Reti di Calcolatori
AA 2010/2011

UNIVERSITÀ DEGLI STUDI DI TRENTO

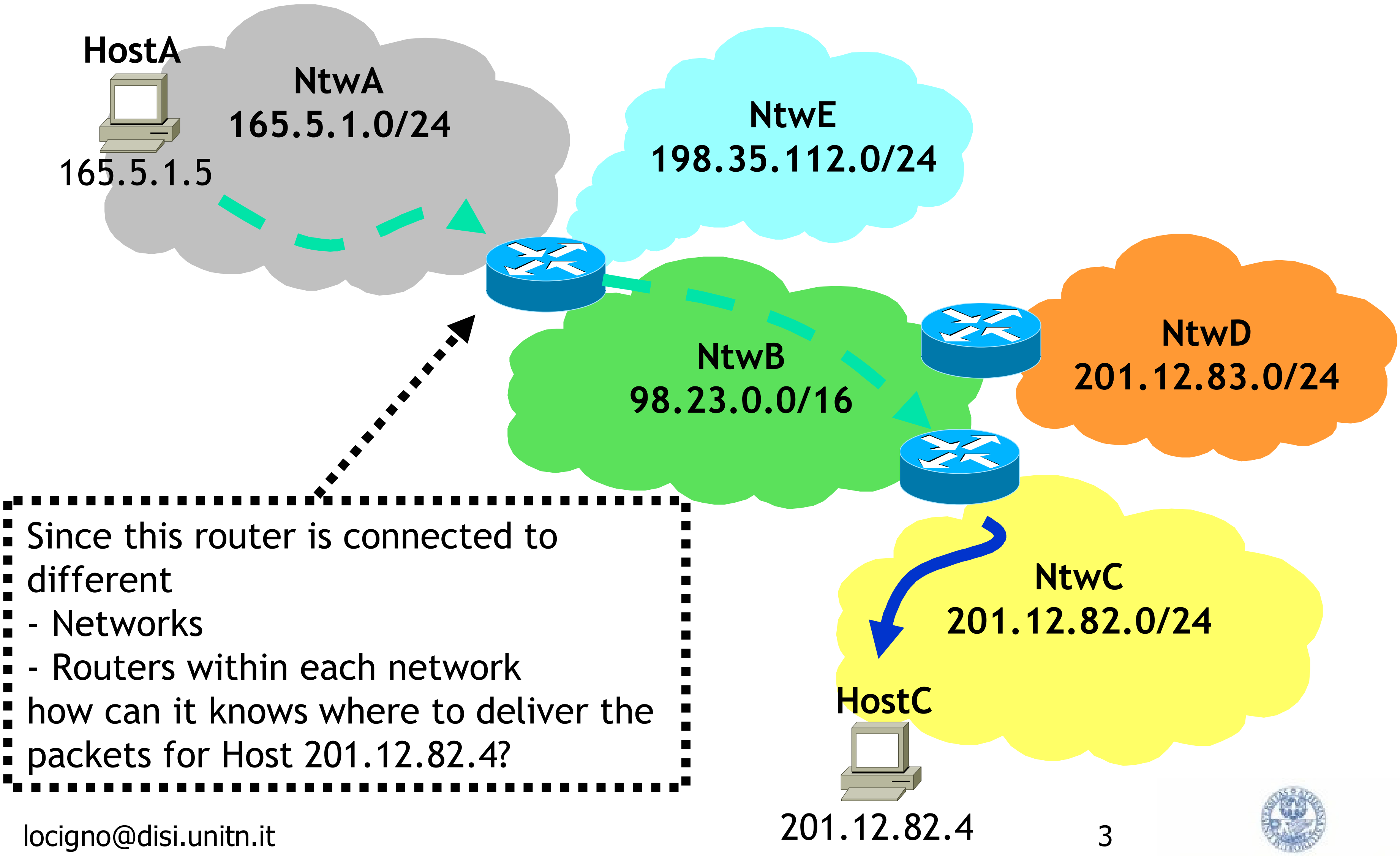
http://disi.unitn.it/locigno/index.php/teaching-duties/computer-networks

Internet Routing

Renato Lo Cigno

Copyright

Direct / Indirect Delivery



Since this router is connected to different
- Networks
- Routers within each network

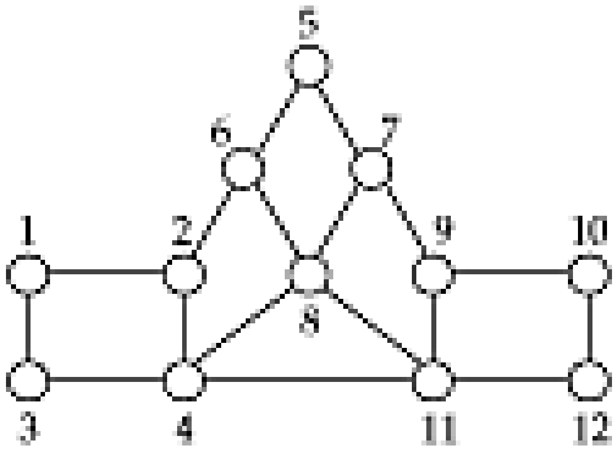
how can it knows where to deliver the packets for Host 201.12.82.4?

Routing: What is it?

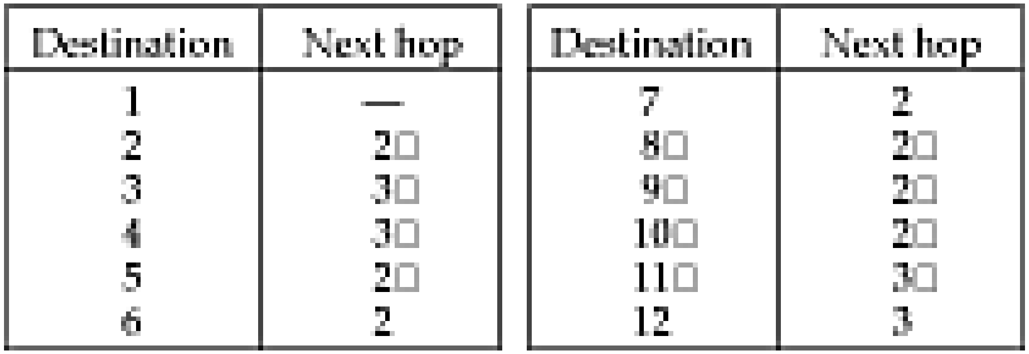
- Process of finding a path from a source to every destination in the network
- Suppose you want to connect to Antarctica from your desktop
  - what route should you take?
  - does a shorter route exist?
  - what if a link along the route goes down?
  - what if you’re on a mobile wireless link?
- Routing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

Basics

- 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

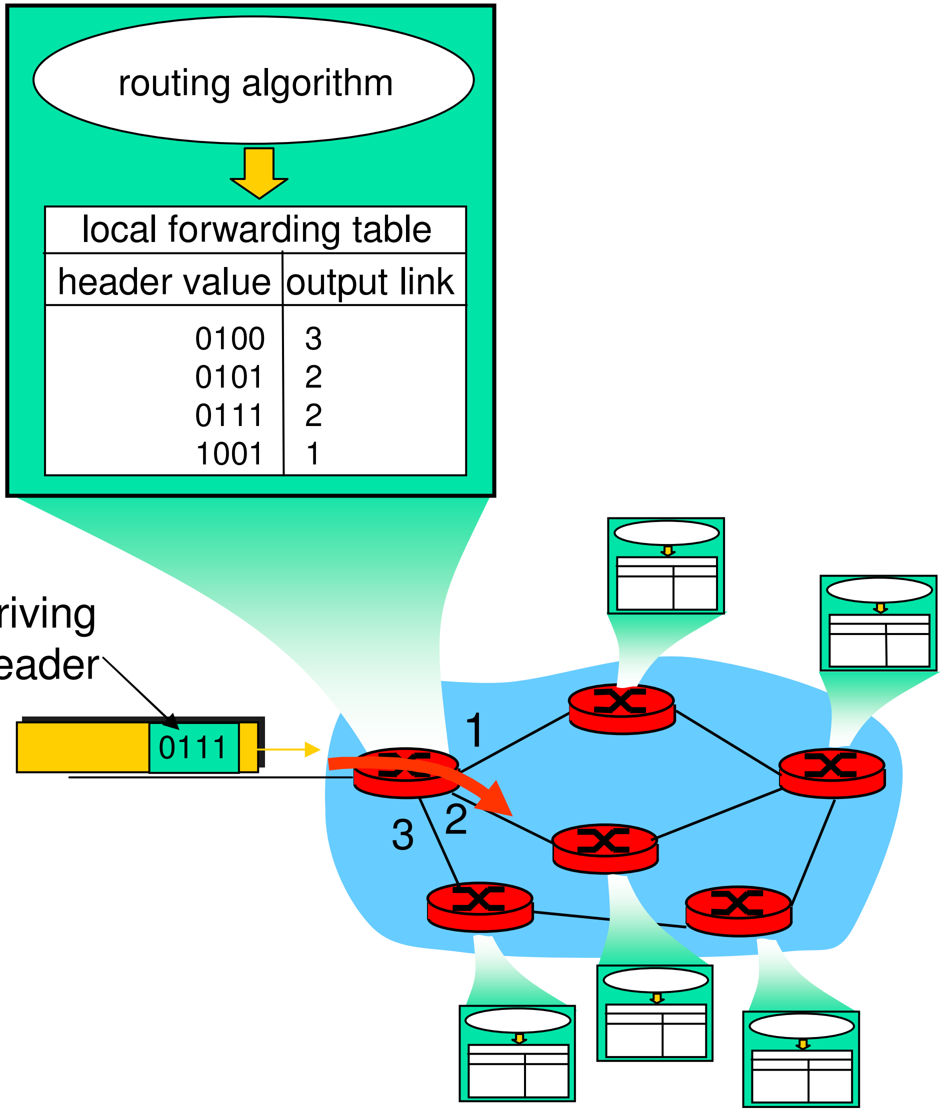
Different degrees of freedom

- 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

Interplay between routing, forwarding

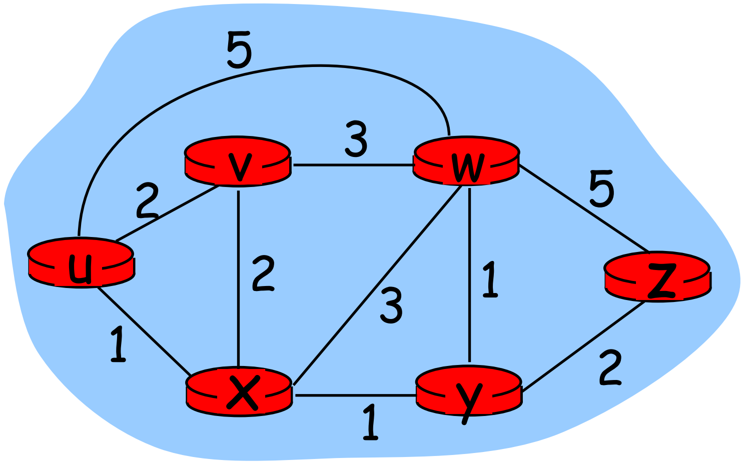


routing algorithm

local forwarding table	
header value	output link
0100	3
0101	2
0111	2
1001	1

value in arriving packet's header

Graph abstraction

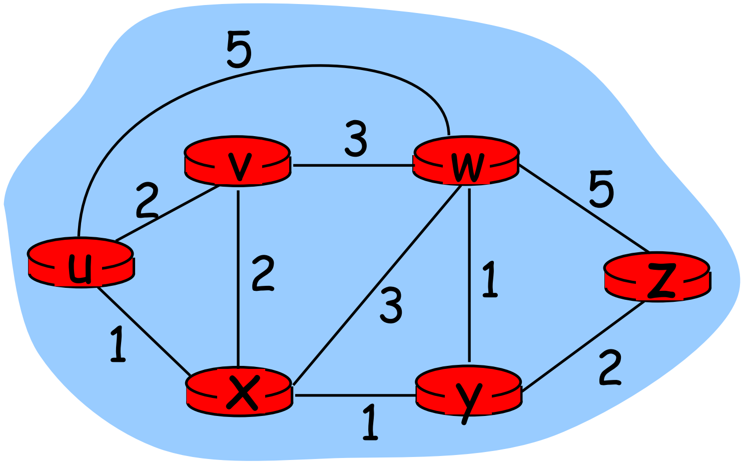


Graph: G = (N,E)

N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Graph abstraction: costs



- c(x,x') = cost of link (x,x')

  - e.g., c(w,z) = 5

- cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

Cost of path $(x_1, x_2, x_3,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Question: What's the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

Distance Vector Algorithms

Consistency criterion

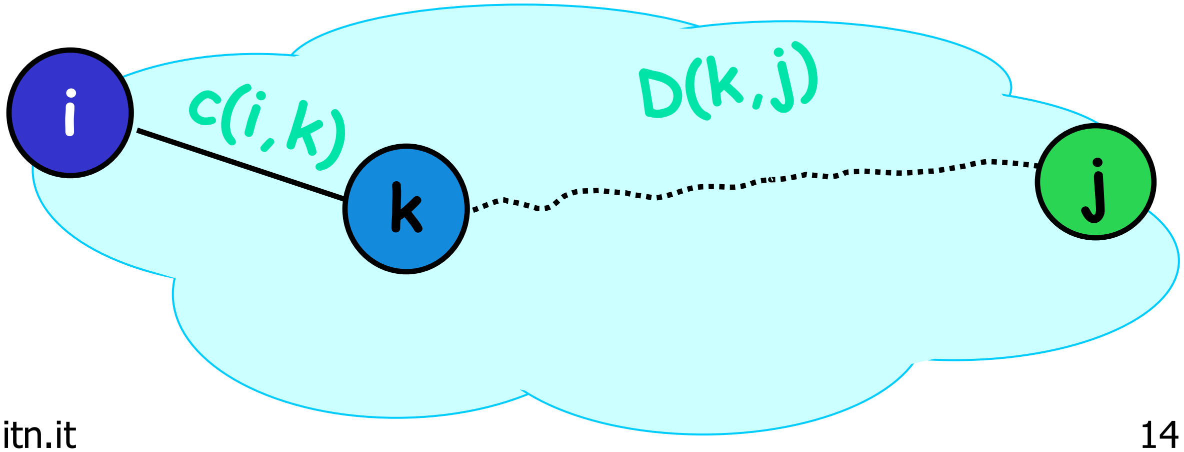
Define

c(i,k) := 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 (i.e. k is one-hop away); and
  - D(i,j) = INFINITY for all other non-neighbors j.
- Note that the set of values D(i,*) is a distance vector at node i.
- The algorithm also 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 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

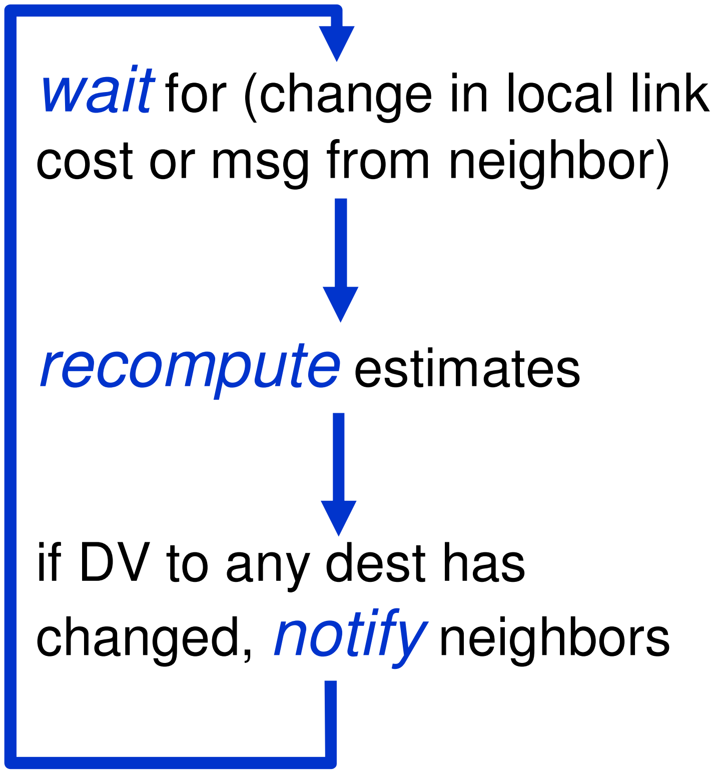
In summary

- 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:



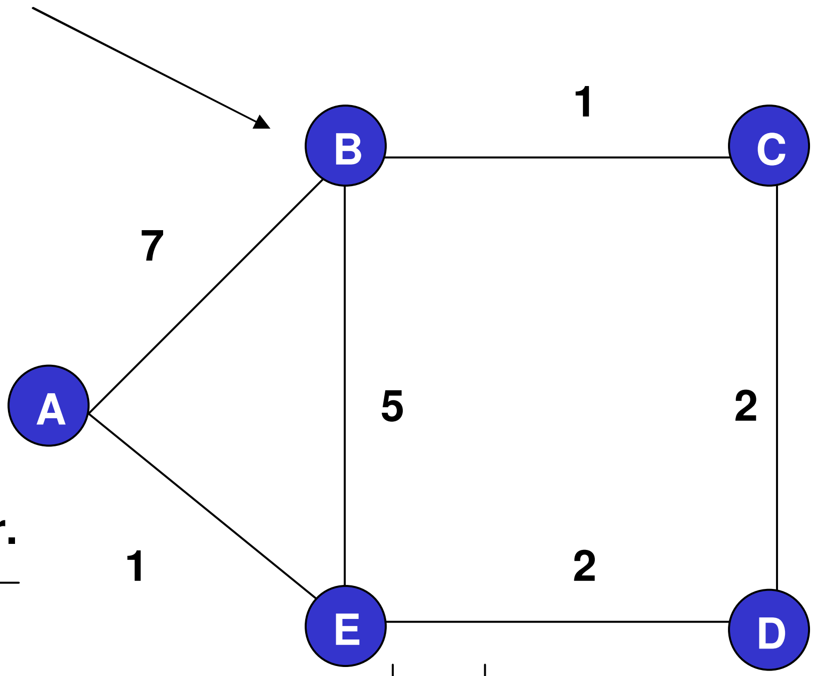
wait for (change in local link cost or msg from neighbor)

recompute estimates

if DV to any dest has changed, *notify* neighbors

Distance Vector: example (starting point)

B	dist	thr.
→A	7	A
→B	0	-
→C	1	C
→D	-	-
→E	5	E



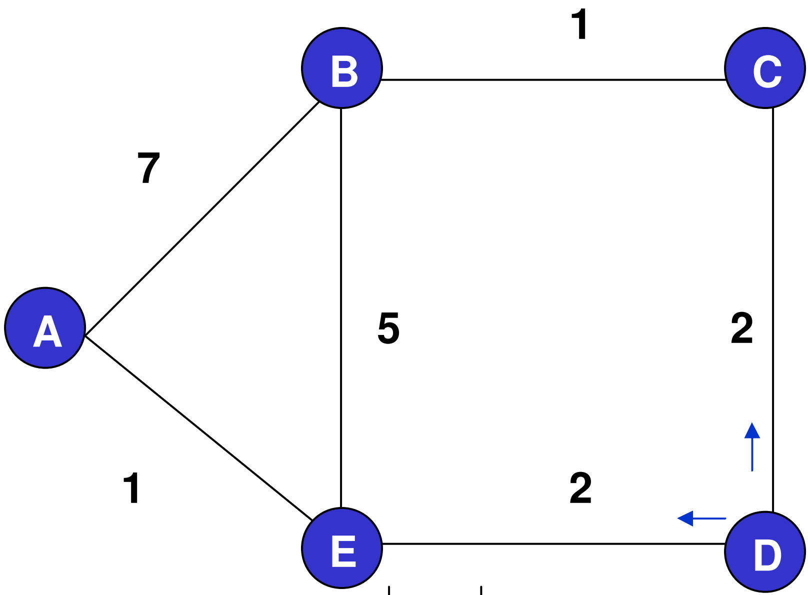
C	dist	thr.
→A	-	-
→B	1	B
→C	0	-
→D	2	D
→E	-	-

A	dist	thr.
→A	0	-
→B	7	B
→C	-	-
→D	-	-
→E	1	E

E	dist	thr.
→A	1	A
→B	5	B
→C	-	-
→D	2	D
→E	0	-

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

Distance Vector: example (running)

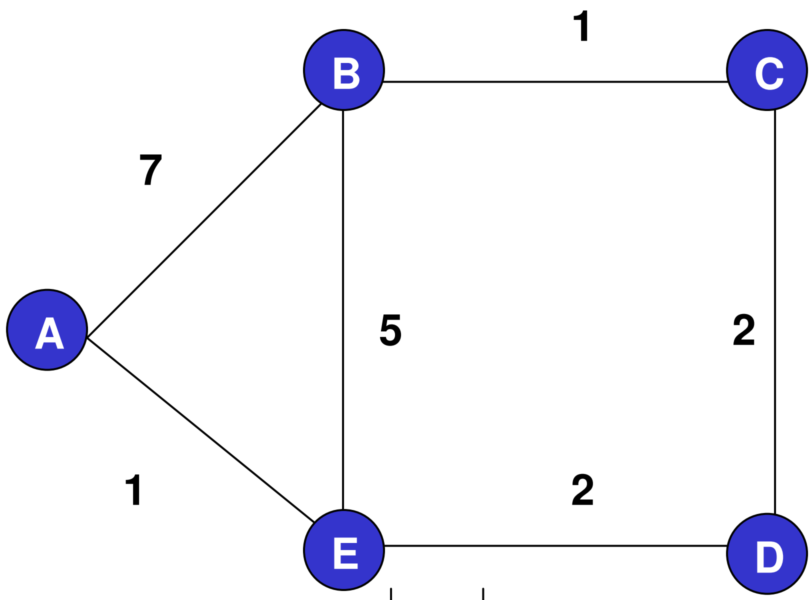


C	dist	thr.
→A	-	-
→B	1	B
→C	0	-
→D	2	D
→E	-	-

E	dist	thr.
→A	1	A
→B	5	B
→C	-	-
→D	2	D
→E	0	-

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

Distance Vector: example (running)



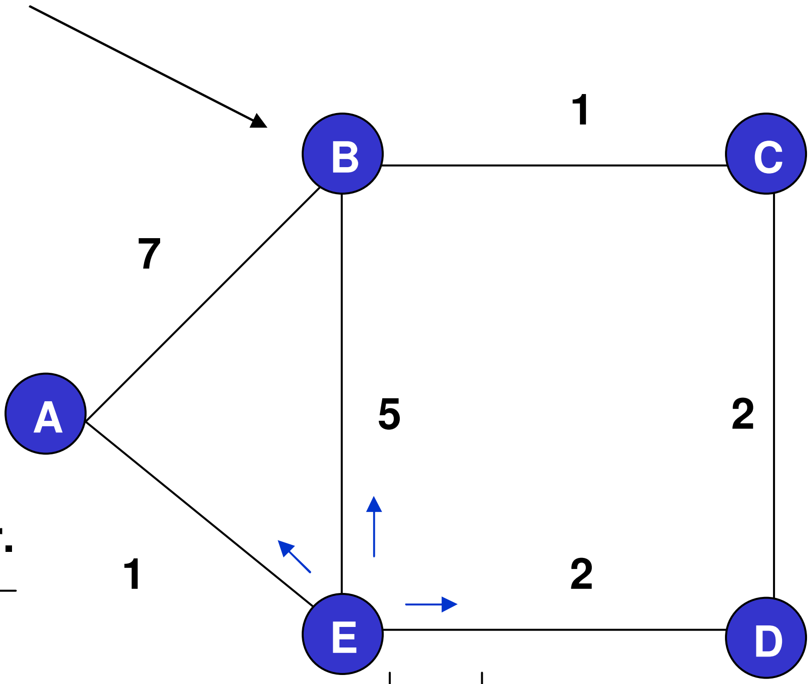
C	dist	thr.
→A	-	-
→B	1	B
→C	0	-
→D	2	D
→E	4	D

E	dist	thr.
→A	1	A
→B	5	B
→C	4	D
→D	2	D
→E	0	-

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

Distance Vector: example (running)

B	dist	thr.
→A	7	A
→B	0	-
→C	1	C
→D	-	-
→E	5	E



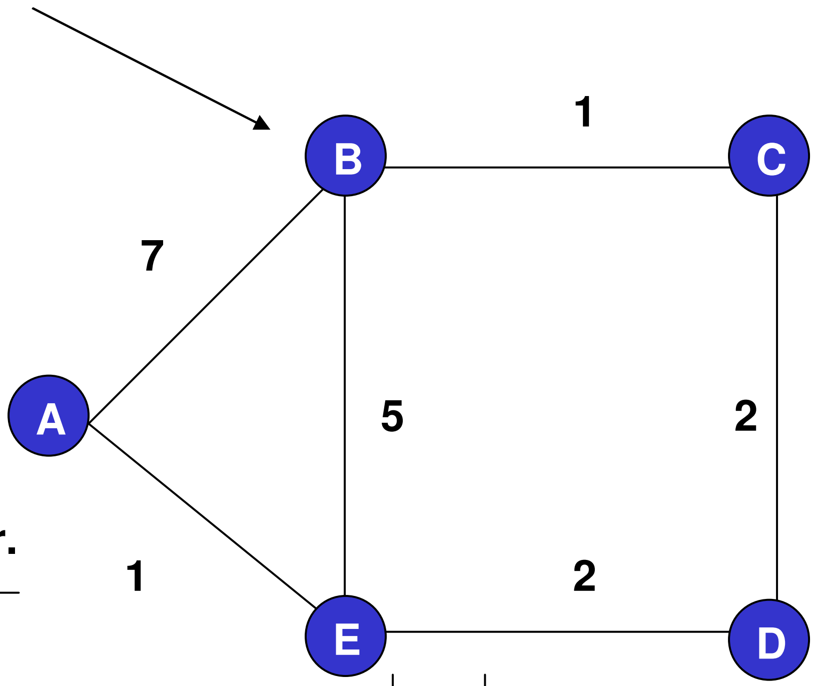
A	dist	thr.
→A	0	-
→B	7	B
→C	-	-
→D	-	-
→E	1	E

E	dist	thr.
→A	1	A
→B	5	B
→C	4	D
→D	2	D
→E	0	-

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

Distance Vector: example (running)

B	dist	thr.
→A	6	E
→B	0	-
→C	1	C
→D	7	E
→E	5	E



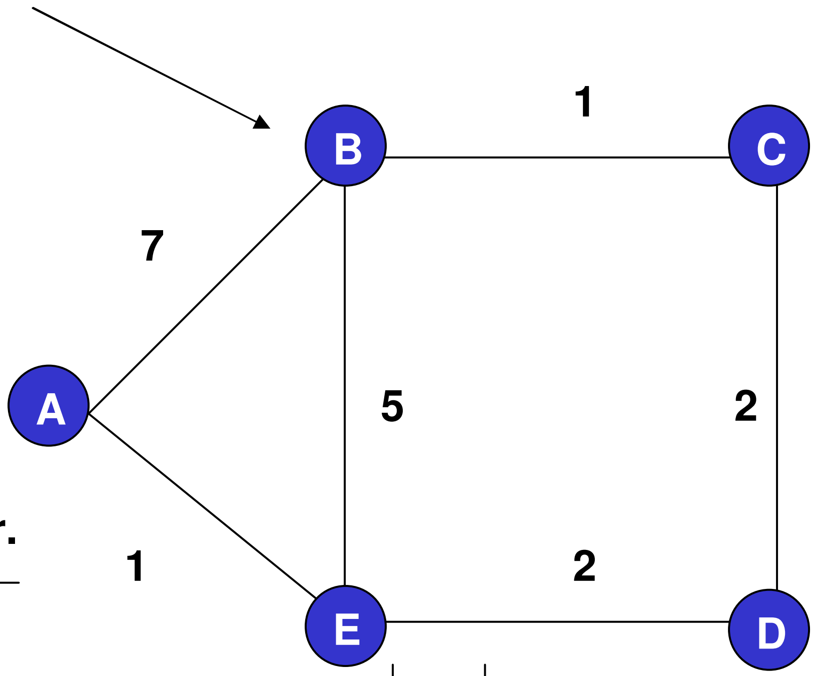
A	dist	thr.
→A	0	-
→B	6	E
→C	5	E
→D	3	E
→E	1	E

E	dist	thr.
→A	1	A
→B	5	B
→C	4	D
→D	2	D
→E	0	-

D	dist	thr.
→A	3	E
→B	7	E
→C	2	C
→D	0	-
→E	2	E

Distance Vector: example (final point)

B	dist	thr.
→A	6	E
→B	0	-
→C	1	C
→D	3	C
→E	5	E



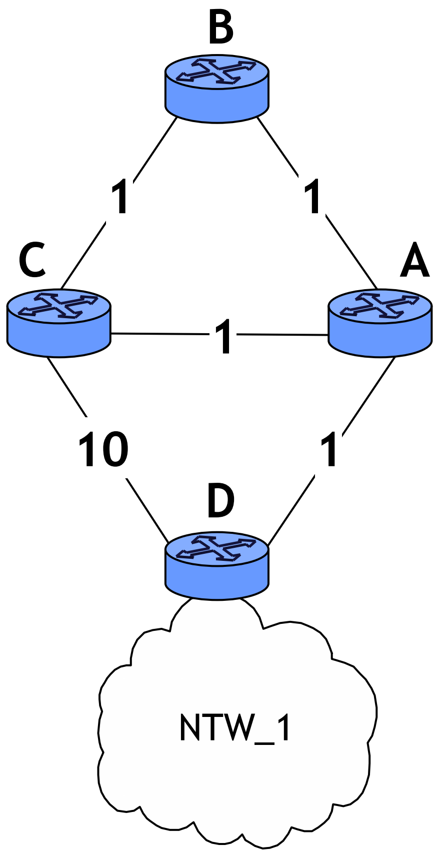
C	dist	thr.
→A	5	D
→B	1	B
→C	0	-
→D	2	D
→E	4	D

A	dist	thr.
→A	0	-
→B	6	E
→C	5	E
→D	3	E
→E	1	E

E	dist	thr.
→A	1	A
→B	5	B
→C	4	D
→D	2	D
→E	0	-

D	dist	thr.
→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

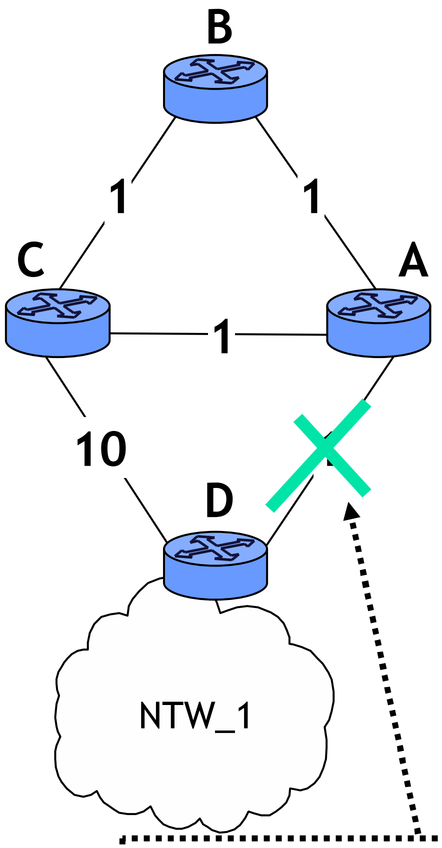
Router B		
Dest	Next	Metric
NTW_1	A	3

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”



Link between B and D fails

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

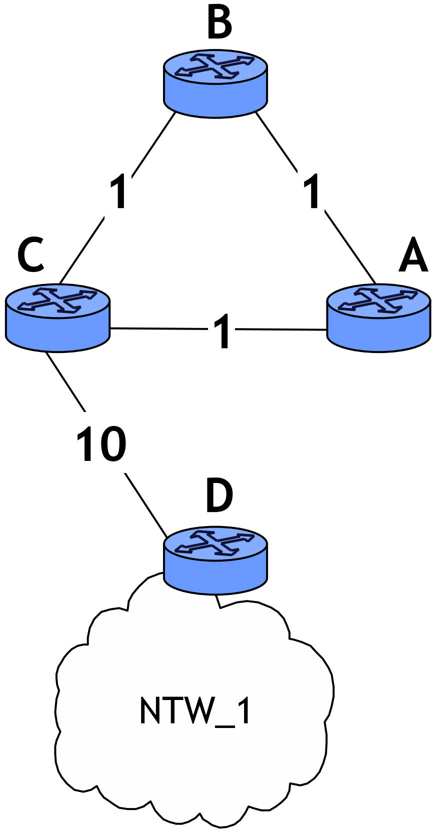
Router C

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 B		
Dest	Next	Metric
NTW_1	C	11

Router C		
Dest	Next	Metric
NTW_1	B	11

Router D		
Dest	Next	Metric
NTW_1	dir	1

Router A		
Dest	Next	Metric
NTW_1	C	12

Router B		
Dest	Next	Metric
NTW_1	C	12

Router C		
Dest	Next	Metric
NTW_1	D	11

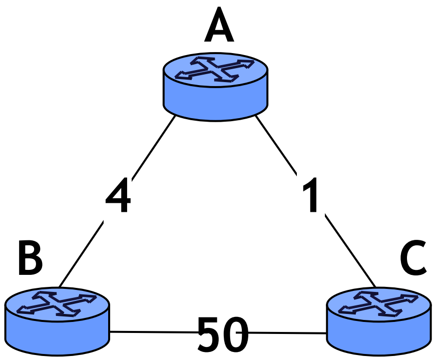
Router D		
Dest	Next	Metric
NTW_1	dir	1

Solution 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

Distance Vector: link cost changes

- 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)

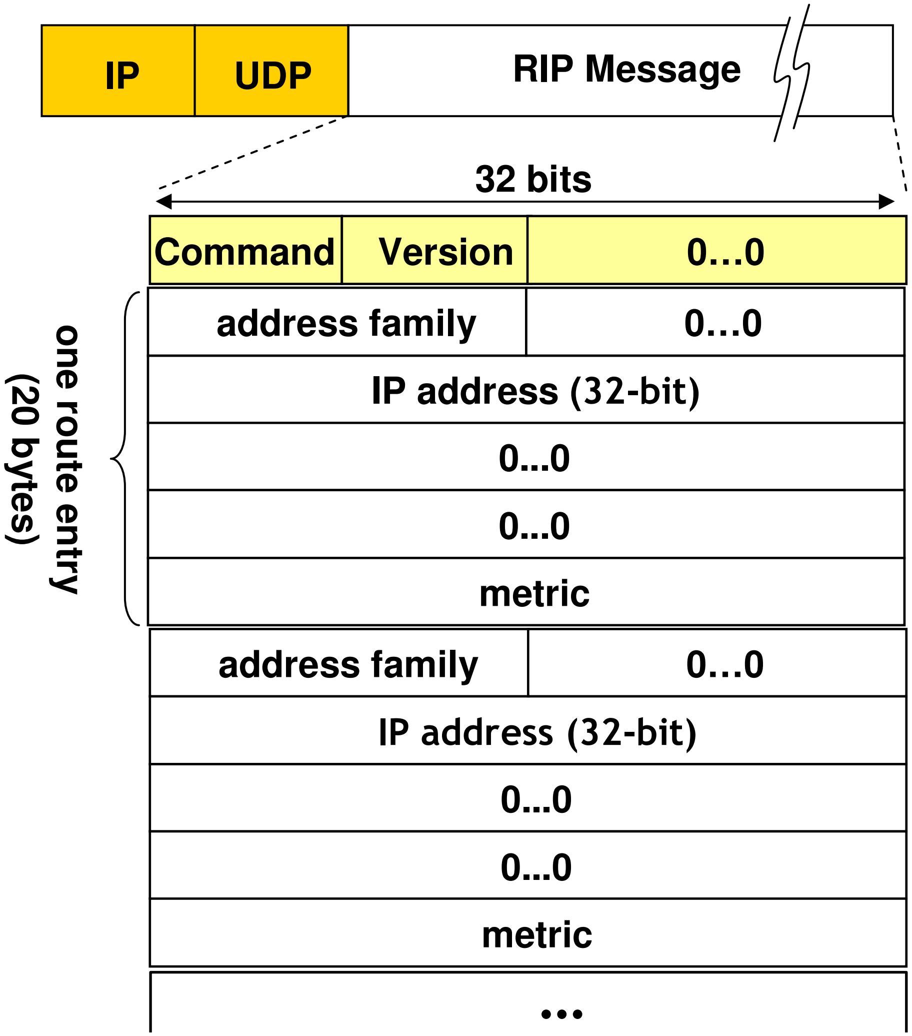
  …with well known issues
  - slow convergence
  - works with limited network size
- Strengths
  - simple to implement
  - simple management
  - widespread use

RIP at a glance

- Metric based on hop count
  - maximum hop count is 15, with “16” equal to “∞”
    - 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, for sending messages
- Changes are propagated across network
- Routes are timeout (set to 16) after 3 minutes if they are not updated

RIP: Message Format

- Command: 1=request 2=response
  - Updates are replies whether asked for or not
  - Initializing node broadcasts request
  - Requests are replied to immediately
- Version: 1
- Address family: 2 for IP
- IP address: non-zero network portion, zero host portion
  - Identifies particular network
- Metric
  - Path distance from this router to network
  - Typically 1, so metric is hop count



IP | UDP | RIP Message

32 bits

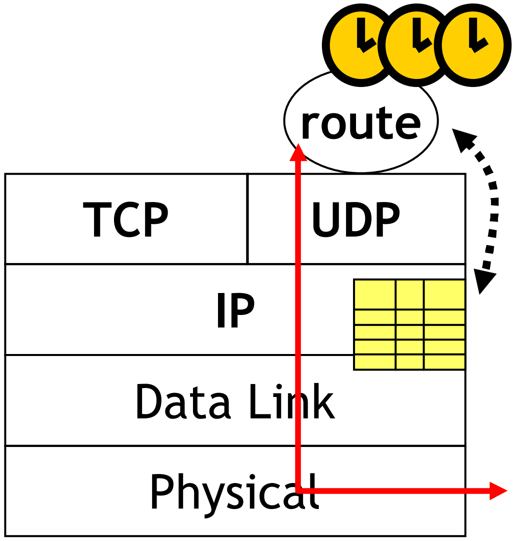
Command	Version	0…0
address family		0…0
IP address (32-bit)		
0…0		
0…0		
metric		
address family		0…0
IP address (32-bit)		
0…0		
0…0		
metric		
…		

one route entry (20 bytes)

(up to 25 total route entries)

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

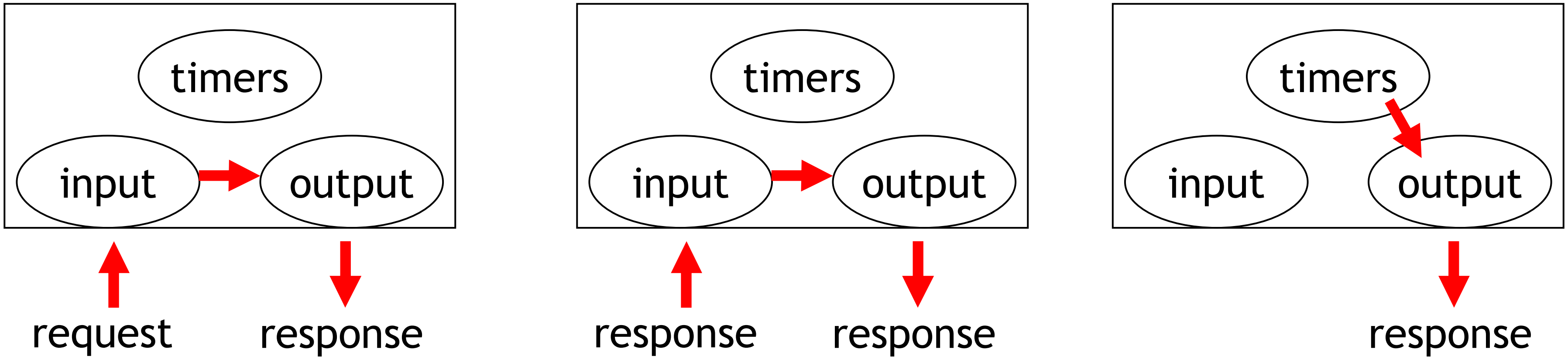


RIP procedures: input processing

- Request Messages
  - they may arrive from routers which have just come up
  - action: the router responds directly to the requestor’s address and port
    - request is processed entry by entry
- Response Messages
  - they may arrive from 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

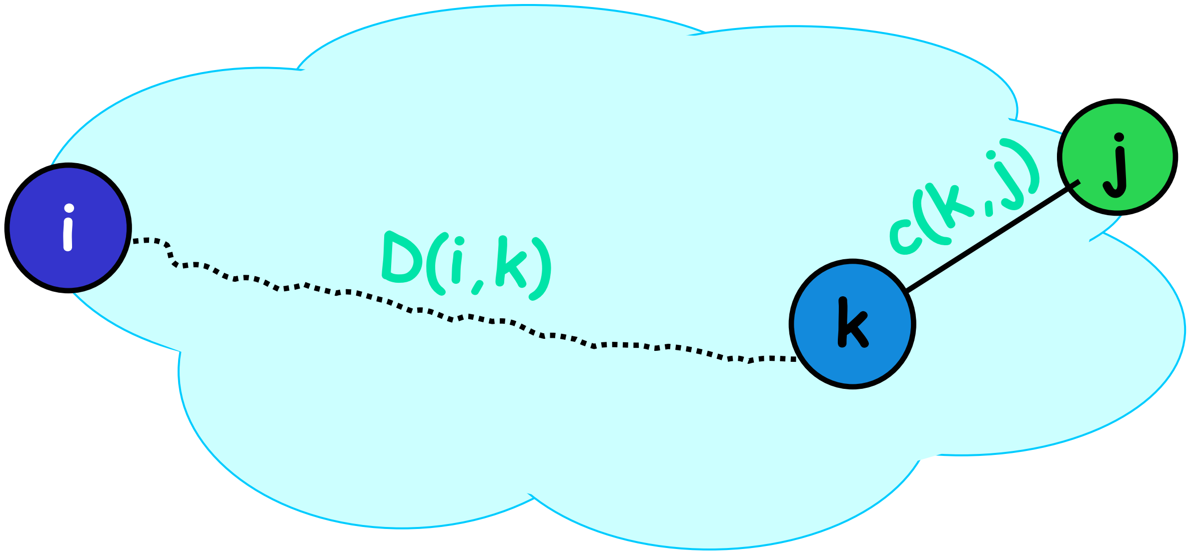
RIP procedures: output processing

- 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



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…

- After each iteration, the algorithm finds a new destination node j and a shortest path to it.
- After m iterations the algorithm has explored paths, which are m hops or smaller from node i.
  - It has an m-hop view of the network just like the distance-vector approach
- The Dijkstra algorithm at node i maintains two sets:
  - set N that contains nodes to which the shortest paths have been found so far, and
  - 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 a 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, the set N contains all the nodes, and set M is empty

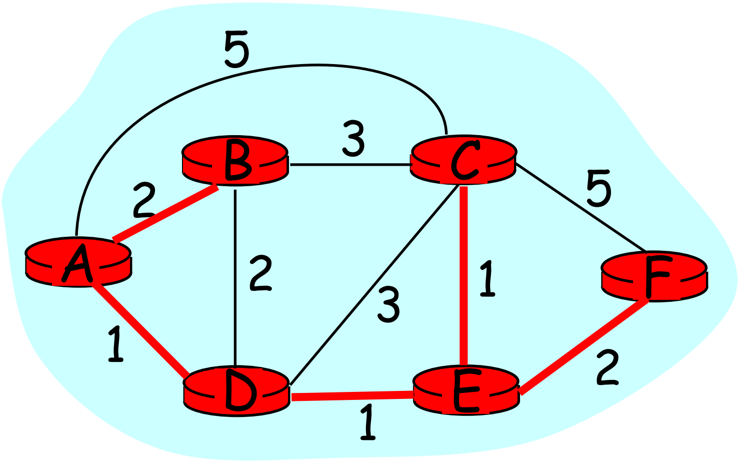
Dijkstra: Iteration

- 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, i.e. D(i,j) = min {l ε M} D(i,l).
  - 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
    - Next-hop(j) = next-hop(p(j))   if p(j) is not i
    - Next-hop(j) = j   if p(j) = i
  - Now, in addition, the distance values of any neighbor k of j in set M is reset as:
    - If D(i,k) < D(i,j) + c(j,k), then

      D(i,k) = D(i,j) + c(j,k), and 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

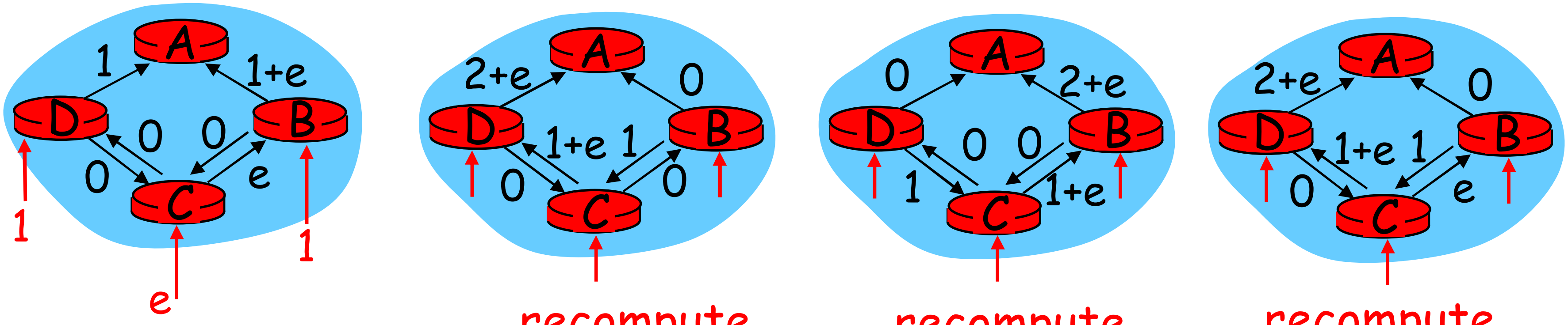
Dijkstra’s algorithm, discussion

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



initially

… recompute routing

… recompute

… recompute

Summary: Distributed Routing Techniques

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?

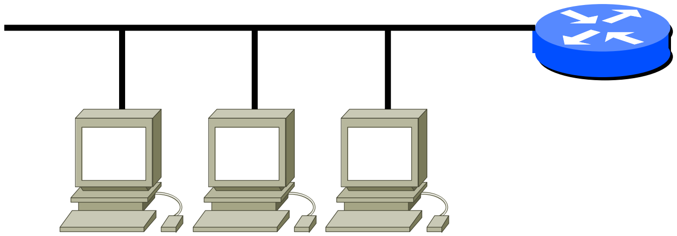
- LS:
  - node can advertise incorrect link cost
  - each node computes only its own table
- DV:
  - DV node can advertise incorrect path cost
  - each node's table used by others
    - error propagate thru network

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

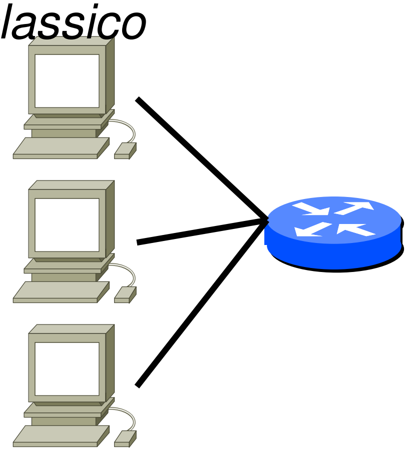
- Hli host sono collocati nelle aree periferiche della rete a sottoreti locali connesse alla attraverso router (default gateway)
- Il modello link state prevede che il database *link state* includa una entry per ogni link tra host e router
- OSPF associa il link di accesso ad una *stub network*
  - una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

Distinzione host/router (2)

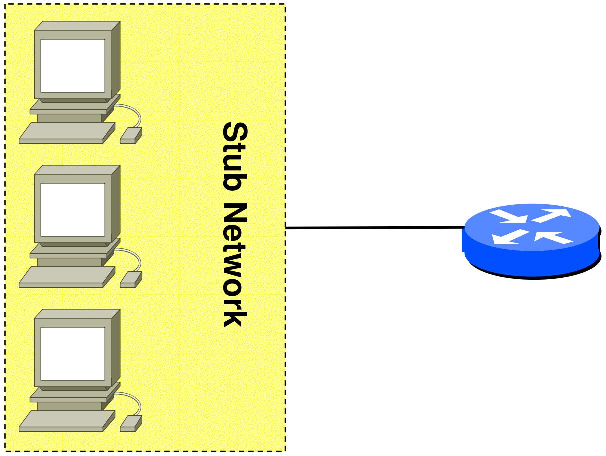


Configurazione fisica

Modello link state classico



Modello OSPF



Il protocollo OSPF

- 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
    - nelle reti broadcast biene usato un indirizzo multicast
- Tutti i messaggi OSPF condividono lo stesso header

<table>
<tr><td><i>Version #</i></td><td><i>Type</i></td><td colspan="2"><i>Packet length</i></td></tr>
<tr><td colspan="4"><i>Router ID</i></td></tr>
<tr><td colspan="4"><i>Area ID</i></td></tr>
<tr><td colspan="2"><i>Checksum</i></td><td colspan="2"><i>Auth Type</i></td></tr>
<tr><td colspan="4"><i>Authentication</i></td></tr>
<tr><td colspan="4"><i>Authentication</i></td></tr>
</table>

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

<table>
<tr><td><em>Version #</em></td><td><em>Type</em></td><td colspan="2"><em>Packet length</em></td></tr>
<tr><td colspan="4"><em>Router ID</em></td></tr>
<tr><td colspan="4"><em>Area ID</em></td></tr>
<tr><td colspan="2"><em>Checksum</em></td><td colspan="2"><em>Auth Type</em></td></tr>
<tr><td colspan="4"><em>Authentication</em></td></tr>
<tr><td colspan="4"><em>Authentication</em></td></tr>
</table>

Messaggi OSPF (3)

- Area ID: identificativo dell’area
  - 0 per la Bacvbone area
- Auth Type: tipo di autenticazione
  - 0 no autenticazione, 1 autenticazione con passwd
- Authentication: password

<table>
<tr><td><i>Version #</i></td><td><i>Type</i></td><td colspan="2"><i>Packet length</i></td></tr>
<tr><td colspan="4"><i>Router ID</i></td></tr>
<tr><td colspan="4"><i>Area ID</i></td></tr>
<tr><td colspan="2"><i>Checksum</i></td><td colspan="2"><i>Auth Type</i></td></tr>
<tr><td colspan="4"><i>Authentication</i></td></tr>
<tr><td colspan="4"><i>Authentication</i></td></tr>
</table>

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		

Hello Protocol: formato pacchetto (3)

- Neighbor: lista di nodi adiacenti da cui ha ricevuto un messaggio di Hello negli ultimi **dead interval** secondi

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

Exchange Protocol: messaggi (1)

- Database Description

<table>
<tr><td colspan="4"><em>Common header (type = 2, db description)</em></td></tr>
<tr><td><em>0</em></td><td><em>0</em></td><td><em>Options</em></td><td><em>0</em></td></tr>
<tr><td colspan="4"><em>DD sequence number</em></td></tr>
<tr><td colspan="4"><em>Link State Type</em></td></tr>
<tr><td colspan="4"><em>Link State ID</em></td></tr>
<tr><td colspan="4"><em>Advertising router</em></td></tr>
<tr><td colspan="4"><em>Link State Sequence Number</em></td></tr>
<tr><td colspan="2"><em>Link State Checksum</em></td><td colspan="2"><em>Link State Age</em></td></tr>
</table>

Exchange Protocol: messaggi (2)

- Link State Request

Common header (type = 3, link state request)
Link State Type
Link State ID
Advertising router

- Link state Update

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

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