# Reti

(già "Reti di Calcolatori")

## Livello Rete
## I protocolli di Routing: RIP, OSPF, BGP

Renato Lo Cigno

http://disi.unitn.it/locigno/index.php/teaching-duties/computer-networks

# Acknowledgement


- *Credits*
  - *Part of the material is based on slides provided by the following authors*
    - *Jim Kurose, Keith Ross, "Computer Networking: A Top Down Approach," 4th edition, Addison-Wesley, July 2007*
    - *Douglas Comer, "Computer Networks and Internets," 5th edition, Prentice Hall*
    - *Behrouz A. Forouzan, Sophia Chung Fegan, "TCP/IP Protocol Suite," McGraw-Hill*, January 2005
- La traduzione, se presente, è in generale opera (e responsabilità) del docente

# Contenuto e temi

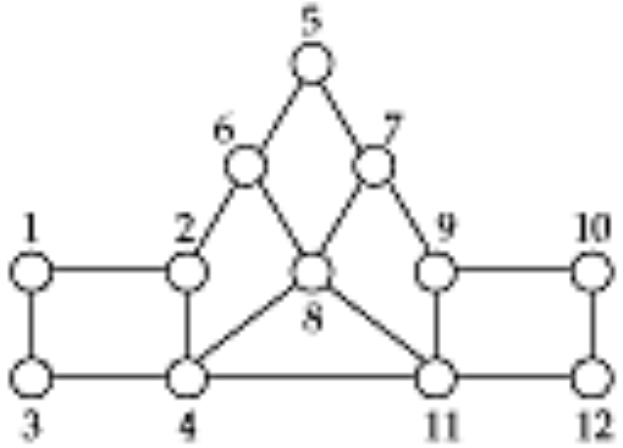
- Spazio di indirizzamento
- Indirizzi IP e loro uso
- Consegna dei pacchetti
- Configurazione dei PC e delle reti
- **Instradamento e Routing**

# Routing: What is it?

- Process of finding a path from a source to every destination in the network
- Suppose you want to connect to Antarctica from your desktop
  - what route should you take?
  - does a shorter route exist?
  - what if a link along the route goes down?
  - what if you're on a mobile wireless link?
- Routing deals with these types of issues

UNIVERSITY OF TRENTO
Department of Information Engineering and Computer Science

- A routing protocol sets up a routing table in routers
  - internal table that says, for each destination, which is the next output to take
- A node makes a local choice depending on global topology: this is the fundamental problem




ROUTING TABLE AT 1

| Destination | Next hop | Destination | Next hop |
|---|---|---|---|
| 1 | — | 7 | 2 |
| 2 | 2 | 8 | 2 |
| 3 | 3 | 9 | 2 |
| 4 | 3 | 10 | 2 |
| 5 | 2 | 11 | 3 |
| 6 | 2 | 12 | 3 |

# Key problem

- How to make correct local decisions?
  - each router must know something about global state
- Global state
  - inherently large
  - dynamic
  - hard to collect
- A routing protocol must intelligently summarize relevant information

# Requirements

- Minimize routing table space
  - fast to look up
  - less to exchange
- Minimize number and frequency of control messages
- Robustness: avoid
  - black holes
  - **loops**
  - **oscillations**
- **Use optimal path**

# Different degrees of freedom

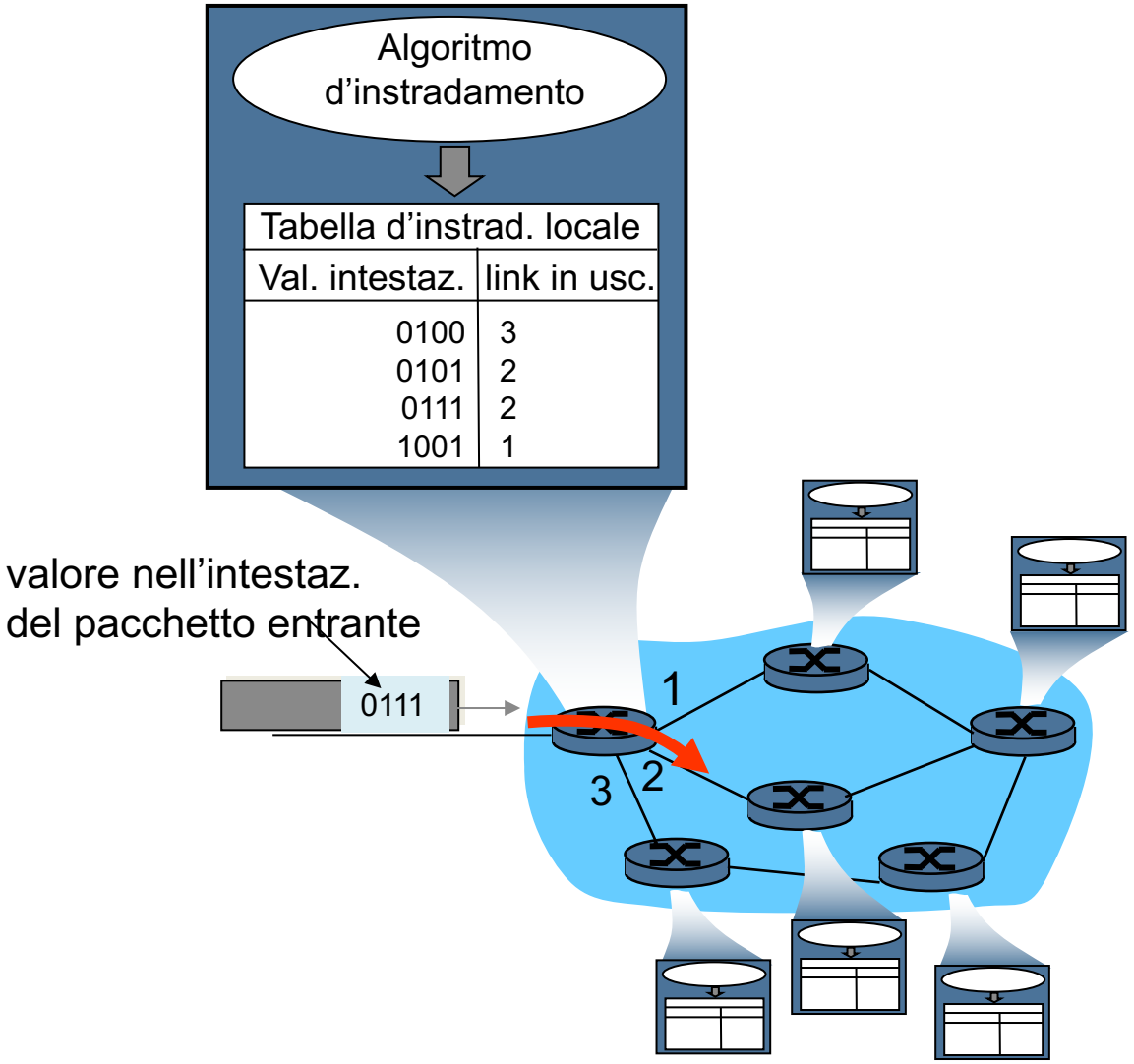


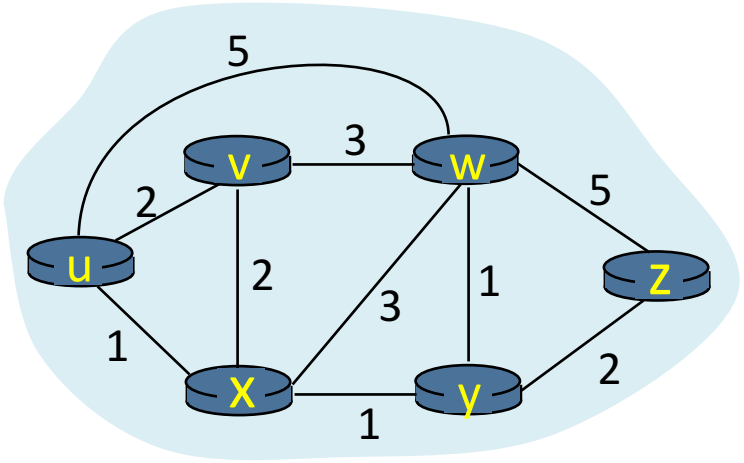
- Centralized vs. distributed routing
  - centralized is simpler, but prone to failure and congestion
- Global vs local information exchange
  - convey global information is expensive
- Static vs dynamic
  - static may work at the edge, not in the core
- Stochastic vs. deterministic
  - stochastic spreads load, avoiding oscillations, but misorders
- Single vs. multiple path
  - primary and alternative paths (compare with stochastic)
- State-dependent vs. state-independent
  - do routes depend on current network state (e.g. delay)

# Dynamic Routing And Routers



- To ensure that all routers maintain information about how to reach each possible destination
  - each router uses a route propagation protocol
    - to exchange information with other routers
  - when it learns about changes in routes
    - updates the local routing table
- Because routers exchange information periodically
  - the local routing table is updated continuously


Algoritmo
d'instradamento
Tabella d'instrad. locale
Val. intestaz. | link in usc.
0100 | 3
0101 | 2
0111 | 2
1001 | 1
valore nell'intestaz.
del pacchetto entrante
0111
1
2
3




Grafo: G = (N,E)

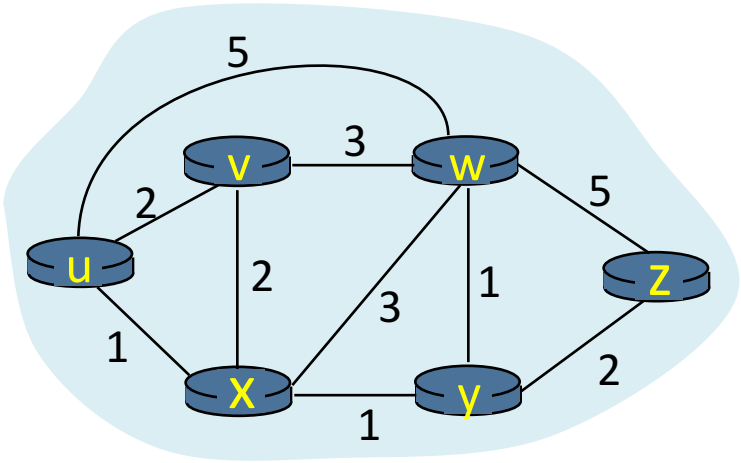
N = insieme di nodi (router) = { u, v, w, x, y, z }

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

**N.B.: Il grafo è un'astrazione utile anche in altri contesti di rete**

**Esempio: P2P, dove N è un insieme di peer ed E è un insieme di collegamenti TCP**

# Grafo di una rete: costi




- c(x,x') = costo del collegamento (x,x')

  - es., c(w,z) = 5

- il costo di un cammino è semplicemente la somma di tutti i costi degli archi lungo il cammino

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: Qual è il cammino a costo minimo tra u e z ?

Algoritmo d'instradamento: determina il cammino a costo minimo
Protocollo d'instradamento: procura all'algoritmo i dati necessari al calcolo e diffonde i risultati

# Algoritmi e protocolli Distance Vector

# Monotonia e coerenza dei cammini minimi

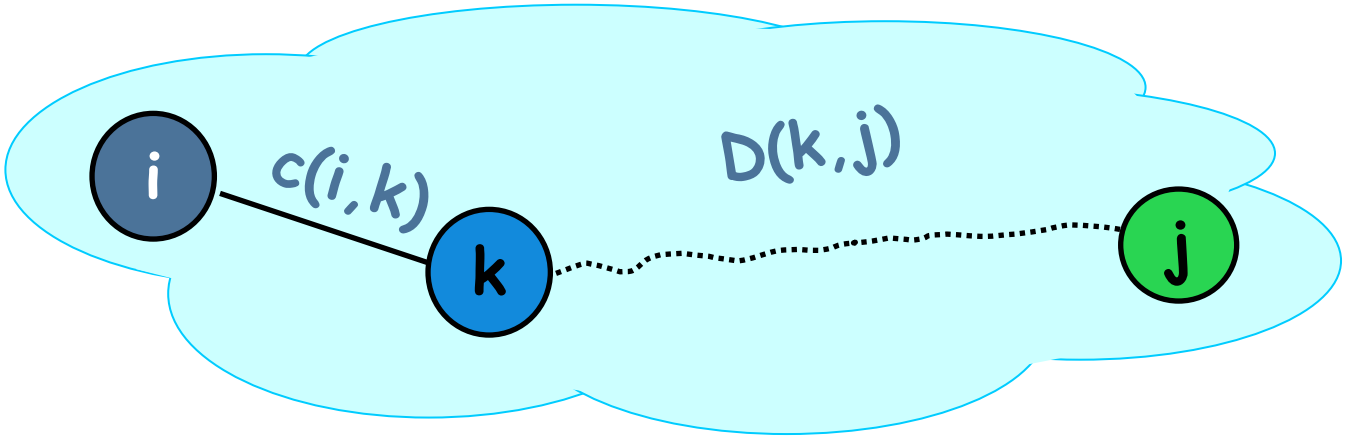
Define

c(i,k) >=0 : cost from i to k (direct connection)

D(i,j) : cost of least-cost path from i to j

➔ The subset of a shortest path is also the shortest path between the two intermediate nodes

- Then, if the shortest path from node i to node j, with distance D(i,j), passes through neighbor k, with link cost c(i,k), we have:
  - D(i,j) = c(i,k) + D(k,j)



# Distance Vector (DV) algorithm



- Initial distance values (iteration 1):
  - D(i,i) = 0
  - D(i,k) = c(i,k) if k is a neighbor (k is one-hop away)
  - D(i,j) = INFINITY for all other non-neighbors j
- The set of values D(i,*) is a *distance vector* at node i
- The algorithm maintains a next-hop value (forwarding table) for every destination j, initialized as:
  - next-hop(i) = i;
  - next-hop(k) = k if k is a neighbor, and
  - next-hop(j) = UNKNOWN if j is a non-neighbor.

# Distance Vector (DV) algorithm



- After every iteration each node i exchanges its distance vectors D(i,*) with all its immediate neighbors.

For any neighbor k

```
if c(i,k) + D(k,j) <  D(i,j) then
  {
    D(i,j) = c(i,k) + D(k,j)
    next-hop(j) = k
  }
```

# In summary

Basic idea:

- From time-to-time, each node sends its own distance vector estimate to neighbors

Asynchronous

- When a node x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

  $D(x,y) \leftarrow \min_v\{c(x,v) + D(v,y)\}$ for each node $y \in N$

- Under minor, natural conditions, the estimate D(x,y) converges to the actual least cost

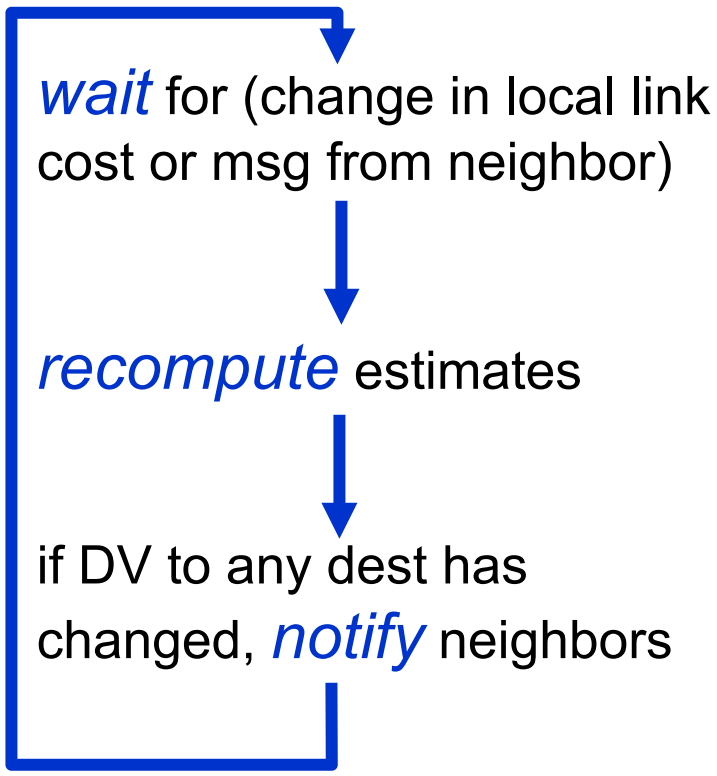
# In summary

- Iterative, asynchronous:

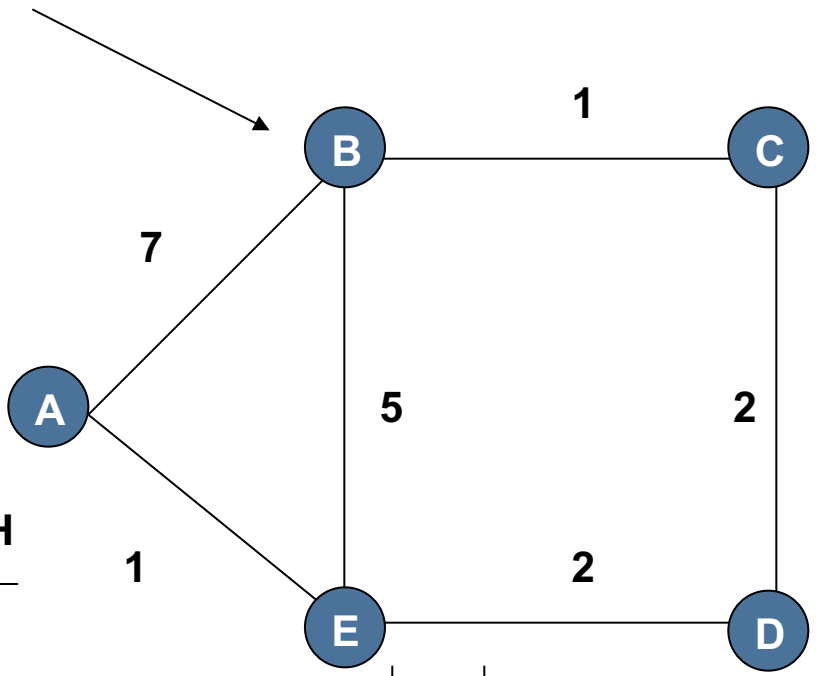
  each local iteration caused by:
  - local link cost change
  - DV update message from neighbor
- Distributed:

  each node notifies neighbors only when its DV changes
  - neighbors then notify their neighbors if necessary

Each node:



# Distance Vector: example (initialization)




| B | dist | NH |
|---|---|---|
| →A | 7 | A |
| →B | 0 | - |
| →C | 1 | C |
| →D | - | - |
| →E | 5 | E |

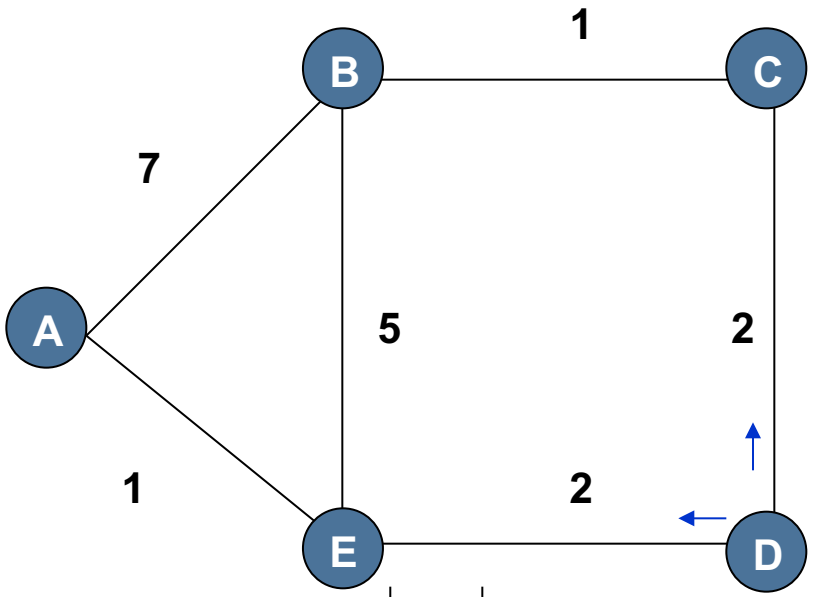
| C | dist | NH |
|---|---|---|
| →A | - | - |
| →B | 1 | B |
| →C | 0 | - |
| →D | 2 | D |
| →E | - | - |

| A | dist | NH |
|---|---|---|
| →A | 0 | - |
| →B | 7 | B |
| →C | - | - |
| →D | - | - |
| →E | 1 | E |

| E | dist | NH |
|---|---|---|
| →A | 1 | A |
| →B | 5 | B |
| →C | - | - |
| →D | 2 | D |
| →E | 0 | - |

| D | dist | NH |
|---|---|---|
| →A | - | - |
| →B | - | - |
| →C | 2 | C |
| →D | 0 | - |
| →E | 2 | E |

# Distance Vector: example (running)

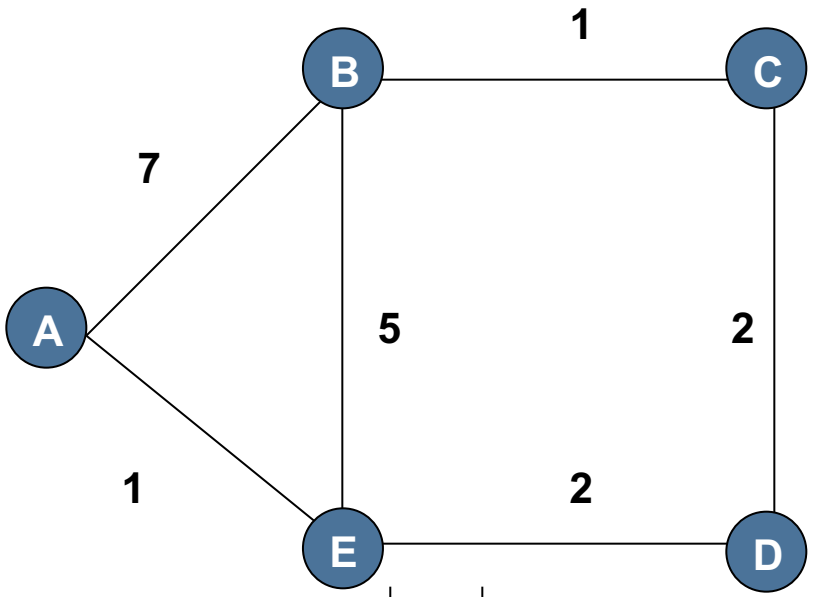



| C | dist | NH |
|---|---|---|
| →A | - | - |
| →B | 1 | B |
| →C | 0 | - |
| →D | 2 | D |
| →E | - | - |

| E | dist | NH |
|---|---|---|
| →A | 1 | A |
| →B | 5 | B |
| →C | - | - |
| →D | 2 | D |
| →E | 0 | - |

| D | dist | NH |
|---|---|---|
| →A | - | - |
| →B | - | - |
| →C | 2 | C |
| →D | 0 | - |
| →E | 2 | E |

# Distance Vector: example (running)



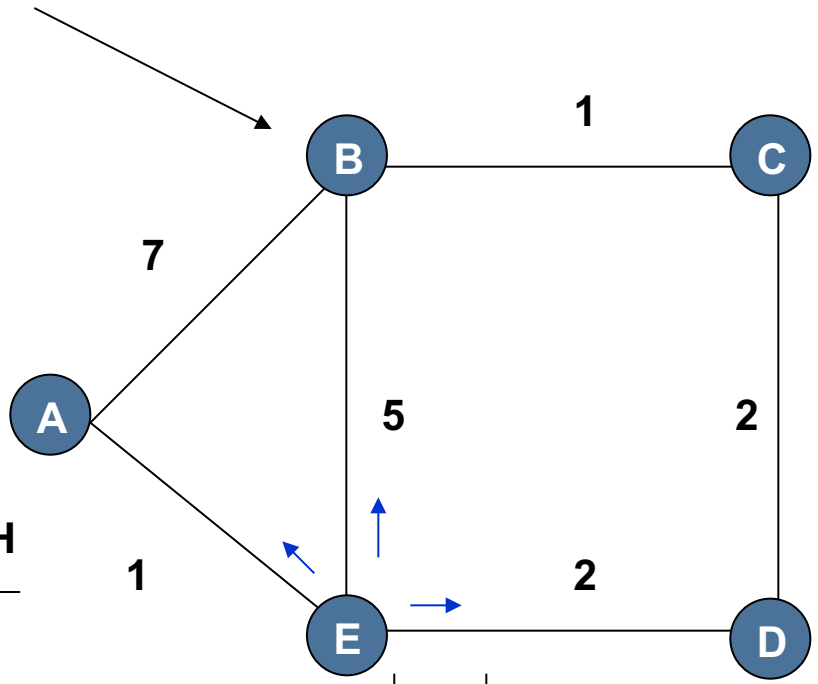

| C | dist | thr. |
|---|---|---|
| →A | - | - |
| →B | 1 | B |
| →C | 0 | - |
| →D | 2 | D |
| →E | 4 | D |

| E | dist | NH |
|---|---|---|
| →A | 1 | A |
| →B | 5 | B |
| →C | 4 | D |
| →D | 2 | D |
| →E | 0 | - |

| D | dist | NH |
|---|---|---|
| →A | - | - |
| →B | - | - |
| →C | 2 | C |
| →D | 0 | - |
| →E | 2 | E |

# Distance Vector: example (running)

| B | dist | NH |
|---|---|---|
| →A | 7 | A |
| →B | 0 | - |
| →C | 1 | C |
| →D | - | - |
| →E | 5 | E |



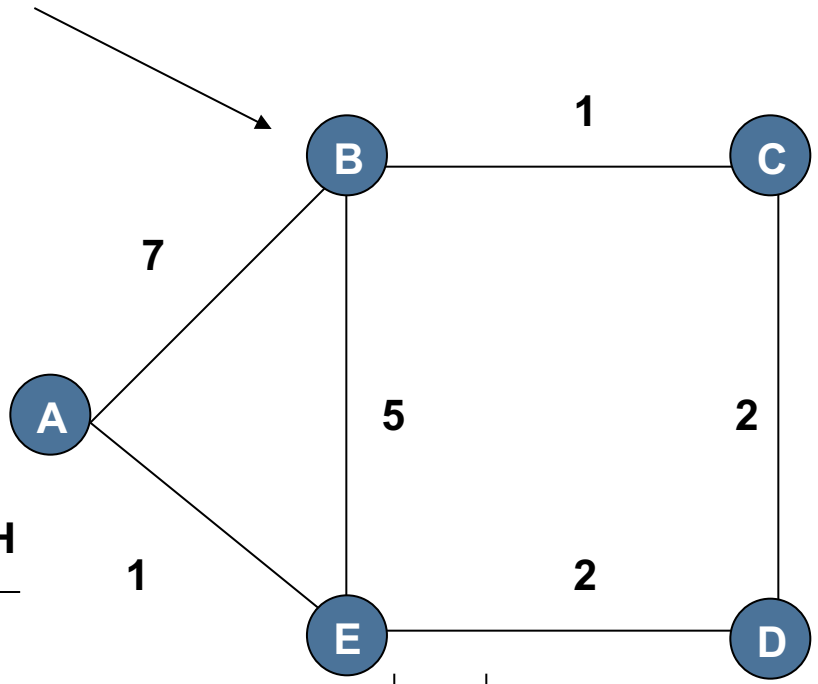

| A | dist | NH |
|---|---|---|
| →A | 0 | - |
| →B | 7 | B |
| →C | - | - |
| →D | - | - |
| →E | 1 | E |

| E | dist | NH |
|---|---|---|
| →A | 1 | A |
| →B | 5 | B |
| →C | 4 | D |
| →D | 2 | D |
| →E | 0 | - |

| D | dist | NH |
|---|---|---|
| →A | - | - |
| →B | - | - |
| →C | 2 | C |
| →D | 0 | - |
| →E | 2 | E |

# Distance Vector: example (running)

| B | dist | NH |
|---|---|---|
| →A | 6 | E |
| →B | 0 | - |
| →C | 1 | C |
| →D | 7 | E |
| →E | 5 | E |

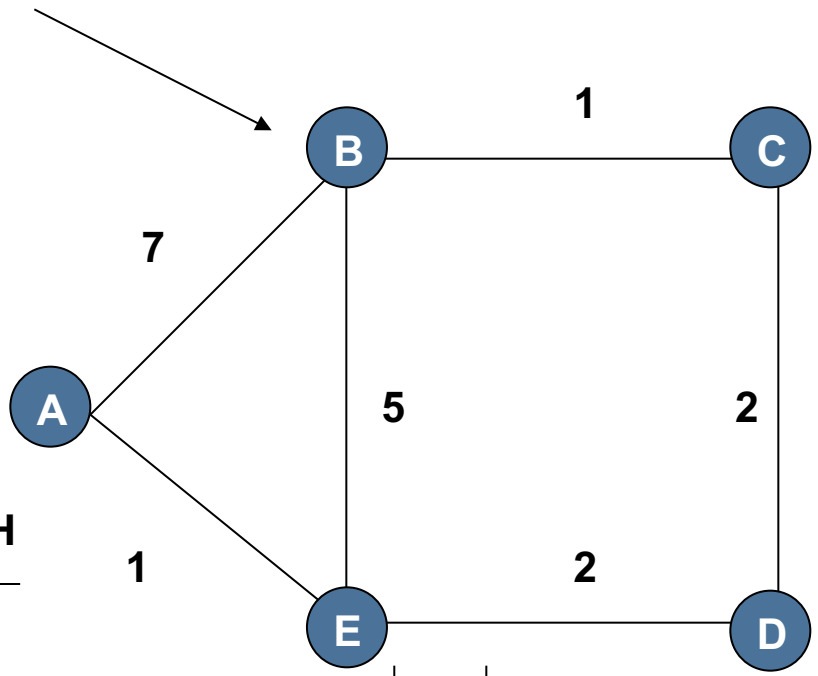



| A | dist | NH |
|---|---|---|
| →A | 0 | - |
| →B | 6 | E |
| →C | 5 | E |
| →D | 3 | E |
| →E | 1 | E |

| E | dist | NH |
|---|---|---|
| →A | 1 | A |
| →B | 5 | B |
| →C | 4 | D |
| →D | 2 | D |
| →E | 0 | - |

| D | dist | NH |
|---|---|---|
| →A | 3 | E |
| →B | 7 | E |
| →C | 2 | C |
| →D | 0 | - |
| →E | 2 | E |

# Distance Vector: example (final point)




| B | dist | NH |
|---|---|---|
| →A | 6 | E |
| →B | 0 | - |
| →C | 1 | C |
| →D | 3 | C |
| →E | 5 | E |

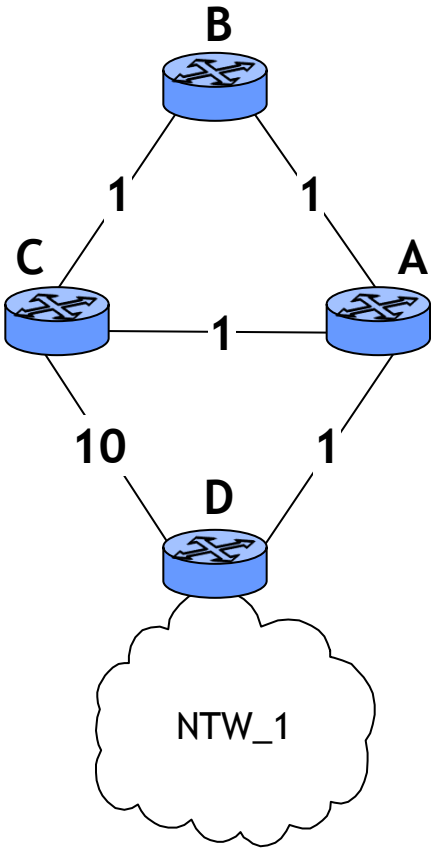
| C | dist | NH |
|---|---|---|
| →A | 5 | D |
| →B | 1 | B |
| →C | 0 | - |
| →D | 2 | D |
| →E | 4 | D |

| A | dist | NH |
|---|---|---|
| →A | 0 | - |
| →B | 6 | E |
| →C | 5 | E |
| →D | 3 | E |
| →E | 1 | E |

| E | dist | NH |
|---|---|---|
| →A | 1 | A |
| →B | 5 | B |
| →C | 4 | D |
| →D | 2 | D |
| →E | 0 | - |

| D | dist | NH |
|---|---|---|
| →A | 3 | E |
| →B | 3 | C |
| →C | 2 | C |
| →D | 0 | - |
| →E | 2 | E |

# Problem: "counting to infinity"




| Router A | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | D | 2 |

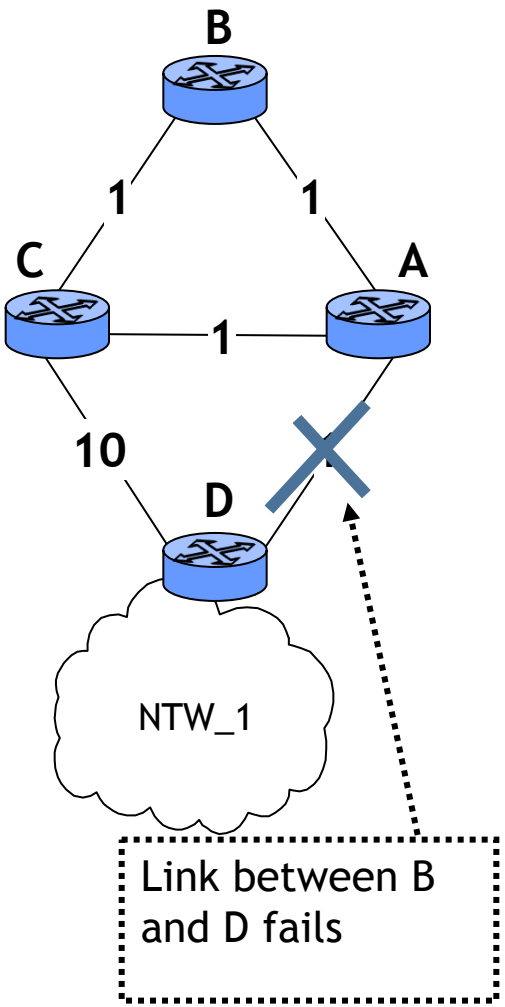
| Router B | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | A | 3 |

| Router C | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | A | 3 |

| Router D | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | dir | 1 |

- Consider the entries in each routing table for network NTW_1
- Router D is directly connected to NTW_1

# Problem: "counting to infinity"




time

| Router A | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | Unr. | - |

| Router B | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | A | 3 |

| Router C | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | A | 3 |

| Router D | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | dir | 1 |

| Router A | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | C | 4 |

| Router B | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | C | 4 |

| Router C | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | B | 4 |

| Router D | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | dir | 1 |

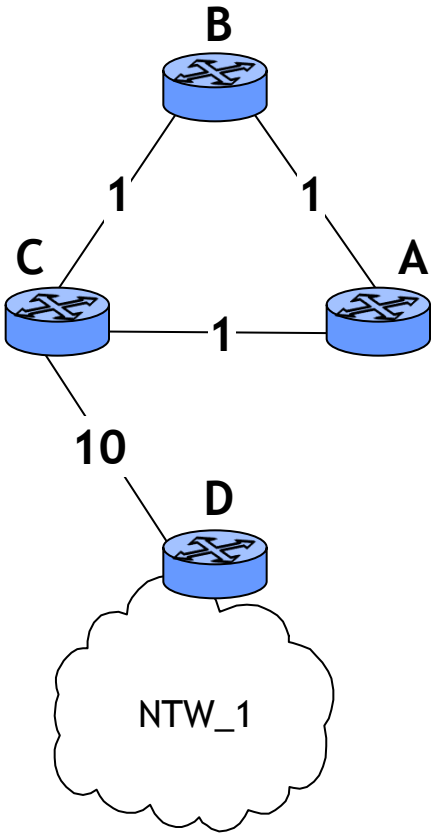
| Router A | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | C | 5 |

| Router B | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | C | 5 |

| Router C | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | B | 5 |

| Router D | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | dir | 1 |

# Problem: "counting to infinity"




time

| Router A | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | C | 11 |

| Router A | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | C | 12 |

| Router B | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | C | 11 |

| Router B | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | C | 12 |

| Router C | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | B | 11 |

| Router C | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | D | 11 |

| Router D | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | dir | 1 |

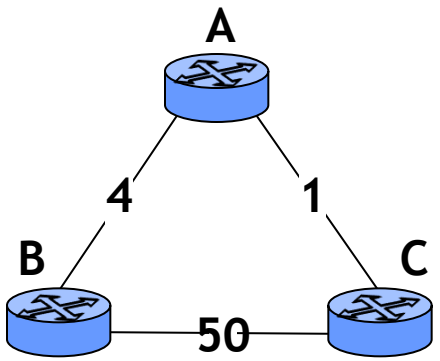
| Router D | | |
|---|---|---|
| Dest | Next | Metric |
| NTW_1 | dir | 1 |

# Solutions to "counting to infinity"



- Maximum number of hops bounded to 15
  - this limits the convergence time
- Split Horizon
  - simple
    - each node *omits* routes learned from one neighbor in update sent to that neighbor
  - with poisoned reverse
    - each node *include* routes learned from one neighbor in update sent to that neighbor, setting their metrics to infinity
      - drawback: routing message size greater than simple Split Horizon

- If link cost changes:
  - good news travels fast
    - good = cost decreases
  - bad news travels slow
    - bad = cost increases




- Exercise
  - try to apply the algorithm in the simple scenario depicted above when
    - the cost of the link A → B changes from 4 to 1
    - the cost of the link A → B changes from 4 to 60

# RIP at a glance

- **R**outing **I**nformation **P**rotocol
- A simple intradomain protocol
- Straightforward implementation of Distance Vector Routing…
  - Distributed version of Bellman-Ford (DBF)

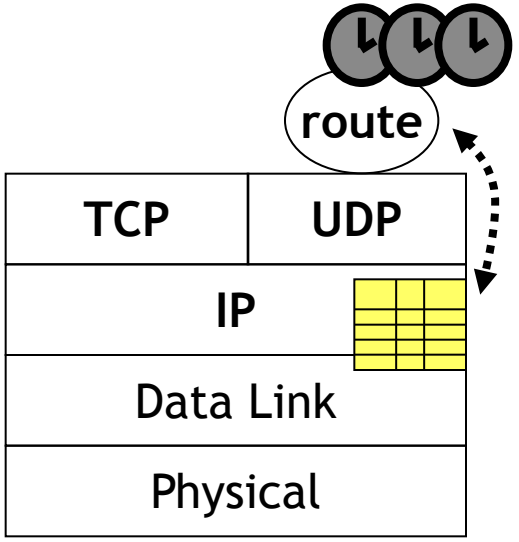
  …with well known issues
  - slow convergence
  - works with limited network size
- Strengths
  - simple to implement
  - simple management
  - widespread use

# RIP at a glance

- Metric based on hop count
  - maximum hop count is 15, with “16” equal to “infinity”
    - imposed to limit the convergence time
  - the network administrator can also assign values higher than 1 to a single hop
- Each router advertises its distance vector every 30 seconds (or whenever its routing table changes) to all of its neighbors
  - RIP uses UDP, port 520, Multicast Group 224.0.0.9
- Changes are propagated across network
- Routes are timeout (set to 16) after 3 minutes if they are not updated

# RIP procedures: introduction

- RIP routing tables are managed by application-level process
  - e.g., *routed* on UNIX machines
- Advertisements are sent in UDP packets (port 520)
- RIP maintains 3 different timers to support its operations
  - Periodic update timer (25-30 sec)
    - used to sent out update messages
  - Invalid timer (180 sec)
    - If update for a particular entry is not received for 180 sec, route is invalidated
  - Garbage collection timer (120 sec)
    - An invalid route in marked, not immediately deleted
    - For next 120 s. the router advertises this route with distance infinity

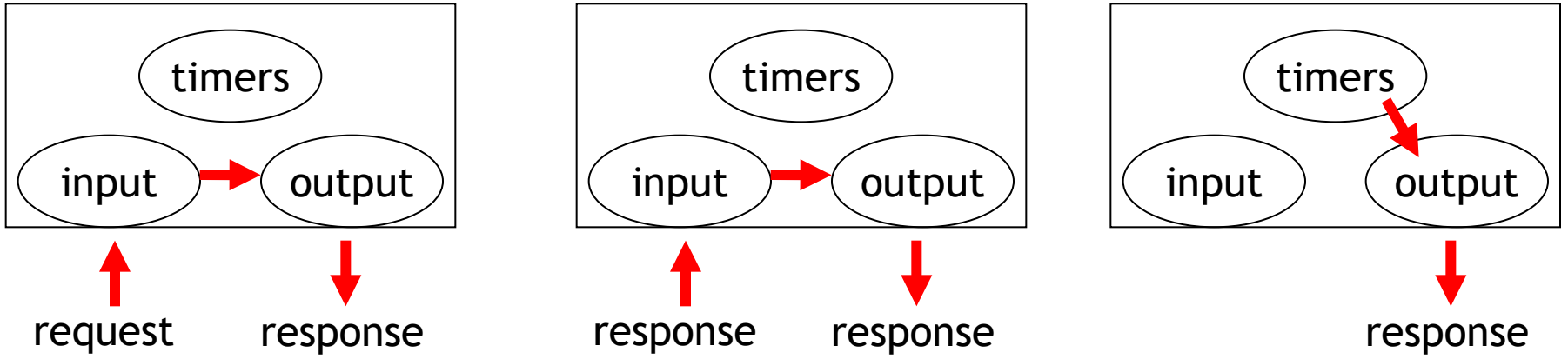


# RIP procedures: input processing

- Request Messages
  - generated by routers that just come up
  - action: the router responds directly to the requestor's address and port
    - request is processed entry by entry
- Response Messages
  - routers that perform regular updates, triggered updates or respond to a specific query
  - action: the router updates its routing table
    - in case of new route or changed routes, the router starts a triggered update procedure

# RIP procedures: output processing

- Output are generated
  - when the router comes up in the network
  - if required by the input processing procedures
  - by regular routing update
- Action: the router generates the messages according to the commands received
  - the messages contain entries from the routing table



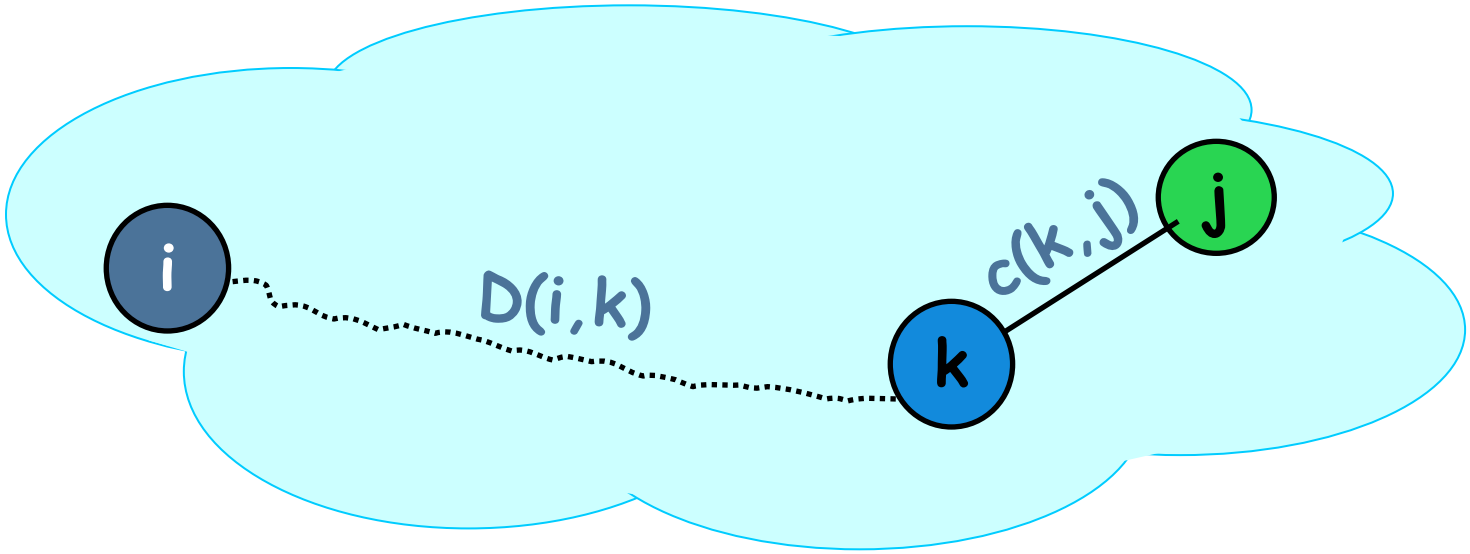
# ALGORITMI E PROTOCOLLI LINK-STATE

Un approccio diverso e “centralizzato” per trovare i cammini minimi

# Link State (LS) Approach



- The link state (Dijkstra) approach is iterative, but it pivots around destinations j, and their predecessors k = p(j)
  - Observe that an alternative version of the consistency condition holds for this case: D(i,j) = D(i,k) + c(k,j)




- Each node i collects all link states c(*,*) first and runs the complete Dijkstra algorithm locally

# Link State (LS) Approach…

- Link states are distributed to all nodes, which build a graph G(N,E) describing the entire network
- Each node applies Dijkstra: minimum shortest paths from itself to all nodes
- At each iteration, the algorithm finds a new destination j and a shortest path to it
- After m iterations the algorithm has explored paths, which are m hops or smaller from node i
- The Dijkstra algorithm at node i maintains two sets:
  - set N that contains nodes to which the shortest paths have been found
  - set M that contains all other nodes
  - For all nodes k, two values are maintained:
    - D(i,k): current value of distance from i to k
    - p(k): the predecessor node to k on the shortest known path from i

# Dijkstra: Initialization

- Initialization:
  - D(i,i) = 0 and p(i) = i;
  - D(i,k) = c(i,k) and p(k) = i if k is a neighbor of i
  - D(i,k) = INFINITY and p(k) = UNKNOWN if k is not neighbor of i
  - Set N = { i }, and next-hop (i) = i
  - Set M = { j | j is not i}

- Initially set N has only the node i and set M has the rest of the nodes
- At the end of the algorithm, set N contains all the nodes, and set M is empty

# Dijkstra: Iteration

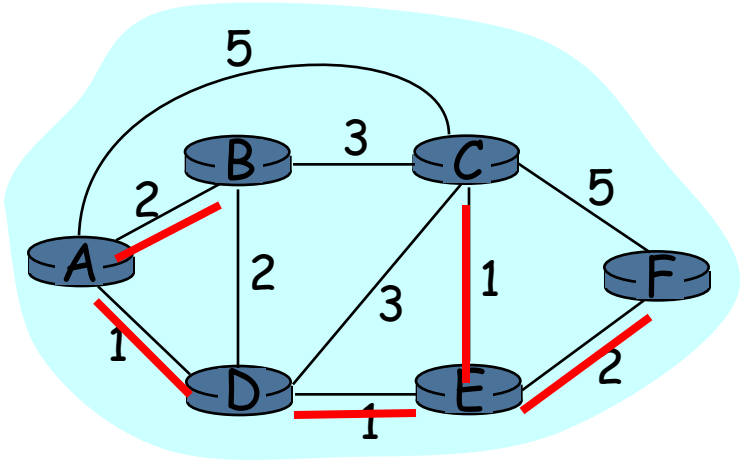
- In each iteration, a new node j is moved from set M into the set N.
  - Node j has the minimum distance among all current nodes in M
    $D(i,j) = \min_{(k \text{ in } M)} D(i,k)$.
  - If multiple nodes have the same minimum distance, any one of them is chosen as j
  - Next-hop(j) = the neighbor of i on the shortest path, or j itself
  - The distance values of any neighbor k of j in set M is recomputed as:
    If $D(i,k) > (D(i,j) + c(j,k))$

    $$D(i,k) = D(i,j) + c(j,k); \quad p(k) = j$$

- This operation is called “relaxing” the edges of node j

# Dijkstra's algorithm: *example*

| Step | set N | D(B),p(B) | D(C),p(C) | D(D),p(D) | D(E),p(E) | D(F),p(F) |
|---|---|---|---|---|---|---|
| 0 | A | 2,A | 5,A | 1,A | infinity | infinity |
| 1 | AD | 2,A | 4,D | | 2,D | infinity |
| 2 | ADE | 2,A | 3,E | | | 4,E |
| 3 | ADEB | | 3,E | | | 4,E |
| 4 | ADEBC | | | | | 4,E |
| 5 | ADEBCF | | | | | |




The shortest-paths spanning tree rooted at A is called an SPF-tree

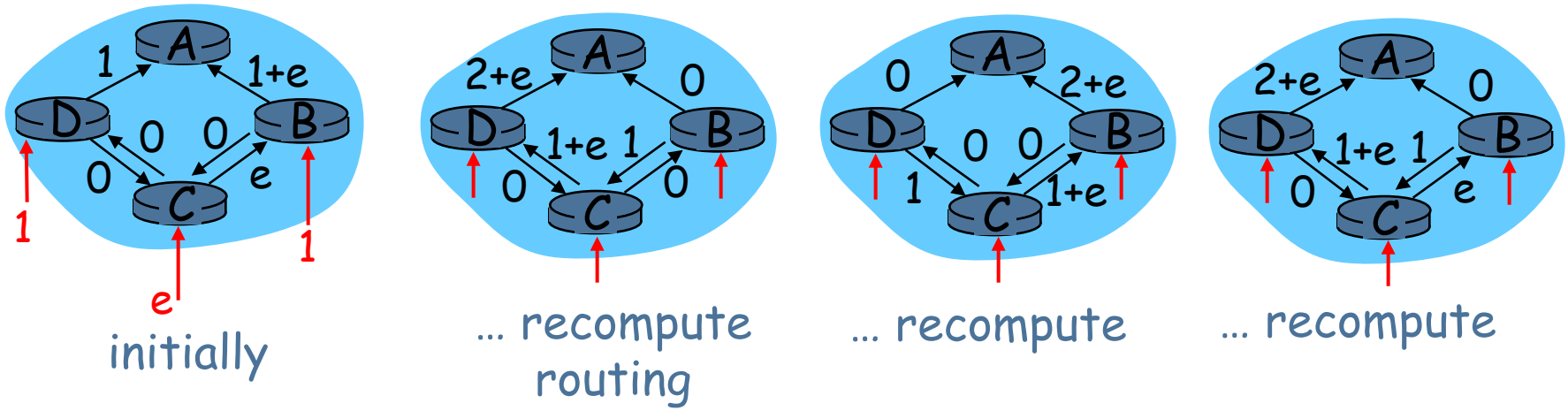
# Dijkstra's algorithm, discussion

Algorithm complexity: n nodes

- each iteration: need to check all nodes, w, not in N
- n(n+1)/2 comparisons: $O(n^2)$
- more efficient implementations possible: O(n log(n))

Oscillations possible:

- e.g., link cost = amount of carried traffic



# Open Shortest Path First (OSPF)

- In alternativa al protocollo RIP di tipo Distance Vector in Internet esiste il protocollo OSPF di tipo Link State
- I tre principali criteri di progettazione del protocollo OSPF sono:
  - distinzione tra host e router
  - reti broadcast
  - suddivisione delle reti di grandi dimensioni

- Hli host sono collocati nelle aree periferiche della rete a sottoreti locali connesse alla attraverso router (default gateway)
- Il modello link state prevede che il database *link state* includa una entry per ogni link tra host e router
- OSPF associa il link di accesso ad una *stub network*
  - una stub network è una sottorete terminale che non fornisce servizio di transito
  - il link di accesso viene identificato dall'indirizzo della sottorete

- Il protocollo OSPF utilizza 3 procedure, chiamati ancora `protocolli', per svolgere le proprie funzioni
  - Hello Protocol
  - Exchange Protocol
  - Flooding Protocol

# Messaggi OSPF (1)

- I messaggi OSPF sono trasportati direttamente all'interno dei pacchetti IP
  - non viene utilizzato il livello di trasporto
  - viene usato l'indirizzo multicast 224.0.0.5
- Tutti i messaggi OSPF condividono lo stesso header

| *Version #* | *Type* | *Packet length* |
|---|---|---|
| *Router ID* | | |
| *Area ID* | | |
| *Checksum* | | *Auth Type* |
| *Authentication* | | |
| *Authentication* | | |

# Messaggi OSPF (2)

- Version # = 2
- Type: indica il tipo di messaggio
- Packet Length: numero di byte del messaggio
- Router ID: indirizzo IP del router di riferimento

| *Version #* | *Type* | *Packet length* |
|---|---|---|
| *Router ID* | | |
| *Area ID* | | |
| *Checksum* | | *Auth Type* |
| *Authentication* | | |
| *Authentication* | | |

# Messaggi OSPF (3)

- Area ID: identificativo dell'area
  - OSPF consente una divisione della rete in "aree" per ridurre la complessità del calcolo dei percorsi e per consentire un instradamento gerarchico.
- Auth Type: tipo di autenticazione
  - 0 no autenticazione, 1 autenticazione con passwd
- Authentication: password

| *Version #* | *Type* | *Packet length* |
|---|---|---|
| *Router ID* | | |
| *Area ID* | | |
| *Checksum* | | *Auth Type* |
| *Authentication* | | |
| *Authentication* | | |

# Il protocollo Hello

- Funzioni:
  - verificare l'operatività dei link
- Messaggi:
  - Hello

| *Common header (type = 1, hello)* | | |
|---|---|---|
| *Network mask* | | |
| *Hello interval* | *Options* | *Priority* |
| *Dead interval* | | |
| *Designated router* | | |
| *Backup Designated router* | | |
| *Neighbor* | | |

# Il protocollo Exchange



- Funzioni:
  - sincronizzazione dei database link state (bring up adjacencies) tra due router che hanno appena verificato l'operatività bidirezionale del link che li connette
  - protocollo client-server
  - messaggi:
    - Database Description Packets
    - Link State Request
    - Link State Update
  - N.B. il messaggio Link State Update viene distribuito in flooding

- Database Description

| *Common header (type = 2, db description)* | | | |
|---|---|---|---|
| *0* | *0* | *Options* | *0* |
| *DD sequence number* | | | |
| *Link State Type* | | | |
| *Link State ID* | | | |
| *Advertising router* | | | |
| *Link State Sequence Number* | | | |
| *Link State Checksum* | | *Link State Age* | |

# Il protocollo di Flooding

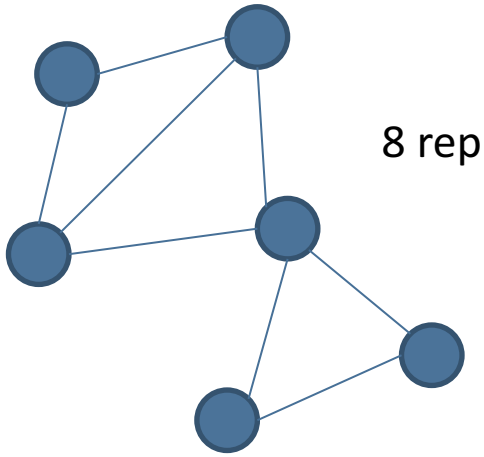


- Funzioni:
  - aggiornare il database link state dell'autonomous system a seguito del cambiamento di stato di un link
  - Garantisce la consegna di tutti I messaggi a tutti, a costo di parecchie repliche
- Messaggi:
  - Link State Update

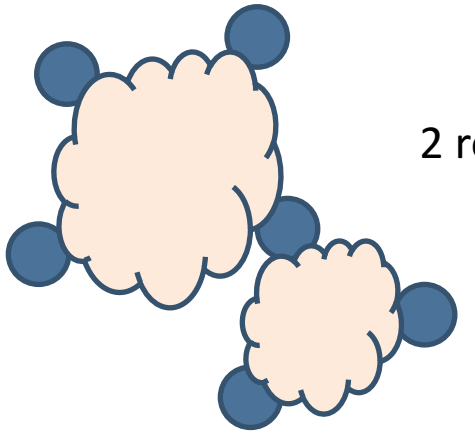
| *Common header (type = 4, link state update)* |
|---|
| *Number of link state advertisement* |
| *Link state advertisement #1* |
| *Link state advertisement #2* |

# Controlled Flooding

- Tutti i nodi inviano i pacchetti ricevuti da una interfaccia su tutte le altre
- In una rete con link punto-punto il numero di pacchetti è pari al numero di link
- In una rete con domini di broadcast, il numero di pacchetti è pari il numero di domini boradcast



8 repliche



2 repliche

# Summary

## Link State

- Topology information is flooded within the routing domain
- Best end-to-end paths are computed locally at each router
- Best end-to-end paths determine next-hops
- Based on minimizing some notion of distance
- Works only if policy is shared and uniform
- Examples: OSPF

## Distance Vector

- Each router knows little about network topology
- Only best next-hops are chosen by each router for each destination network.
- Best end-to-end paths result from composition of all next-hop choices
- Does not require any notion of distance
- Does not require uniform policies at all routers
- Examples: RIP

# Comparison of LS and DV algorithms

Message complexity

- LS: with n nodes, E links, O(nE) msgs sent
- DV: exchange between neighbors only
  - convergence time varies

Speed of Convergence

- LS: O(n2) algorithm requires O(nE) msgs
  - may have oscillations
- DV: convergence time varies
  - may be routing loops
  - count-to-infinity problem

Robustness: what happens if router malfunctions?

- LS:
  - node can advertise incorrect link cost
  - each node computes only its own table
- DV:
  - DV node can advertise incorrect path cost
  - each node's table used by others
    - error propagate thru network

# Instradamento e Topologia Globali in Internet

Come si effettua il trasferimento e l'instradamento complessivo in tutta la rete

Qualche considerazione sulle proprietà globali del sistema

Fin qui abbiamo visto la rete come una collezione di router interconnessi

- Ciascun router era indistinguibile dagli altri
- Visione omogenea della rete

*… nella pratica le cose non sono così semplici*

Scala: con milioni di destinazioni:

- Archiviare le informazioni d'instradamento su ciascuna sottorete richiederebbe un'enorme quantità di memoria
- Il traffico generato dagli aggiornamenti LS non lascerebbero banda per i pacchetti di dati!

Autonomia amministrativa:

- Internet = la rete delle reti
- Da un punto di vista ideale, ciascuno dovrebbe essere in grado di amministrare la propria rete nel modo desiderato, pur mantenendo la possibilità di connetterla alle reti esterne

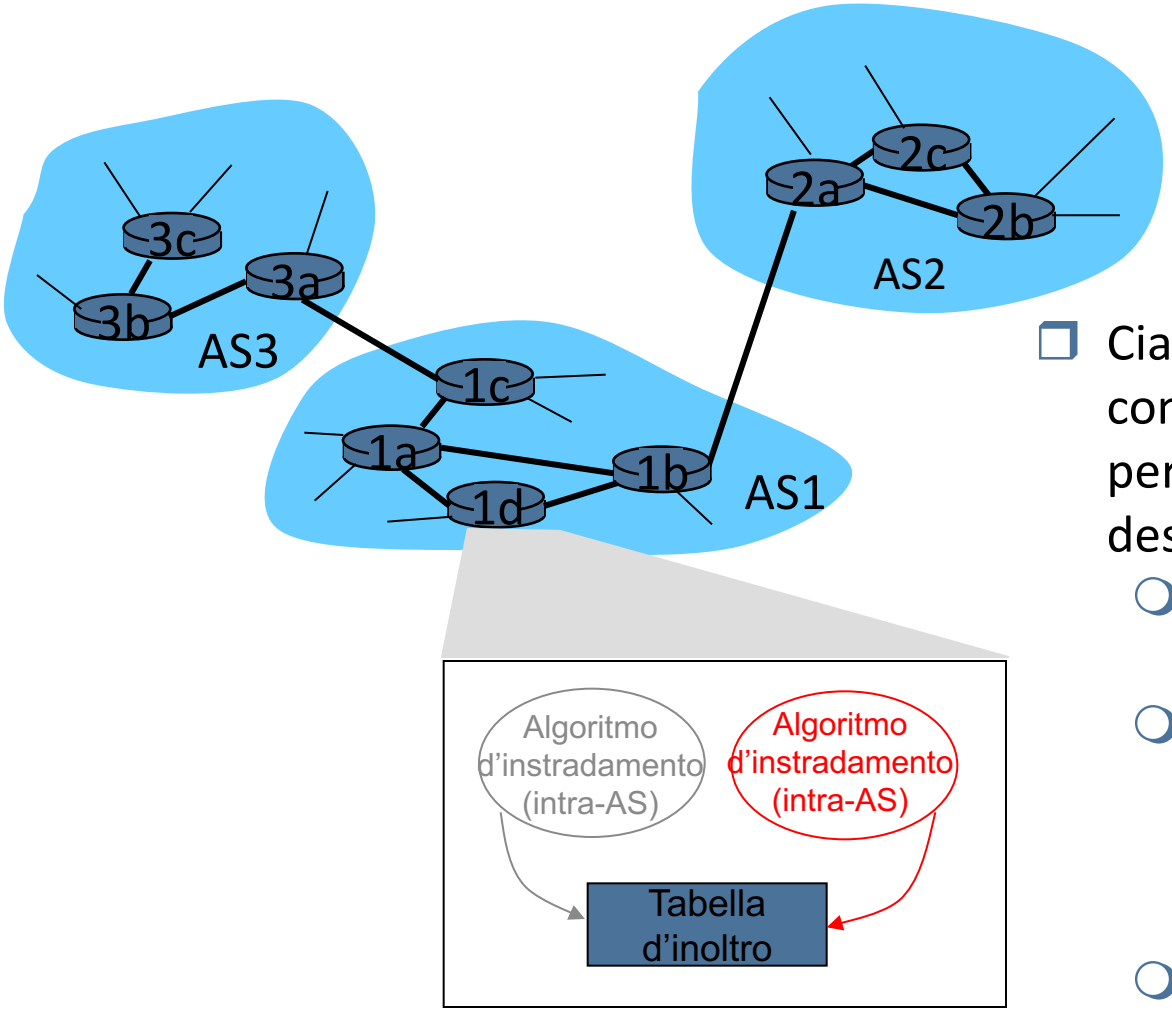
# Instradamento gerarchico

- Organizzazione di router in *sistemi autonomi (AS, autonomous system).*
- I router di un gruppo autonomo eseguono lo stesso algoritmo d'instradamento.
  - Protocollo d'instradamento interno al sistema autonomo (intra-AS).
  - I router appartenenti a differenti AS possono eseguire protocolli d'instradamento intra-AS diversi

Border Routers

- Hanno il compito aggiuntivo d'inoltrare pacchetti a destinazioni esterne.

# Sistemi autonomi interconnessi



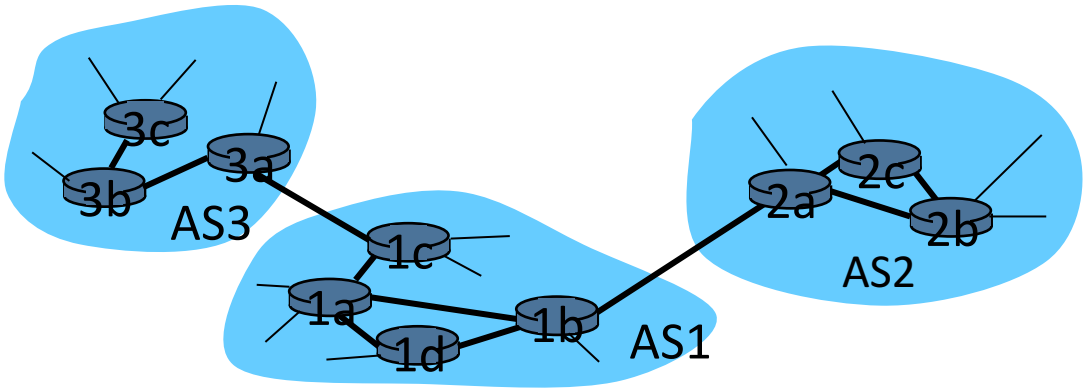

- Ciascun sistema autonomo sa come inoltrare pacchetti lungo il percorso ottimo verso qualsiasi destinazione interna al gruppo
  - I sistemi AS2 e AS3 hanno tre router ciascuno
  - I protocolli d'instradamento dei tre sistemi autonomi non sono necessariamente gli stessi
  - I router 1b, 1c, 2a e 3a sono gateway

UNIVERSITY OF TRENTO
Department of Information Engineering and Computer Science

- Supponiamo che un router in AS1 riceva un datagramma la cui destinazione ricade al di fuori di AS1
  - Il router dovrebbe inoltrare il pacchetto verso uno dei due gateway. Ma quale??

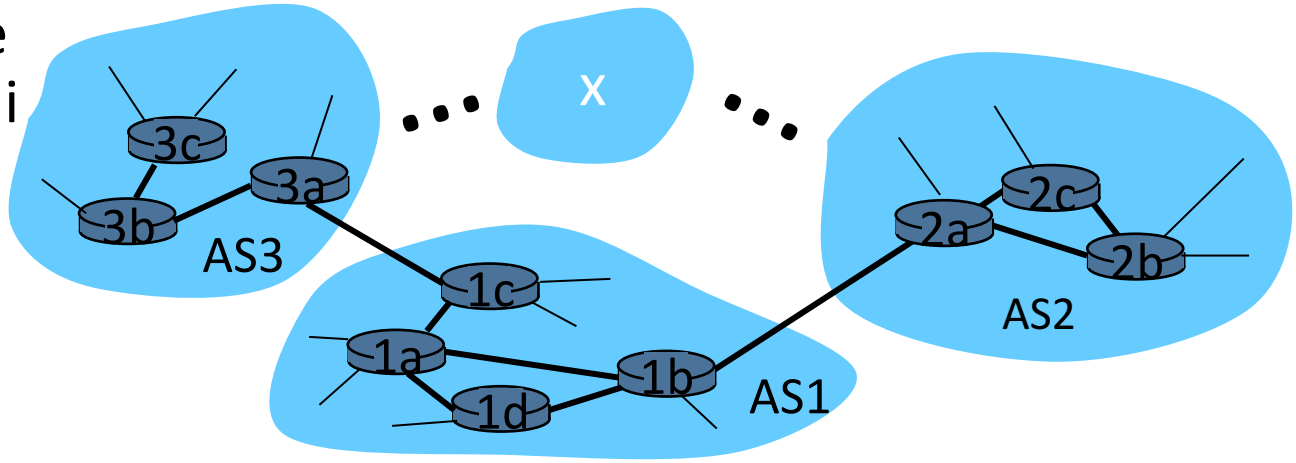
AS1 deve:

1. Sapere quali destinazioni sono raggiungibili attraverso AS2 e quali attraverso AS3
2. Informare tutti i router all'interno del sistema in modo che ciascuno possa configurare la propria tabella d'inoltro per gestire destinazioni esterne



# Esempio: scegliere fra più AS

- Supponiamo inoltre che AS1 apprenda dal protocollo d'instradamento tra sistemi autonomi che la sottorete *x* è raggiungibile da AS2 *e* da AS3
- Al fine di configurare la propria tabella d'inoltro, il router 1D dovrebbe determinare a quale gateway, 1b o 1c, indirizzare i pacchetti destinati alla sottorete *x*
- Anche questo è un compito che spetta al protocollo d'instradamento inter-AS!
- La scelta in genere si basa su decisioni locali di tipo economico, commerciale, di politica (policy)

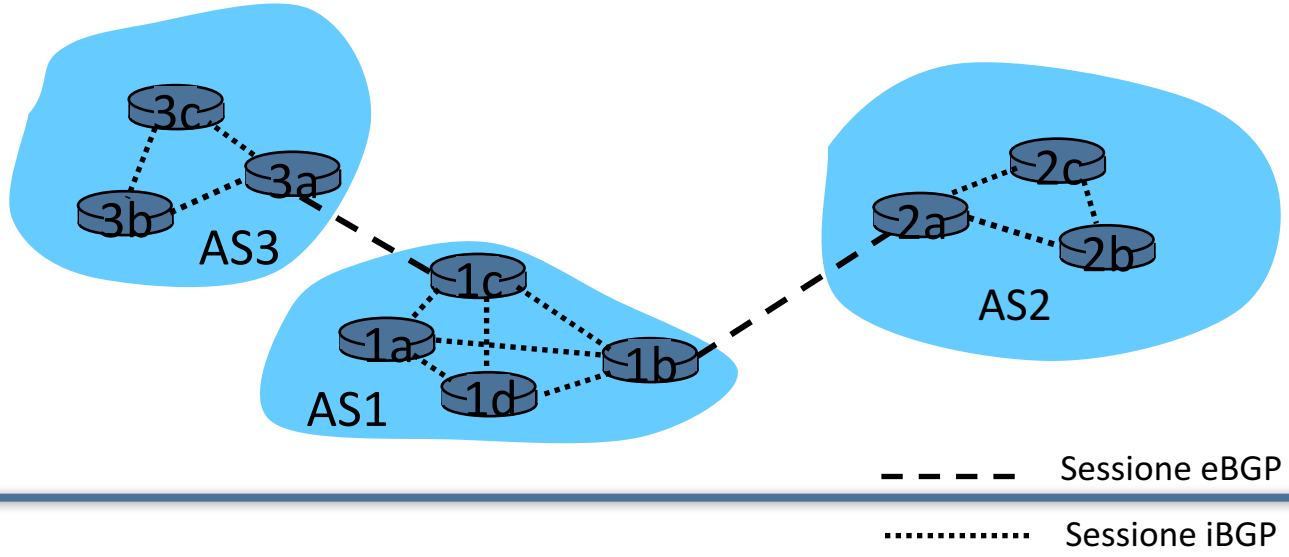


# Border Gateway Protocol (BGP)



- A differenza dei protocolli intra-AS esiste un solo protocollo inter-AS: BGP appunto
- Lo scopo del protocollo è distribuire informazioni di raggiungibilità di tutte le destinazioni
- BGP non cerca un percorso ottimo, lascia che i diversi AS prendano le loro decisioni in base al all'elenco di AS che bisogna attraversare per raggiungere una destinazione
- BGP si divide in due parti
- eBGP: (exterior), il protocollo di routing vero e proprio che distribuisce la raggiungibilità di tutte le destinazioni
- iBGP: (interior), tramite il quale un router che partecipa a eBGP comunica a tutti i nodi del suo AS dove instradare i pacchetti per le varie destinazioni

- BGP si appoggia a una maglia di connessioni TCP: due router connessi sono peer BGP, e la connessione è detta sessione BGP
- Quando AS2 annuncia un prefisso a AS1, AS2 sta *promettendo* che inoltrerà i datagrammi su un percorso verso il prefisso cui sono destinati.
  - AS2 può aggregare più prefissi nel suo annuncio
- E` sufficiente che un router per AS partecipi a eBGP



# Attributi del percorso e rotte BGP



- Quando un router annuncia un prefisso per una sessione BGP, include anche un certo numero di attributi BGP
  - prefisso + attributi = “rotta”
- Due dei più importanti attributi sono:
  - AS-PATH: elenca i sistemi autonomi attraverso i quali è passato l’annuncio del prefisso: es. AS67-AS17-AS21
  - NEXT-HOP: indirizzo del router (interfaccia0 da usare per accedere ad AS-PATH
- Quando un border router riceve un annuncio di rotta, utilizza le proprie politiche d’importazione per decidere se accettare o filtrare la rotta

# Selezione dei percorsi BGP

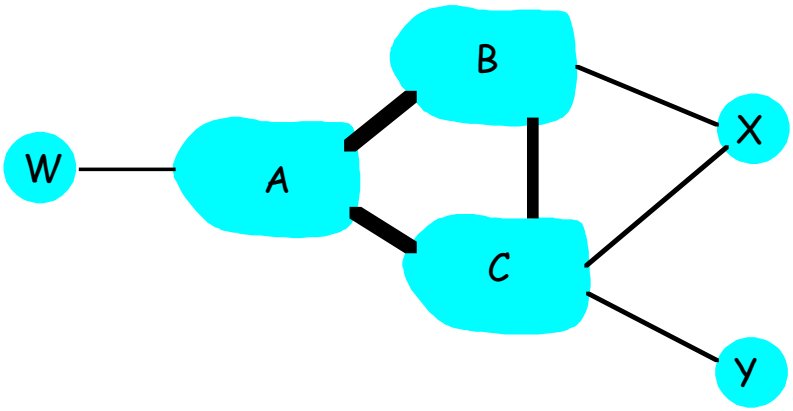
- Un router può ricavare più di una rotta verso un determinato prefisso, e deve quindi sceglierne una
- Regole di eliminazione:
  1. Alle rotte viene assegnato come attributo un valore di preferenza locale. Si selezionano quindi le rotte con i più alti valori di preferenza locale
  2. Si seleziona la rotta con valore AS-PATH più breve escludendo quelle che contengono AS con cui non ci sono accordi commerciali
  3. Si seleziona quella il cui router di NEXT-HOP è più vicino
  4. Se rimane ancora più di una rotta, il router fa una scelta casuale

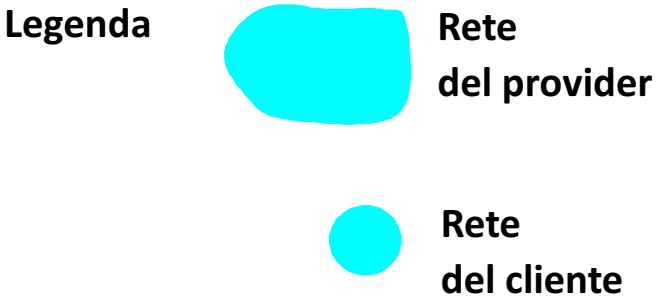
# Messaggi BGP



- I messaggi BGP vengono scambiati attraverso TCP.
- Messaggi BGP:
  - OPEN: apre la connessione TCP e autentica il mittente
  - UPDATE: annuncia il nuovo percorso (o cancella quello vecchio)
  - KEEPALIVE mantiene la connessione attiva in mancanza di UPDATE
  - NOTIFICATION: riporta gli errori del precedente messaggio; usato anche per chiudere il collegamento.

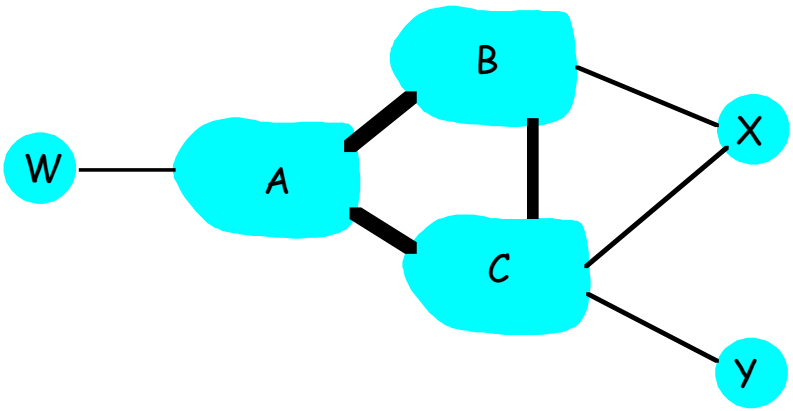
# Politiche d'instradamento BGP

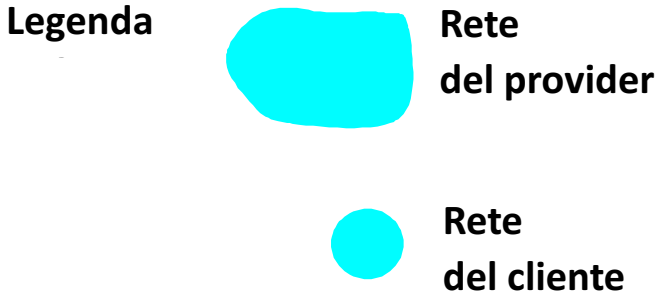



- A, B, C sono reti di provider di dorsale
- X, W, Y sono reti stub
- X è una rete stub a più domicili
  - X non vuole che il traffico da B a C le passi attraverso
  - ... e così X non annuncerà a B la rotta verso C

# Politiche d'instradamento BGP




- A annuncia a B il percorso AW.
- B annuncia a X il percorso BAW.
- B deve annunciare a C del percorso BAW?
  - Certo che no! B non ha nessun "interesse" nella rotta CBAW poiché né W né C sono clienti di B
  - B vuole costringere C ad instradare verso W attraverso A
  - B vuole instradare *solo* da/verso i suoi clienti!

UNIVERSITY OF TRENTO
Department of Information Engineering and Computer Science

Politiche:

- Inter-AS: il controllo amministrativo desidera avere il controllo su come il traffico viene instradato e su chi instrada attraverso le sue reti.
- Intra-AS: unico controllo amministrativo, e di conseguenza le questioni di politica hanno un ruolo molto meno importante nello scegliere le rotte interne al sistema

Scala:

- L'instradamento gerarchico fa "risparmiare" sulle tabelle d'instradamento, e riduce il traffico dovuto al loro aggiornamento

Prestazioni:

- Intra-AS: orientato alle prestazioni
- Inter-AS: le politiche possono prevalere sulle prestazioni

# Topologia di Internet



- Troppo complessa per poterla “disegnare”
  - Anche solo gli AS sono oltre 50.000, i prefissi di destinazione oltre 1 milione (a livello BGP, quindi aggregati)
- Caratteristiche “small world”
  - Pochi nodi con tantissime connessioni
  - Tantissimi nodi con poche connessioni e “stub”
  - Pochi “hop” (a livello AS) per arrivare ovunque
- Proprietà del grafo interessanti e complesse
  - esiste una intera “scienza” di analisi delle caratteristiche della topologia di Internet