DIPARTIMENTO DI INFORMATICA
Via Salaria 113 - 00198, Roma

SAPIENZA UNIVERSITÀ DI ROMA

Chapter 4
Network Layer

Reti di Elaboratori
Corso di Laurea in Informatica
Università degli Studi di Roma "La Sapienza"
Canale A-L

Prof.ssa Chiara Petrioli

Thanks also to Antonio Capone, Politecnico di Milano, Giuseppe Bianchi and Francesco LoPresti, Un. di Roma Tor Vergata

Previous lecture. Summary:

Distributed Belman Ford

- Based on Distributed Bellman Ford Equation

Cost associated to the (X,Z) link

$D^X(Y,Z)$ = distance *from* X *to* Y, *via* Z as next hop

$= c(X,Z) + \min_w \{D^Z(Y,w)\}$

- $D^x(Y,Z)$ recomputed:
  - Upon reception of updates from the neighbors
  - Upon link cost change
- $\min_z D^X(Y,Z)$ communicated to the neighbors whenever its value changes, or periodically

Distance Vector Routing: overview

Iterative, asynchronous: each local iteration caused by:

- local link cost change
- message from neighbor: its least cost path change from neighbor

Distributed:

- each node notifies neighbors *only* when its least cost path to any destination changes
  - neighbors then notify their neighbors if necessary

Each node:

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

↓

recompute distance table

↓

if least cost path to any dest has changed, *notify* neighbors

Distributed Bellman Ford correctness

- Completely asynchronous
- Starting from arbitrary estimates of the cost of the ‘best route’ from node i to the destination, if:
  - links weights are constant for enough time for the protocol to converge
  - stale info expire after a while
  - once in a while updated info are sent from a node to its neighbors

the Distributed Bellman Ford algorithm converges, i.e. each node correctly estimates the cost of the best route to the destination

Bellman-Ford

Given a graph G=(N,A) and a node *s* finds the shortest path from *s* to every node in N.

A shortest walk from *s* to i subject to the constraint that the walk contains at most h arcs and goes through node *s* only once, is denoted shortest(<=h) walk and its length is D^h_i.

Bellman-Ford rule:

Initiatilization $D^h_s=0$, for all h; $w_{i,k}$ = infinity if (i,k) NOT in A; $w_{k,k}=0$; **D^0_i=infinity for all i!=s.**

Iteration:

$$D^{h+1}_i=\min_k [w_{i,k} + D^h_k]$$

Assumption: non negative cycles **(this is the case in a network!!)**

The Bellman-Ford algorithm first finds the one-arc shortest walk lengths, then the two-arc shortest walk length, then the three-arc…etc. →distributed version used for routing

Previous lecture. Summary:

Distributed Belman Ford

- Based on Distributed Bellman Ford Equation

Cost associated to the (X,Z) link

$D^X(Y,Z)$ = distance *from* X *to* Y, *via* Z as next hop

$$D^X(Y,Z) = c(X,Z) + \min_w \{D^Z(Y,w)\}$$

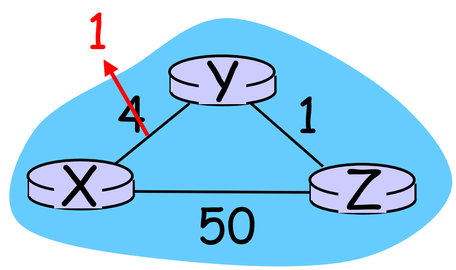
- $D^x(Y,Z)$ recomputed:
  - Upon reception of updates from the neighbors
  - Upon link cost change
- $\min_z D^X(Y,Z)$ communicated to the neighbors whenever its value changes, or periodically
- How long does it take for the algorithm to converge? ‘good news travel fast, bad news may not→count to infinity’

Distance Vector: link cost changes

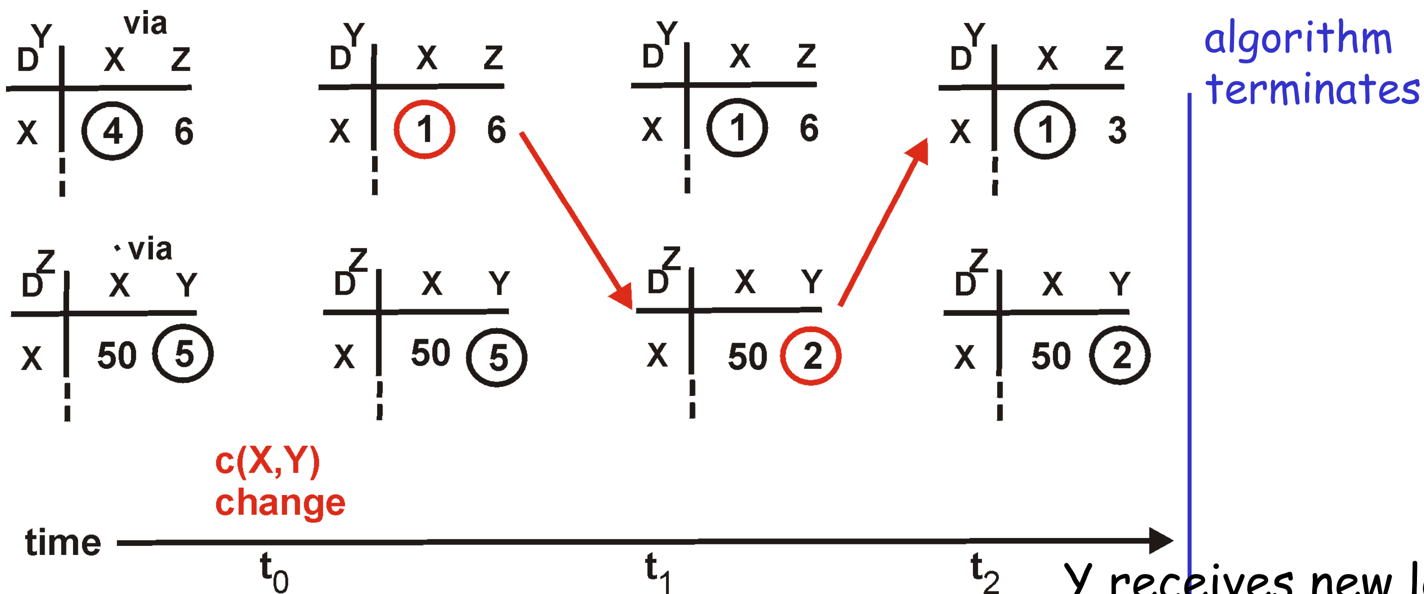
Subtitle: Distributed Bellman Ford converges *but* how fast?

Link cost changes:

- node detects local link cost change
- updates distance table (line 15)
- if cost change in least cost path, notify neighbors (lines 23,24)



"good news travels fast"



Ydetects change informs neighbors

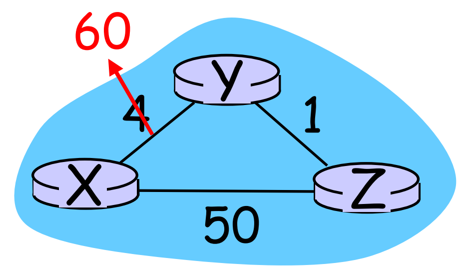
Z receives update ppdates table, new least Cost→informs neighbors

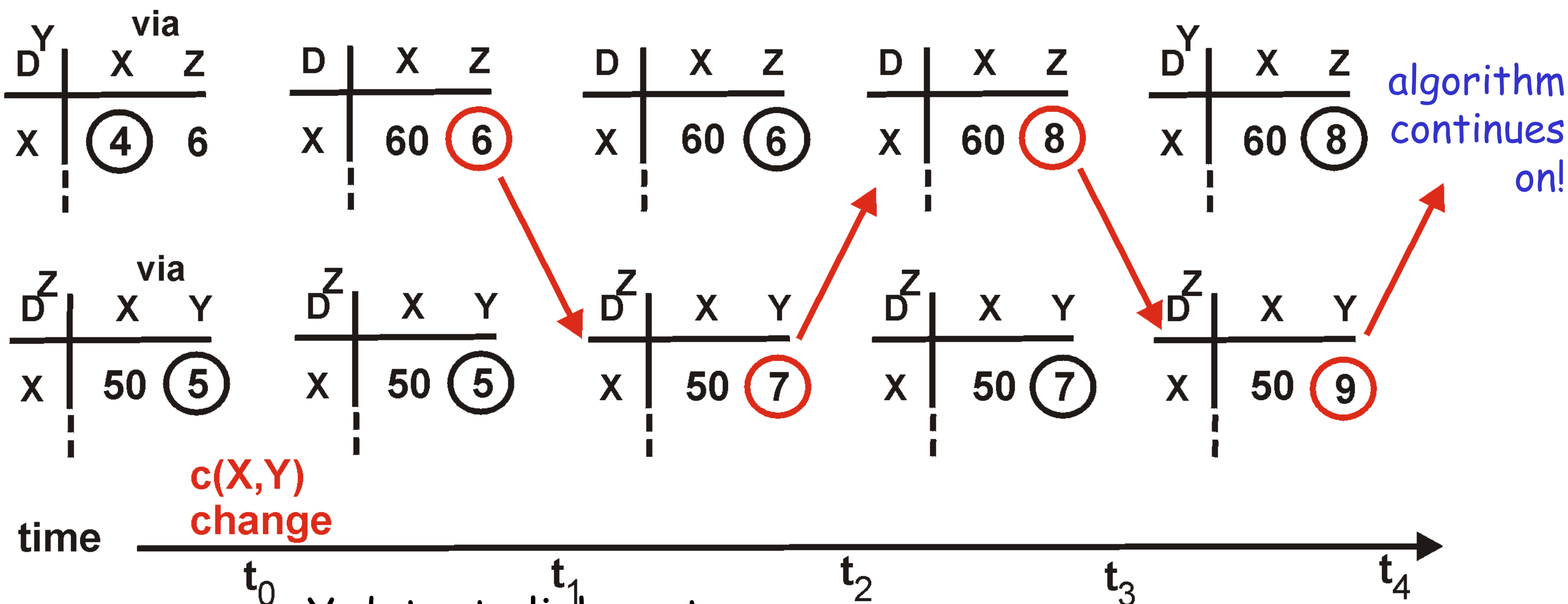
Y receives new least cost; no modification in the routing table, No updates sent

Distance Vector: link cost changes

Link cost changes:

- good news travels fast
- *bad news travels slow* - "count to infinity" problem!





Y detects link cost Increase but think can Reach X through Z at a total cost of 6 (wrong!!)

The path is Y-Z-Y-X

Count-to-infinity –an everyday life example

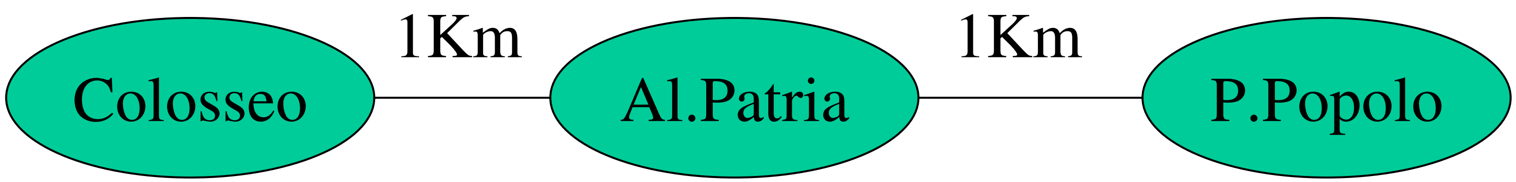
Which is the problem here?

the info exchanged by the protocol!! 'the best route to X I have has the following cost…' (no additional info on the route)

A Roman example…

-assumption: there is only one route going from Colosseo to Altare della Patria: Via dei Fori Imperiali. Let us now consider a network, whose nodes are Colosseo., Altare della Patria, Piazza del Popolo

Count-to-infinity –everyday life example (2/2)



The Colosseo. and Alt. Patria nodes exchange the following info

• Colosseo says 'the shortest route from me to P. Popolo is 2 Km'

• Alt. Patria says 'the shortest path from me to P. Popolo is 1Km'

Based on this exchange from Colosseo you go to Al. Patria, and from there to Piazza del Popolo OK **Now due to the big dig they close Via del Corso (Al. Patria—P.Popolo)**

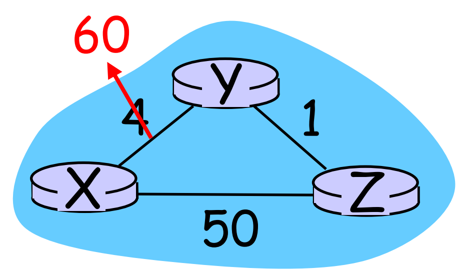
• Al. Patria thinks 'I have to find another route from me to P.Popolo. Look there is a route from Colosseo to P.Popolo that takes 2Km, I can be at Colosseo in 1Km → I have found a 3Km route from me to P.Popolo!!' Communicates the new cost to Colosseo that updates 'OK I can go to P.Popolo via Al. Patria in 4Km'

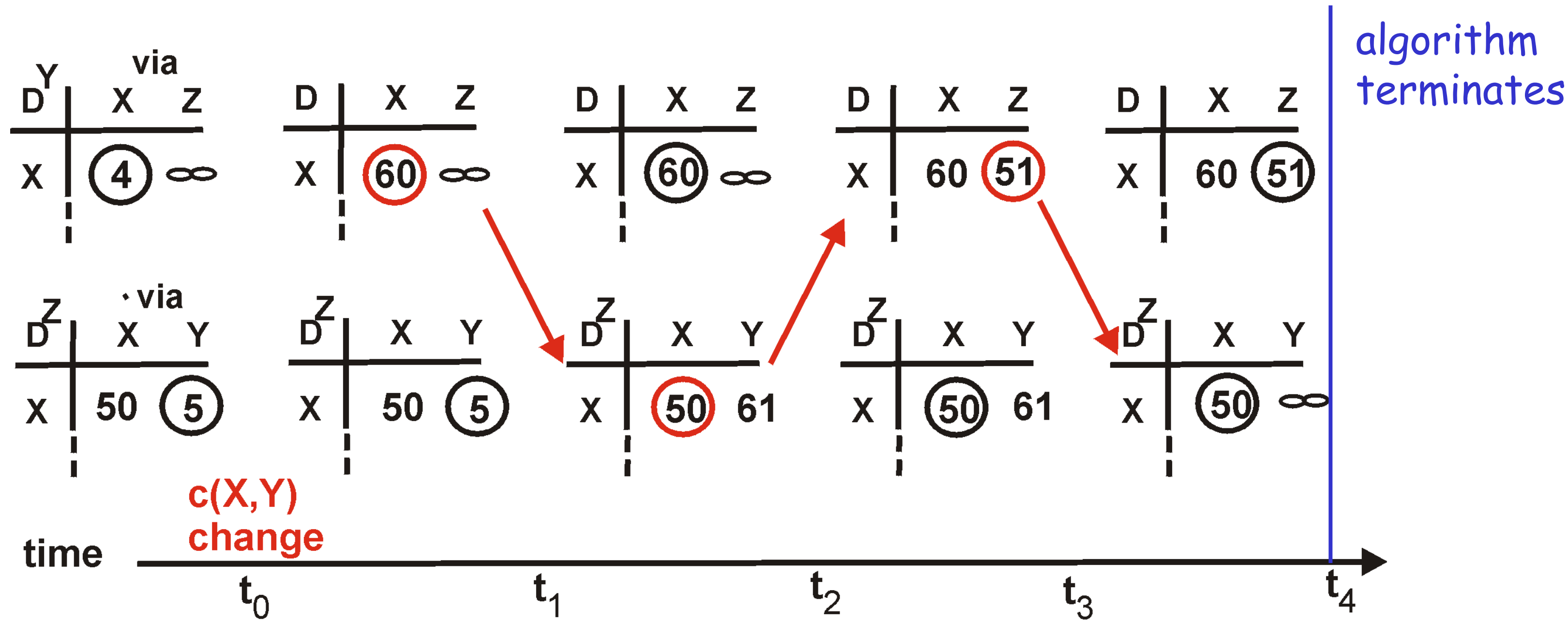
VERY WRONG!! Why is it so? I didn't know that the route from Colosseo to P.Popolo was going through Via del Corso from Al.Patria to P.Popolo (which is closed)!!

Distance Vector: poisoned reverse

If Z routes through Y to get to X :

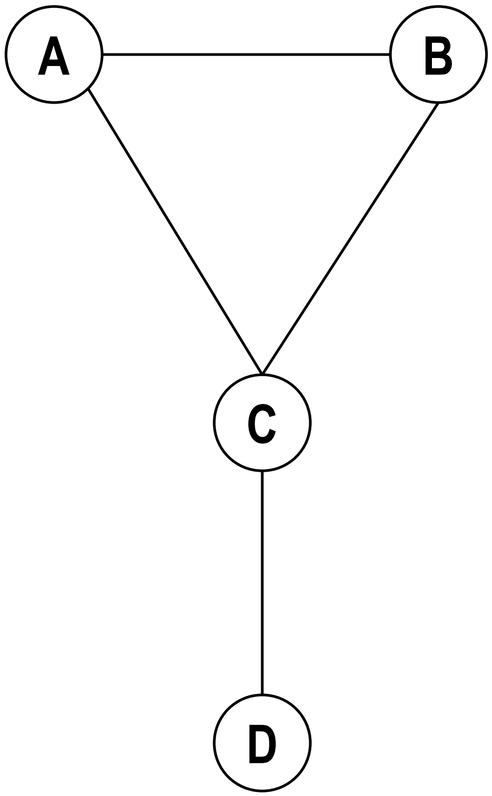
- Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- will this completely solve count to infinity problem?





Infinity is advertized by Y (poisoned reverse)

Split horizon poison reverse failure



Line CD goes down…

1) because of split horizon rule,
   A and B tell C that dist(D)=inf
2) C concludes that D is unreachable
   and reports this to A and B
3) but A knows from B that dist(D)=2, and
   sets its dist=3
4) similarly, B knows from A distance from D…

C estimates new value 4; A and B again through C estimate a value of 5….then again 1)
… etc until distance = infinite

Regardless the hack used, there is always a network topology that makes the trick fail!

Comparison of LS and DV algorithms

Message complexity

- LS: global exchange of information
- DV: exchange between neighbors only
  - convergence time varies

Speed of Convergence

- LS: $O(n^2)$ algorithm
  - 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

Chapter 4: Network Layer

- 4. 1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What's inside a router
- 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 Broadcast and multicast routing

Hierarchical Routing

Our routing study thus far - idealization
- all routers identical
- network "flat"

… *not* true in practice

scale: with 200 million destinations:
- can't store all dest's in routing tables!
- routing table exchange would swamp links!

administrative autonomy
- internet = network of networks
- each network admin may want to control routing in its own network

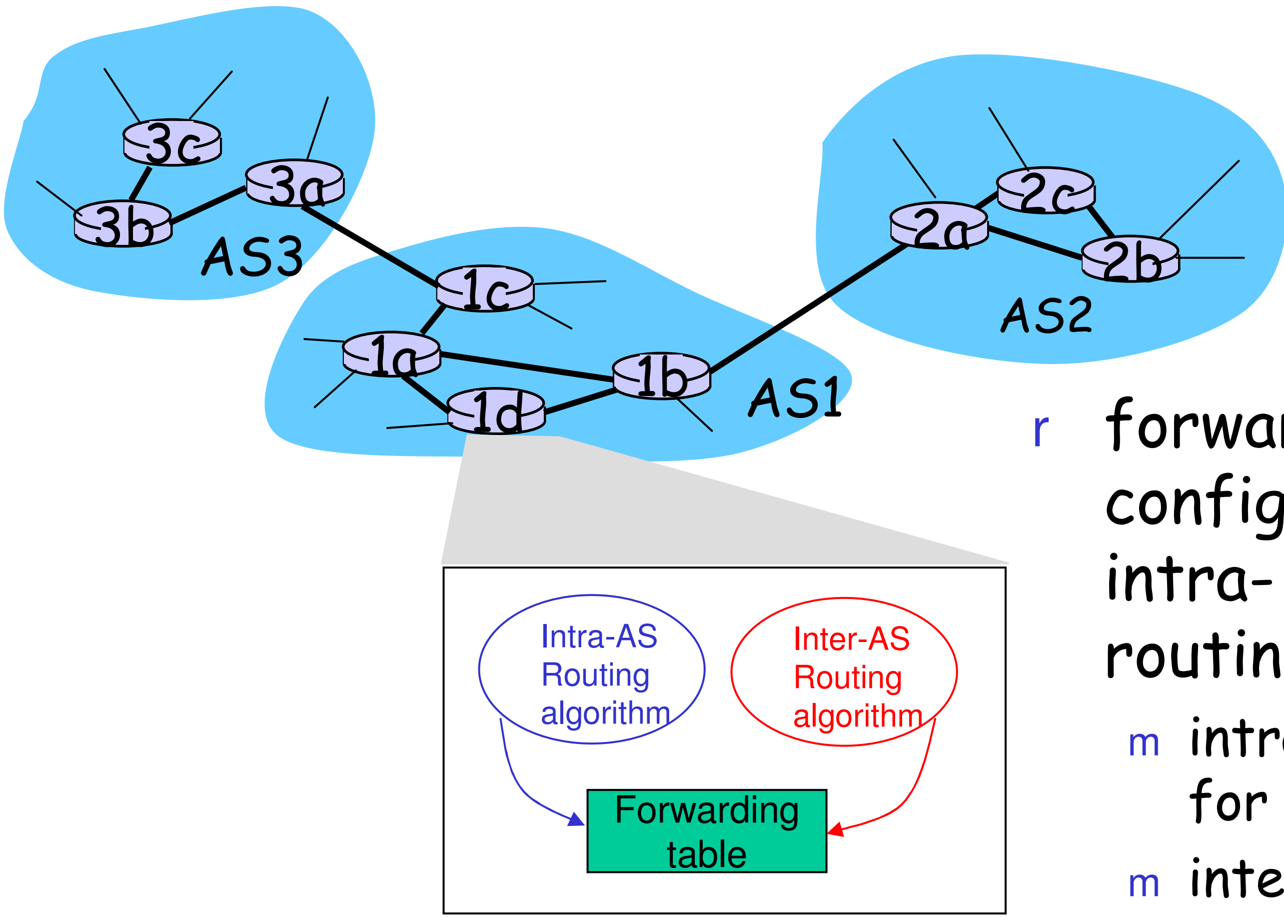
Hierarchical Routing

- aggregate routers into regions, "autonomous systems" (AS)
- routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol

Gateway router

- Direct link to router in another AS

Interconnected ASes



- forwarding table configured by both intra- and inter-AS routing algorithm
  - intra-AS sets entries for internal dests
  - inter-AS & intra-As sets entries for external dests

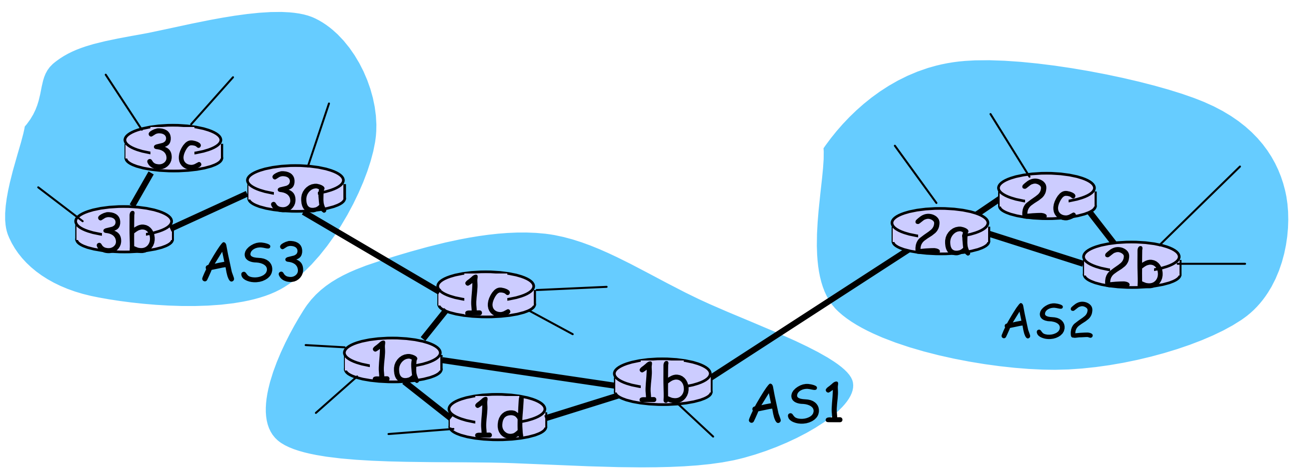
Inter-AS tasks

- suppose router in AS1 receives datagram destined outside of AS1:
  - router should forward packet to gateway router, but which one?

AS1 must:

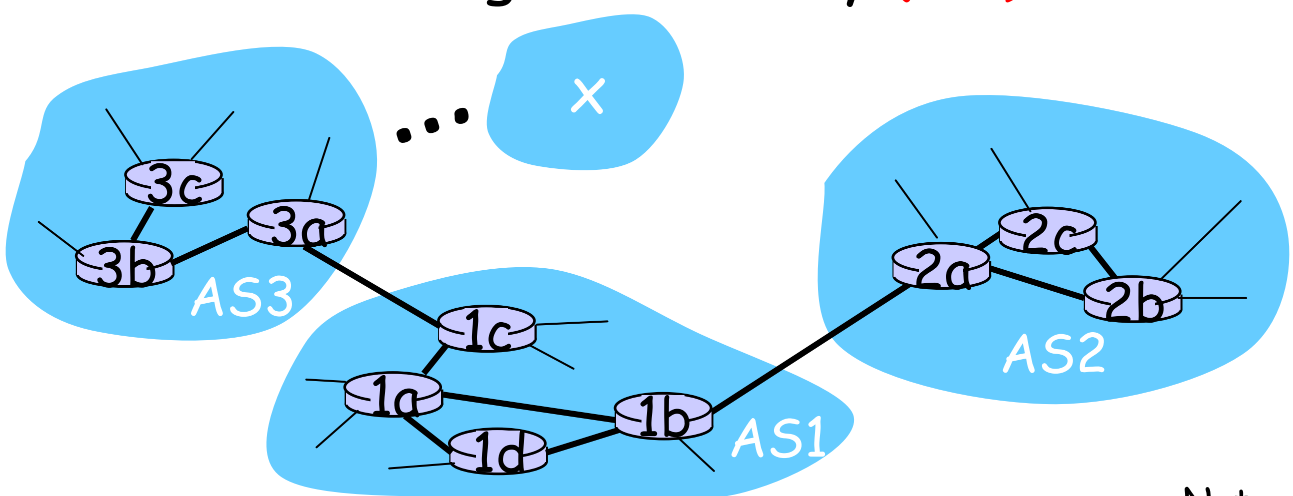
1. learn which dests are reachable through AS2, which through AS3
2. propagate this reachability info to all routers in AS1

Job of inter-AS routing!



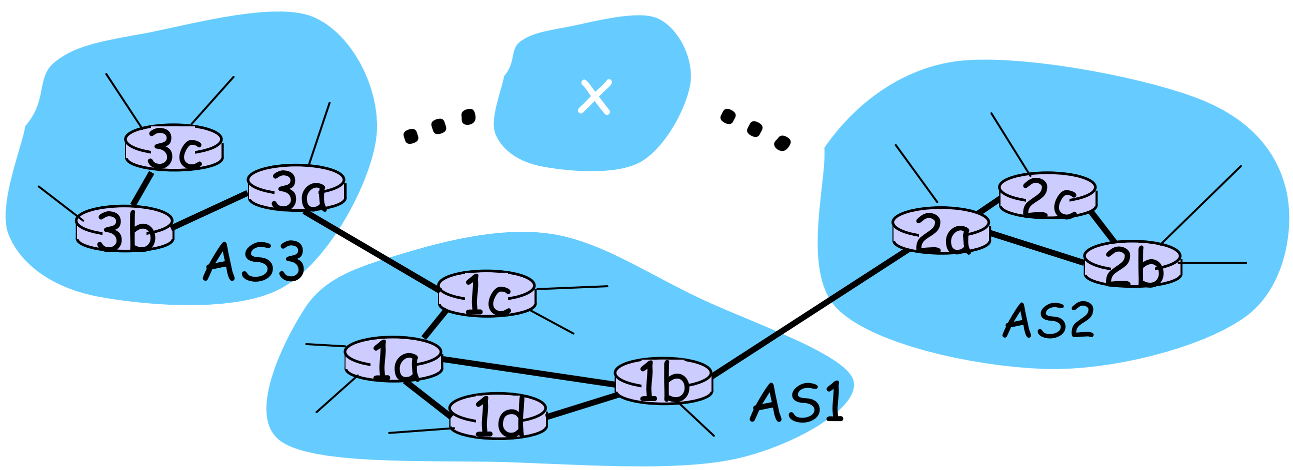
Example: Setting forwarding table in router 1d

- suppose AS1 learns (via inter-AS protocol) that subnet *x* is reachable via AS3 (gateway 1c) but not via AS2.
- inter-AS protocol propagates reachability info to all internal routers.
- router 1d determines from intra-AS routing info that its interface *I* is on the least cost path to 1c.
  - installs forwarding table entry *(x,I)*



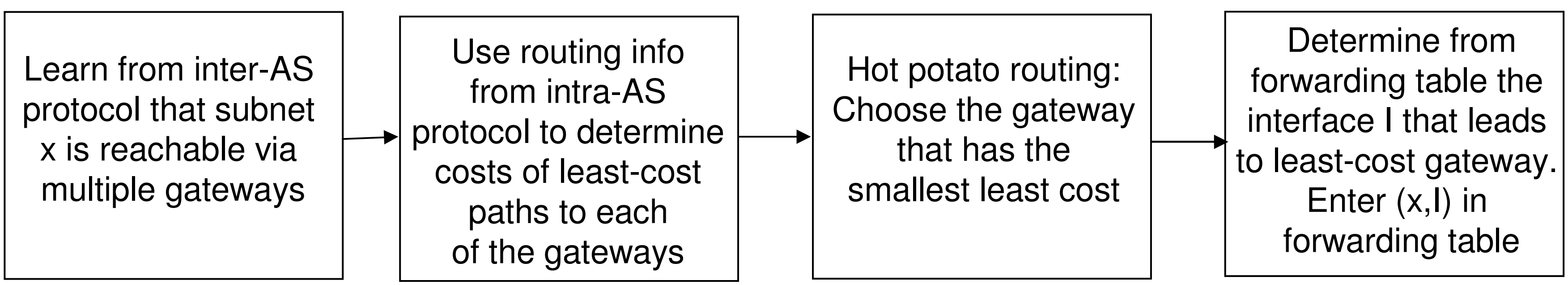
Example: Choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- to configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest x.
  - this is also job of inter-AS routing protocol!



Example: Choosing among multiple ASes

- now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- to configure forwarding table, router 1d must determine towards which gateway it should forward packets for dest x.
    - this is also job of inter-AS routing protocol!
- hot potato routing: send packet towards closest of two routers.



Chapter 4: Network Layer

- 4. 1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What's inside a router
- 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 Broadcast and multicast routing

Intra-AS Routing

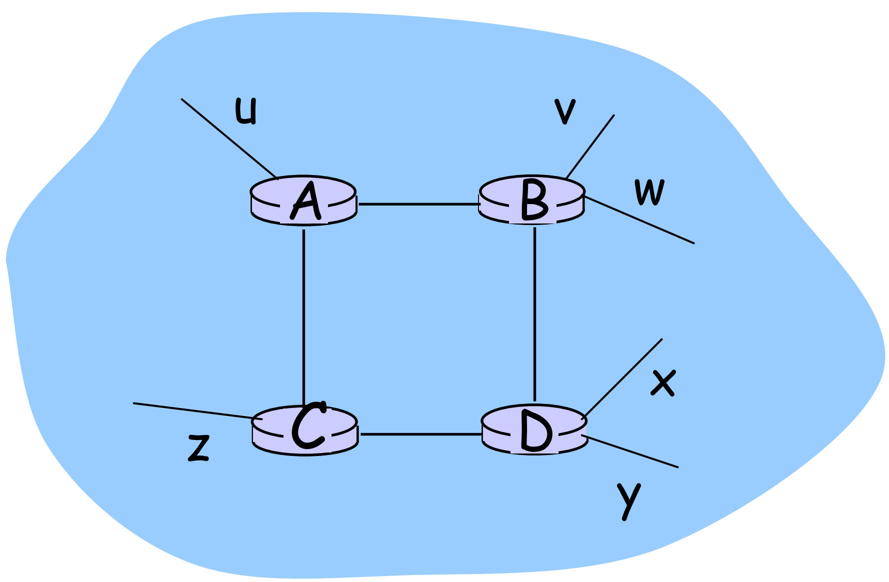
- also known as Interior Gateway Protocols (IGP)
- most common Intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

Chapter 4: Network Layer

- 4. 1 Introduction
- 4.2 Virtual circuit and datagram networks
- 4.3 What's inside a router
- 4.4 IP: Internet Protocol
  - Datagram format
  - IPv4 addressing
  - ICMP
  - IPv6
- 4.5 Routing algorithms
  - Link state
  - Distance Vector
  - Hierarchical routing
- 4.6 Routing in the Internet
  - RIP
  - OSPF
  - BGP
- 4.7 Broadcast and multicast routing

RIP (Routing Information Protocol)

- distance vector algorithm
- included in BSD-UNIX Distribution in 1982
- distance metric: # of hops (max = 15 hops)



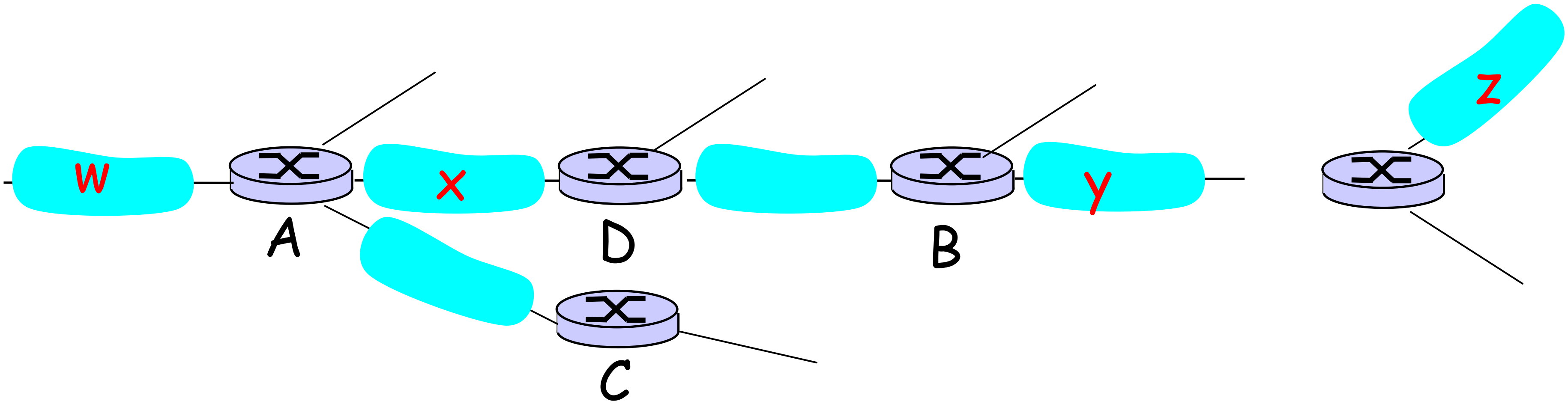
From router A to subnets:

destination	hops
u	1
v	2
w	2
x	3
y	3
z	2

RIP advertisements

- *distance vectors:* exchanged among neighbors every 30 sec via Response Message (also called advertisement)
- each advertisement: list of up to 25 destination subnets within AS

RIP: Example



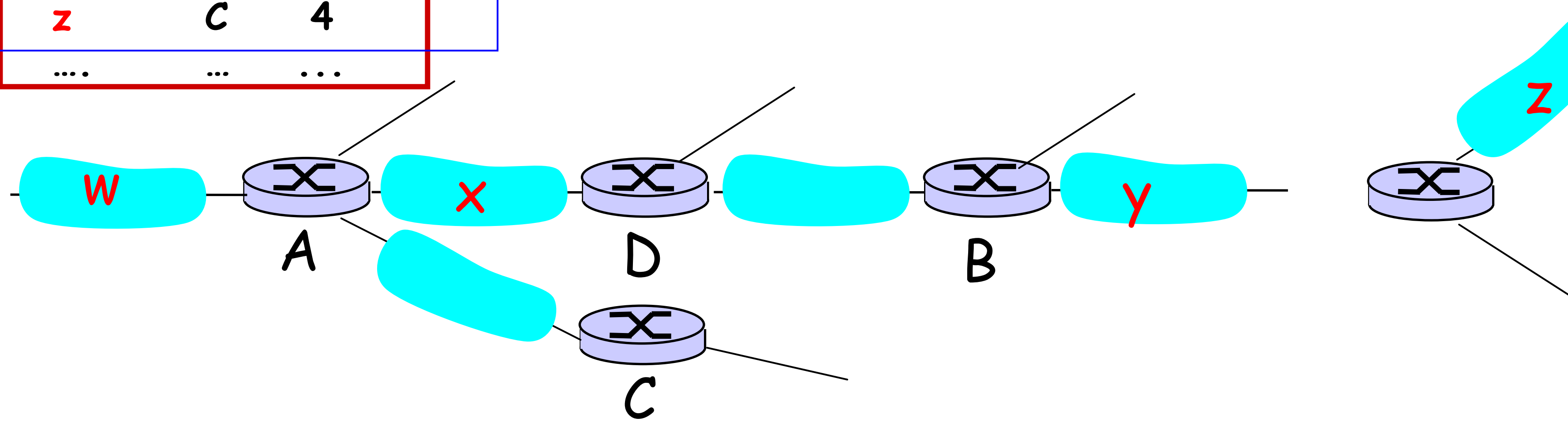
Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	B	7
x	--	1
….	….	….

Routing/Forwarding table in D

RIP: Example

Dest	Next	hops
w	-	1
x	-	1
z	C	4
….	…	...

Advertisement from A to D



Destination Network	Next Router	Num. of hops to dest.
w	A	2
y	B	2
z	~~B~~ A	~~7~~ 5
x	--	1
….	….	....

Routing/Forwarding table in D

Differences wrt Bellman Ford

- Since count to infinity not solved upper bound on the network size
- Info on the cost of going through destination X through neighbor Z is maintained ONLY IF the path through Z is the current "best" (min cost) path
  - Different way of updating costs
    - Suppose current route to dest has cost D and goes through G
    - if an update arrives from X!=G then updates route ONLY IF cost is <D
    - if an update arrives from G always update route cost
    - PROBLEM: what if the router we go through crashes?
      - Route cost aging MUST be adopted
- Cost is maintained for each subnetwork (rather than node)
- Periodic exchange of messages

The RIP algorithm (from RFC)

- Keep a table with an entry for every possible destination in the system. The entry contains the distance D to the destination, and the first router G on the route to that network. Conceptually, there should be an entry for the entity itself, with metric 0, but this is not actually included.

- Periodically, send a routing update to every neighbor. The update is a set of messages that contain all of the information from the routing table. It contains an entry for each destination, with the distance shown to that destination.

- When a routing update arrives from a neighbor G', add the cost associated with the network that is shared with G'. (This should be the network over which the update arrived.) Call the resulting distance D'. Compare the resulting distances with the current routing table entries. If the new distance D' for N is smaller than the existing value D, adopt the new route. That is, change the table entry for N to have metric D' and router G'. If G' is the router from which the existing route came, i.e., G' = G, then use the new metric even if it is larger than the old one.

RIP: Link Failure and Recovery

If no advertisement heard after 180 sec --> neighbor/link declared dead

- routes via neighbor invalidated
- new advertisements sent to neighbors
- neighbors in turn send out new advertisements (if tables changed)
- link failure info quickly (?) propagates to entire net
- *poison reverse* used to prevent ping-pong loops (infinite distance = 16 hops)

Differences with Bellman Ford

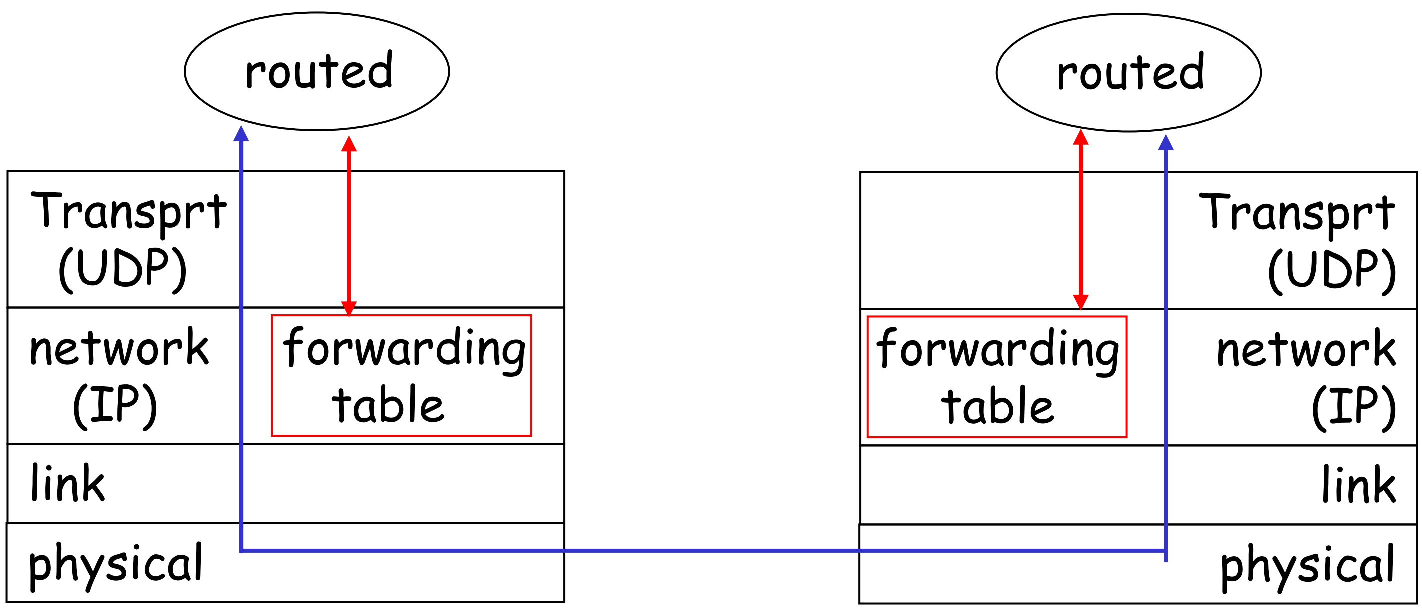
- Split Horizon
  - with Poison Reverse
  - Simple split horizon (omits cost of reaching destination when advertising through the router it goes through)
- Clear implementation with point to point links. But consider the possibility that A and C are connected by a broadcast network such as an Ethernet, and there are other routers on that network. Is it a problem?
  - If A has a route through C, it should indicate that D is unreachable when talking to any other router on that network. The other routers on the network can get to C themselves. They would never need to get to C via A.
  - If A's best route is really through C, no other router on that network needs to know that A can reach D. This is fortunate, because it means that the same update message that is used for C can be used for all other routers on the same network. Thus, update messages can be sent by broadcast.

An additional way to speed up convergence

- Triggered updates
  - Whenever a router changes the metric for a route it is required to send update messages almost immediately
    - must be implemented for deleted routes

RIP Table processing

- RIP routing tables managed by **application-level** process called route-d (daemon)
  - port number 520
- advertisements sent in UDP packets, periodically repeated



Packet format

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  command (1)  |  version (1)  |       must be zero (2)        |
+---------------+---------------+-------------------------------+
|                                                               |
~                         RIP Entry (20)                        ~
|                                                               |
+---------------+---------------+---------------+---------------+

 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| address family identifier (2) |      must be zero (2)        |
+-------------------------------+-------------------------------+
|                        IPv4 address (4)                       |
+---------------------------------------------------------------+
|                        must be zero (4)                       |
+---------------------------------------------------------------+
|                        must be zero (4)                       |
+---------------------------------------------------------------+
|                           metric (4)                          |
+---------------------------------------------------------------+
```