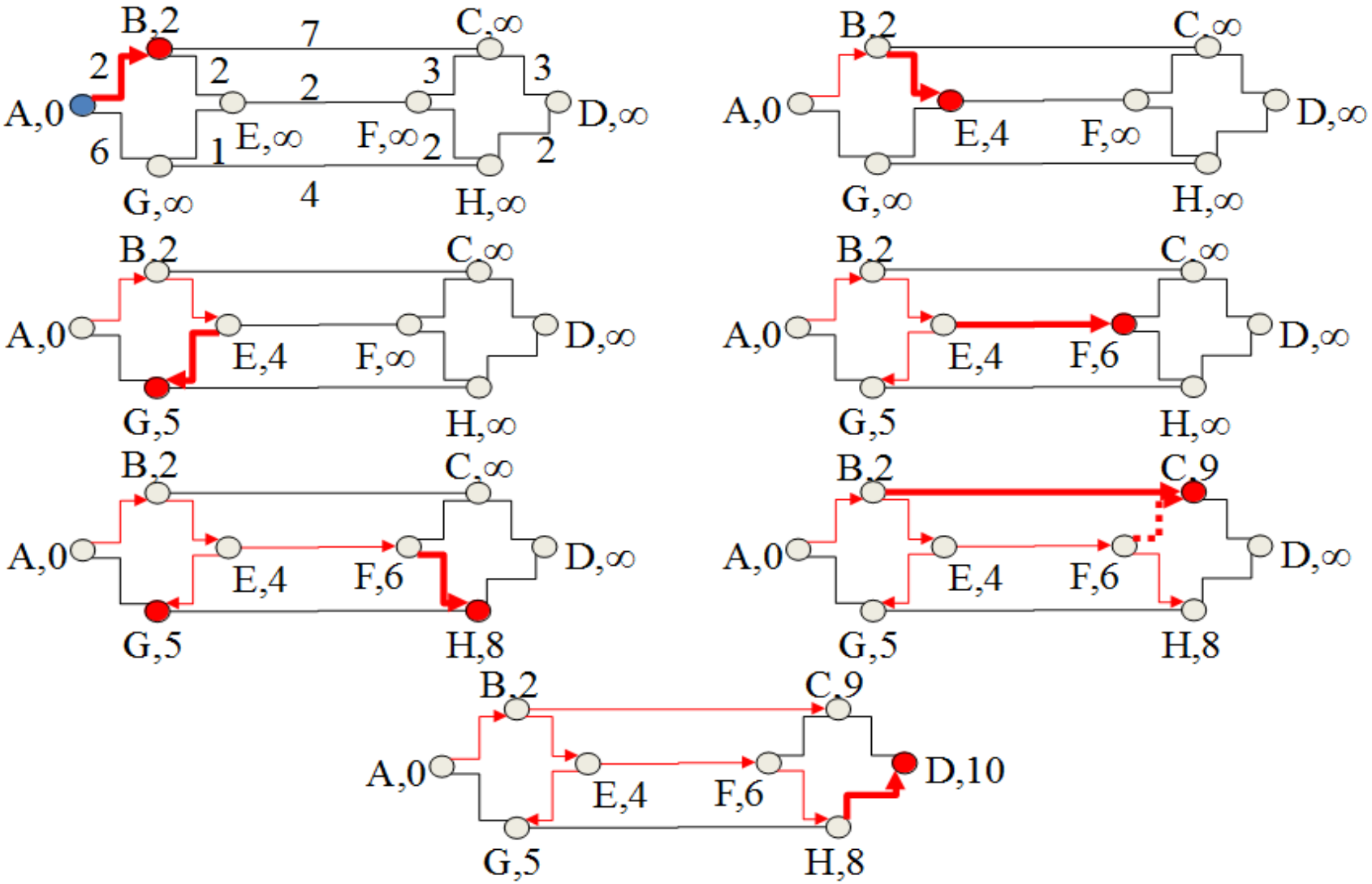


L'ALGORITMO DI DIJKSTRA





L'ALGORITMO DI DIJKSTRA



OSPF: OPEN SHORTEST PATH FIRST

- il protocollo di routing maggiormente utilizzato

- storia

  1989: RFC 1131 OSPF versione 1

  1991: RFC 1247 OSPF versione 2

  1994: RFC 1583 OSPF versione 2 (rivista)

  1997: RFC 2178 OSPF versione 2 (rivista)

  1998: RFC 2328 OSPF versione 2 (versione corrente)

OSPF: OPEN SHORTEST PATH FIRST

- ogni router conosce lo stato delle linee (costi) che lo collegano ai nodi vicini
- ogni router invia a tutti gli altri router le informazioni relative ai links ad esso connessi (link state advertisement)
- quando un router ha ricevuto lo stato dell'intera rete, crea una mappa completa della topologia della rete
- ogni router mantiene le informazioni ricevute in una struttura dati, il link-state database (perchè mantiene informazioni sullo stato dei links dell'intera rete)
- quando un nuovo router viene inserito nella rete, l'informazione viene aggiornata.
- i diversi routers possono avere temporaneamente diverse visioni della rete, ma le diverse visioni convergono mediante l'invio degli advertisements

OSPF: OPEN SHORTEST PATH FIRST

- ogni router R
  - utilizza il suo link state database per calcolare il cammino di minor costo che lo collega ad un qualsiasi altro router del sistema autonomo
  - crea uno shortest path tree (SPT) indica il cammino minimo da R ad un qualsiasi altro router del sistema autonomo
  - l'albero contiene la stessa informazione contenuta nel database, ma vista 'dal punto di vista' di R
- Lo SPT viene modificato dinamicamente se si ricevono nuovi advertisements che notificano la modifica della topologia della rete

•Ogni router applica l'algoritmo di Dijkstra per il calcolo dei cammini minimi per il calcolo dello SPT

OSPF: OPEN SHORTEST PATH FIRST

- Each router knows directly connected networks

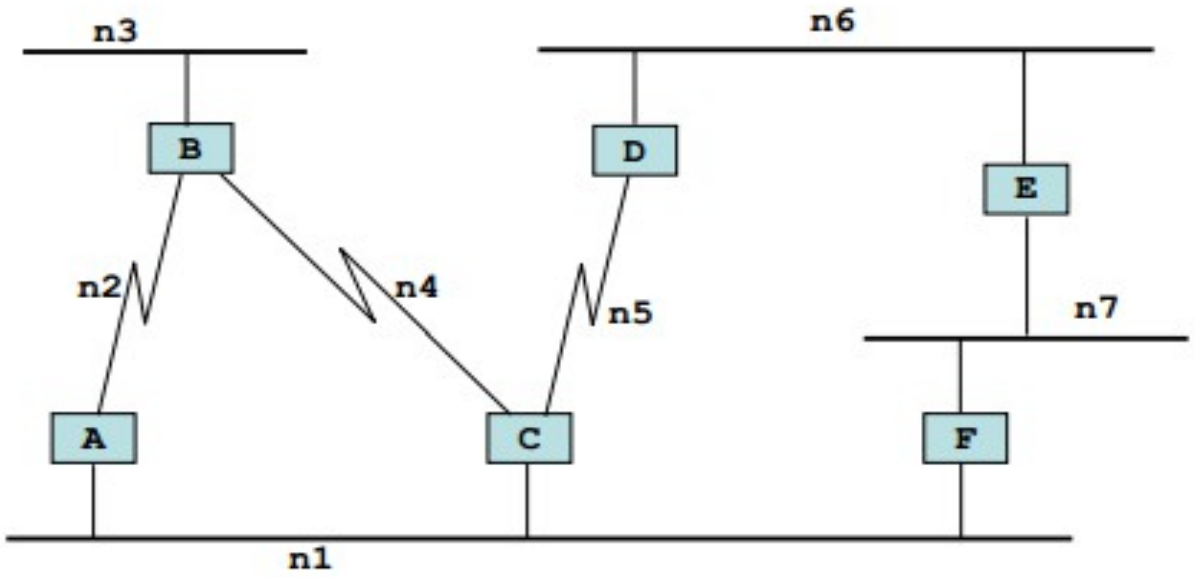


TABELLE DI ROUTING INIZIALI

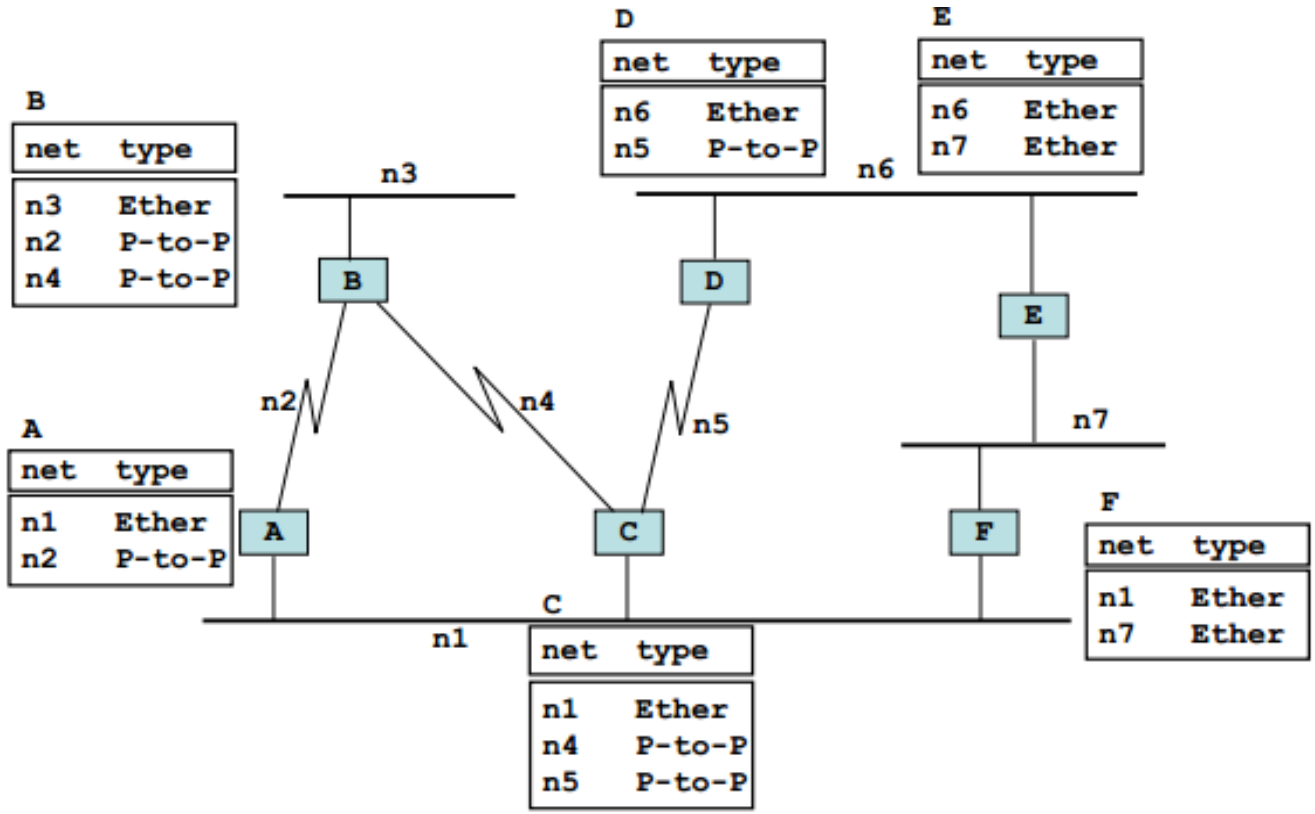
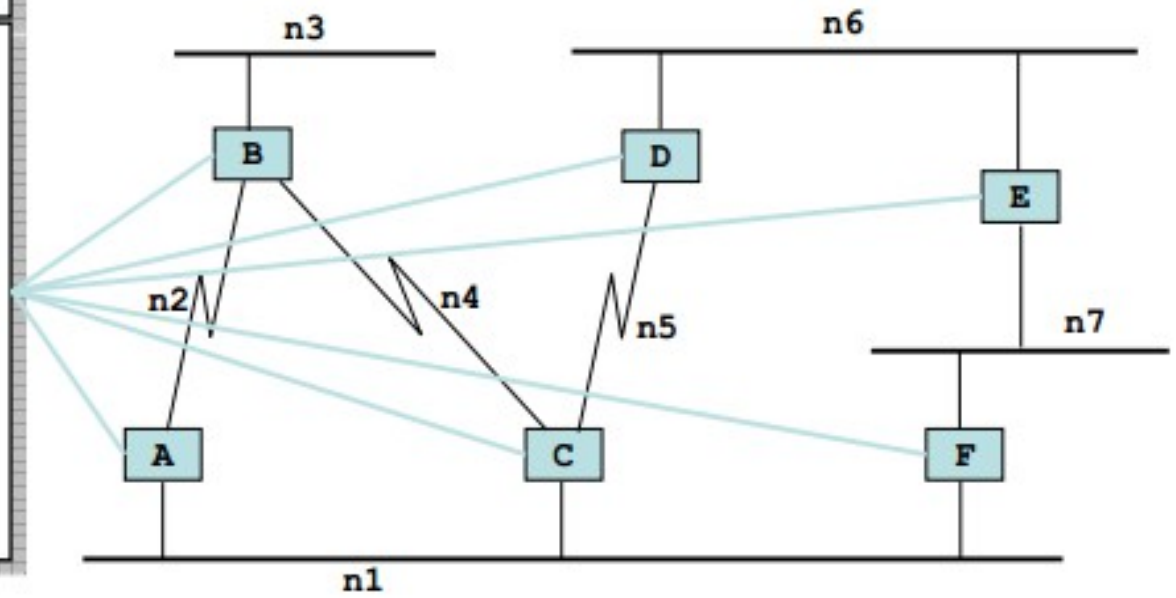


TABELLE DI ROUTING INIZIALI

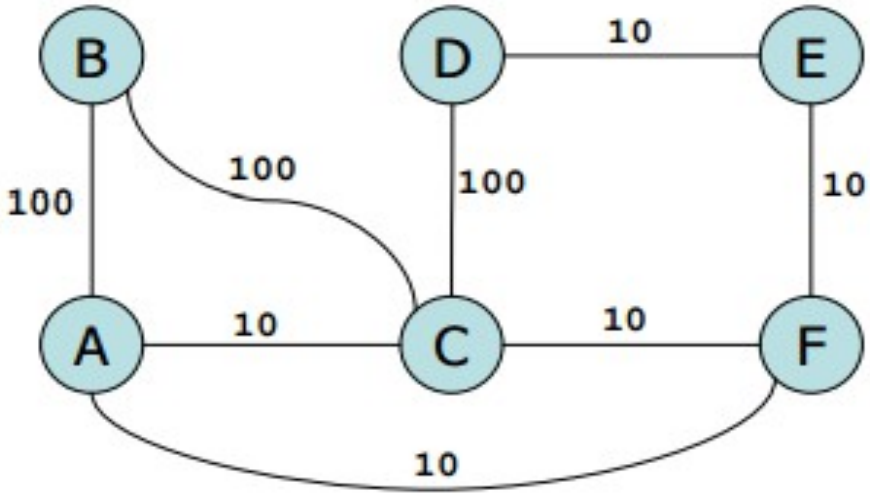
- The local metric information is flooded to all routers
- After convergence, all routers have the same information

rtr	net	cost
A	n1	10
A	n2	100
B	n3	10
B	n2	100
B	n4	100
C	n1	10
C	n4	100
C	n5	100
D	n6	10
D	n5	100
E	n6	10
E	n7	10
F	n1	10
F	n7	10

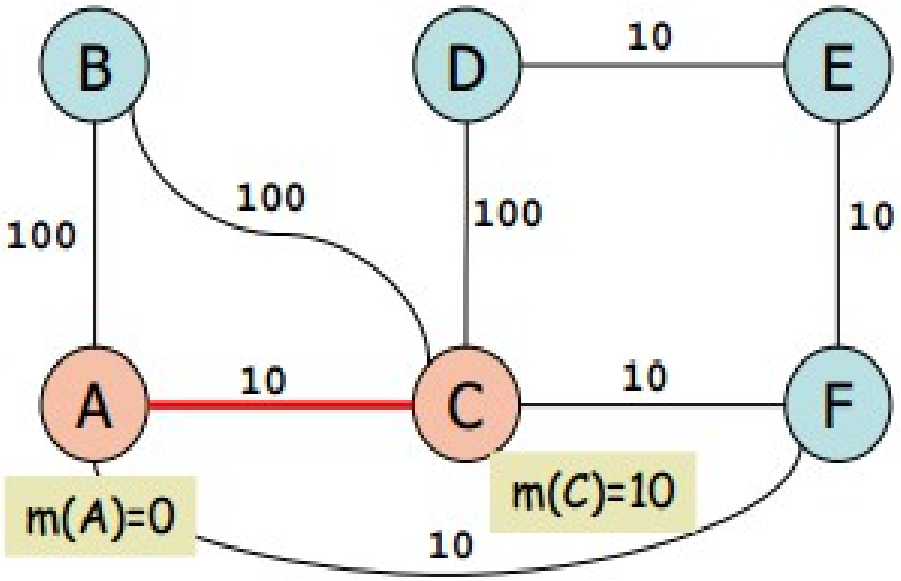


IL GRAFO SEMPLIFICATO

- Only arrows with metrics between routers
- Every node executes the shortest path computation on the graph – same graph, but different sources



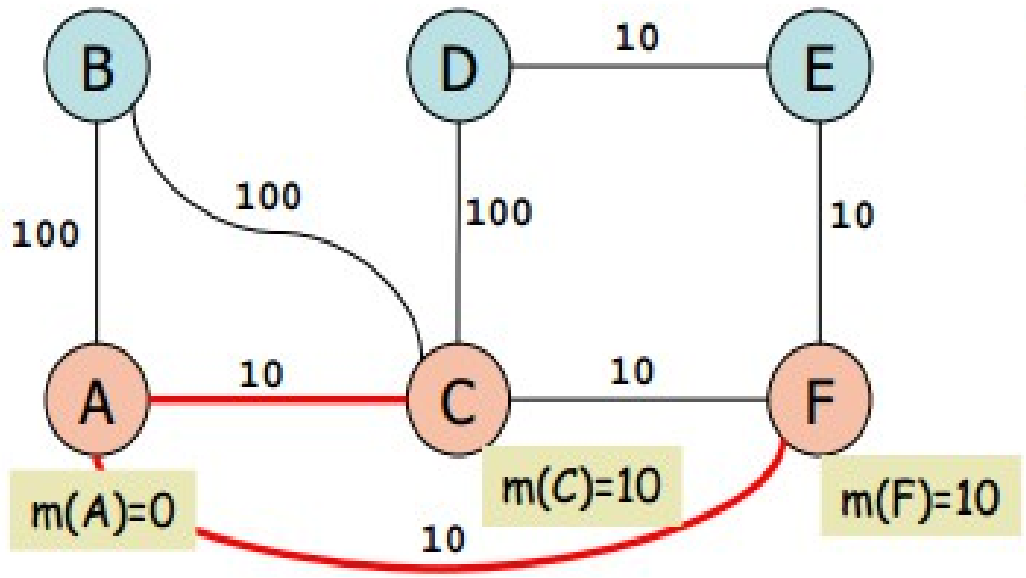
ESEMPIO: DIJKSTRA IN *A*



init: M = { A }

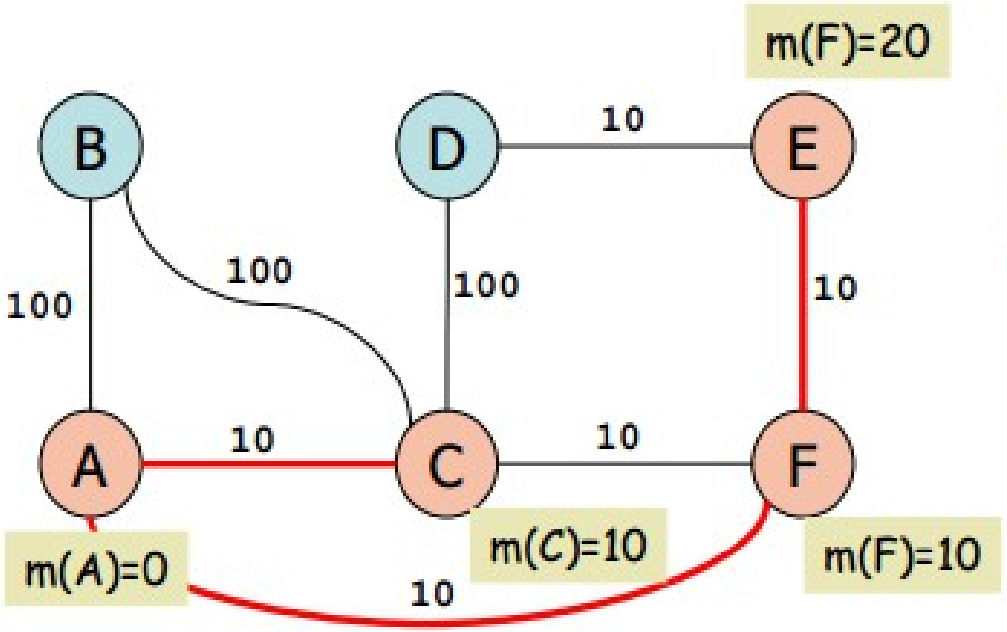
step 1:
i0=A
j0=C
m(C)=10
M = {A, C}

ESEMPIO: DIJKSTRA IN *A*

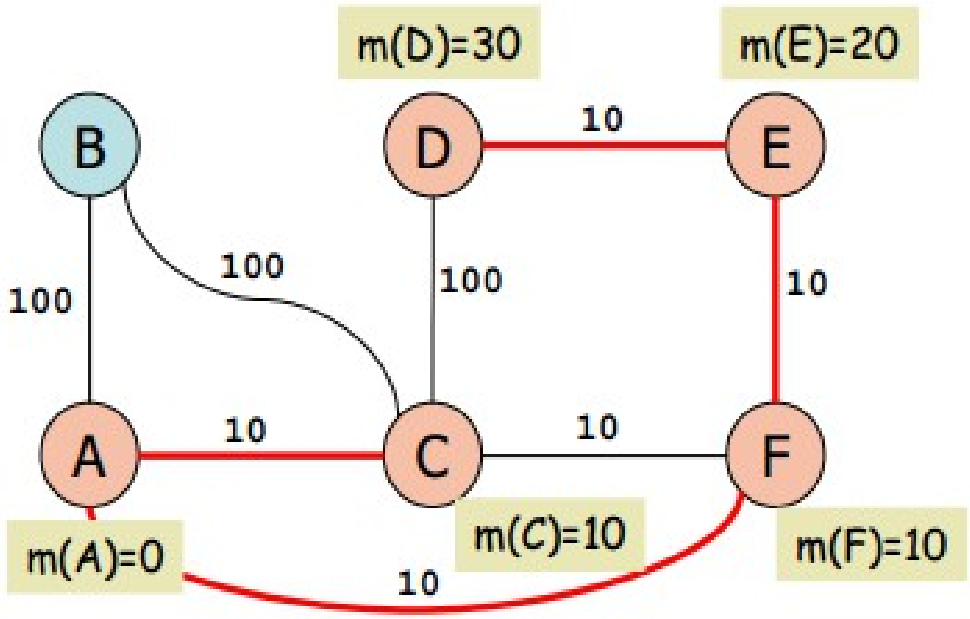


i0=*A*
j0=F
m(F)=10
M = {*A*,*C*,F}

ESEMPIO: DIJKSTRA IN *A*

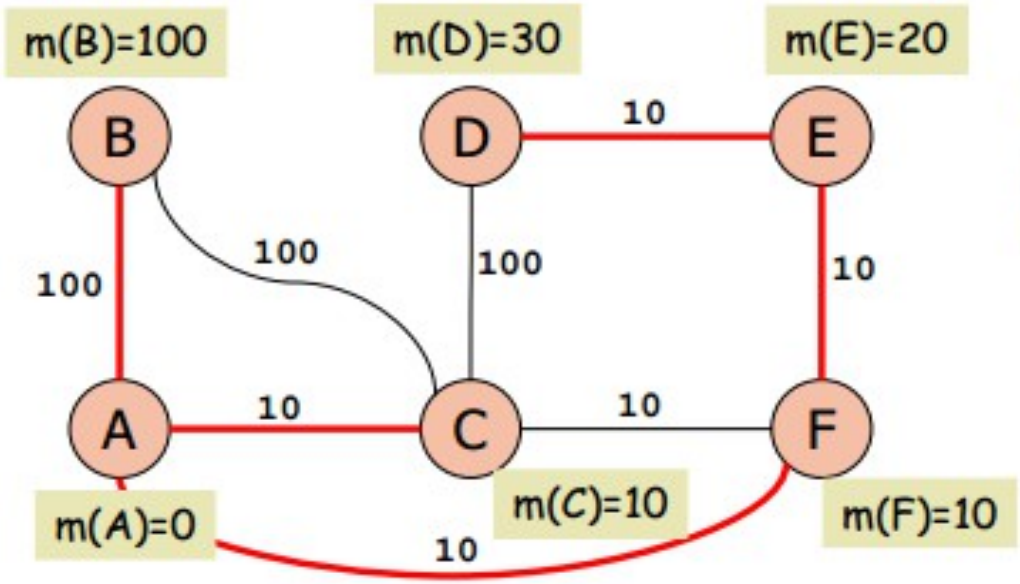


ESEMPIO: DIJKSTRA IN *A*



i0=E
j0=D
m(D)=40
M = {A,C,F,E,D}

ESEMPIO: DIJKSTRA IN *A*

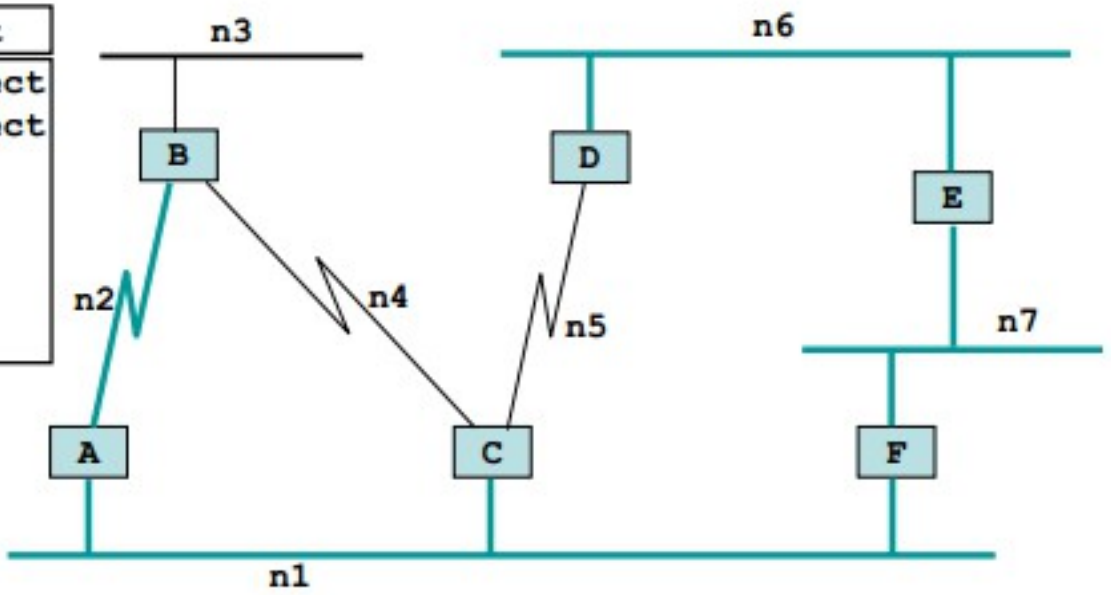


i0=*A*
j0=B
m(B)=100
M = {*A*,*C*,F,E,D,B}

ROUTING TABLE DI A

A

net	next
n1	direct
n2	direct
n3	B
n4	C
n5	C
n6	F
n7	F



LINK STATE: SOMMARIO

- tutti i nodi calcolano il loro database con la topologia della rete
- il database rappresenta l'intera rete
- fortemente sincronizzato
- tutti i nodi calcolano il loro albero con i migliori path verso tutte le destinazioni
- la tabelle di routing sono costruite a partire dall'albero ed utilizzate per individuare il next hop di routin
- LS versus DV
  - LS evita i problemi di convergenza di DV
  - LS più complesso di DV
  - LS maggior overhead di DV
  - LS centralizzato DV distribuito