CS 356: Computer Network Architectures

Lecture 12: Dynamic Routing: Routing Information Protocol

Chap. 3.3.1, 3.3.2

Xiaowei Yang
xwy@cs.duke.edu

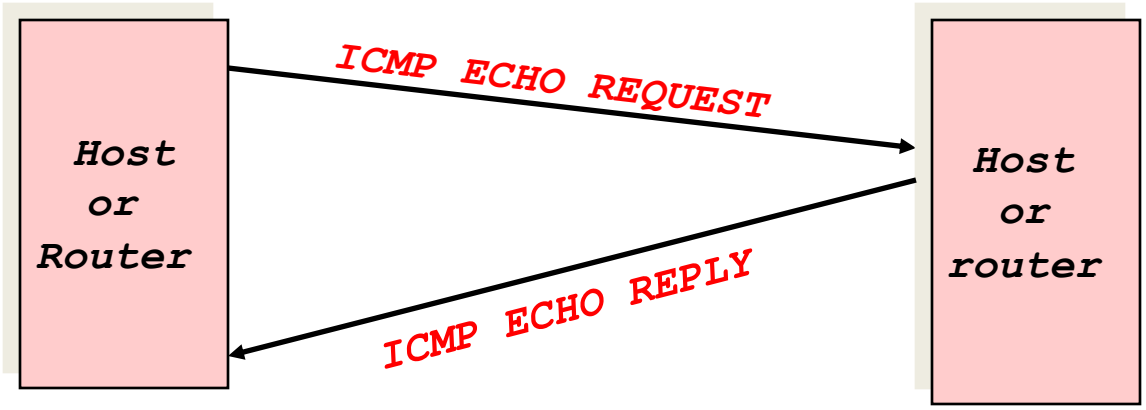
Today

- ICMP applications
- Dynamic Routing
  - Routing Information Protocol

ICMP applications

- Ping
  - ping www.duke.edu
- Traceroute
  - traceroute nytimes.com
- MTU discovery

Ping: Echo Request and Reply



Type (= 8 or 0)	Code (=0)	Checksum
identifier		sequence number
Optional data		

- Ping's are handled directly by the kernel
- Each Ping is translated into an ICMP Echo Request
- The Ping'ed host responds with an ICMP Echo Reply

Traceroute

- xwy@linux20$ traceroute -n 18.26.0.1
  - traceroute to 18.26.0.1 (18.26.0.1), 30 hops max, 60 byte packets
  - 1 152.3.141.250 4.968 ms 4.990 ms 5.058 ms
  - 2 152.3.234.195 1.479 ms 1.549 ms 1.615 ms
  - 3 152.3.234.196 1.157 ms 1.171 ms 1.238 ms
  - 4 128.109.70.13 1.905 ms 1.885 ms 1.943 ms
  - 5 128.109.70.138 4.011 ms 3.993 ms 4.045 ms
  - 6 128.109.70.102 10.551 ms 10.118 ms 10.079 ms
  - 7 18.3.3.1 28.715 ms 28.691 ms 28.619 ms
  - 8 18.168.0.23 27.945 ms 28.028 ms 28.080 ms
  - 9 18.4.7.65 28.037 ms 27.969 ms 27.966 ms
  - 10 128.30.0.246 27.941 ms * *

Traceroute algorithm

- Sends out three UDP packets with TTL=1,2,…,n, destined to a high port
- Routers on the path send ICMP Time exceeded message with their IP addresses until n reaches the destination distance
- Destination replies with port unreachable ICMP messages

Path MTU discovery algorithm

- Send packets with DF bit set
- If receive an ICMP error message, reduce the packet size

Today

- ICMP applications
- Dynamic Routing
  - Routing Information Protocol

Dynamic Routing

- There are two parts related to IP packet handling:
  1. Forwarding
  2. Routing: distributed computation

Static versus Dynamic routing

- Two approaches:
  - Static Routing (Lab 2)
  - Dynamic Routing
    - Routes are calculated by a routing protocol
    - Graph algorithms
  - Why do we need a distributed protocol to setup routing tables?

Static routing

- Setting up host routing tables
  - Route add

- xwy@linux20$ netstat -nr
- Kernel IP routing table

```
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
152.3.140.0     0.0.0.0         255.255.254.0   U         0 0          0 eth0
0.0.0.0         152.3.140.61    0.0.0.0         UG        0 0          0 eth0
```

- If a destination has the same network number as the host, send directly to the destination; otherwise, send to default router

Protocols versus algorithms

- Routing protocols establish forwarding tables at routers

- A routing protocol specifies
  - What messages are sent
  - When are they sent
  - How are they handled

- At the heart of any routing protocol is a distributed algorithm that determines the path from a source to a destination

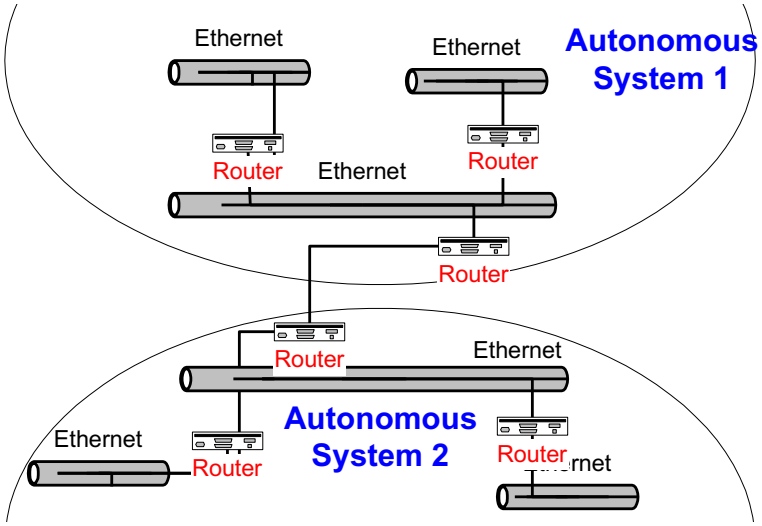
What distributed routing algorithms common routing protocols use

Routing protocol	Distributed algorithm
Routing information protocol (RIP)	Distance vector

Intra-domain routing versus inter-domain routing

- The Internet is a network of networks
- Administrative autonomy
  - internet = network of networks
  - each network admin may want to control routing in its own network
- Scale: with 200 million destinations:
  - can't store all destinations in routing tables!
  - routing table exchange would swamp links
  - Solution: using hierarchy to scale

Autonomous systems



- Aggregate routers into regions, "autonomous systems" (AS) or domain
- Routers in the same AS run the same routing protocol
  - "intra-AS" or intra-domain routing protocol
  - routers in different AS can run different intra-AS routing protocol

Autonomous Systems

- An **autonomous system** is a region of the Internet that is administered by a single entity

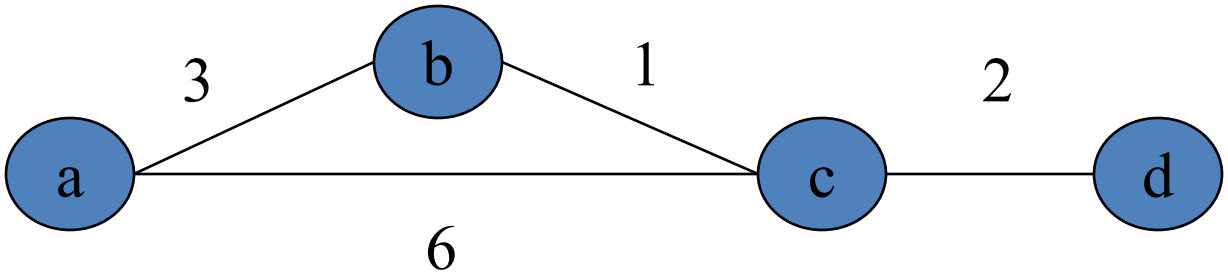
- Examples of autonomous regions are:
  - Duke's campus network
  - at&t's backbone network
  - Regional Internet Service Provider (NC regional)

- **intradomain routing**
- **interdomain routing**

- RIP, OSPF, IGRP, and IS-IS are intra-domain routing protocols

- BGP is the only inter-domain routing protocol

RIP and OSPF computes shortest paths



- Shortest path routing algorithms
  - **Goal**: Given a network where each link is assigned a cost. Find the path with the least cost between two nodes
  - Shortest path routing is provably loop-free
    - Why?

Distance vector algorithm

- A decentralized algorithm
  - Each node has a partial view
    - Neighbors
    - Link costs to neighbors
- Distance vector
- Path computation is iterative and mutually dependent
  1. A router sends its known distances to each destination (distance vector) to its neighbors
  2. A router updates the distance to a destination from all its neighbors' distance vectors
  3. A router sends its updated distance vector to its neighbors
  4. The process repeats until all routers' distance vectors do not change (this condition is called convergence).

A router updates its distance vectors using bellman-ford equation

Bellman-Ford Equation

Define

$d_x(y)$:= cost of the least-cost path from x to y

Then

- $d_x(y) = \min_v\{c(x,v) + d_v(y)\}$, where min is taken over all neighbors of node x

Distance vector algorithm: initialization

- Let $D_x(y)$ be the estimate of least cost from x to y

- Initialization:
  - Each node x knows the cost to each neighbor: c(x,v). For each neighbor v of x, $D_x(v) = c(x,v)$
  - $D_x(y)$ to other nodes are initialized as infinity

- Each node x maintains a distance vector (DV):
  - $\mathbf{D}_x = [D_x(y): y \in N]$

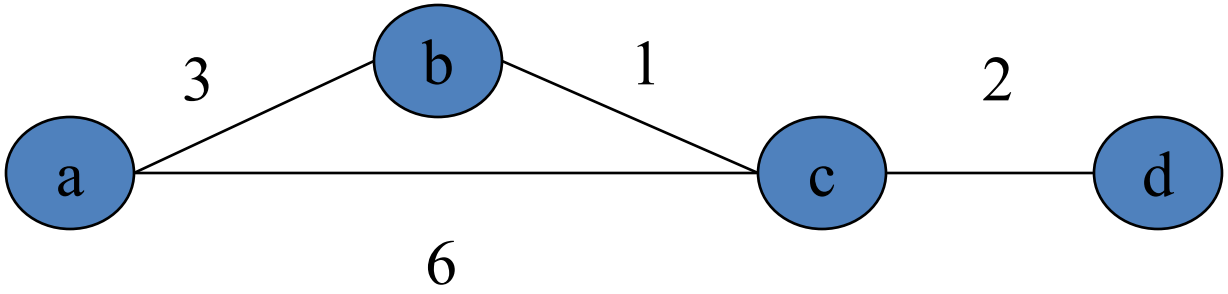
Distance vector algorithm: updates

- Each node x sends its distance vector to its neighbors, either periodically, or triggered by a change in its DV

- When a node x receives a new DV estimate from a neighbor v, it updates its own DV using the B-F equation:
  - If $c(x,v) + D_v(y) < D_x(y)$ then
    - $D_x(y) = c(x,v) + D_v(y)$
    - Sets the next hop to reach the destination y to the neighbor v
    - Notify neighbors of the change

- The estimate $D_x(y)$ *will converge to the actual least cost* $d_x(y)$

Distance vector algorithm: an example

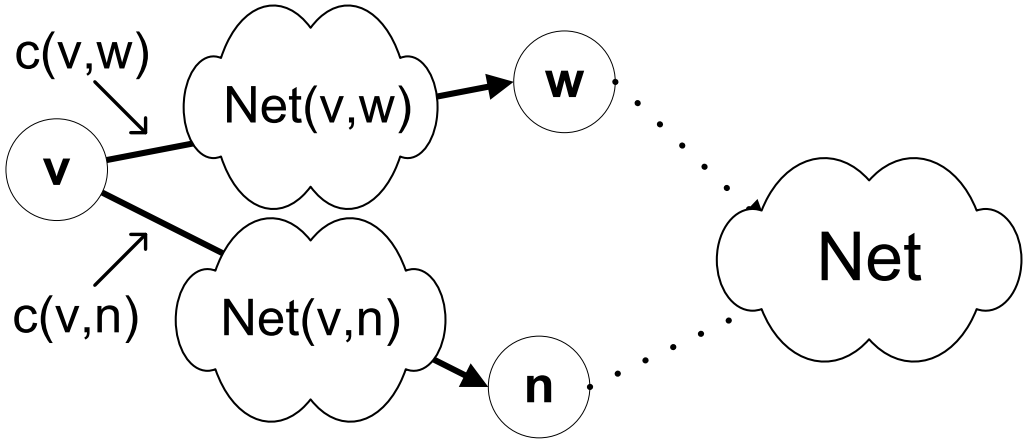


- t = 0
- a = ((a, 0), (b, 3), (c, 6))
- b = ((a, 3), (b, 0), (c,1))
- c = ((a, 6), (b, 1), (c, 0) (d, 2))
- d = ((c, 2), (d, 0))

- t = 1
- a = ((a, 0), (b, 3), (c, 4), (d, 8))
- b = ((a, 3), (b, 0), (c,1), (d, 3))
- c = ((a, 4), (b, 1), (c, 0), (d, 2))
- d = ((a, 8), (b, 3), (c, 2), (d,0))

- t = 2
- a = ((a, 0), (b, 3), (c, 4), (d, 6))
- b = ((a, 3), (b, 0), (c,1), (d, 3))
- c = ((a, 4), (b, 1), (c, 0), (d, 2))
- d = ((a, 6), (b, 3), (c, 2), (d,0))

Mapping an abstract graph to the physical network

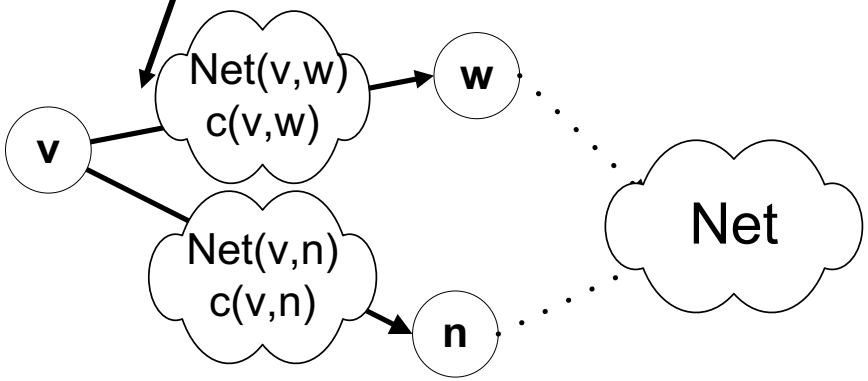


- Nodes (e.g., v, w, n) are routers, identified by IP addresses, e.g. 10.0.0.1
- Nodes are connected by either a directed link or a broadcast link (Ethernet)
- Destinations are IP networks, represented by the network prefixes, e.g., 10.0.0.0/16
  - Net(v,n) is the network directly connected to router v and n.
- Costs (e.g. c(v,n)) are associated with network interfaces.
  - Router1(config)# router rip
  - Router1(config-router)# offset-list 0 out 10 Ethernet0/0
  - Router1(config-router)# offset-list 0 out 10 Ethernet0/1

Distance vector routing protocol: Routing Table

c(v,w): cost to transmit on the interface to network Net(v,w)

Net(v,w): Network address of the network between v and w



RoutingTable of

Dest	via (next hop)	
Net	n	D(v,Net)

D(v,net) is v's cost to Net

Distance vector routing protocol: Messages

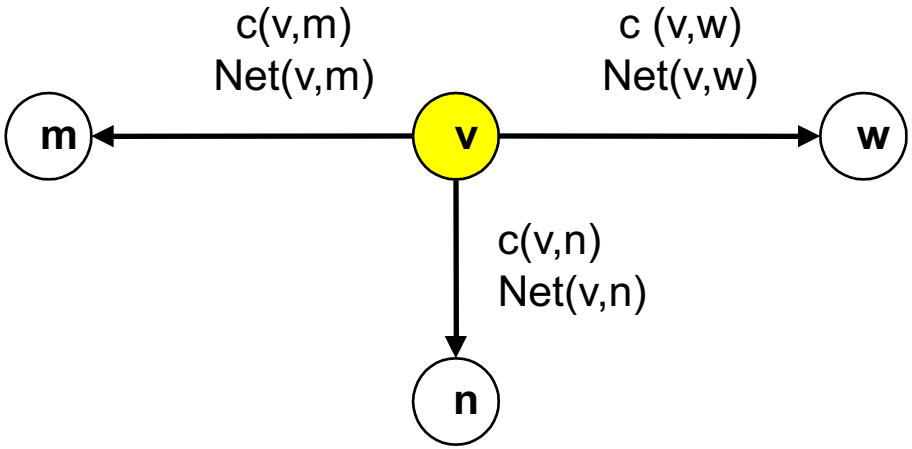
RoutingTable of node v

Dest	via (next hop)	cost
Net	n	D(v,Net)

- Nodes send messages to their neighbors which contain distance vectors
- A message has the format: **[Net , D(v,Net)]** means ***"My cost to go to Net is D (v,Net)"***

Initiating Routing Table I

- Suppose a new node v becomes active
- The cost to access directly connected networks is zero:
  - D (v, Net(v,m)) = 0
  - D (v, Net(v,w)) = 0
  - D (v, Net(v,n)) = 0

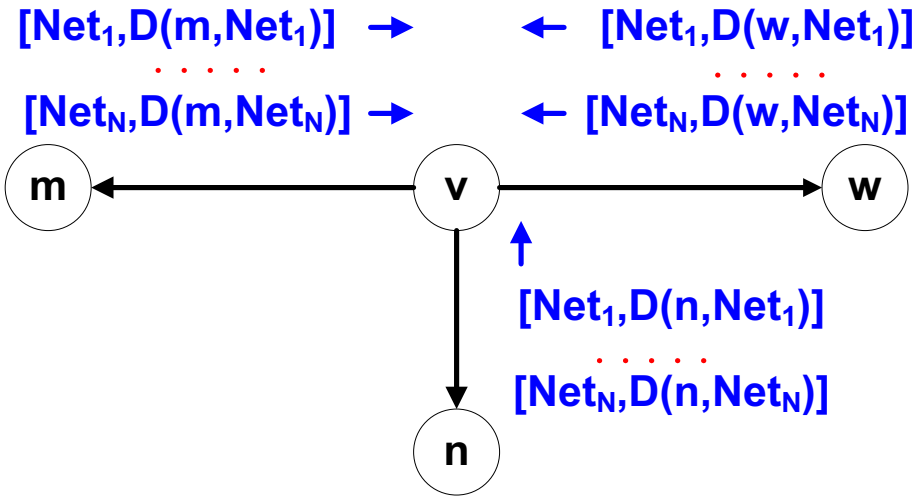


RoutingTable

Dest	via (next hop)	cost
Net(v,m)	-	0
Net(v,w)	-	0
Net(v,n)	-	0

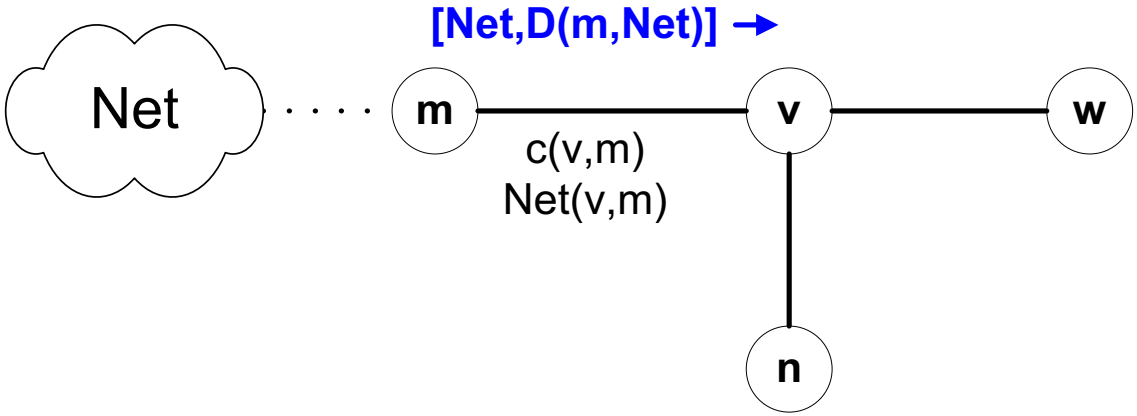
Initiating Routing Table III

- Node v receives the routing tables from other nodes and builds up its routing table



Updating Routing Tables I

- Suppose node v receives a message from node m: **[Net,D(m,Net)]**

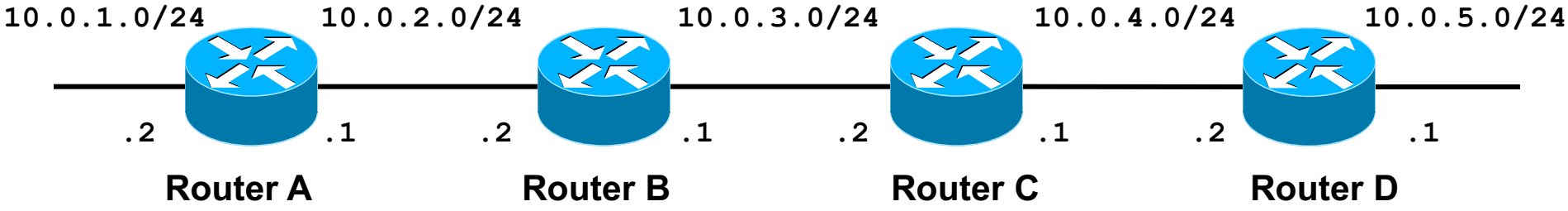


Node v updates its routing table and sends out further messages if the message reduces the cost of a route:

```
if  ( D(m,Net) + c (v,m) < D (v,Net) ) {
      D^new (v,Net) := D (m,Net) + c (v,m);
      Update routing table;
      send message [Net, D^new (v,Net)] to all neighbors
}
```

Example

Assume: - link cost is 1, i.e., c(v,w) = 1
- all updates, updates occur simultaneously
- Initially, each router only knows the cost of connected interfaces



Router A

Net	via	cost
t=0:		
10.0.1.0	-	0
10.0.2.0	-	0
t=1:		
10.0.1.0	-	0
10.0.2.0	-	0
10.0.3.0	10.0.2.2	1
t=2:		
10.0.1.0	-	0
10.0.2.0	-	0
10.0.3.0	10.0.2.2	1
10.0.4.0	10.0.2.2	2

Router B

Net	via	cost
t=0:		
10.0.2.0	-	0
10.0.3.0	-	0
t=1:		
10.0.1.0	10.0.2.1	1
10.0.2.0	-	0
10.0.3.0	-	0
10.0.4.0	10.0.3.2	1
t=2:		
10.0.1.0	10.0.2.1	1
10.0.2.0	-	0
10.0.3.0	-	0
10.0.4.0	10.0.3.2	1
10.0.5.0	10.0.3.2	2

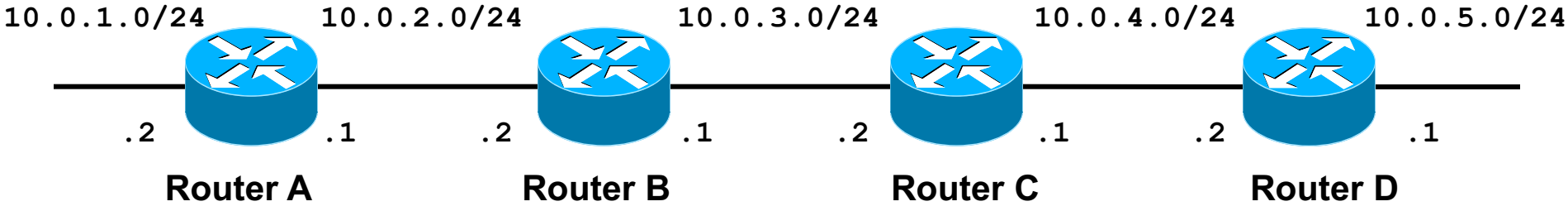
Router C

Net	via	cost
t=0:		
10.0.3.0	-	0
10.0.4.0	-	0
t=1:		
10.0.2.0	10.0.3.1	1
10.0.3.0	-	0
10.0.4.0	-	0
10.0.5.0	10.0.4.2	1
t=2:		
10.0.1.0	10.0.3.1	2
10.0.2.0	10.0.3.1	1
10.0.3.0	-	0
10.0.4.0	-	0
10.0.5.0	10.0.4.2	1

Router D

Net	via	cost
t=0:		
10.0.4.0	-	0
10.0.5.0	-	0
t=1:		
10.0.3.0	10.0.4.1	1
10.0.4.0	-	0
10.0.5.0	-	0
t=2:		
10.0.2.0	10.0.4.1	2
10.0.3.0	10.0.4.1	1
10.0.4.0	-	0
10.0.5.0	-	0

Example



Net	via	cost
t=2:		
10.0.1.0	*–*	*0*
10.0.2.0	*–*	*0*
10.0.3.0	*10.0.2.2*	*1*
10.0.4.0	*10.0.2.2*	*2*
t=3:		
10.0.1.0	–	0
10.0.2.0	–	0
10.0.3.0	10.0.2.2	1
10.0.4.0	10.0.2.2	2
10.0.5.0	10.0.2.2	3

Net	via	cost
t=2:		
10.0.1.0	*10.0.2.1*	*1*
10.0.2.0	*–*	*0*
10.0.3.0	*–*	*0*
10.0.4.0	*10.0.3.2*	*1*
10.0.5.0	*10.0.3.2*	*2*
t=3:		
10.0.1.0	10.0.2.1	1
10.0.2.0	–	0
10.0.3.0	–	0
10.0.4.0	10.0.3.2	1
10.0.5.0	10.0.3.2	2

Net	via	cost
t=2:		
10.0.1.0	*10.0.3.1*	*2*
10.0.2.0	*10.0.3.1*	*1*
10.0.3.0	*–*	*0*
10.0.4.0	*–*	*0*
10.0.5.0	*10.0.4.2*	*1*
t=3:		
10.0.1.0	10.0.3.1	2
10.0.2.0	10.0.3.1	1
10.0.3.0	–	0
10.0.4.0	–	0
10.0.5.0	10.0.4.2	1

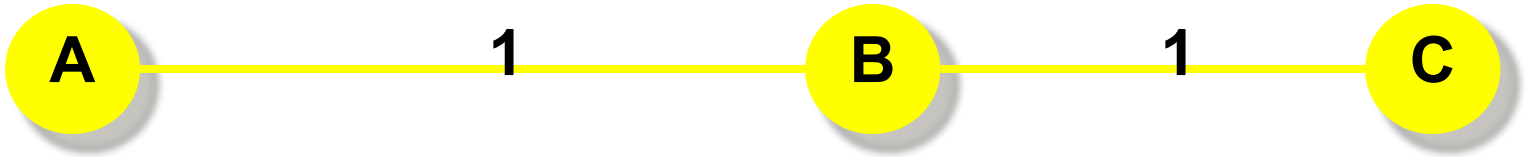
Net	via	cost
t=2:		
10.0.2.0	*10.0.4.1*	*2*
10.0.3.0	*10.0.4.1*	*1*
10.0.4.0	*–*	*0*
10.0.5.0	*–*	*0*
t=3:		
10.0.1.0	10.0.4.1	3
10.0.2.0	10.0.4.1	2
10.0.3.0	10.0.4.1	1
10.0.4.0	–	0
10.0.5.0	–	0

Now, routing tables have converged !

Characteristics of Distance Vector Routing Protocols

- **Periodic Updates:** Updates to the routing tables are sent at the end of a certain time period. A typical value is 30 seconds
- **Triggered Updates:** If a metric changes on a link, a router immediately sends out an update without waiting for the end of the update period
- **Full Routing Table Update**: Most distance vector routing protocol send their neighbors the entire routing table (not only entries which change)
- **Route invalidation timers:** Routing table entries are invalid if they are not refreshed. A typical value is to invalidate an entry if no update is received after 3-6 update periods.

The Count-to-Infinity Problem



A's Routing Table

to	via (next hop)	cost
C	B	2

B's Routing Table

to	via (next hop)	cost
C	C	1

now link B-C goes down

C	B	2

C	2

→

←

C	∞

C	-	∞

C	∞

→

←

C	3

C	A	3

C	B	4

C	4

→

←

C	∞

C	-	∞

Count-to-Infinity

- The reason for the count-to-infinity problem is that each node only has a “next-hop-view”
- For example, in the first step, A did not realize that its route (with cost 2) to C went through node B

- How can the Count-to-Infinity problem be solved?

Solutions to Count-to-Infinity

- The reason for the count-to-infinity problem is that each node only has a "next-hop-view"
- For example, in the first step, A did not realize that its route (with cost 2) to C went through node B
- How can the Count-to-Infinity problem be solved?
- **Solution 1:** Always advertise the entire path in an update message to avoid loops (**Path vectors**)
  - BGP uses this solution

Count-to-Infinity

- The reason for the count-to-infinity problem is that each node only has a "next-hop-view"

- For example, in the first step, A did not realize that its route (with cost 2) to C went through node B

- How can the Count-to-Infinity problem be solved?
- **Remedy 2:** Never advertise the cost to a neighbor if this neighbor is the next hop on the current path **(Split Horizon)**
  - Example: A would not send the first routing update to B, since B is the next hop on A's current route to C
  - Split horizon with poison reverse
    - Sends to the next hop neighbor an invalid route (C, ∞)
  - Only solve the problem if routing loops involve only two nodes
- Remedy 3: Has a small infinity (16) so that routing messages will not bounce forever

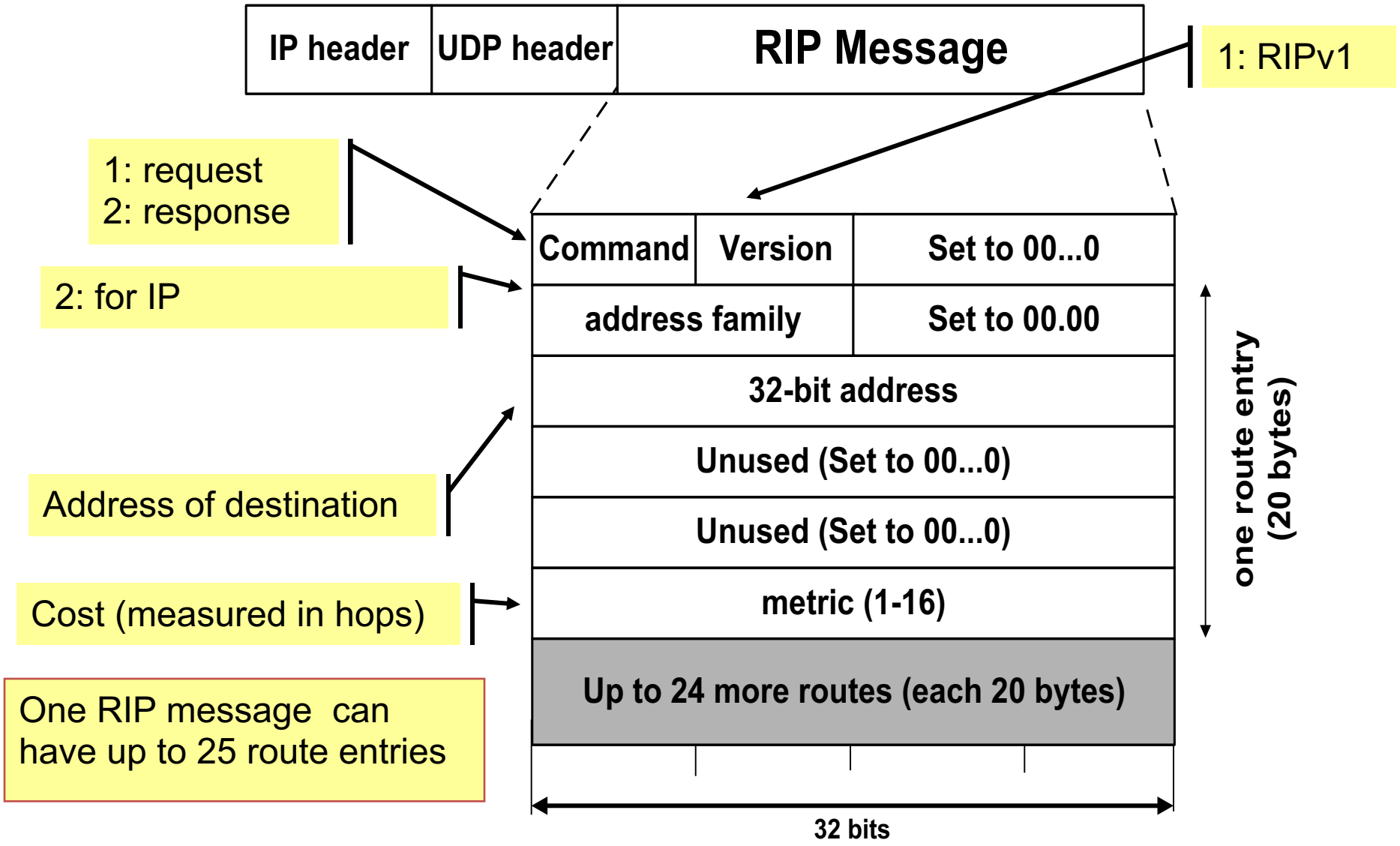
RIP - Routing Information Protocol

- A simple intra-domain protocol
- Straightforward implementation of Distance Vector Routing
- Each router advertises its distance vector every 30 seconds (or whenever its routing table changes) to all of its neighbors
- RIP always uses 1 as link metric
- Maximum hop count is 15, with “16” equal to “∞”
- Routes are timeout (set to 16) after 3 minutes if they are not updated

RIP - History

- Late 1960s : Distance Vector protocols were used in the ARPANET
- Mid-1970s: XNS (Xerox Network system) routing protocol is the ancestor of RIP in IP
- 1982 Release of **routed** for BSD Unix
- 1988 RIPv1 (RFC 1058)
  - classful routing
- 1993 RIPv2 (RFC 1388)
  - adds subnet masks with each route entry
  - allows classless routing
- 1998 Current version of RIPv2 (RFC 2453)

RIPv1 Packet Format

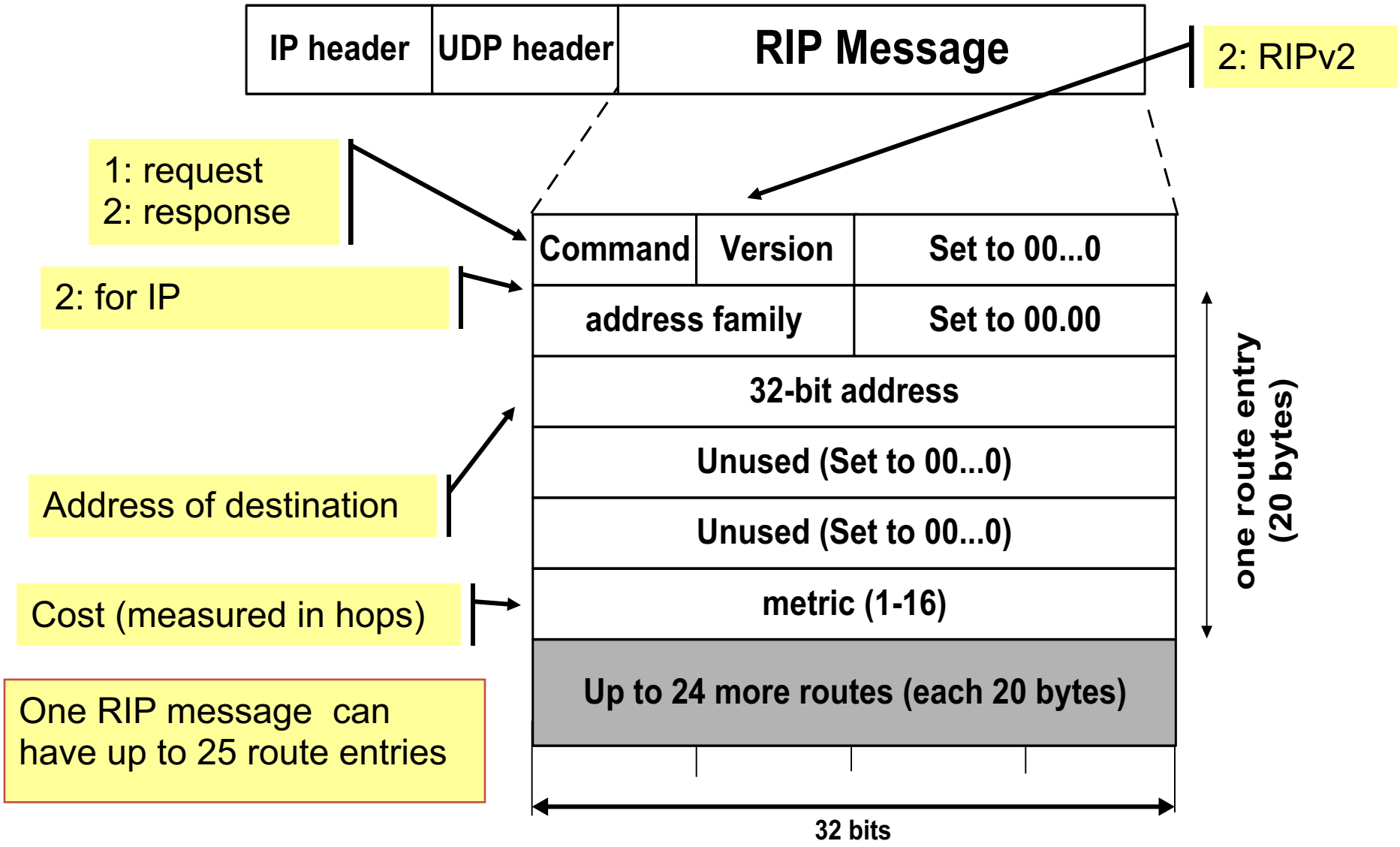


RIPv2

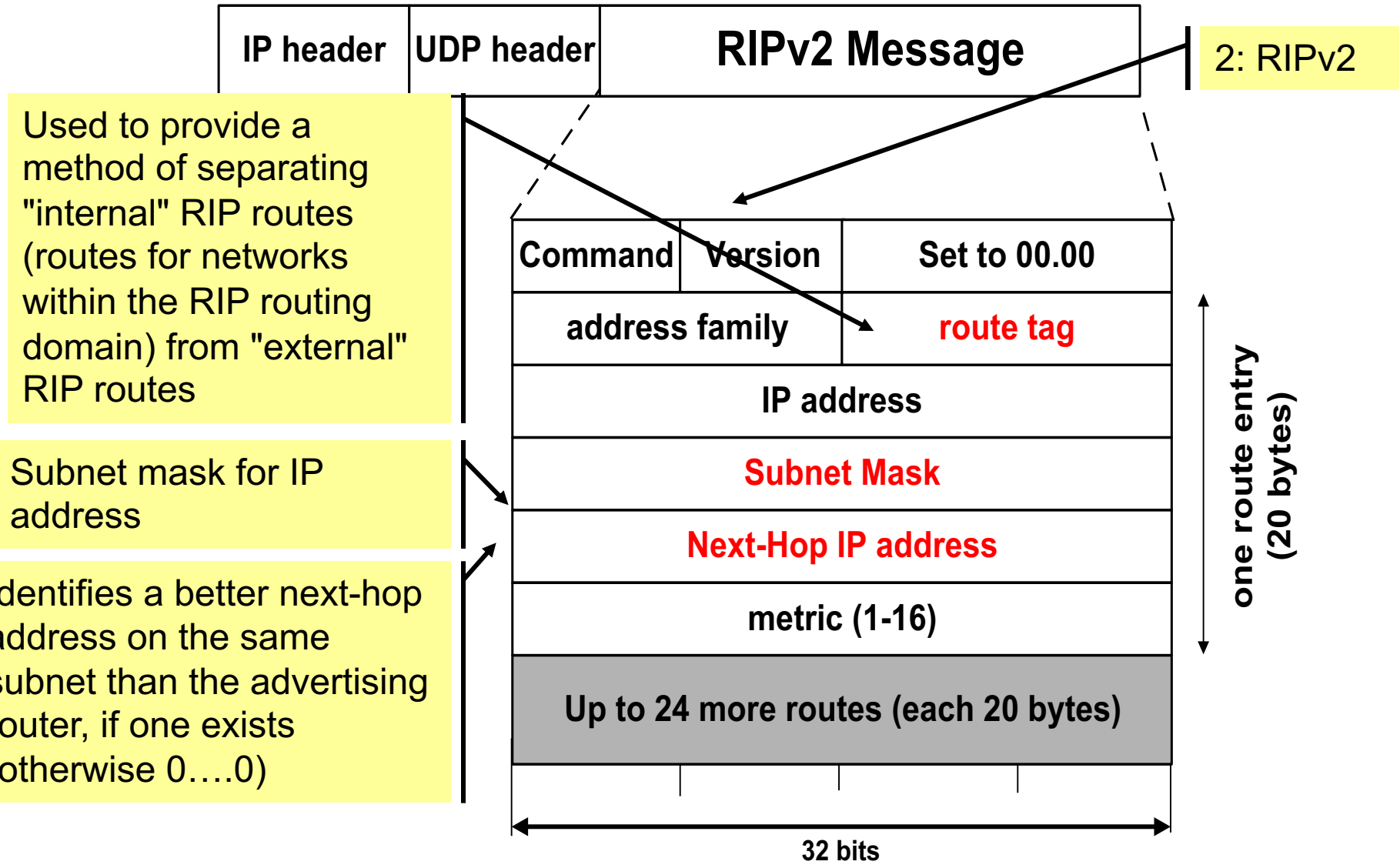
- RIPv2 is an extends RIPv1:
  - Subnet masks are carried in the route information
  - Authentication of routing messages
  - Route information carries next-hop address
  - Uses IP multicasting to send routing messages

- Extensions of RIPv2 are carried in unused fields of RIPv1 messages

RIPv2 Packet Format



RIPv2 Packet Format



RIP Messages

- This is the operation of RIP in **`routed`**. Dedicated port for RIP is UDP port 520.

- Two types of messages:
  - **Request messages**
    - used to ask neighboring nodes for an update
  - **Response messages**
    - contains an update

Routing with RIP

- **Initialization:** Send a **request packet** (command = 1, address family=0..0) on all interfaces:
    - RIPv1 uses broadcast if possible,
    - RIPv2 uses multicast address 224.0.0.9, if possible

  requesting routing tables from neighboring routers

- **Request received**: Routers that receive above request send their entire routing table

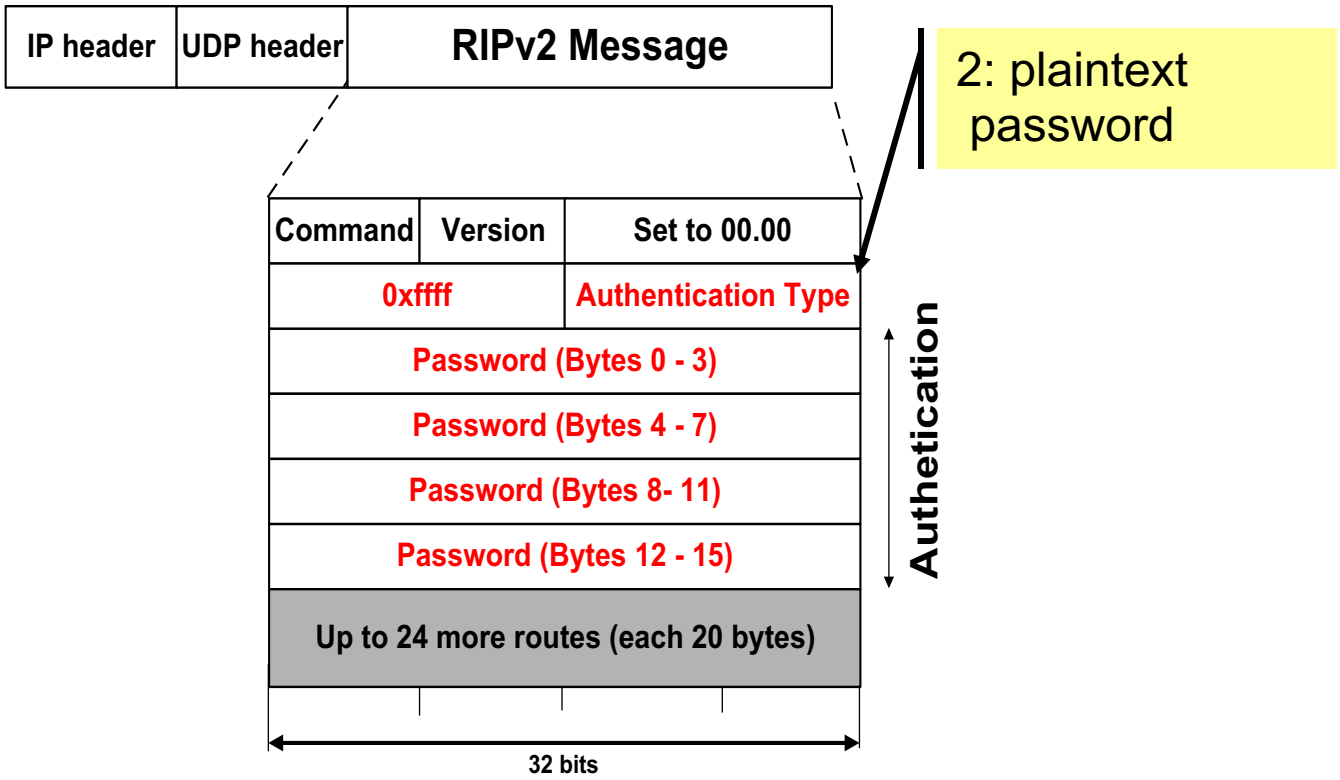
- **Response received**: Update the routing table

- **Regular routing updates**: Every 30 seconds, send all or part of the routing tables to every neighbor in an response message

- **Triggered Updates:** Whenever the metric for a route change, send the entire routing table

RIP Security

- Issue: Sending bogus routing updates to a router
- RIPv1: No protection
- RIPv2: Simple authentication scheme



RIP Problems

- RIP takes a long time to stabilize
  - Even for a small network, it takes several minutes until the routing tables have settled after a change
- RIP has all the problems of distance vector algorithms, e.g., count-to-Infinity
    - » RIP uses split horizon to avoid count-to-infinity
- The maximum path in RIP is 15 hops

Summary

- Dynamic Routing
  - RIP