CS118 Discussion 1B, Week 7

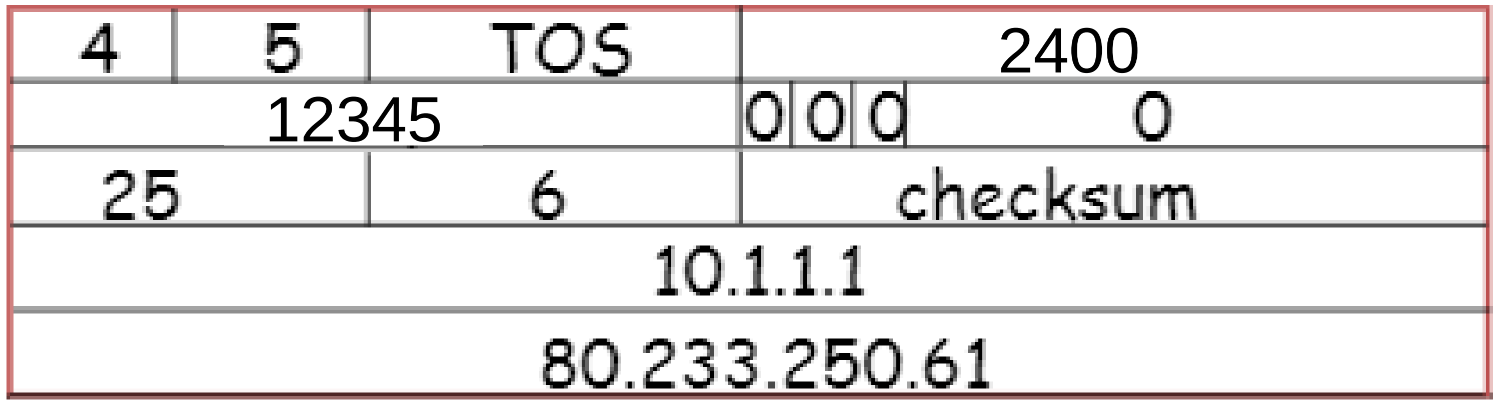
Zhehui Zhang
HAINES A2 / Friday / 12:00pm-1:50pm

Outline

- Network data plane
  - Fragmentation, DHCP, NAT, IPv6, Openflow
- Network control plane
  - Routing
    - Link state routing
    - Distance vector routing
- Project 2

Quick question on fragmentation

- Consider following IP packet

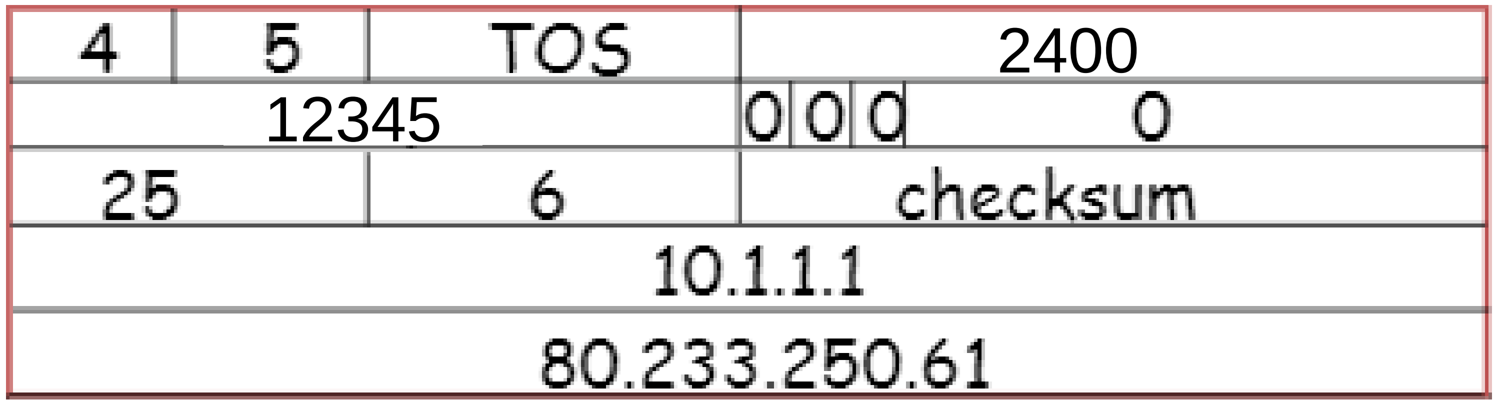


4	5	TOS	2400
12345		0 0 0	0
25	6	checksum	
10.1.1.1			
80.233.250.61			

- Assume MTU = 1450 Bytes. Show the header length, total length, identification, flags, fragment offset, TTL, and IP payload size.

Quick question

- Consider following IP packet



4	5	TOS	2400	
12345			0 0 0	0
25		6	checksum	
10.1.1.1				
80.233.250.61				

- Assume MTU = 1450 Bytes. Show the header length, total length, identification, flags, fragment offset, TTL, and IP payload size.

For the first packet: 5 (20 bytes), 1444 bytes, ID = 12345, MF = 1, Offset = 0, TTL = 25, 1424 bytes.
For the second packet: 5 (20 bytes), 976 bytes, ID = 12345, MF = 0, Offset = 178, TTL = 25, 956 bytes.

DHCP: Dynamic Host Configuration Protocol

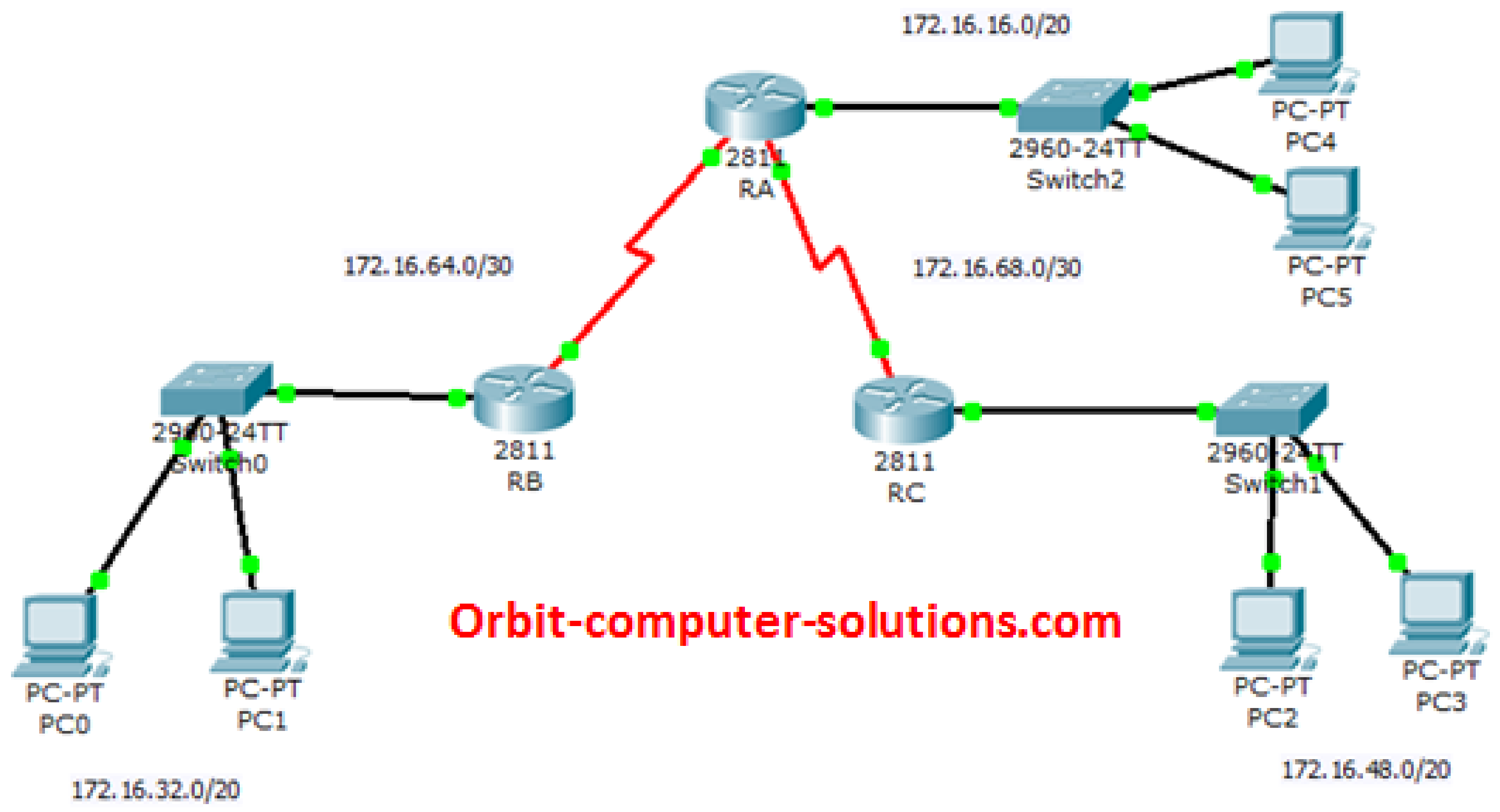
- Dynamically allocates the following info to a host
  - IP address on subnet for the host
  - IP address for default router (“first-hop” router)
  - Subnet mask
  - IP address and name for DNS caching resolver
- Allows address reuse

DHCP: operations

- Host broadcasts “DHCP discovery” msg [optional]
- DHCP server responds with “DHCP offer” msg [optional]
- Host requests IP address: “DHCP request” msg
- DHCP server sends address: “DHCP ack” msg

Important example on Chapter 4 slides 45—46!

Quick question



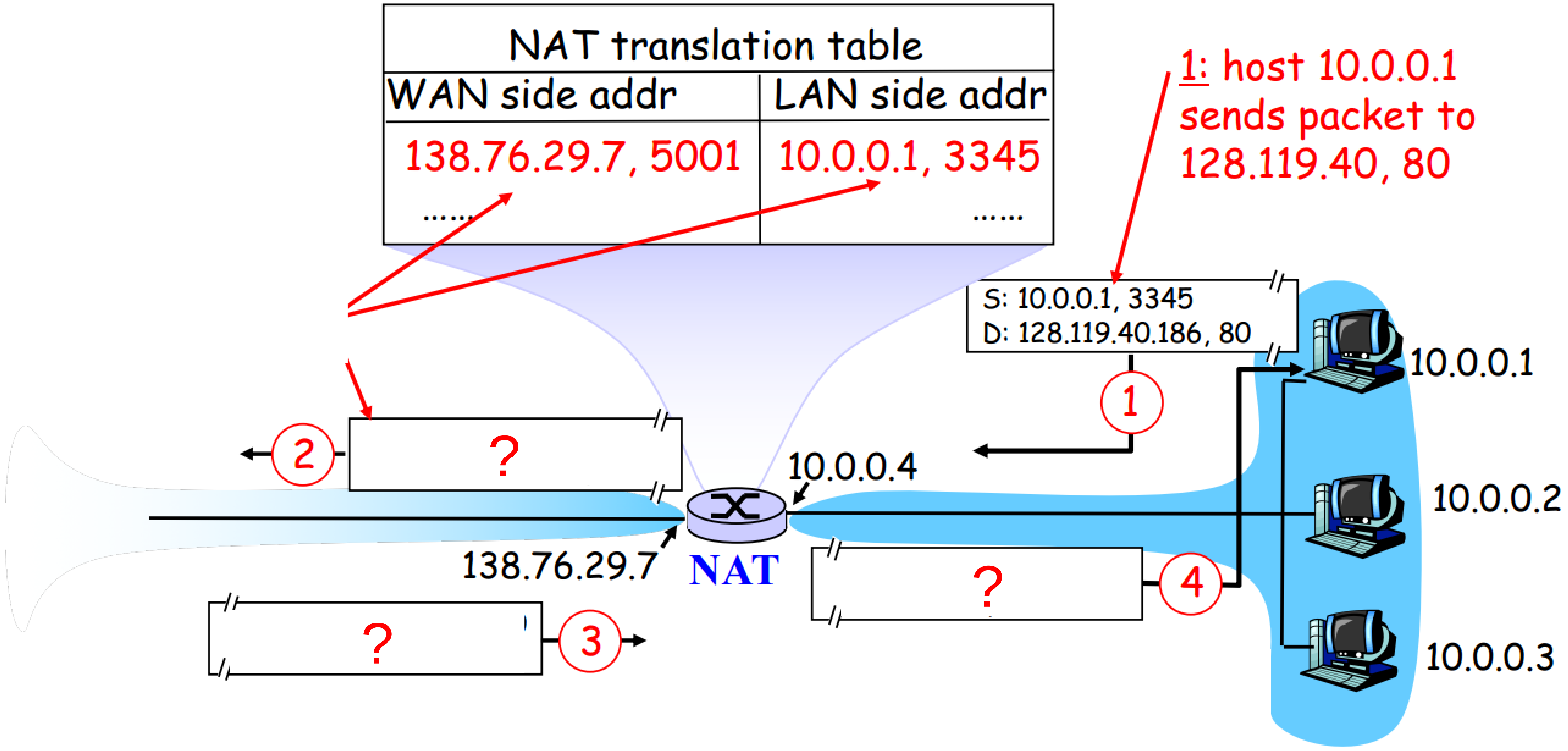
- What information in DHCP will be updated if PC0 move to PC1?
- How about PC0 move to PC3?

NAT (network address translation)

- Depletion of IPv4 addresses — short-term solution
  - IP tunneling?
- Use private IP addresses
- Side-benefit: security
- How to achieve?
  - <public IP:port> — <private IP:port> mapping

Quick question

•



NAT: downside

- Increased complexity
- Single point of failure
- Cannot run services inside a NAT box
  - Why?

IPv6

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version| Traffic Class |           Flow Label                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Payload Length        |  Next Header  |   Hop Limit   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                         Source Address                        +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                                                               +
|                                                               |
+                      Destination Address                      +
|                                                               |
+                                                               +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

IPv6 Header Format (RFC 2460)

IPv6/IPv4 differences

- Fixed-length 40 byte header
    - length field excludes header
    - Header Length field eliminated
- Address length: 128 bits
- Priority: usage yet to be finalized
- Flow Label: identify packets in same flow
- Next header: identify upper layer protocol for data
- Options: outside of the basic header, indicated by Next Header field
- Header Checksum: removed

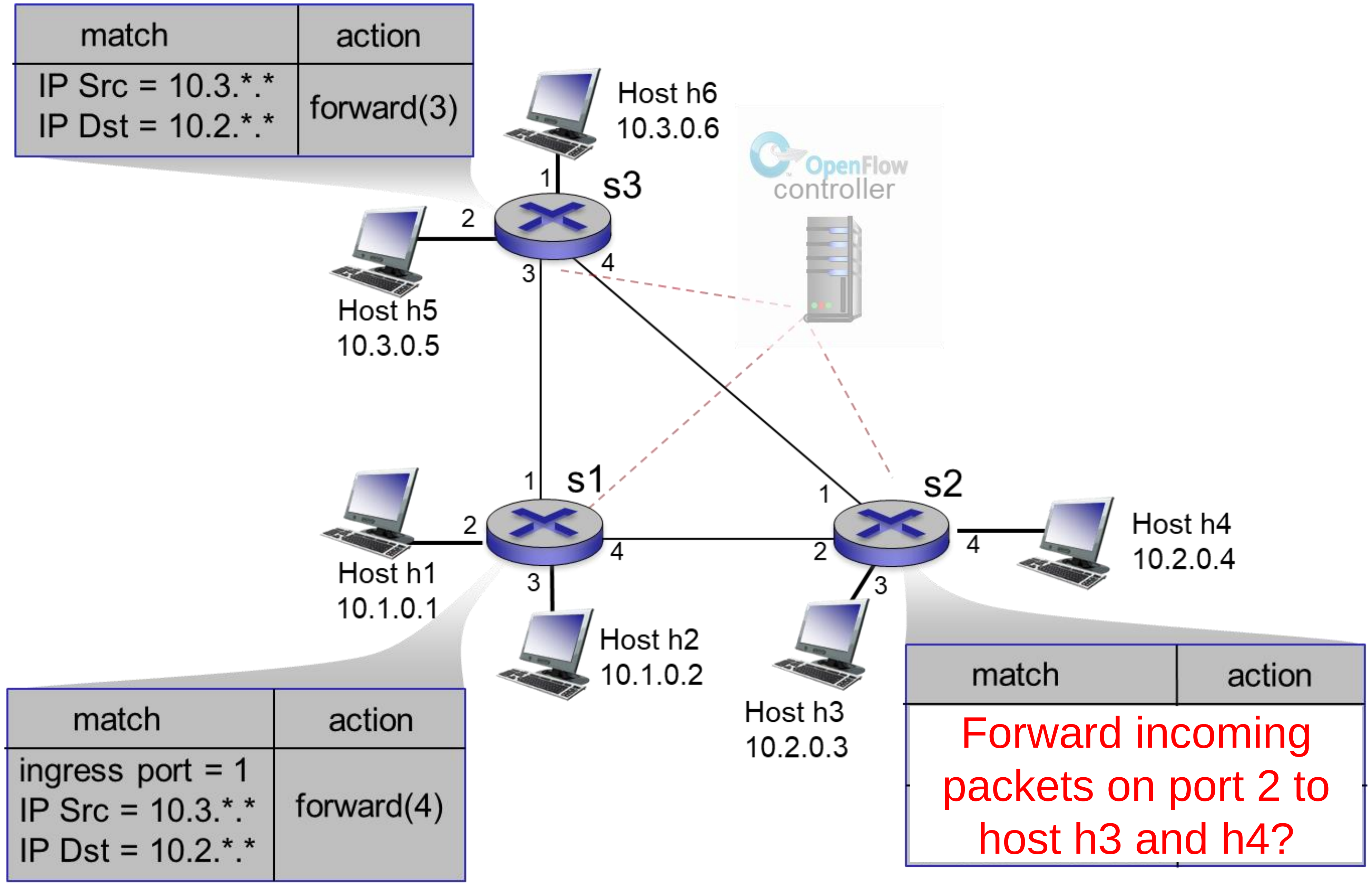
IPv6 address format (optional)

- Colon-Hex: 2607:F010:03f9:0000:0000:0000:0004:0001
  - Can skip leading zeros of each word: 2607:F010:3f9:0:0:0:4:1
  - Can skip one sequence of zero words (compressed representation), e.g., 2607:f010:3f9::4:1
  - Can leave the last 32 bits in dot-decimal: 2607:f010:3f9::0.4.0.1
  - Can specify a prefix by /length: 2607:f010:3f9::/64

Special IPv6 addresses (optional)

- ::/128 - Unspecified
- ::1/128 - Loopback
- ::ffff:0:0/96 - IP4-mapped address
- 2002::/16 - 6to4
- ff00::/8 - Multicast
- fe80::/10 - Link-Local Unicast

Open flow example



match	action
IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(3)

match	action
ingress port = 1 IP Src = 10.3.*.* IP Dst = 10.2.*.*	forward(4)

match	action
Forward incoming packets on port 2 to host h3 and h4?	

Routing: concepts

- Global or decentralized information?
  - global: all routers have complete topology, link cost info
    - algorithm?

Routing: concepts

- Global or decentralized information?
  - global: all routers have complete topology, link cost info
    - “link state” algorithms

Routing: concepts

- Global or decentralized information?
  - global: all routers have complete topology, link cost info
    - “link state” algorithms
  - decentralized: router knows physically-connected neighbors, link costs to neighbors; iterative process of computation, exchange of info with neighbors
    - algorithm?

Routing: concepts

- Global or decentralized information?
  - global: all routers have complete topology, link cost info
    - “link state” algorithms
  - decentralized: router knows physically-connected neighbors, link costs to neighbors; iterative process of computation, exchange of info with neighbors
    - “distance vector” algorithms

Link state routing

- Dijkstra’s algorithm
  - net topology, link costs known to all nodes
  - computes least cost paths from one node (‘source”) to all other nodes
  - iterative: after k iterations, know least cost path to k destinations

Link state routing: algorithm

```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

 Loop
   find w not in N' such that D(w) is a minimum
   add w to N'
   update D(v) for all v adjacent to w and not in N':
     [Link cost update heuristic from Dijkstra algo.]
 until all nodes in N'
```

c(x, y): link cost from node x to y; c(x, y) = ∞ if not direct neighbors
D(v): current value of cost of path from source to destination v
p(v): predecessor node along path from source to v
N': set of nodes whose least cost path definitively known

Link state routing: algorithm

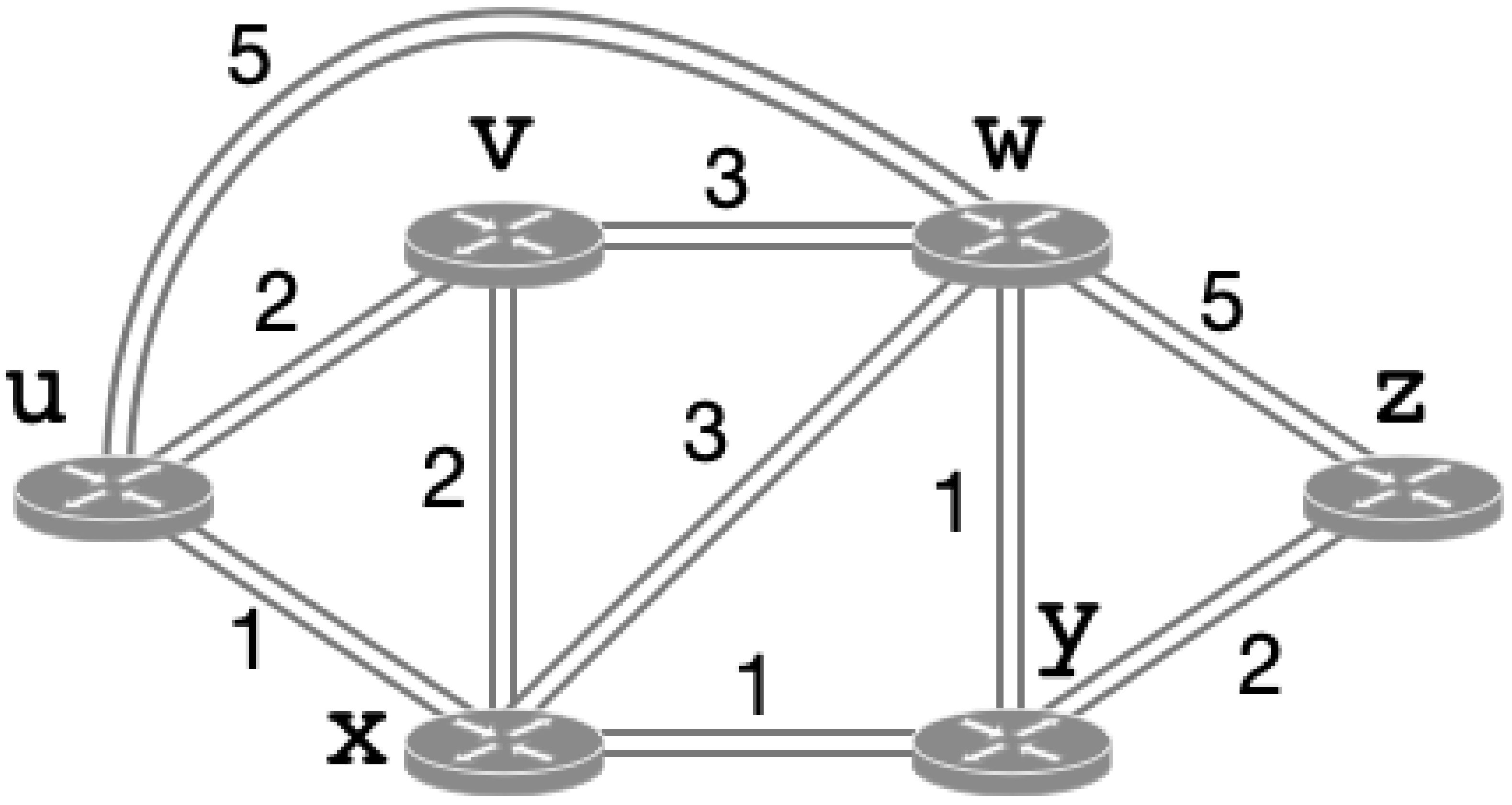
```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

 Loop
   find w not in N' such that D(w) is a minimum
   add w to N'
   update D(v) for all v adjacent to w and not in N':
     D(v) = min( D(v), D(w) + c(w,v) )
 until all nodes in N'
```

c(x, y): link cost from node x to y; c(x, y) = ∞ if not direct neighbors
D(v): current value of cost of path from source to destination v
p(v): predecessor node along path from source to v
N': set of nodes whose least cost path definitively known

Link state routing: example

- Using link state routing to setup a forwarding table for node `u`



Let’s work it out

N’	D(v), p(v)	D(w), p(w)	D(x), p(x)	D(y), p(y)	D(z), p(z)
u	2, u	5, u	1, u	∞	∞
ux	2, u	4, x		2, x	∞
uxy	2, u	3, y			4, y
uxyv		3, y			4, y
uxyvw					4, y
uxyvwz					

Let’s work it out

N’	D(v), p(v)	D(w), p(w)	D(x), p(x)	D(y), p(y)	D(z), p(z)
u	2, u	5, u	1, u	∞	∞
ux	2, u	4, x		2, x	∞
uxy	2, u	3, y			4, y
uxyv		3, y			4, y
uxyvw					4, y
uxyvwz					

Link state routing: complexity

- size: 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(nlogn)

Distance vector routing

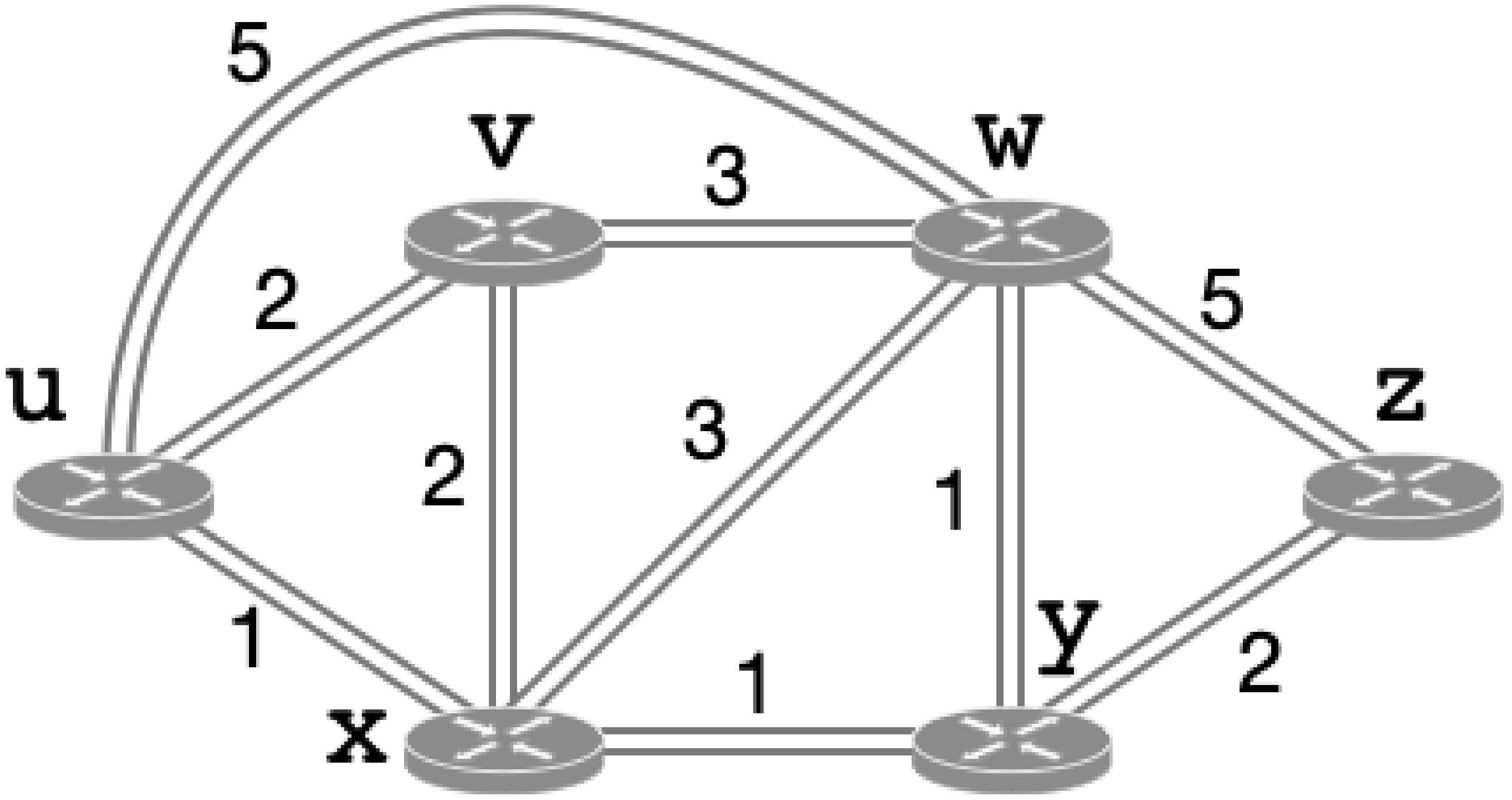
- Bellman-Ford equation (dynamic programming)
- let
- dx(y) := cost of least-cost path from x to y
- then
- dx(y) = ?

Distance vector routing

- Bellman-Ford equation (dynamic programming)
- let
-     dx(y) := cost of least-cost path from x to y
- then
-     dx(y) = min_v {c(x,v) + dv(y) }, v: neighbors of x

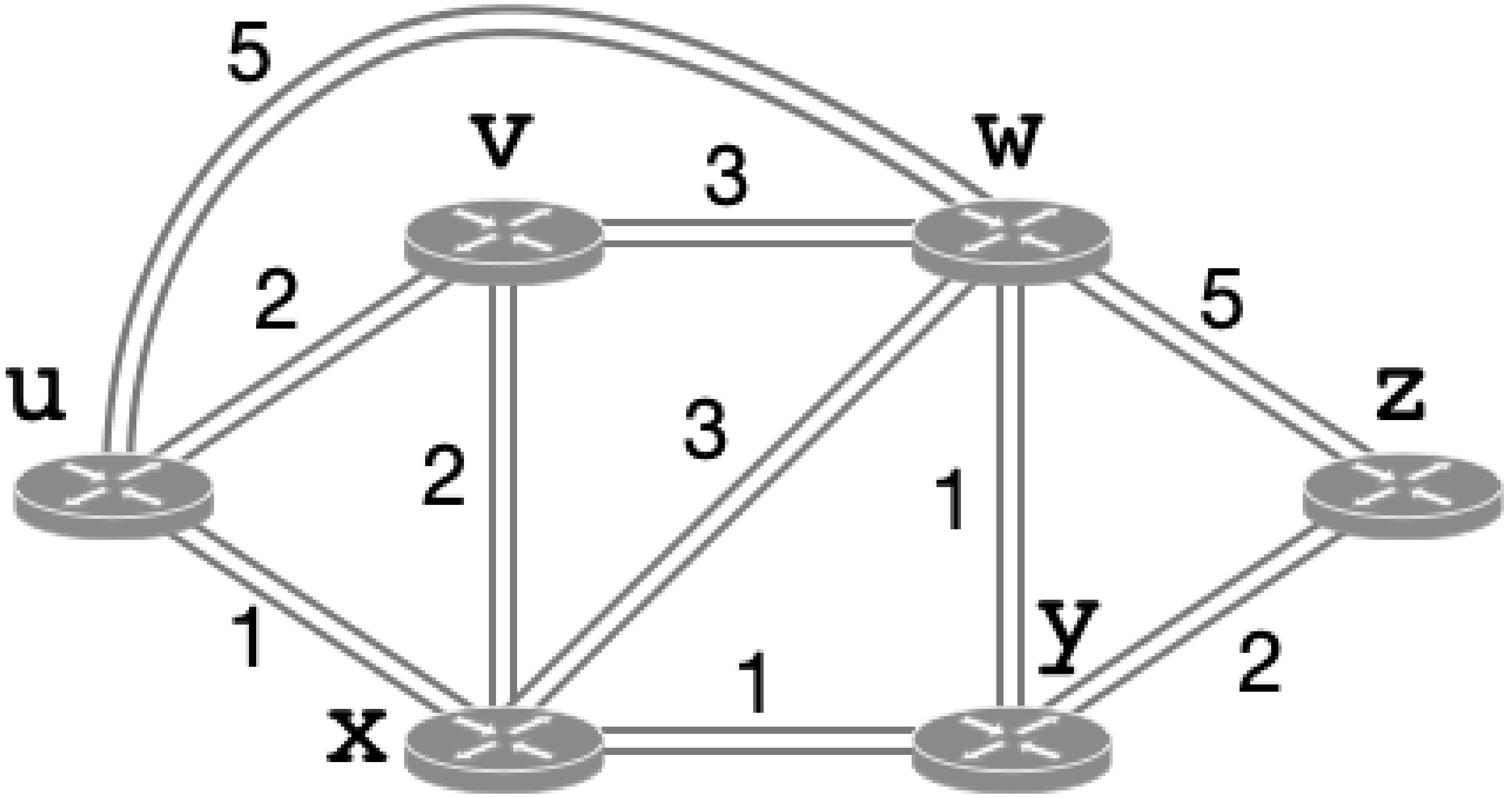
Distance vector routing: example

- What's the cost of least-cost path for `u → z`?



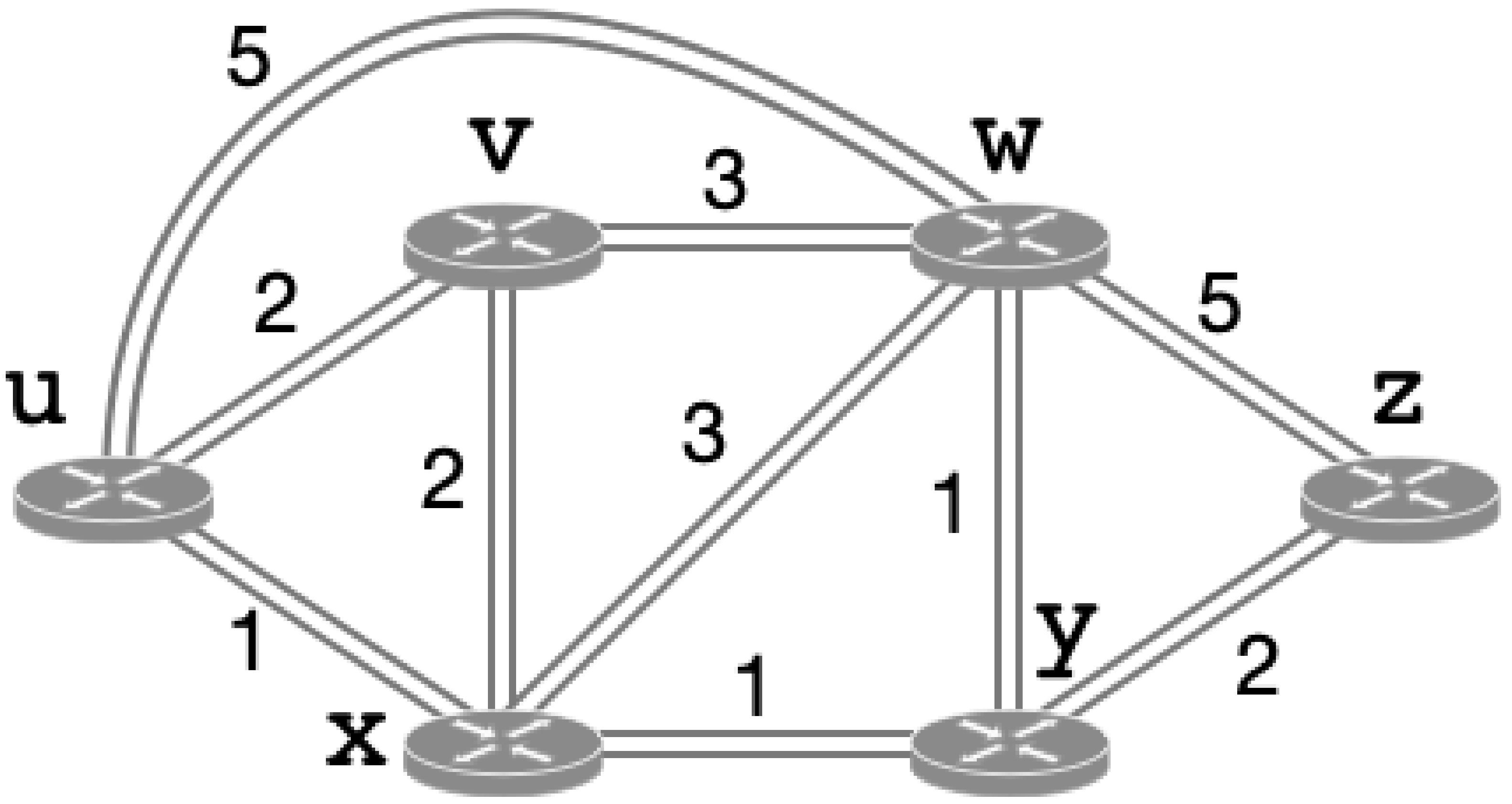
Let's work it out

- clearly:
  - dv(z) = ?, dx(z) = ?, dw(z) = ?



Let’s work it out

- clearly:
  - dv(z) = 5, dx(z) = 3, dw(z) = 3
- According to B-F equation:
  - du(z) = min { ? }



Let’s work it out

- clearly:
  - dv(z) = 5, dx(z) = 3, dw(z) = 3
- According to B-F equation:
  - du(z) = min {c(v, x) + dv(z), c(u,x) + dx(z), c(u,w) + dw(z)}

Let’s work it out

- clearly:
  - $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$
- According to B-F equation:
  - $d_u(z) = \min \{c(u, v) + d_v(z), c(u,x) + d_x(z), c(u,w) + d_w(z)\}$
    - $= \min \{2 + 5, 1 + 3, 5 + 3\} = 4$

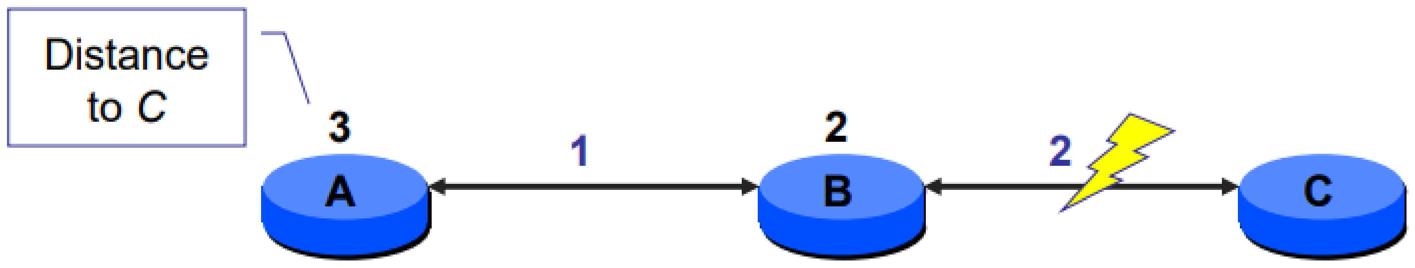
Distance vector routing: key idea

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

- when x receives new DV estimate from neighbor, it updates its own DV using B-F equation.

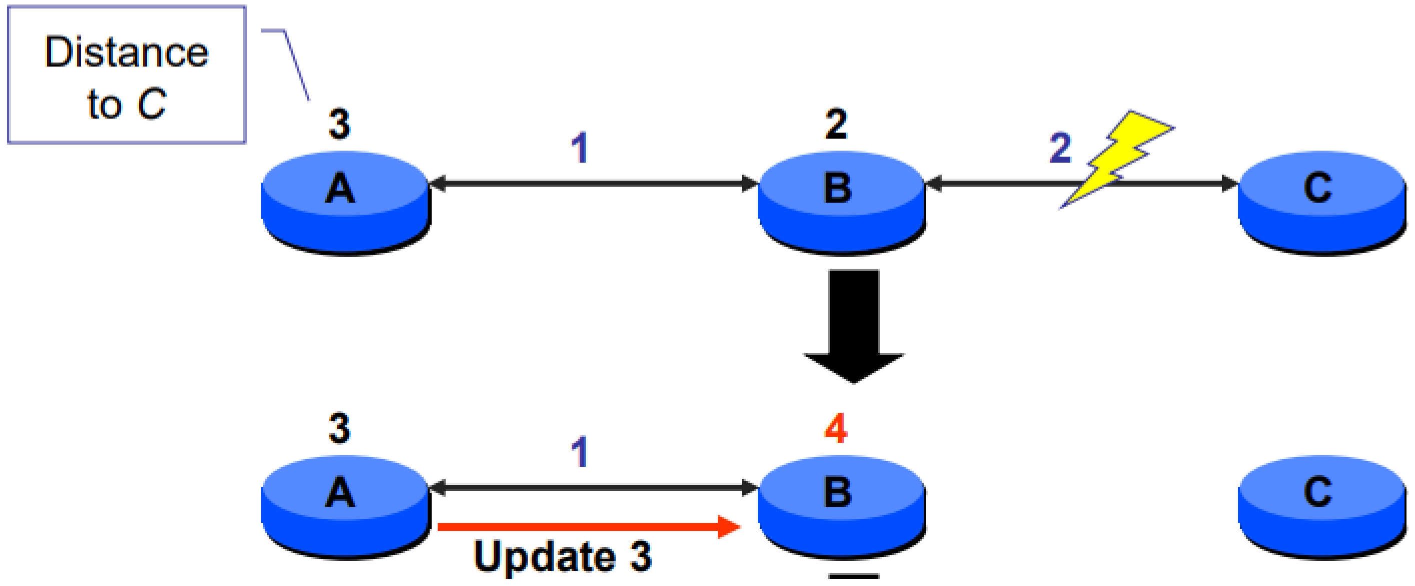
Distance vector routing: caveat

- Count-to-infinity problem.
- Can you work out an example?



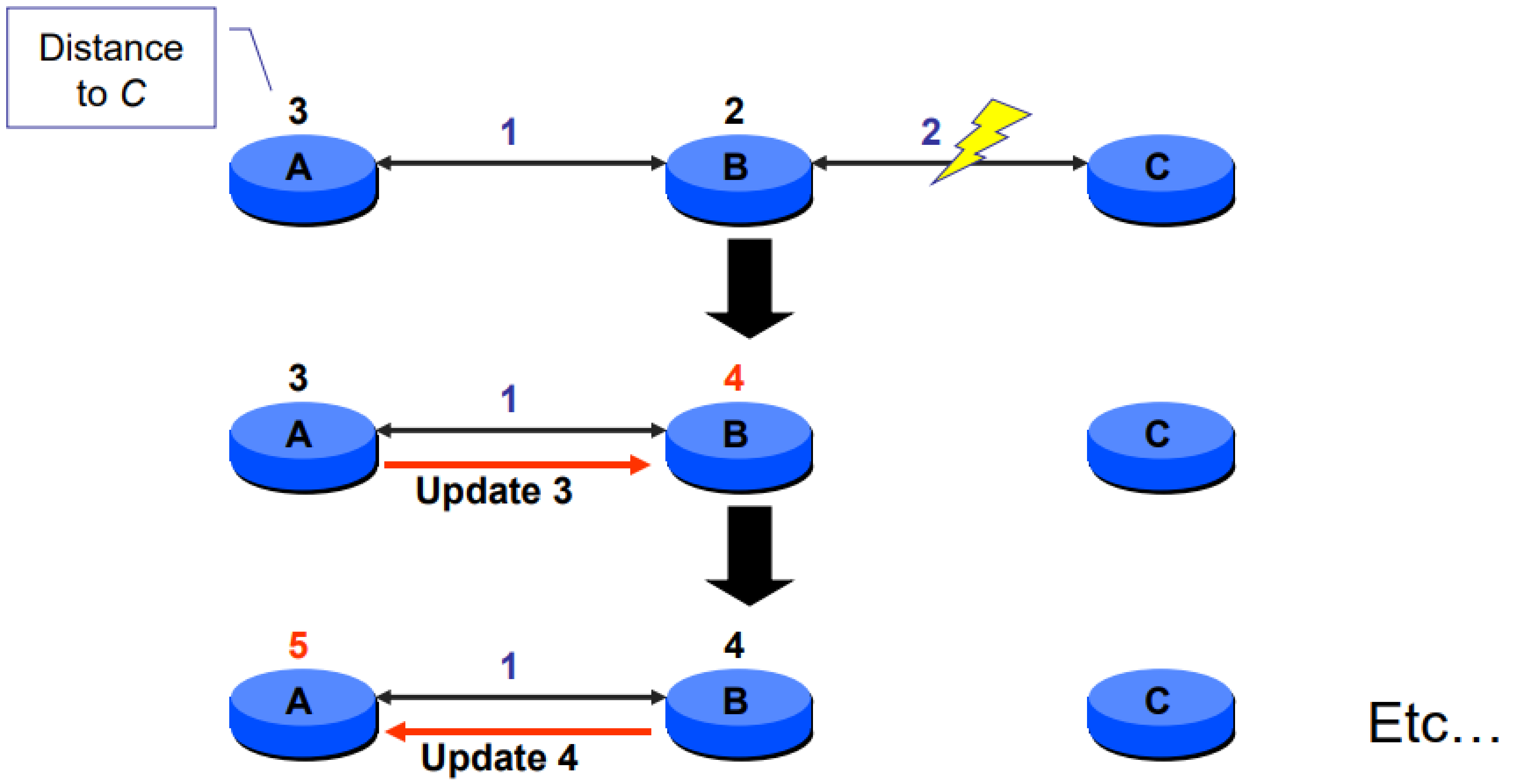
Distance vector routing: caveat

- Count-to-infinity problem.
- Can you work out an example?



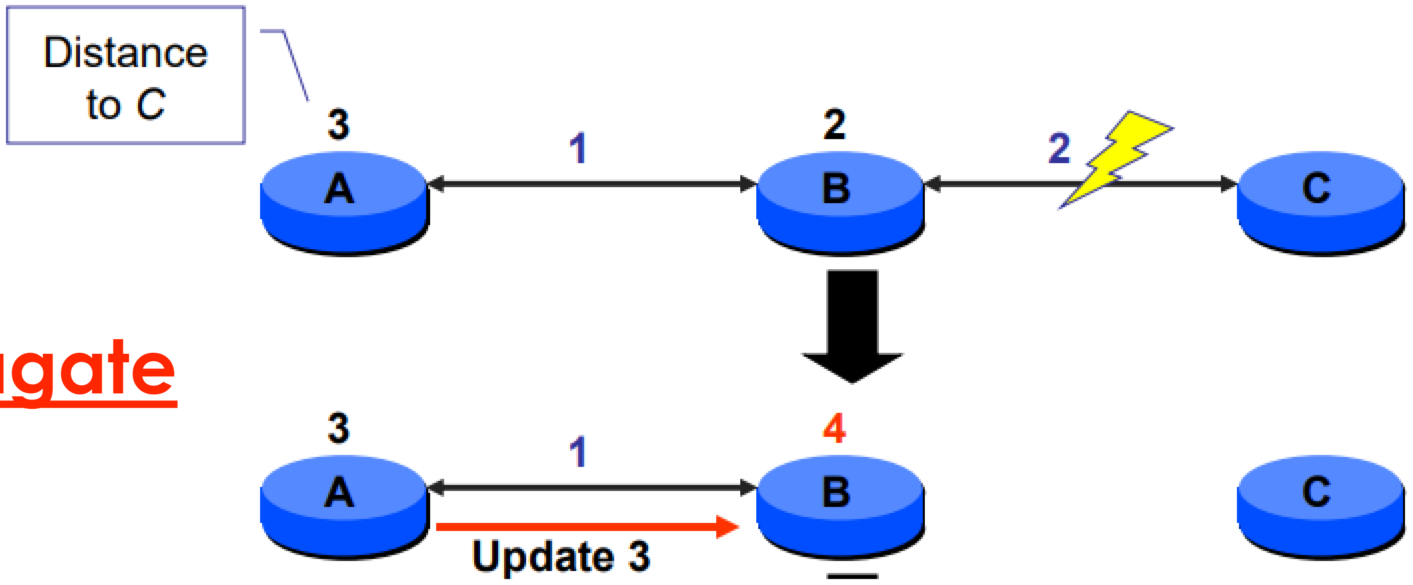
Distance vector routing: caveat

- Count-to-infinity problem.
- Can you work out an example?



Distance vector routing: caveat

- Count-to-infinity problem.
- Can you work out an example?
- Can you propose a solution?
    - basic idea?



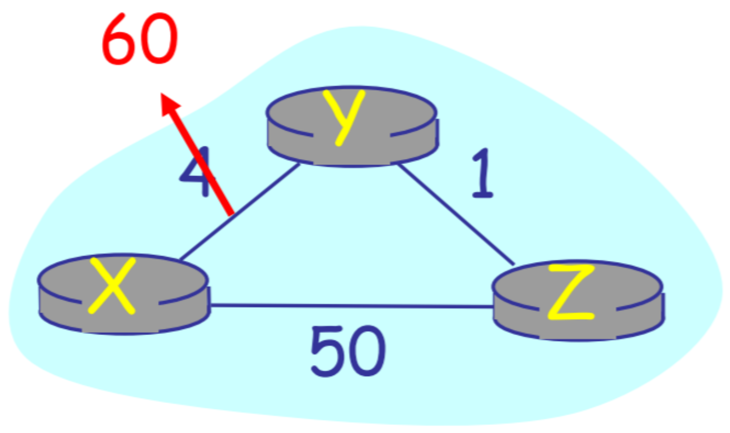
A should not propagate its distance to B!

Distance vector routing: split horizon

- Previous solution idea:
  - split horizon
    - if A reaches C through B, A should not tell B that B can reach C
    - Then B will not attempt to go through A to reach C
  - Are we good?

Distance vector routing: split horizon

- Previous solution idea:
  - split horizon
    - if A reaches C through B, A should not tell B that B can reach C
    - Then B will not attempt to go through A to reach C
  - Are we good?



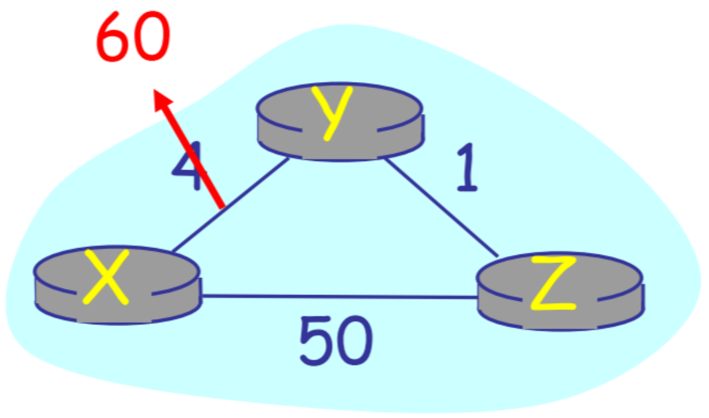
Distance vector routing: poison reverse

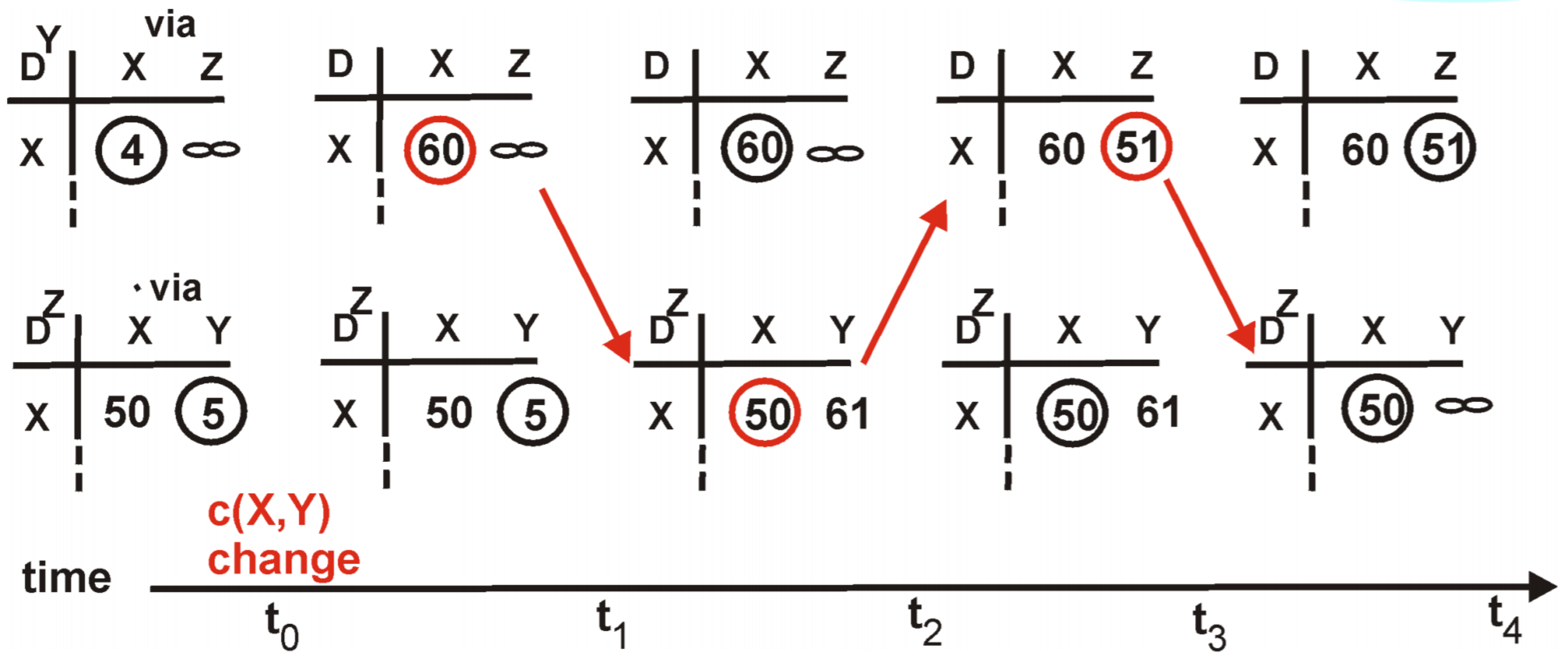
- Split horizon + poison reverse
    - if A reaches D through C:
        - A tells C that A’s distance to D is infinite
        - Then C will not attempt to go through A to reach D
        - In practice, infinite == 16 hops

Distance vector routing: poison reverse

If *Z* routes through *Y* to get to *X*:

- *Z* tells *Y* its (*Z*'s) distance to *X* is infinite (so *Y* won't route to *X* via *Z*)





Link State v.s. Distance Vector

	Link state	Distance vector
message complexity	with n nodes, E links, O(nE) msgs sent	exchange between neighbors only (convergence time varies)
convergence speed	O(n2) algorithm requires O(nE) msgs	convergence time varies (may be routing loops)
robustness	node can advertise incorrect link cost; each node computes only its own table	DV node can advertise incorrect path cost; error propagate thru network
implementation	OSPF	RIP

Summary

- Link-state routing (Dijkstra) algorithm:
    - each node computes the shortest paths to all the other nodes based on the complete topology map
- Distance Vector (Bellman-Ford) routing algorithm:
    - each node computes the shortest paths to all the other nodes based on its neighbors distance to all destinations