.DIEM

Corso di laurea in
Ingegneria Informatica

Reti di Calcolatori
A.A. 2024/25

Routing algorithms

Docenti: Vincenzo Auletta
Diodato Ferraioli

FUNZIONALITÀ DEL LIVELLO DI RETE

Ricorda: due funzionalità:

- *inoltro:* trasferisci pacchetti da un interfaccia ad un'altra di un router

data plane

- *routing:* determina la rotta intrapresa dai pacchetti dalla sorgente alla destinazione

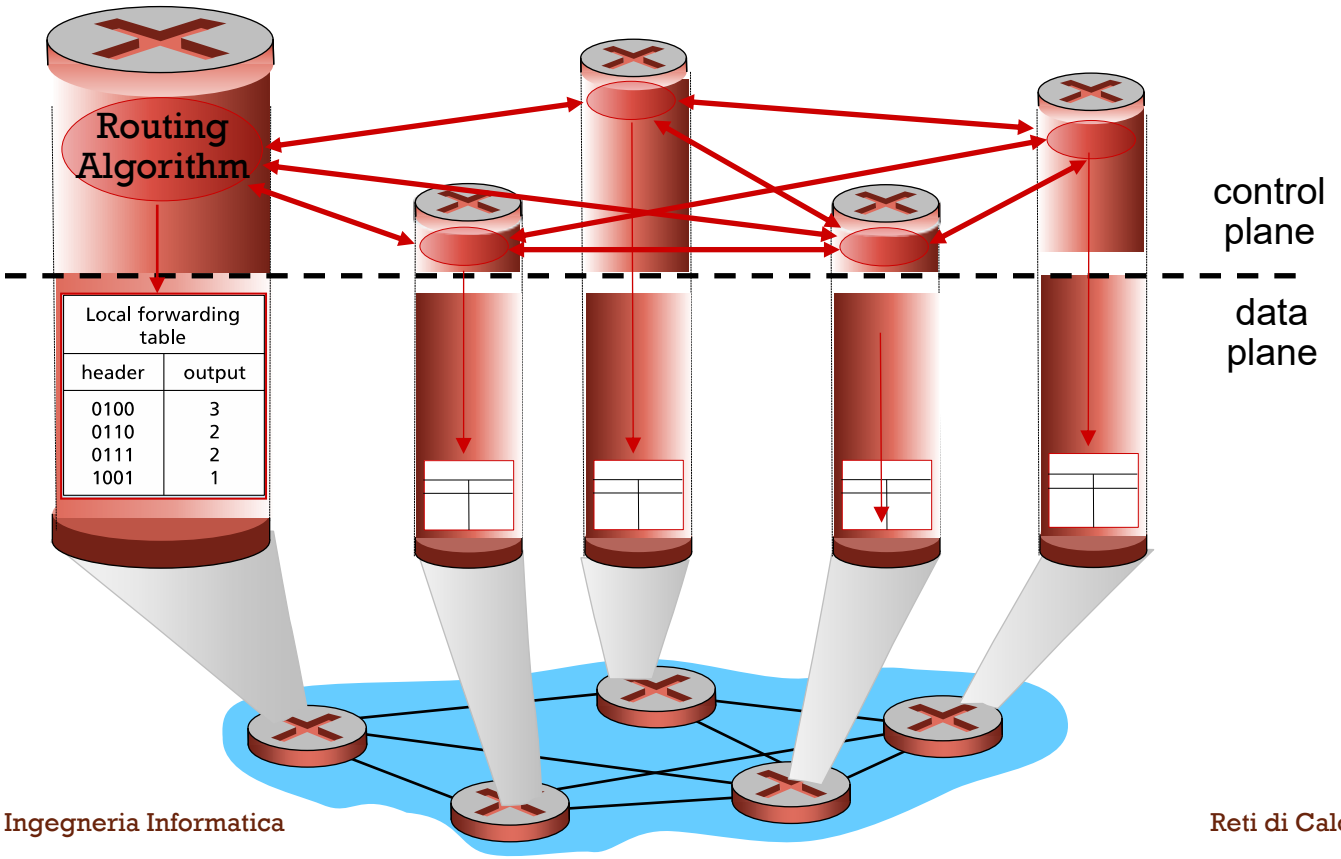
control plane

Due approcci per strutturare il piano di controllo:

- Controllo per-router (traditionale)
- Controllo logicamente centralizzato (software defined networking)

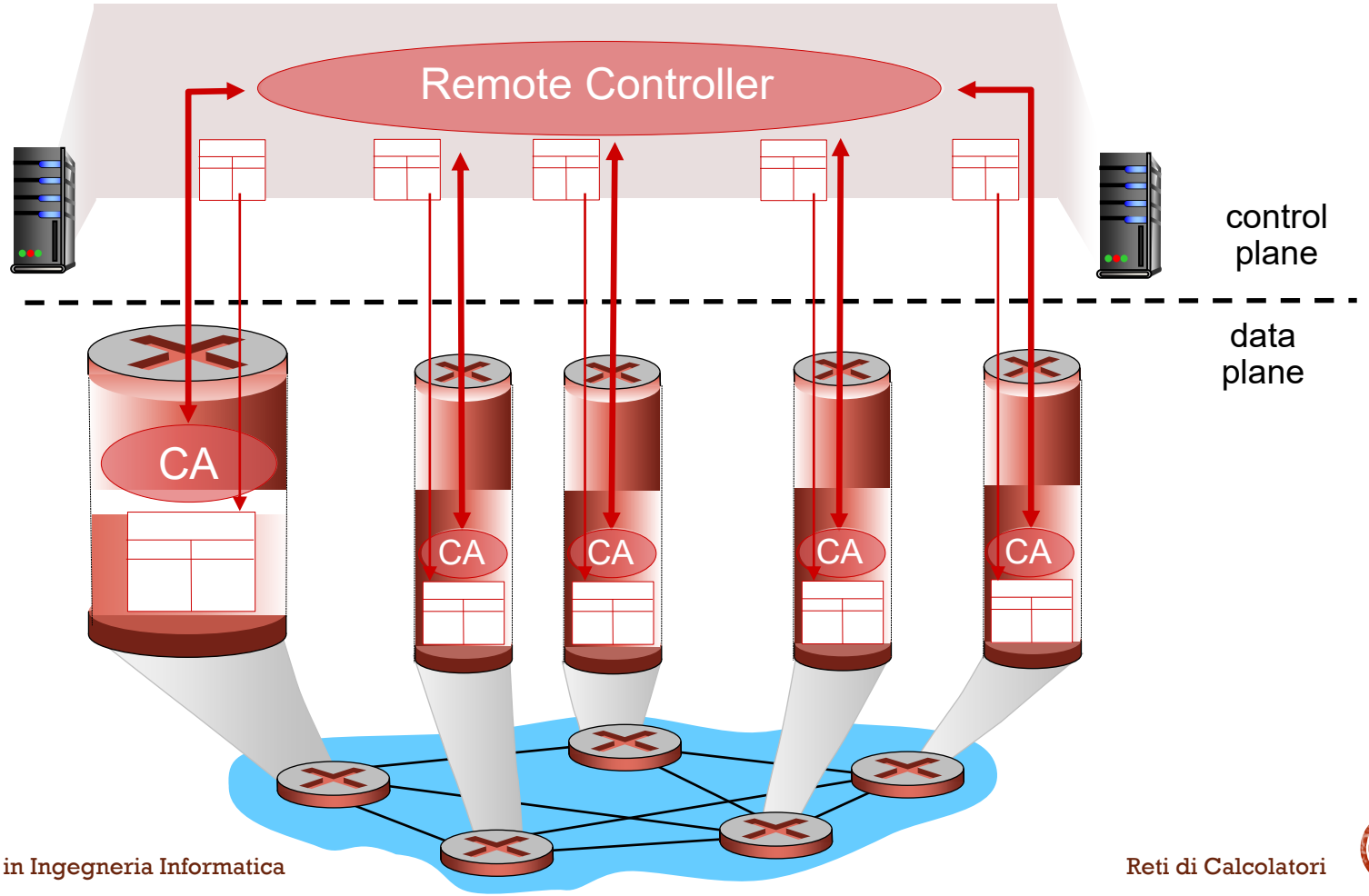
PIANO DI CONTROLLO PER-ROUTER

Ogni router ha componenti che eseguono un algoritmo di routing e interagisce con gli altri router per computare le tabelle di inoltro



PIANO DI CONTROLLO LOGICAMENTE CENTRALIZZATO

Un controller distinto (solitamente remoto) interagisce con gli agenti di controllo locale (CAs) presenti nei router per computare le tabelle di inoltro

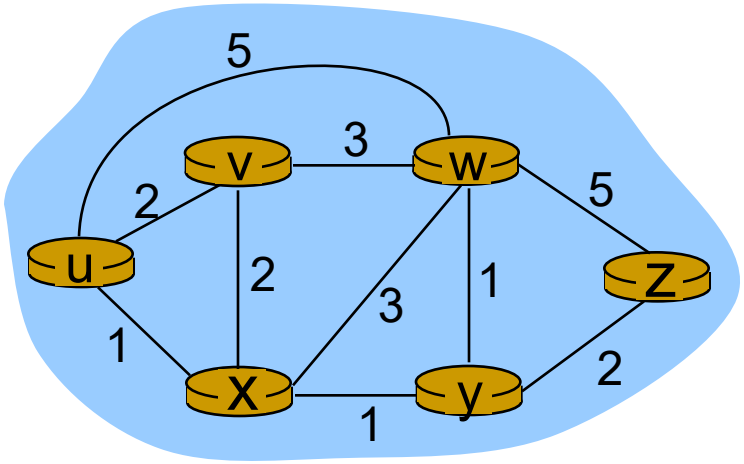


PROTOCOLLI DI ROUTING

Obiettivo: determinare cammini (o rotte) “buone”, dall’host mittente all’host ricevente, attraverso una rete di routers

- cammino: la sequenza di router che i pacchetti attraverseranno per andare dalla data sorgente alla data destinazione
- “buono”? meno “costoso”, più “veloce”, “meno congestionato”

GRAFI: ASTRAZIONE DELLA RETE

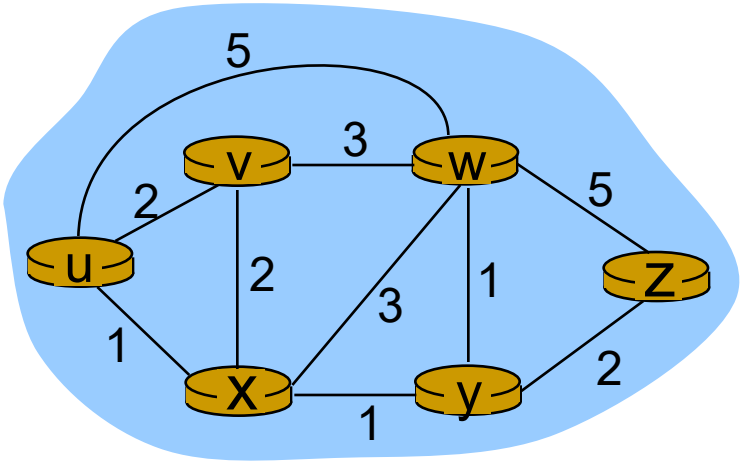


grafo: G = (N,E)

N = insieme di routers = { u, v, w, x, y, z }

E = insieme di links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

GRAFI: COSTI



c(x,x') = costo del link (x,x')
e.g., c(w,z) = 5

costo potrebbe essere sempre 1, o inversamente relato alla bandwidth, o inversamente relato alla congestione

costo di un cammino $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Domanda chiave: qual è il path di minor costo tra u e z ?
routing algorithm: algoritmo che trova il path di costo minimo

CLASSIFICAZIONE DEGLI ALGORITMI DI ROUTING

Q: informazioni globali o decentralizzate?

global:

- tutti i router hanno informazioni complete sulla topologia e il costo dei link
- Algoritmi "link state"

decentralized:

- i router conoscono solo i vicini a cui sono fisicamente connessi e il costo di tali link
- un processo iterativo di computazione, tramite lo scambio di info con i vicini
- "distance vector" algorithms

Q: statico o dinamico?

statico:

- le rotte cambiano lentamente nel tempo

dinamico:

- le rotte cambiano più frequentemente
  - Aggiornamenti periodici in risposta al cambio di costo dei link

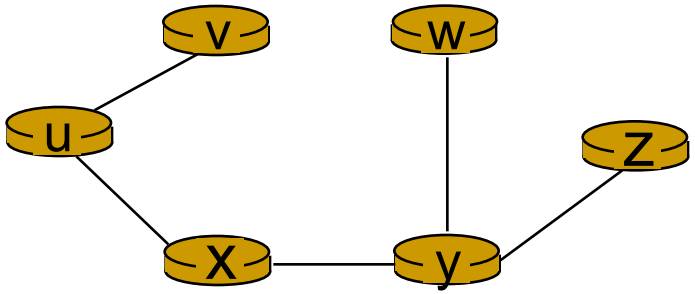
UN ALGORITMO DI ROUTING LINK-STATE

L'algoritmo di Dijkstra

- La topologia della rete e il costo dei link sono noti a tutti i nodi
  - Si può ottenere attraverso un "link state broadcast"
  - I nodi hanno le stesse info
- Computa il cammino di costo minimo da una sorgente a tutti gli altri nodi
  - Da cui si ottiene la *forwarding table* per quel nodo
- iterativo: dopo k iterazioni, si conosce il path di costo minimo verso k destinazioni

COME COSTRUIRE LE TABELLE DI INOLTRO?

L'algoritmo di Dijkstra restituisce l'albero dei cammini minimi da u a tutti gli altri nodi



La tabella di inotro risultante per il router u:

destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

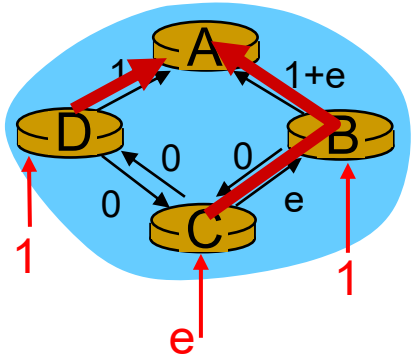
L'ALGORITMO DI DIJKSTRA: DISCUSSIONE

Complessità dell'algoritmo: n nodi

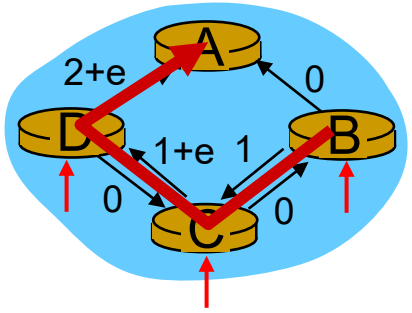
- Ad ogni iterazione si controllano tutti i nodi per cui il cammino minimo da u non è ancora noto
- n(n+1)/2 controlli: $O(n^2)$
- Sono possibili implementazioni più efficienti: O(n log n)

Sono possibili oscillazioni:

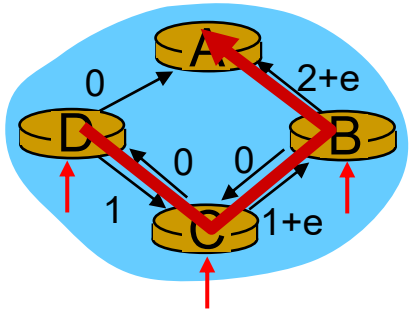
- e.g., quando il costo dei link dipende dal traffico trasportato



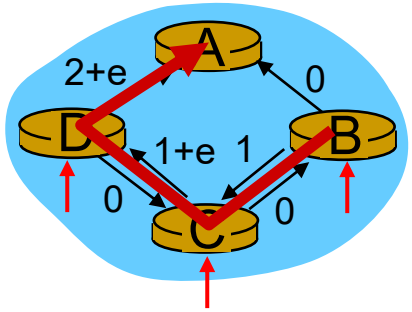
inizialmente



Dati questi costi, trova nuove rotte…. e quindi nuovi costi



Dati questi costi, trova nuove rotte…. e quindi nuovi costi



Dati questi costi, trova nuove rotte…. e quindi nuovi costi

ALGORITMI DISTANCE VECTOR

Equazione di Bellman-Ford

sia

$d_x(y)$:= il costo del cammino minimo da x a y

allora

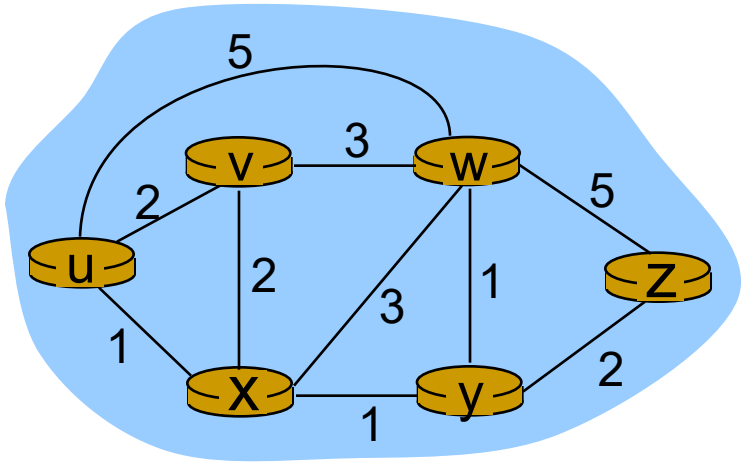
$$d_x(y) = min_v\{c(x,v) + d_v(y)\}$$

costo dal vicino v alla destinazione y

costo da x al vicino v

min tra tutti i vicini v di x

BELLMAN-FORD: ESEMPIO



Se u conosce che

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

allora:

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

Il nodo che ottiene il minimo è il next hop nel cammino minimo, che verrà usato nella forwarding table

ALGORITMO DISTANCE VECTOR

- Sia $D_x(y)$ = stima del minor costo da x a y
  - x conserva il distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- Il nodo x conosce il costo del link a ciascun vicino v: $c(x,v)$
- Di tanto in tanto, ogni nodo invia il proprio distance vector ai vicini
- Il nodo x quando riceve un nuovo distance vector dai vicini, aggiorna il proprio distance vector usando l'equazione di Bellman-Ford
- Se il distance vector di x cambia, allora x invia il nuovo distance vector ai suoi vicini

CONFRONTO DI ALGORITMI LS E DV

message complexity

- ***LS:*** con n nodi, E links, inviati O(nE) msgs
- ***DV:*** scambio solo tra i vicini
  - tempo di convergenza varia

Velocità di convergenza

- ***LS:*** $O(n^2)$ con O(nE) msgs
  - Potrebbe avere oscillazioni
- ***DV:*** Variabile
  - Potrebbe creare routing loops
  - count-to-infinity problem

robustness: cosa succede se il router non funziona bene?

LS:

- nodo può pubblicizzare costi di *link* scorretti
- ogni nodo computa soltanto la *propria* tabella di inoltro

DV:

- il nodo può pubblicizzare costi di *cammini* scorretti
- La tabella di ogni nodo è usata da altri
  - errori si propagano attraverso la rete