Routing e protocolli di routing

Routing: nozioni base

1. CALCOLO DEL PERCORSO o INSTRADAMENTO o ROUTING = Determinazione del percorso ottimale dei messaggi in base alle informazioni contenute nella ROUTING TABLE. Centrale nella determinazione del percorso ottimale è il modo in cui si creano e si aggiornano le tabelle di routing. I protocolli di ROUTING svolgono questo tipo di attività.
2. INOLTRO o FORWARDING = Trasporto del pacchetto in arrivo sulla porta di input del router verso l'interfaccia di output alla rete (al router) cui è destinato il pacchetto o al router di default

NOTA: nella tabella di routing di ogni nodo di rete puo' esistere una linea che specifica un router di **default**. Questo è solitamente un router con conoscenza della struttura di rete maggiore che sul nodo corrente.

Inoltro di datagrammi IP

L'INOLTRO dei datagrammi verso l'host destinazione avviene secondo modalità standard:

- Acquisito l'indirizzo IP di un datagrama da inoltrare, il router controlla, attraverso la propria netmask, se è relativo ad un host della propria rete;
- Se l'indirizzo appartiene alla stessa rete del router, l'IP del router utilizzerà i servizi dello strato inferiore (data-link) per spedire il pacchetto direttamente all'host destinatario.
- Se l'indirizzo appartiene ad un'altra rete, il router consulterà la propria Routing Table, che associa ad ogni rete l'indirizzo del router di frontiera delle reti connesse.
- Se il router (che è collegato a più reti) ha una connessione alla rete dove è collegato l'host destinatario, gli inoltra direttamente il pacchetto, altrimenti lo passa al router più vicino, fino a raggiungere un router in grado di consegnare il pacchetto all'host destinatario.

Esempio di routing table su host linux

$ netstat -rn

Kernel IP routing table

Destination	*Gateway*	*Mask*	*Flags*	*Iface*
153.18.16.0	0.0.0.0	255.255.240.0	U	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	lo
0.0.0.0	153.18.31.254	0.0.0.0	UG	eth0

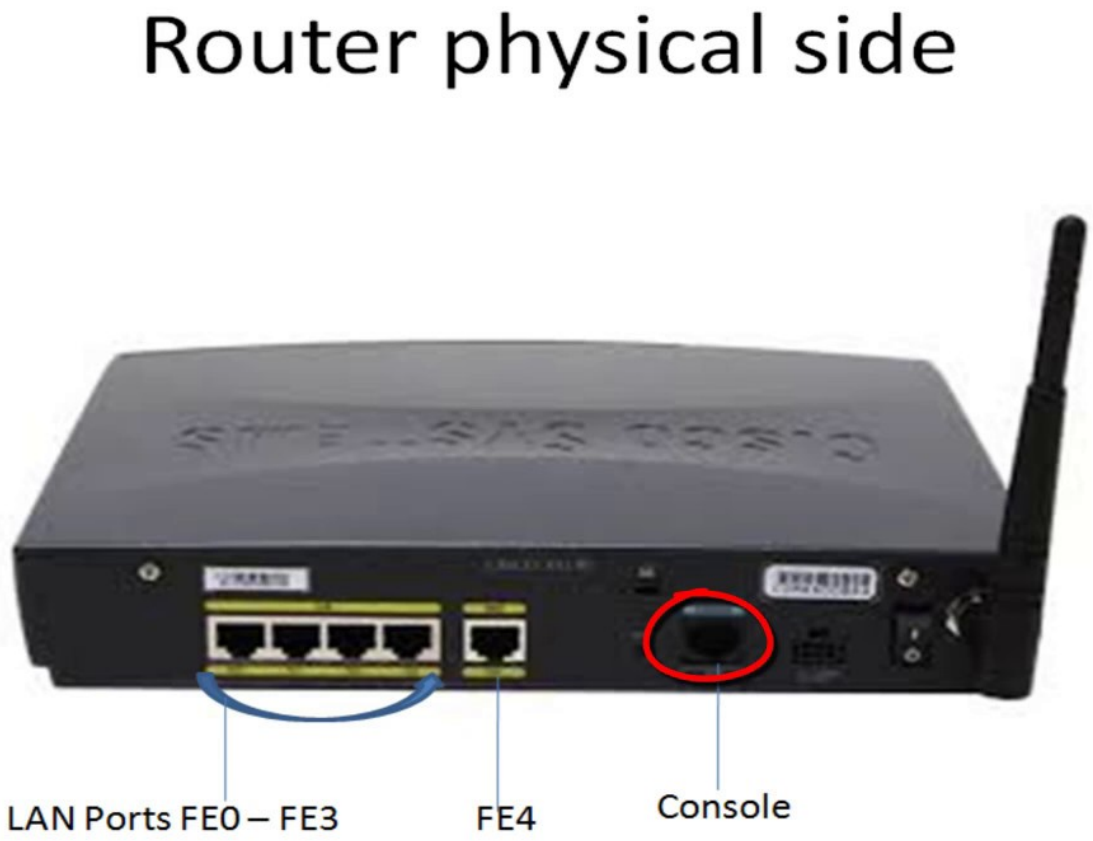
Esempio di traceroute

```
traceroute to www.ebay.com (66.135.200.161), 30 hops max, 40 byte packets
 1  193.205.128.1 (193.205.128.1)  0.486 ms  0.586 ms  0.693 ms
 2  193.205.131.42 (193.205.131.42)  0.263 ms  0.267 ms  0.281 ms
 3  193.205.131.33 (193.205.131.33)  0.588 ms  0.589 ms  0.580 ms
 4  ru-unian-re1-an.an.garr.net (193.206.140.193)  0.558 ms  0.529 ms  0.449 ms
 5  re1-an-rx1-bo1.bo1.garr.net (90.147.81.5)  8.738 ms  8.734 ms  8.757 ms
 6  rx1-bo1-rx1-mi1.mi1.garr.net (90.147.80.109)  11.426 ms  10.927 ms  10.879 ms
 7  rx1-mi1-r-mi1.mi1.garr.net (90.147.80.93)  7.663 ms  6.907 ms  6.892 ms
 8  xe-0-3-2.csr1.lin1.gblx.net (208.50.25.117)  8.053 ms xe-0-3-1.csr1.lin1.gblx.net
(64.215.30.89)  8.016 ms  8.055 ms
 9  xe5-2-0-10G.scr1.FRA4.gblx.net (67.16.138.49)  16.702 ms  16.739 ms  16.703 ms
10  lag1.ar4.fra4.gblx.net (67.16.145.238)  15.974 ms  19.411 ms  23.920 ms
11  sprint-1.ar3.fra4.gblx.net (64.215.195.210)  16.356 ms  17.026 ms  17.014 ms
```

Router

Router

Router



Routing: implementazione

La funzione di routing deve invece 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

Caratteristiche del routing

I vari algoritmi e protocolli di routing si differenziano per le modalità con cui la tabella di instradamento viene creata ed 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 o Dinamico
- Centralizzato o Distribuito

Routing statico

Le tabelle di instradamento sono:

- invarianti nel tempo
- indipendenti dalle condizioni di traffico nella rete: usano algoritmi non adattativi

Le tabelle di instradamento vengono create in fase di configurazione del router:

- grande lavoro di configurazione

Le tabelle vengono modificate **con l'intervento di un operatore** solo in caso di variazioni strutturali o topologiche della rete (inserimento o caduta di nodi, collegamenti)

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 come inserimento di nuovi nodi o collegamenti, caduta di un nodo o collegamento per guasto

– condizioni di traffico: si evita la scelta di percorsi che comprendono collegamenti congestionati

Routing centralizzato

Un unico nodo centrale:

– raccoglie tutte le informazioni sullo stato e la topologia della rete (tramite un opportuno protocollo)

– 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

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**!!)

Cosa implementa un router moderno?

L’instradamento è una delle funzioni del livello Internet

- dal punto di vista dell’utilizzo della rete i router implementano le funzionalità fino allo livello Internet

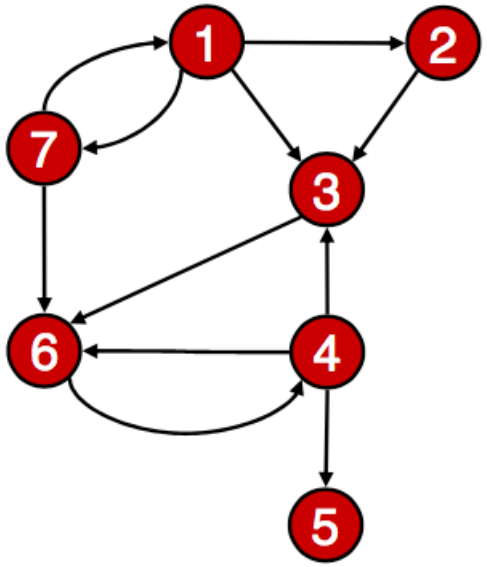
I protocolli di routing si basano su uno scambio di informazioni tra applicazioni

- dal punto di vista del controllo della rete i router implementano anche funzionalità di livello superiore (sono calcolatori specializzati). Ad esempio implementano i protocolli di routing, oppure server web per la interfaccia di configurazione.

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

Grafi

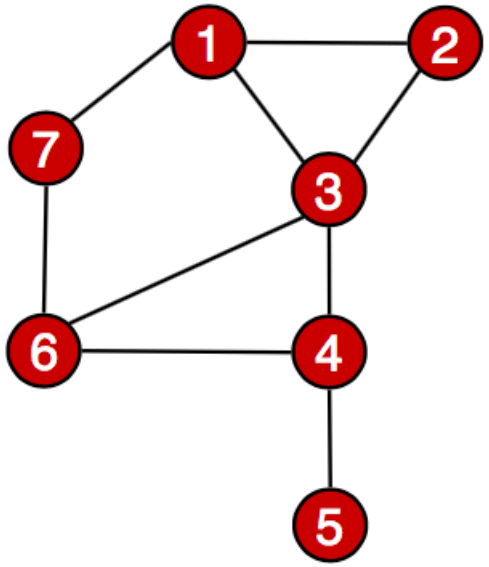


Grafo orientato

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)}

Dimensioni: |V|=7, |E|=11



Grafo non orientato

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

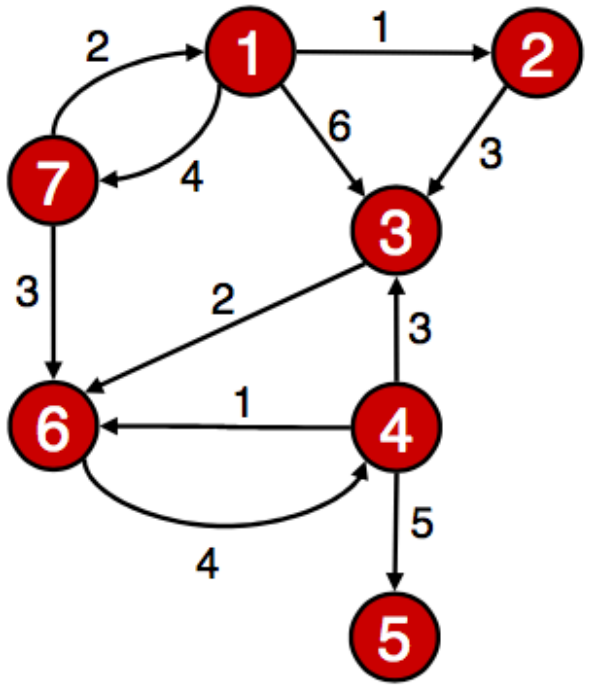
E = {(1,2),(1,3),(1,7),
(2,3),(3,4),(3,6),
(4,5),(4,6),(6,7)}

Dimensioni: |V|=7, |E|=9

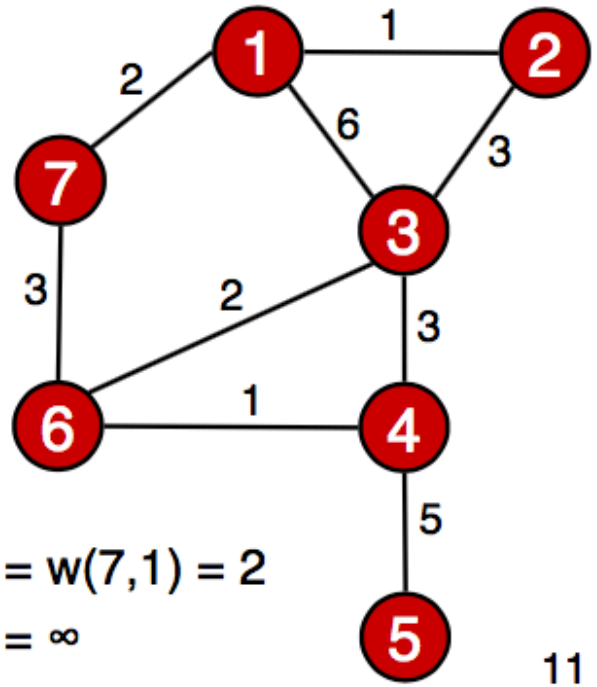
Grafo pesato

- 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) = ∞
- Per semplicità si assume w(i,j) > 0 per ogni arco (i,j) $\in$ E

Grafi pesati



w(1,7) = 4
w(7,1) = 2
w(1,2) = 1
w(2,1) = ∞



w(1,7) = w(7,1) = 2
w(1,6) = ∞

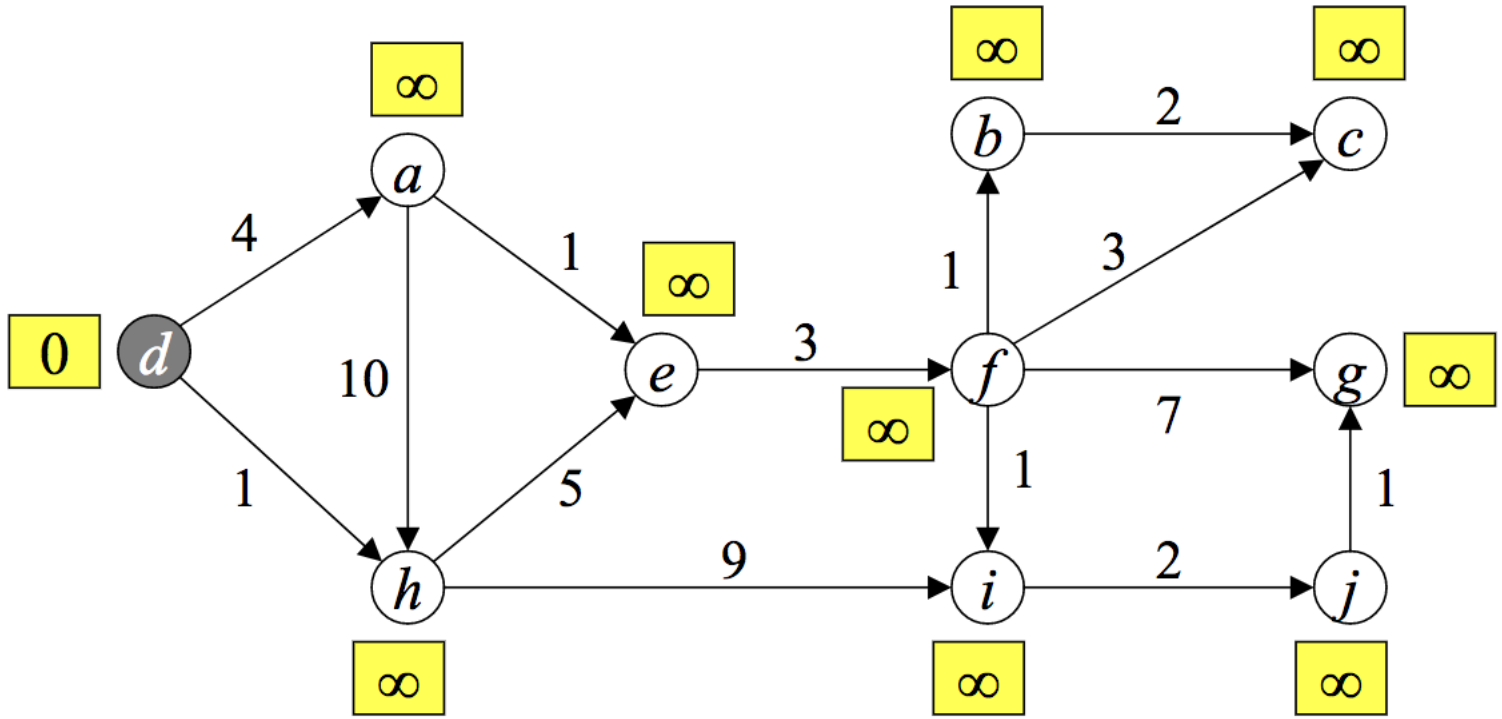
Grafi pesati: peso di un cammino e distanze

- Il peso di un cammino è la somma dei pesi dei suoi archi.
- La distanza di un vertice dall'altro è il minimo dei pesi tra tutti i cammini che congiungono il primo al secondo (se esistono).

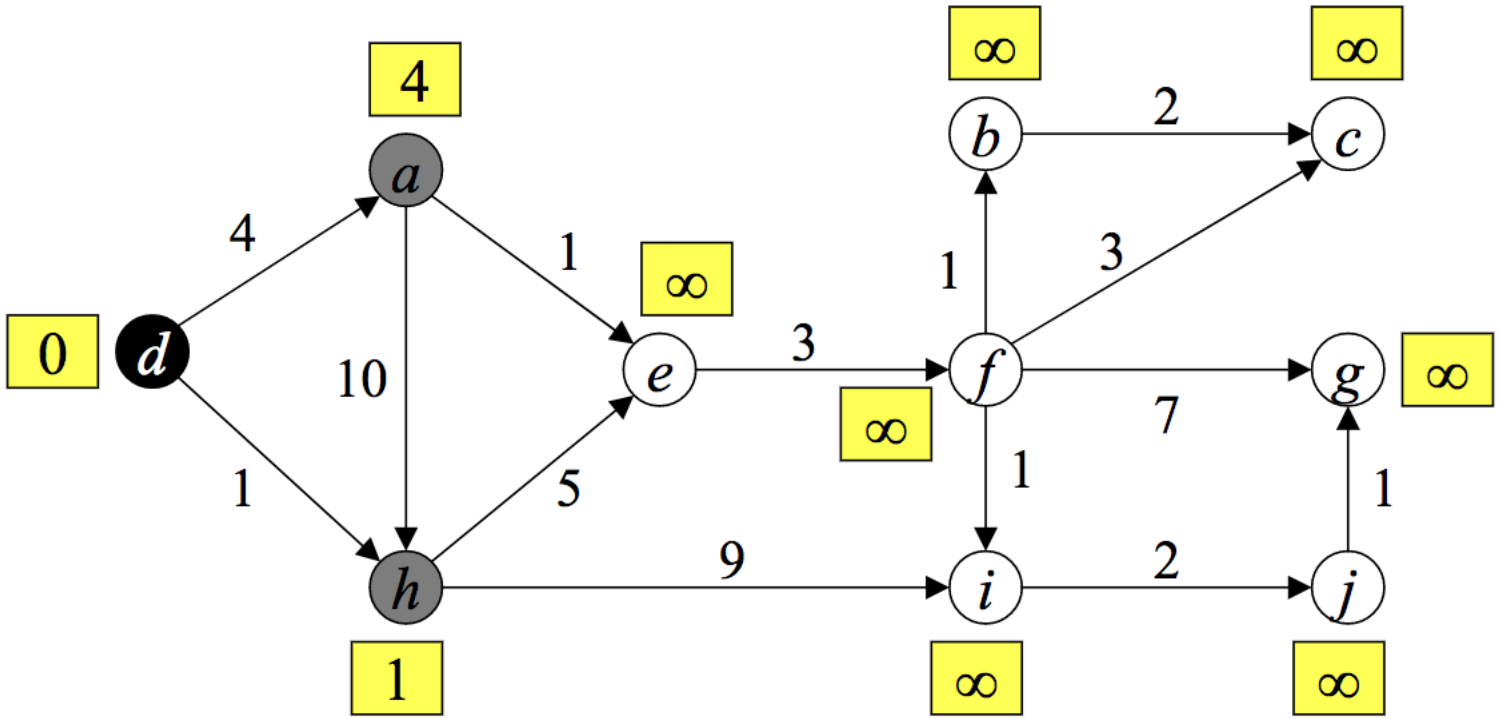
Algoritmo di Dijkstra

- Si possono pensare diversi algoritmi per calcolare il cammino più breve tra due nodi di un grafo
- Per cammino più breve **non** si intende quello con il numero minore di hop, ma il cammino con il costo minore, ovvero quello con distanza minore.
- Vediamo l'algoritmo di **Dijkstra** (1959):
  - fissiamo un nodo sorgente
  - ad ogni nodo si associa una label che contiene la distanza dal nodo sorgente lungo il miglior path conosciuto
  - allo step zero dell'algoritmo non so nulla per cui tutte le label sono a infinito
  - quando scopro che una label è il costo minimo possibile, la label diventa permanente.

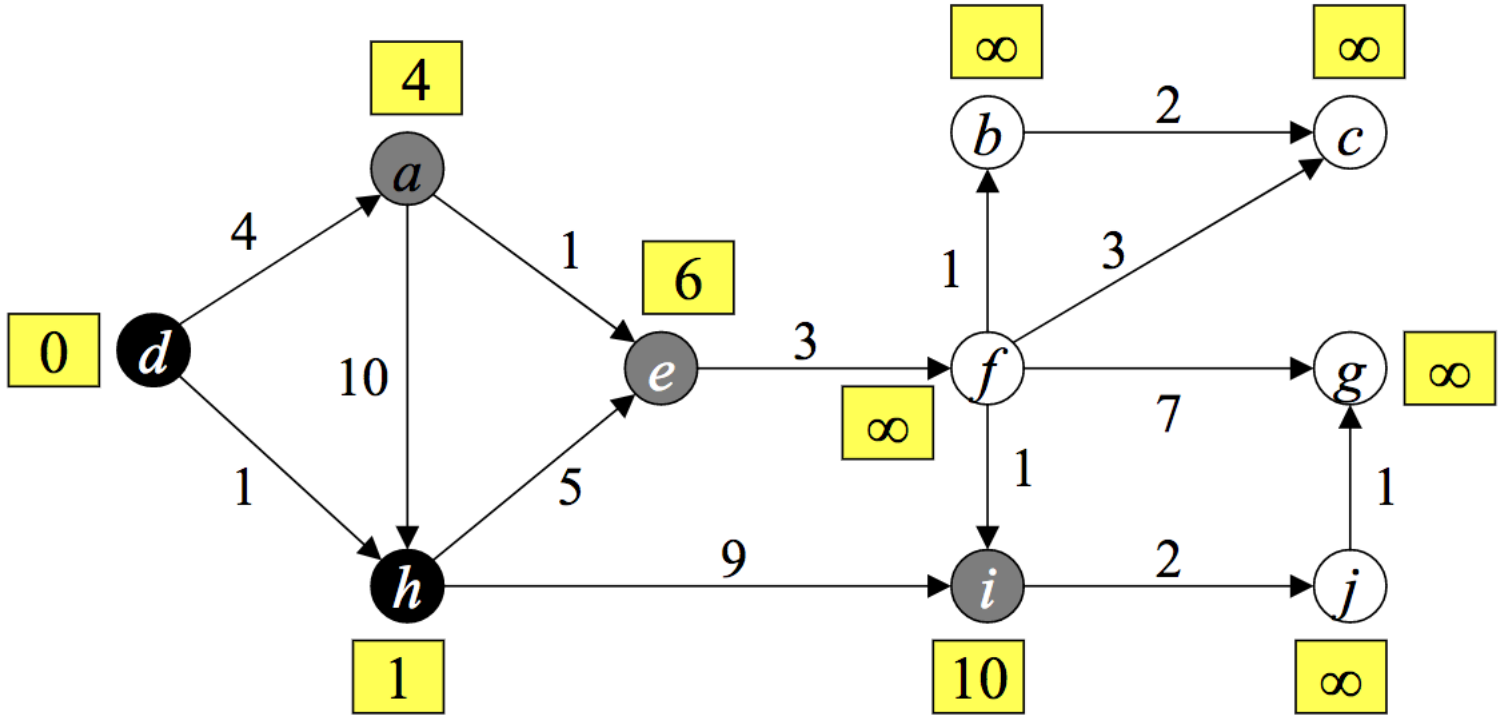
Algoritmo di Dijkstra in azione: passo 0



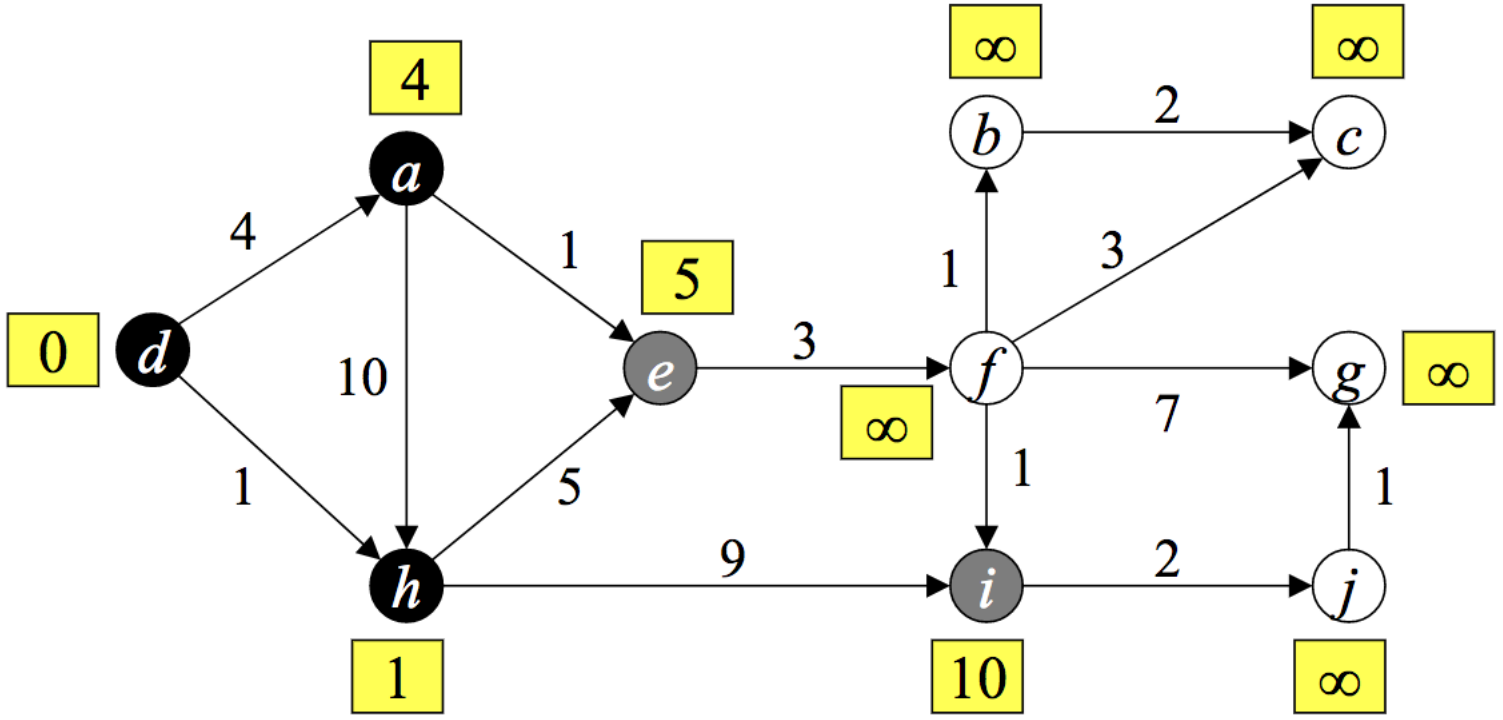
Algoritmo di Dijkstra in azione: passo 1



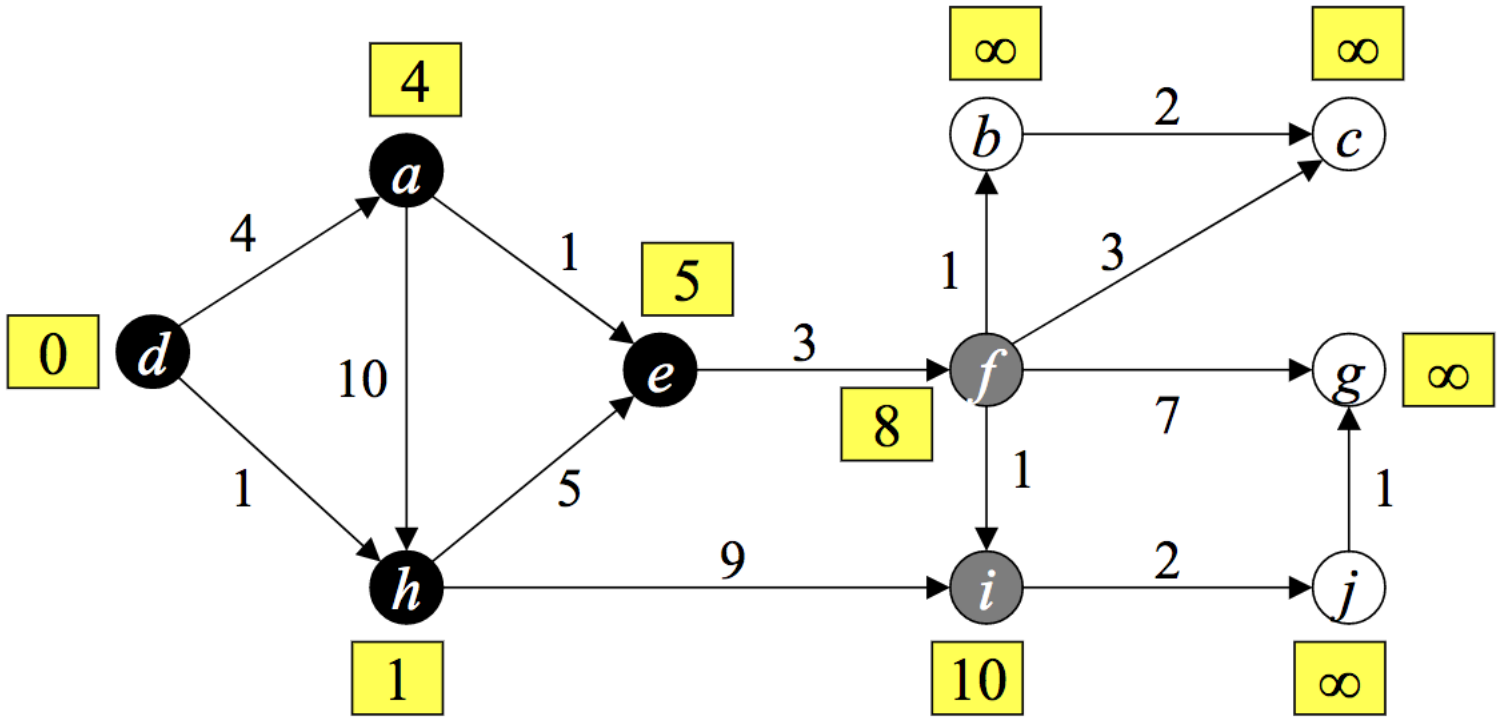
Algoritmo di Dijkstra in azione: passo 2



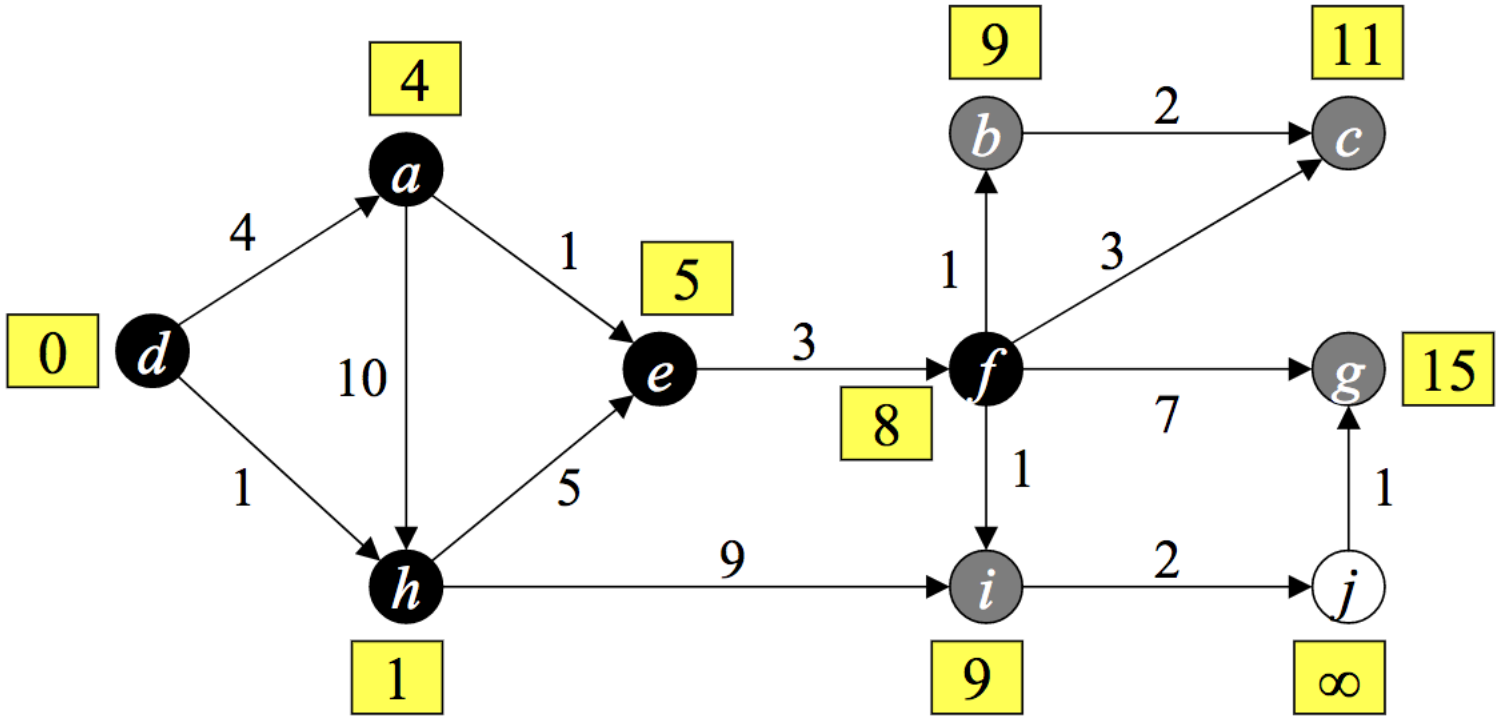
Algoritmo di Dijkstra in azione: passo 3



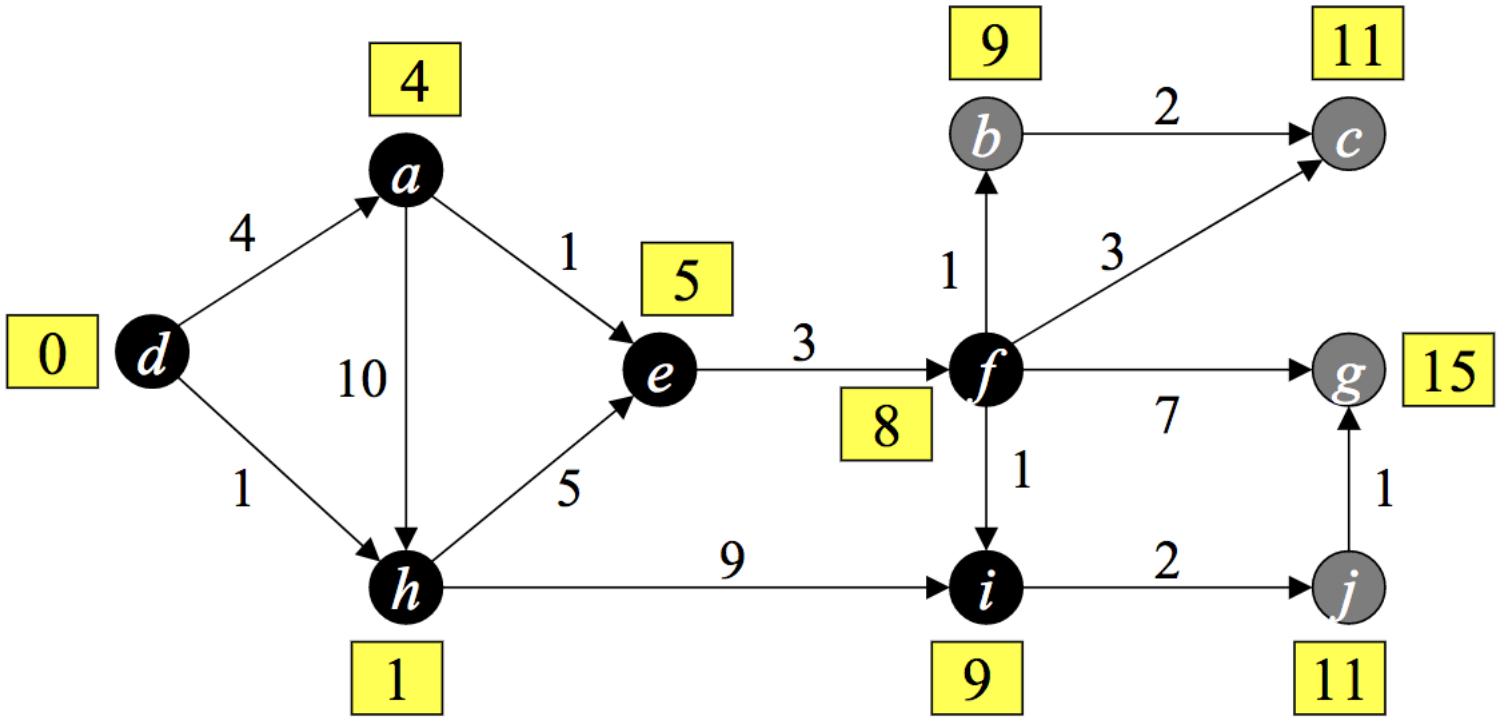
Algoritmo di Dijkstra in azione: passo 4



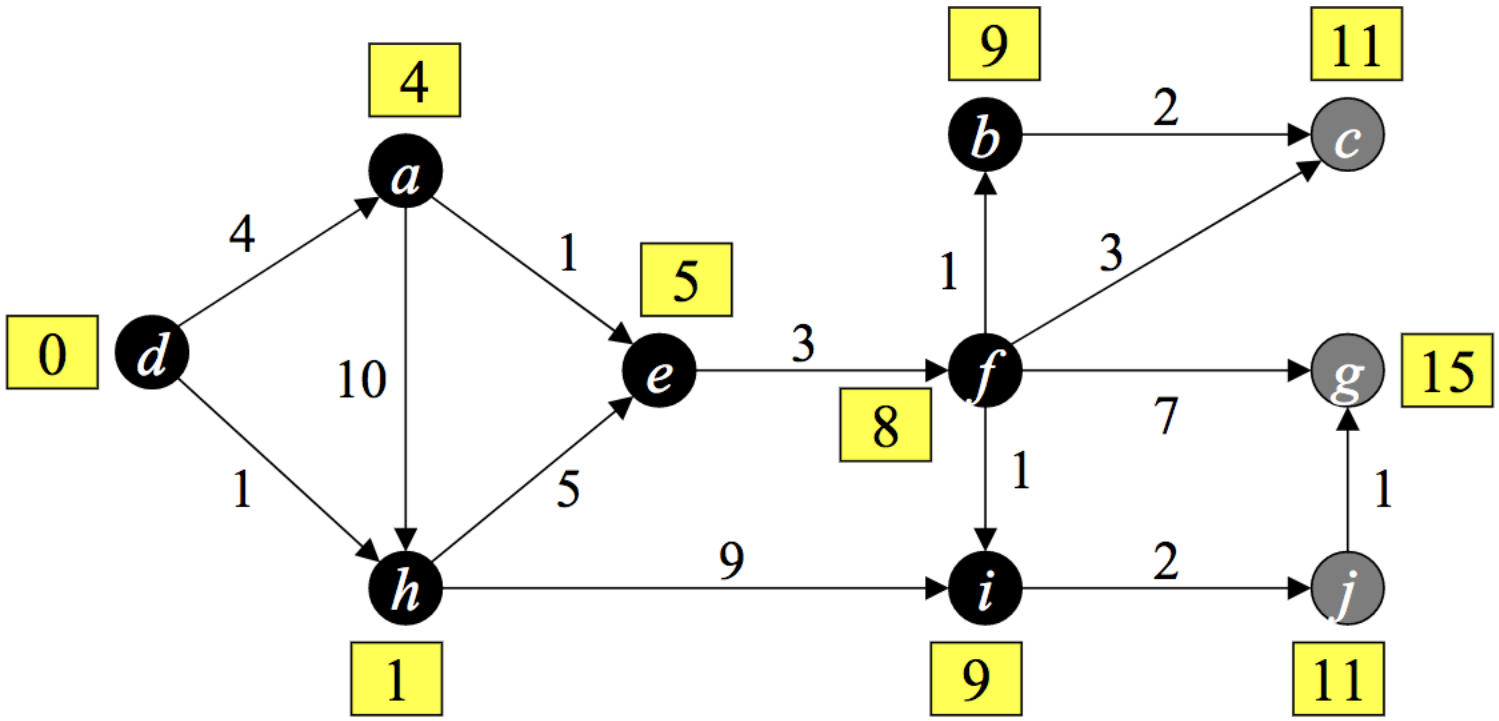
Algoritmo di Dijkstra in azione: passo 5



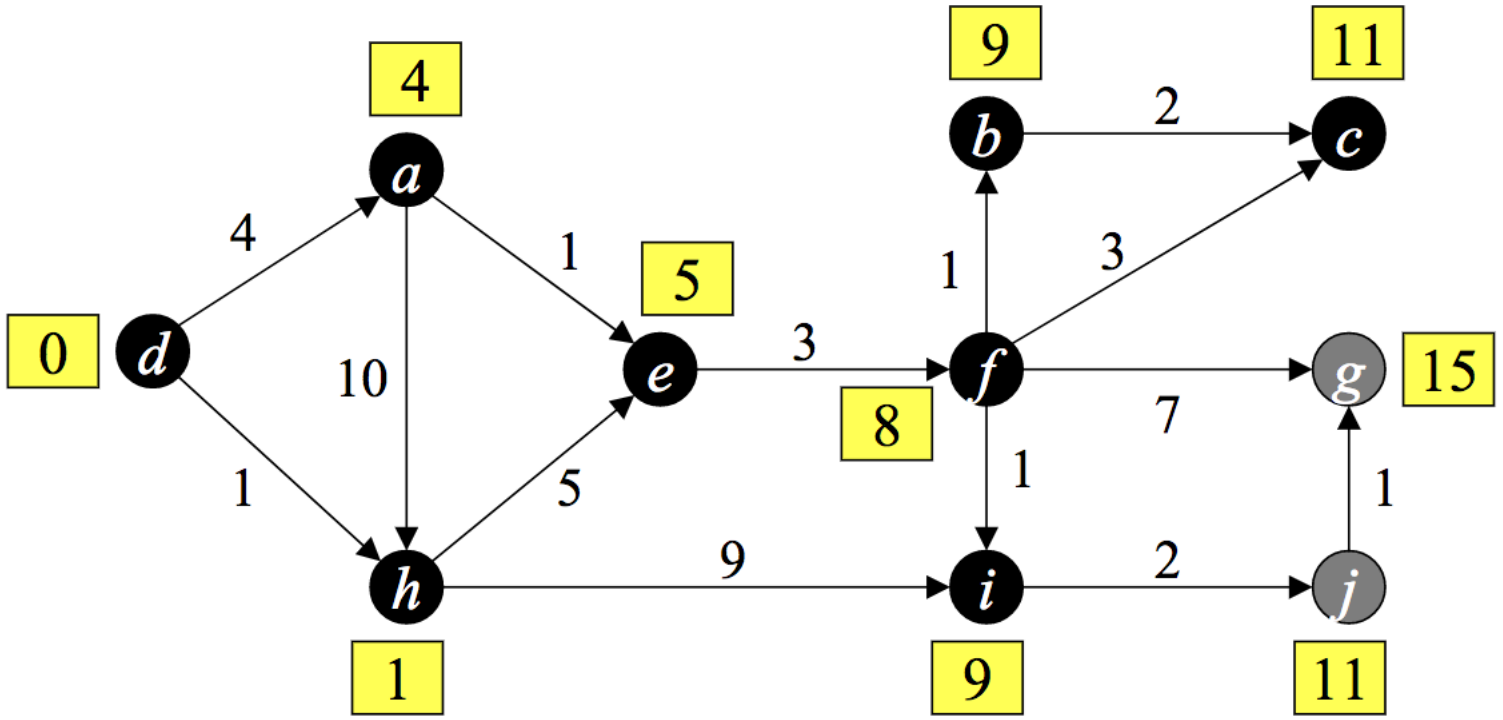
Algoritmo di Dijkstra in azione: passo 6



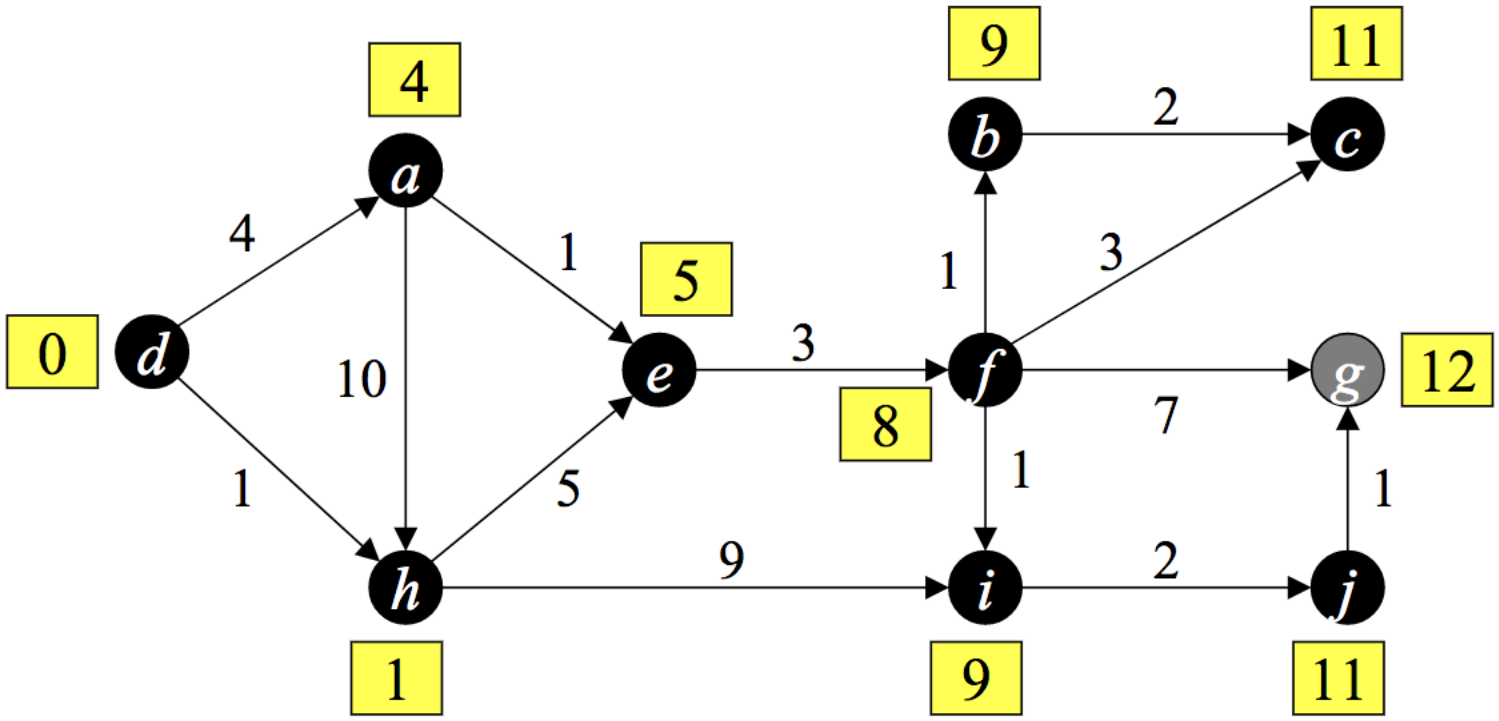
Algoritmo di Dijkstra in azione: passo 7



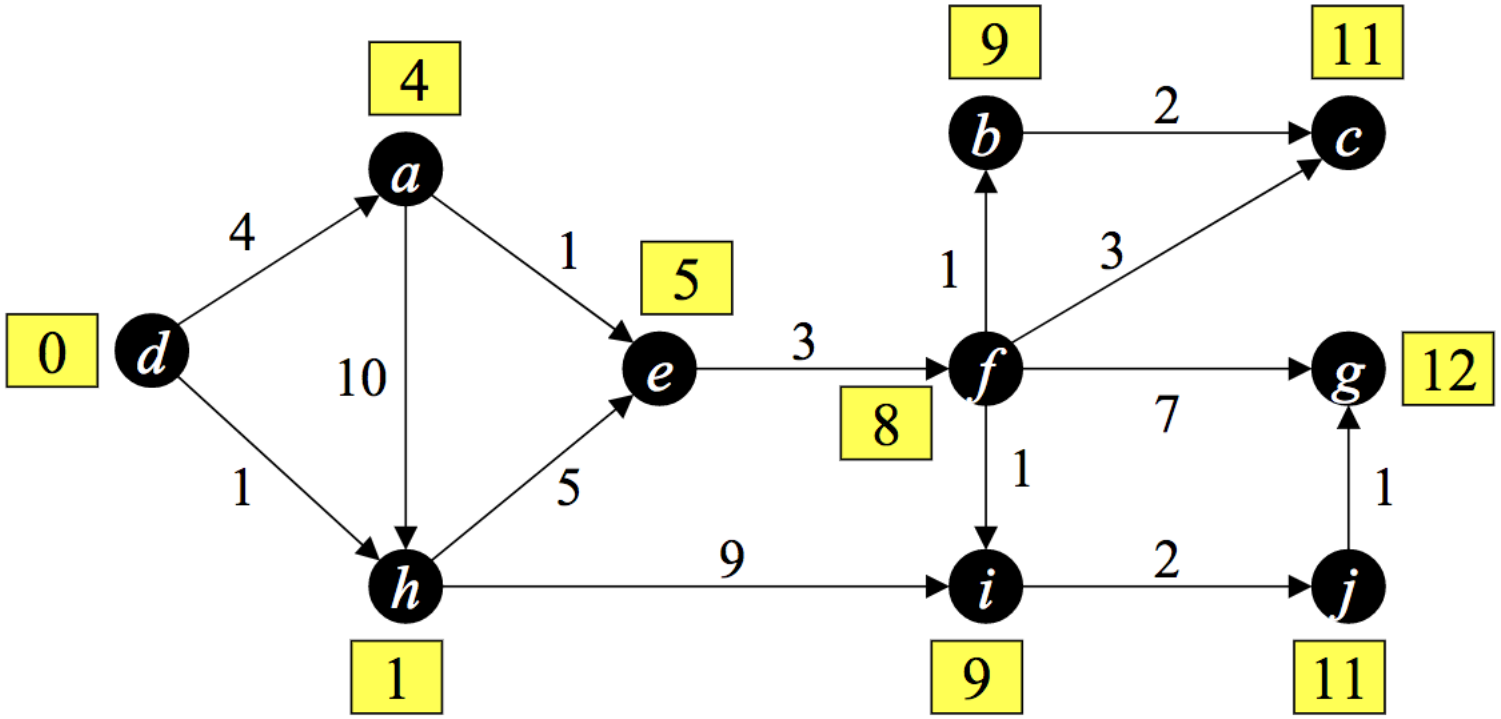
Algoritmo di Dijkstra in azione: passo 8



Algoritmo di Dijkstra in azione: passo 9



Algoritmo di Dijkstra in azione: passo 10



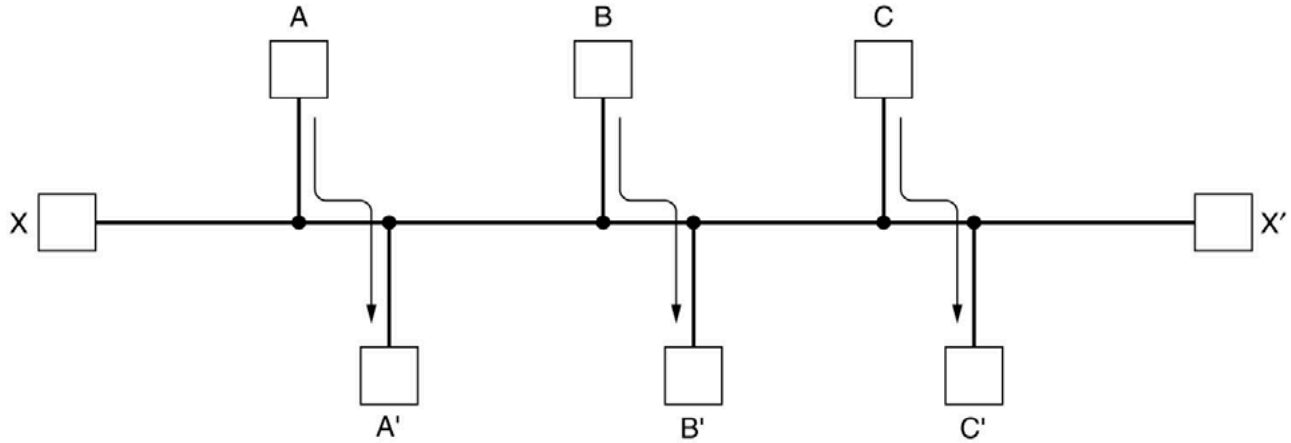
Algoritmo di Dijkstra in azione: pseudocodice

```
 1  function Dijkstra(Graph, source):
 2
 3      create vertex set Q
 4
 5      for each vertex v in Graph:             // Initialization
 6          dist[v] ← INFINITY                  // Unknown distance from source to v
 7          prev[v] ← UNDEFINED                 // Previous node in optimal path from source
 8          add v to Q                          // All nodes initially in Q (unvisited nodes)
 9
10      dist[source] ← 0                        // Distance from source to source
11
12      while Q is not empty:
13          u ← vertex in Q with min dist[u]    // Node with the least distance
14                                                  // will be selected first
15          remove u from Q
16
17          for each neighbor v of u:           // where v is still in Q.
18              alt ← dist[u] + length(u, v)
19              if alt < dist[v]:               // A shorter path to v has been found
20                  dist[v] ← alt
21                  prev[v] ← u
22
23      return dist[], prev[]
```

Ottimizzazione del routing

Abbiamo visto che il routing deve essere ottimale ovvero:

- Questo può essere in contrasto con il criterio precedente di Equità.
- Equità vs Ottimizzazione: ad esempio se il traffico **A-A’**, **B-B’** , **C-C’** è sufficiente a saturare il link orizzontale io ho ottimizzato il troughput globale a spese del traffico **X-X’:**



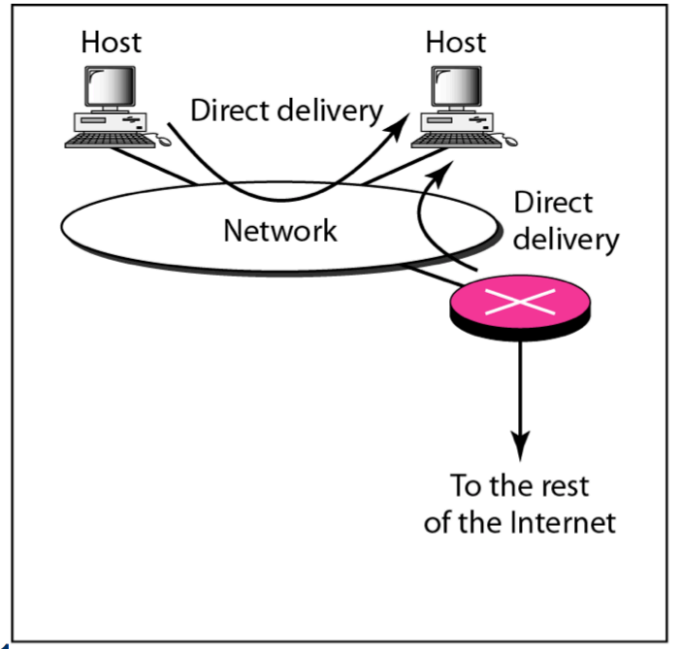
Ottimizzazione del routing

Devo anche decidere quale aspetto ottimizzare:

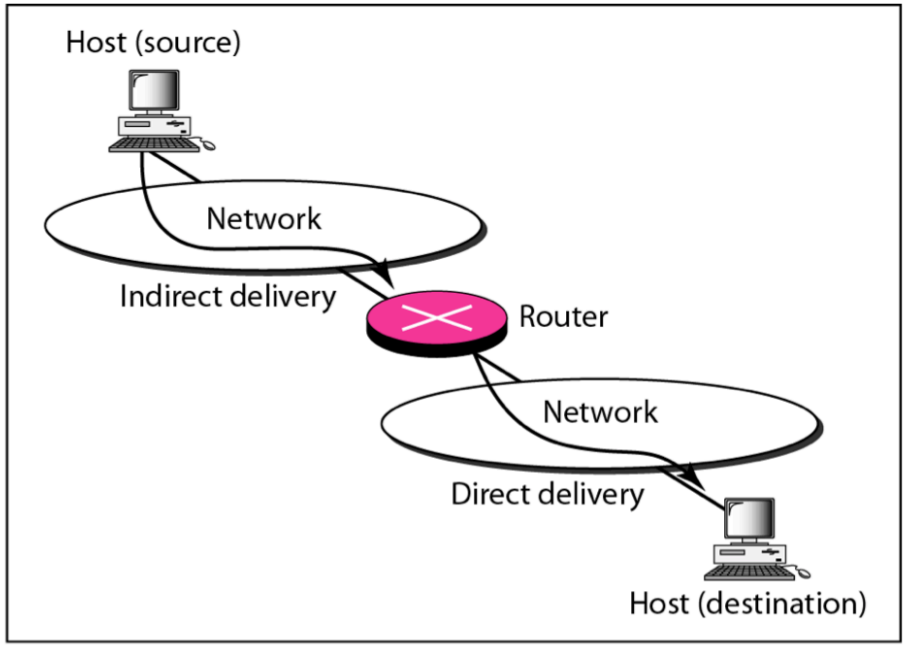
- Minimizzo il packet delay?
- Massimizzo il throughput?
- Minimizzo il costo del servizio di trasporto? (es vado su una flat ADSL)
- A volte si decide di minimizzare il numero di hops per minimizzare lo spreco di banda e questo aiuta a minimizzare il delay e ad usare il massimo throughput.

Consegna diretta e indiretta

- Quando mittente e destinazione sono nella stessa rete, il mittente può consegnare direttamente il pacchetto
- Altrimenti, se la consegna è indiretta si deve cercare nella tabella di routing l'indirizzo del prossimo router intermedio



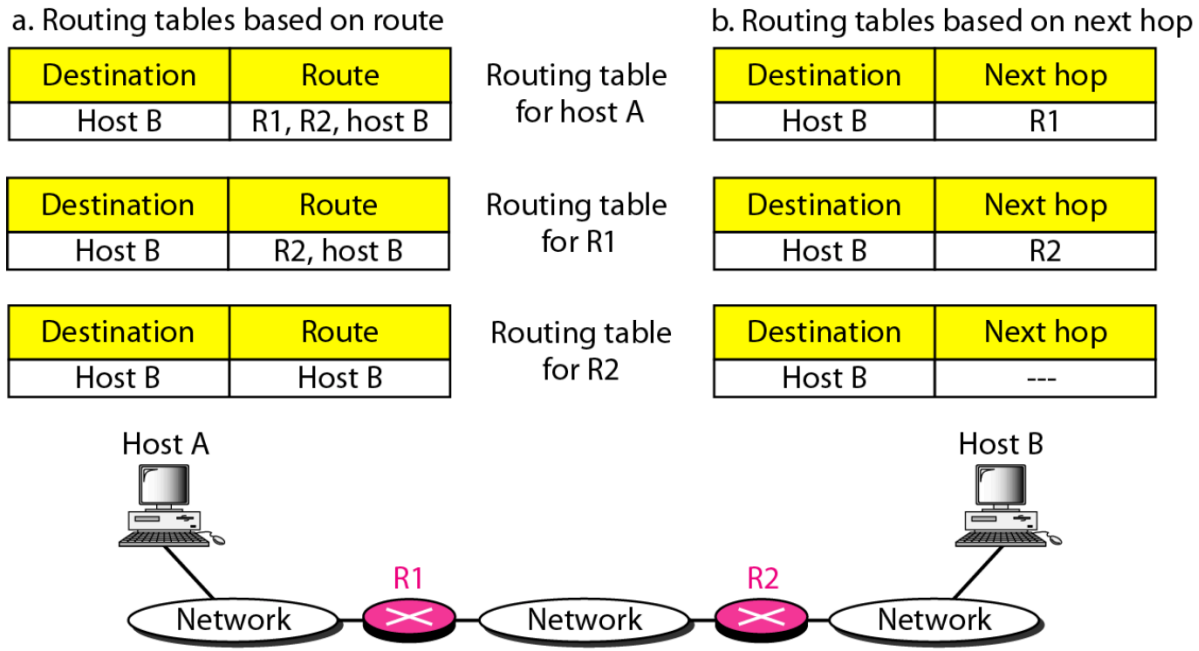
a. Direct delivery



b. Indirect and direct delivery

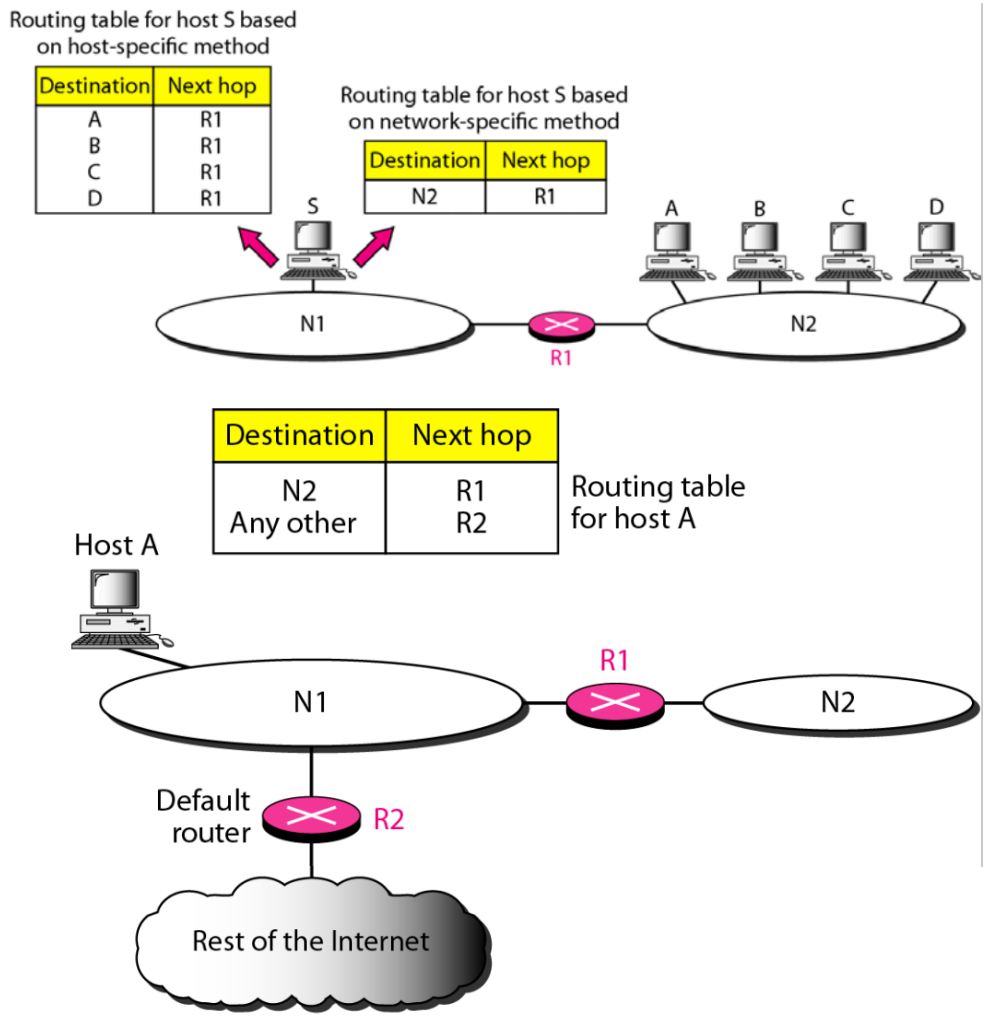
Routing table

- Come posso mettere nella routing table le informazioni per raggiungere tutte le possibili destinazioni di internet?
- Next-Hop. Non serve mettere le informazioni relative al percorso completo. Sufficiente specificare come arrivare al Next Hop



Aggregazioni e default route

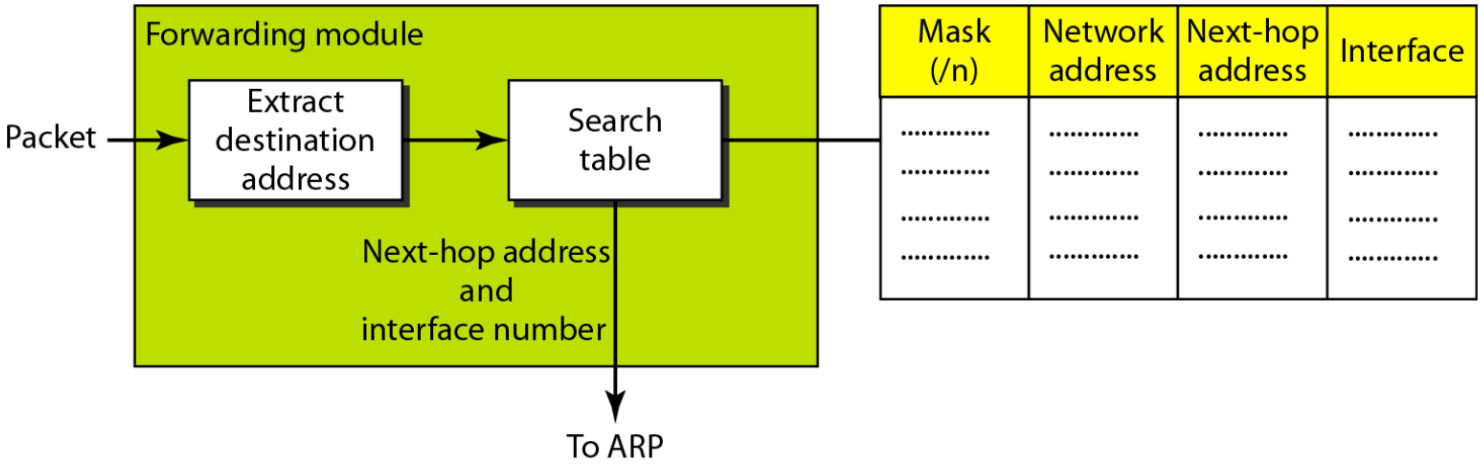
- Non occorre che io metta tutte i possibili host di destinazione nella routing table
- Basta che io inserisca le info su come raggiungere la rete per tutti gli host di quella rete
- Potrei anche specificare solo alcune reti e dare per tutte le altre reti l'indirizzo di una default route su cui saranno inoltrati tutti i pacchetti che non hanno una router



Routing table in IPv4

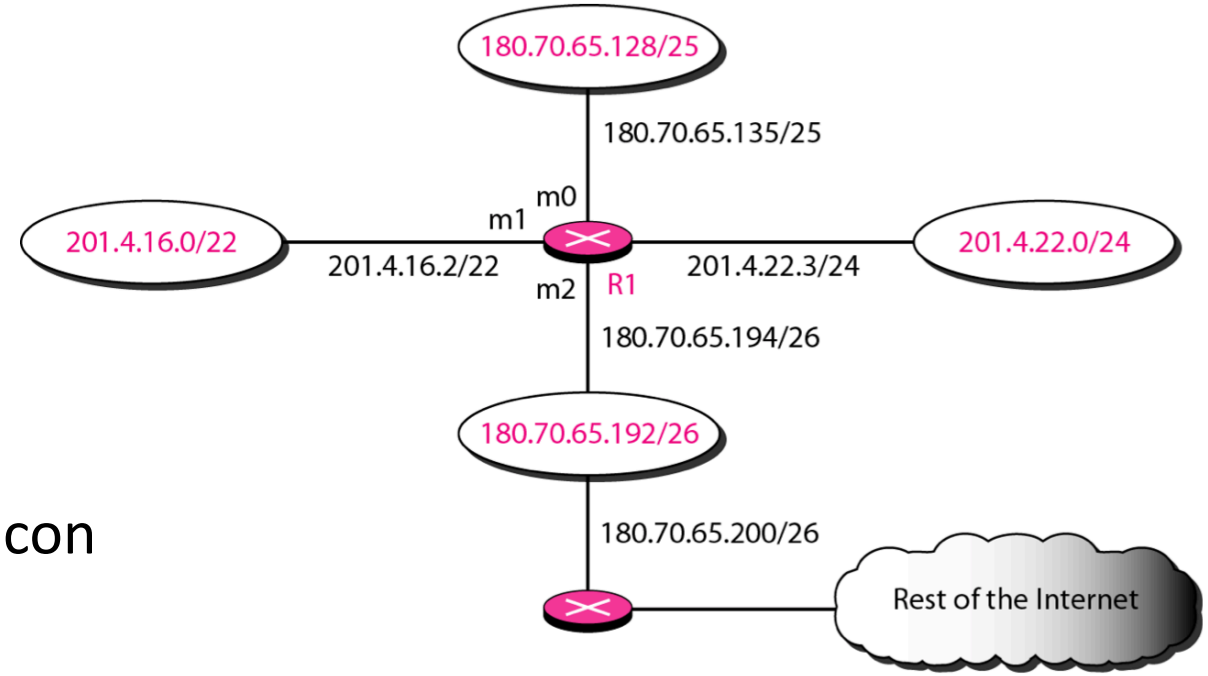
Se uso indirizzi Classless devo avere per ogni entry almeno 4 informazioni (4 colonne)

1) la maschera
2) l'indirizzo del blocco di indirizzi di destinazione
3) il next hop (l'indirizzo del prossimo router)
4) l'interfaccia da usare



Esempio

Mask	*Network Address*	*Next Hop*	*Interface*
/26	180.70.65.192	—	m2
/25	180.70.65.128	—	m0
/24	201.4.22.0	—	m3
/22	201.4.16.0	....	m1
Any	Any	180.70.65.200	m2



Dove va un pacchetto che arriva a R1 con destinazione:

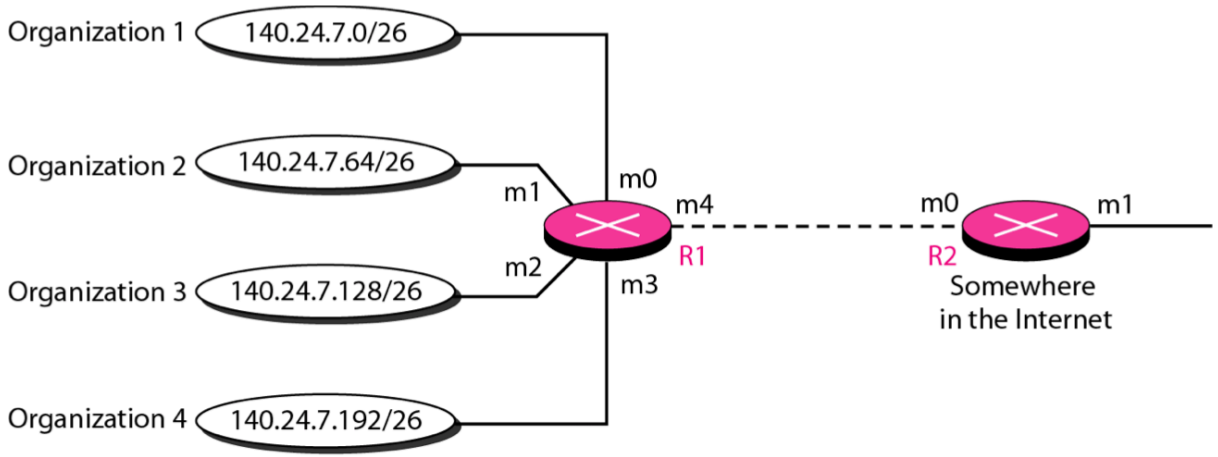
1. 180.70.65.140 ?
2. 201.4.22.35?
3. 18.24.32.78 ?

Soluzione

1. Applico la mask di **26 bit** a 180.70.65.140 ottengo 180.70.65.128 che **non corrisponde** alla prima riga. Applico allora la mask a **25 bit** e ottengo **180.70.65.128 che corrisponde** alla seconda riga. Mando il pacchetto verso **m0** (con una ARP)
2. Prima riga 201.4.22.35/26 ottengo 201.4.22.0 e poi , seconda riga 201.4.22.35/25 ottengo 201.4.22.0. Nessun match. Con la terza riga **201.4.22.35/24** invece ho un **“match”.** Mando il pacchetto a **m3** (via ARP)
3. **18.24.32.78 [/26, /25, /24] non ho un “match” quindi uso next hop di default**

Aggregazione

- L'aggregazione mi permette di indicare con l'indirizzo di rete tutti gli host in quella rete. Posso usarla anche per aggregare reti che hanno lo stesso next hop, come fa R2 in figura:



Mask	Network address	Next-hop address	Interface
/26	140.24.7.0	----------	m0
/26	140.24.7.64	----------	m1
/26	140.24.7.128	----------	m2
/26	140.24.7.192	----------	m3
/0	0.0.0.0	Default	m4

Routing table for R1

Mask	Network address	Next-hop address	Interface
/24	140.24.7.0	----------	m0
/0	0.0.0.0	Default	m1

Routing table for R2

Formato tabella di routing

Abbiamo visto che deve avere almeno 4 colonne ma dipende dal costruttore del router che può aggiungere altre info. Vediamo un tipico esempio:

- Mask
- Indirizzo di destinazione
- Indirizzo next hop
- Interface

Mask	Network address	Next-hop address	Interface	Flags	Reference count	Use
.....	.....	.....	.....	.....	.....	.....

Formato tabella di routing

- Flag
  - **U U**p vuol dire che il Next Hop è funzionante
  - **G G**ateway il Next Hop è in un'altra rete, quindi consegna indiretta
  - **H H**ost specific. L'indirizzo specificato è quello di un host
  - **D** A**dd**ed by redirection. Riga inserita in seguito ad un messaggio ICMP di redirezione
  - **M m**odified by redirection. Riga modificata in seguito ad un messaggio ICMP di redirezione
- Contatore: numero di utenti che stanno usando questo link (connessioni contemporanee)
- Use: Mostra quanti pacchetti sono stati spediti alle destinazioni di questa riga

Netstat

I comandi Linux o Windows netstat permettono di vedere le informazioni di routing

- Es: netstat -rn serve per avere la tabella di routing (-r) in formato numerico (-n)
- Gateway equivale a routing e indica il next hop
- Ifconfig invece serve per avere informazioni sull'interfaccia di rete

$ netstat -rn

Kernel IP routing table

Destination	Gateway	Mask	Flags	Iface
153.18.16.0	0.0.0.0	255.255.240.0	U	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	lo
0.0.0.0	153.18.31.254	0.0.0.0	UG	eth0

$ ifconfig eth0

```
eth0   Link encap:Ethernet  HWaddr 00:B0:D0:DF:09:5D
inet addr:153.18.17.11  Bcast:153.18.31.255  Mask:255.255.240.0
. . .
```

Rete corrispondente

$ netstat -rn

Kernel IP routing table

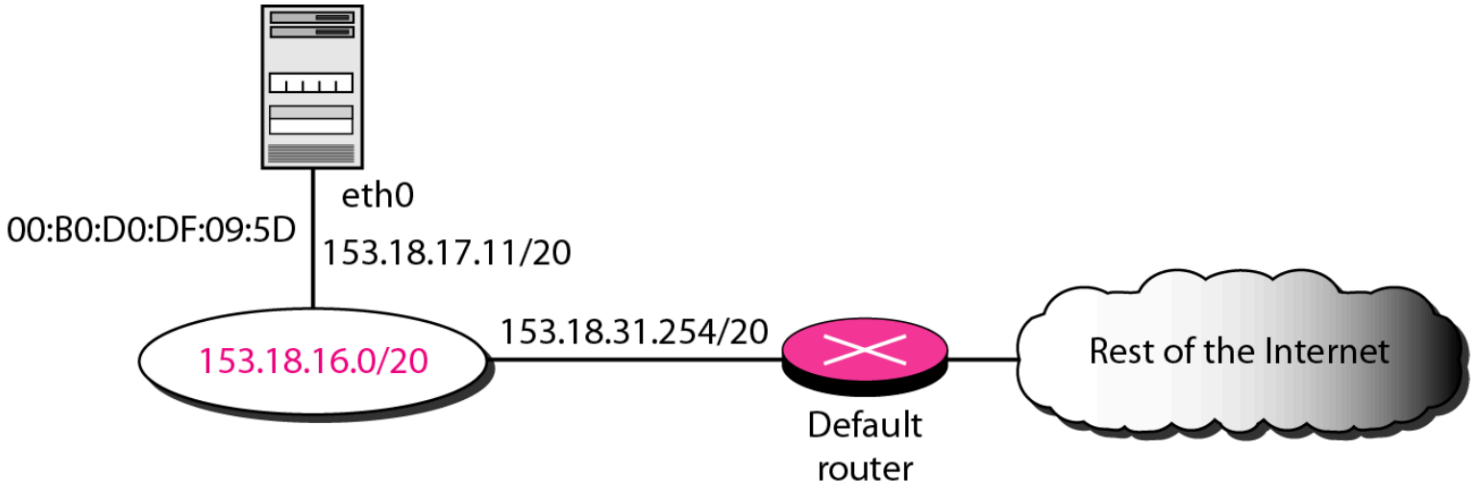
Destination	Gateway	Mask	Flags	Iface
153.18.16.0	0.0.0.0	255.255.240.0	U	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	lo
0.0.0.0	153.18.31.254	0.0.0.0	UG	eth0

$ ifconfig eth0

eth0 Link encap:Ethernet HWaddr 00:B0:D0:DF:09:5D

inet addr:153.18.17.11 Bcast:153.18.31.255 Mask:255.255.240.0

. . .



Come funziona il routing dinamico?

- Cambiano le regole di routing per riflettere cambiamente di topologia e di traffico
- Alcuni prendono informazioni solo dai router adiacenti mentre altri da tutti i router
- Usano diverse metriche:
  - Distanze in km
  - Distanza in numero di hops
  - Stime del tempo di attraversamento (= latenza)
  - Bandwidth
  - Costie conomici

Shortest Path Routing

- Costruisco un grafo della subnet in cui ogni nodo è un router e ogni linea è il link tra i due router
- Per trovare la route migliore l'algoritmo cerca il path più corto (**shortest path**)
- La lunghezza del path la posso misurare in **hops** oppure in **lunghezza in km** dei link ma posso anche prendere **l'accodamento medio** o la **latenza** di un link, per cui il path più corto risulta essere il più veloce e non il più corto in kilometri .

Algoritmo di Dijkstra!!!

Più in pratica….

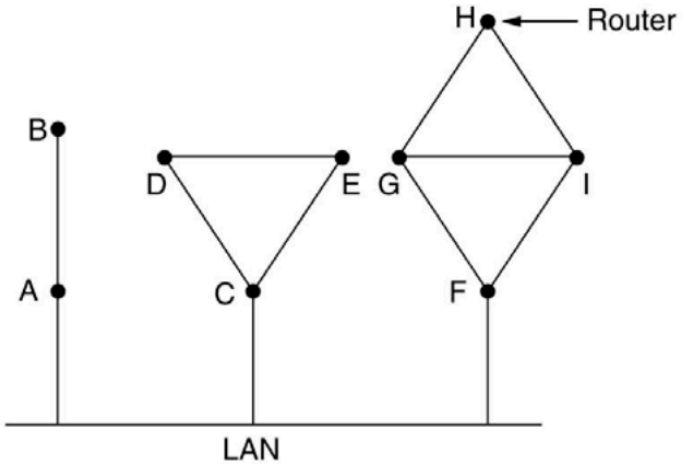
Ogni router deve:

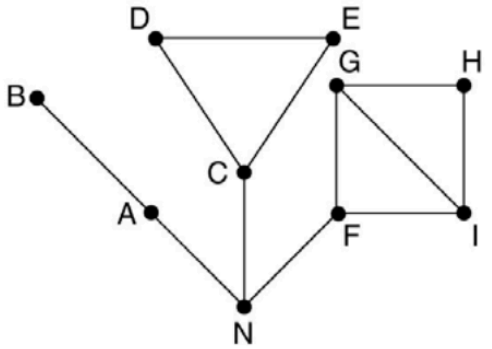
- Scoprire i suoi vicini e imparare il loro indirizzo di rete
- Misurare il delay o il costo verso ciascun vicino
- Costruire un pacchetto che dice a tutti cosa ha imparato
- Mandare questo pacchetto a tutti gli altri router
- Calcolare il shortest path verso ogni router
- Costruire le tabelle di routing

Conoscere i vicini

Al boot il router vuole conoscere i suoi vicini.

- Manda loro un pacchetto HELLO su ogni linea punto-punto.
- Il router all'altro capo risponde dicendo chi è
- I nomi devono essere globalmene unici
- Se due o più router sono collegati ad una LAN posso modellare la LAN come un router virtuale. Il fatto che posso andare da A a C via LAN lo modello come un path A*N*C dove N è la LAN





Misurare i costi

Ogni router deve conoscere o stimare il delay verso ognuno dei suoi vicini:

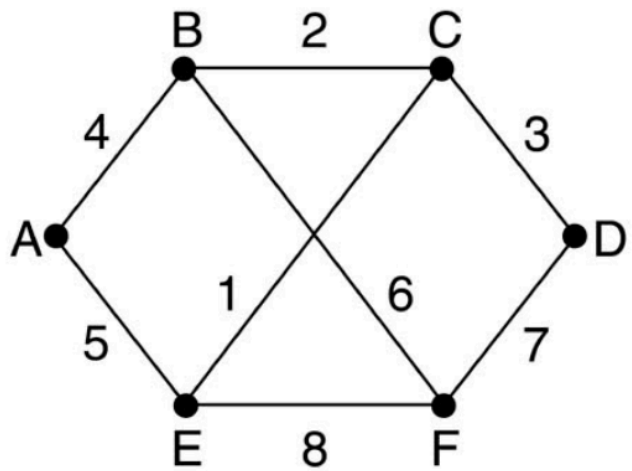
- Lo può fare mandando un pacchetto di ECHO che dall'altra parte viene rimandato immediatamente, misurando round trip time diviso per due ho una buona stima del delay
- Posso ripetere la misura e prendere una media
- Questo implica che i delay siano simmetrici ma questo non è sempre vero

Creo i Link State Packet

I Link state packets sono implementati dall0 Open Shortest Path First (OSPF) protocol.

- Il pacchetto contiene il mittente, un numero di sequenza e anzianità e la lista dei vicini.
- Per ogni vicino contiene il costo verso quel vicino

Fare i pacchetti è facile. Il difficile è decidere quando. Li costruisco periodicamente o solo quando succede qualcosa di significativo? Es. Una linea (o un vicino) scompare (o ricompare) o alcune metriche cambiano significativamente?



Link State Packets

A
Seq.
Age
B	4
E	5

B
Seq.
Age
A	4
C	2
F	6

C
Seq.
Age
B	2
D	3
E	1

D
Seq.
Age
C	3
F	7

E
Seq.
Age
A	5
C	1
F	8

F
Seq.
Age
B	6
D	7
E	8

Distribuzione degli LSP

- E’ la parte più delicata perché i router che ricevono i primi pacchetti possono cambiare le loro routes, quindi abbiamo router con diverse versioni della topologia, inconsistenze, loop e macchine non raggiungibili
- Ogni pacchetto viene mandato in **flooding verso tutte le linee** esclusa quella di provenienza
- Ogni pacchetto ha un numero di sequenza che viene incrementato per ogni pacchetto mandato. Quando arriva un nuovo pacchetto Link State (LSP) se è nuovo viene forwardato, ma se è un duplicato viene scartato, se arriva con un numero più basso di quello in uso viene scartato come obsoleto.

Problemi

- Se il numero di sequenza va in overflow tutti si confondono. Quindi uso numeri a 32 bit che con un LSP al secondo ci mette 137 anni a wrappare
- Se un router crasha perde traccia della sequenza e ricomincia da 0. Il prossimo pacchetto viene scartato come duplicato
- Se un numero di sequenza si corrompe e per esempio ricevo 65540 (0x1004) invece che 4 (0x0004) i pacchetti da 5 a 65540 vengono buttati come obsoleti

Soluzione. Metto un campo “Age” a 60 che viene decrementato ogni secondo. Quando arriva a zero le informazioni di quel router vengono scartate. Di solito un pacchetto arriva ogni 10 secondi per cui le informazioni vanno in timeout solo se il router è down o perde sei pacchetti di fila

Calcolo delle route

- Quando il router ha tutto un insieme di LSP può costruirsi l'intero grafo
- Ogni link è rappresentato due volte, una per ogni direzione e posso fare una media o usare i due valori separatamente
- Posso quindi usare l'algoritmo di Dijkstra per calcolarmi lo Shortest Path per ogni destinazione. Mi creo la nuova routing table e ricomincio le normali operazioni

Il protocollo **OSPF** molto usato in Internet usa Link State Routing

Un altro protocollo LSR è IS-IS (Intermediate System – Intermediate System) progettato per DECnet e adottato in seguito da ISO per il protocollo connectionless CLNP. Ora IS-IS viene usato anche in qualche backbone IP.