

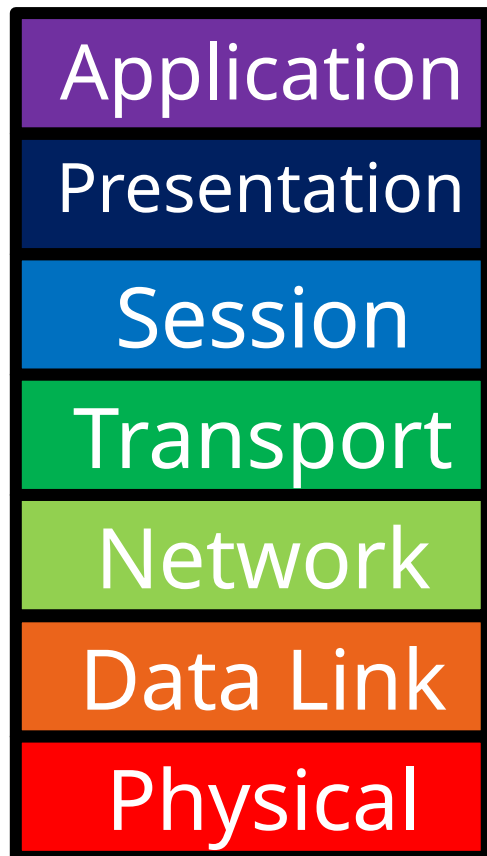
# Computer Networks

## Lecture 8: Interconnecting LANs

Based on slides from D. Choffnes Northeastern U. and P. Gill from StonyBrook University  
Revised Autumn 2015 by S. Laki

# Just Above the Data Link Layer

2

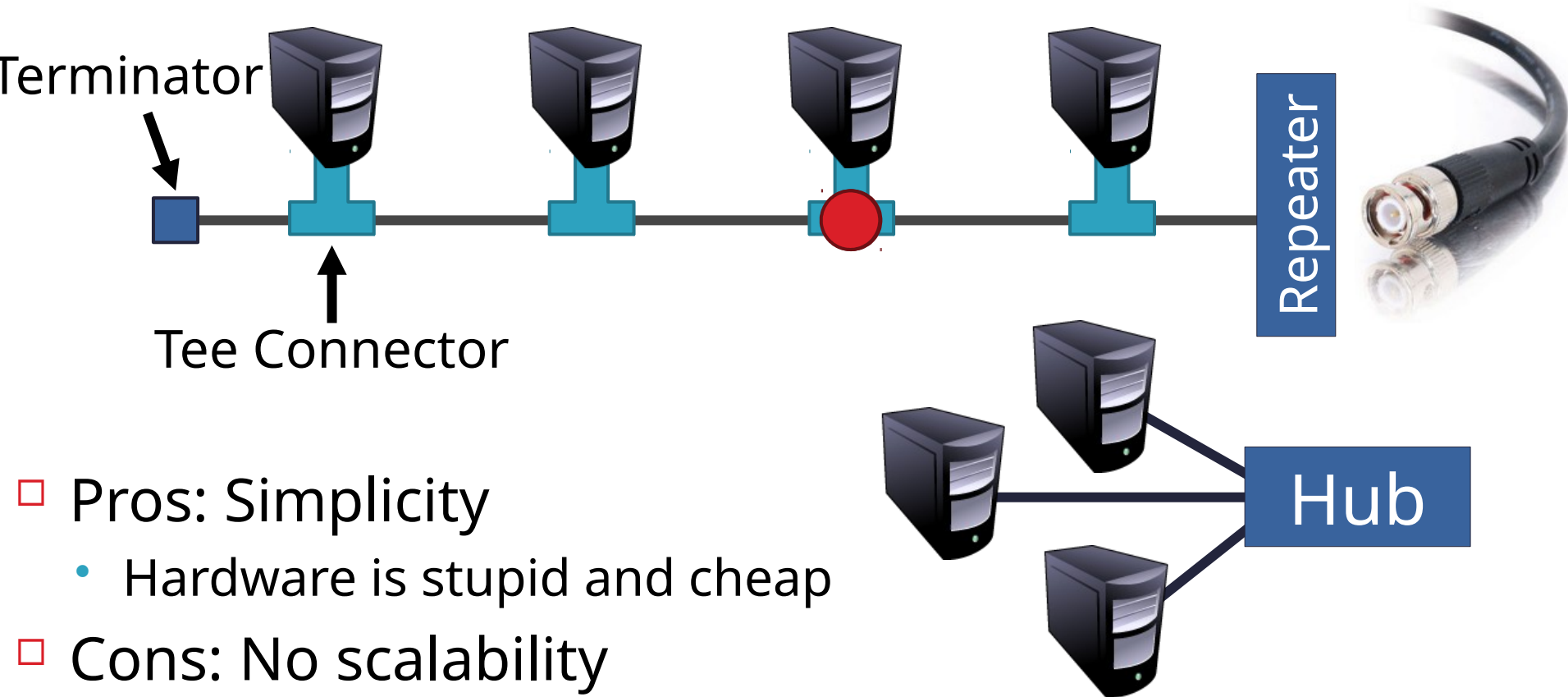


- Bridging
  - How do we connect LANs?
- Function:
  - Route packets between LANs
- Key challenges:
  - Plug-and-play, self configuration
  - How to resolve loops

# Recap

3

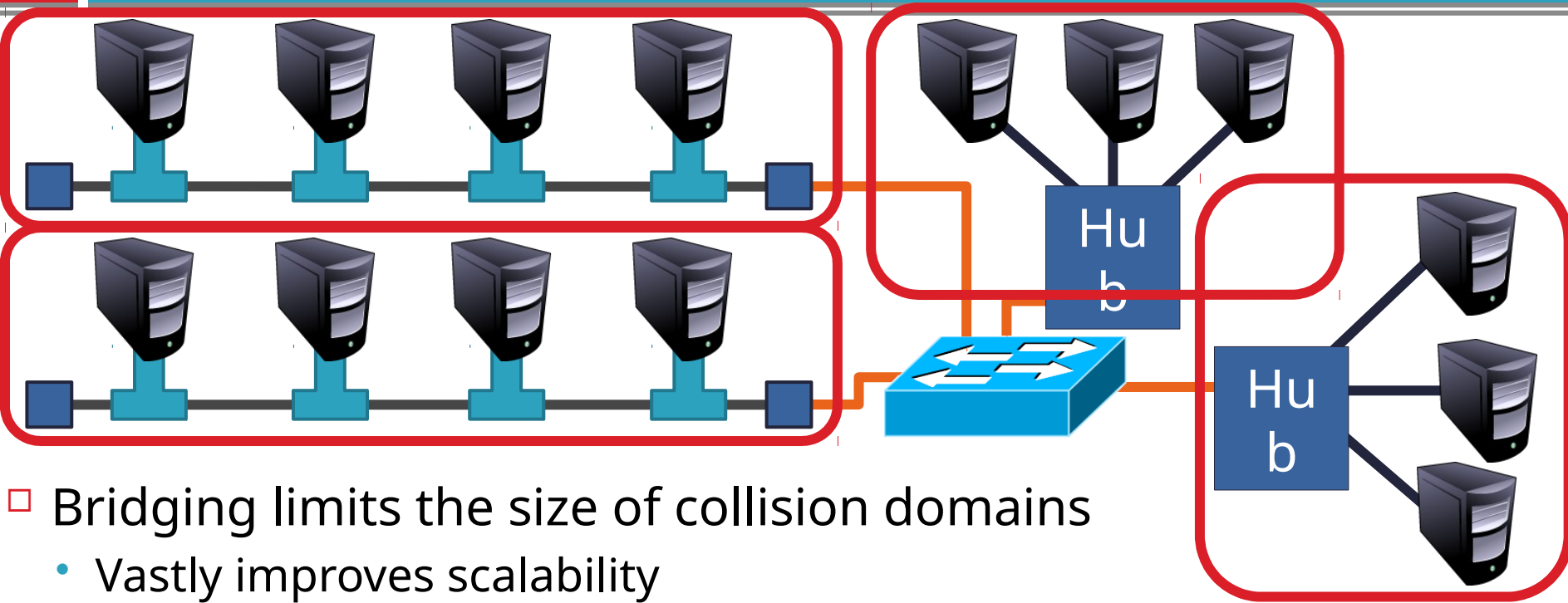
- Originally, Ethernet was a broadcast technology



- Pros: Simplicity
  - Hardware is stupid and cheap
- Cons: No scalability
  - More hosts = more collisions = pandemonium

# Bridging the LANs

4



- Bridging limits the size of collision domains
  - Vastly improves scalability
  - Question: could the whole Internet be one bridging domain?
- Tradeoff: bridges are more complex than hubs
  - Physical layer device vs. data link layer device
  - Need memory buffers, packet processing hardware, routing tables

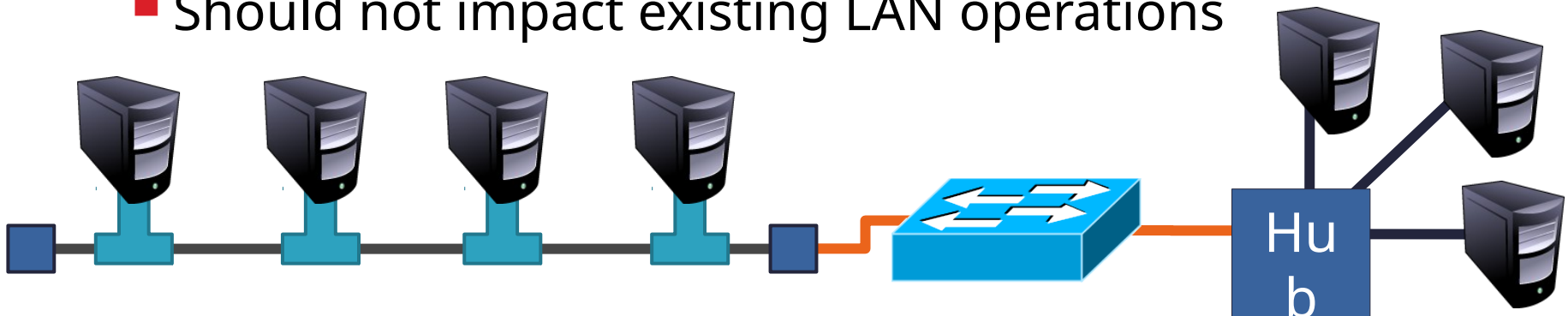
# Bridges

5

- ❑ Original form of Ethernet switch

❑ Consistent with IEEE 802 LAN/LAN+ standards

- ❑
  1. Forwarding of frames
  2. Learning of (MAC) Addresses
  3. Spanning Tree Algorithm (to handle loops)
- No hardware or software changes on hosts/hubs
- Should not impact existing LAN operations



# Frame Forwarding Tables

6

- Each bridge maintains a forwarding table

MAC Address	Port	Age
00:00:00:00:00:AA	1	1 minute
00:00:00:00:00:BB	2	7 minutes
00:00:00:00:00:CC	3	2 seconds
00:00:00:00:00:DD	1	3 minutes



# Learning Addresses

7

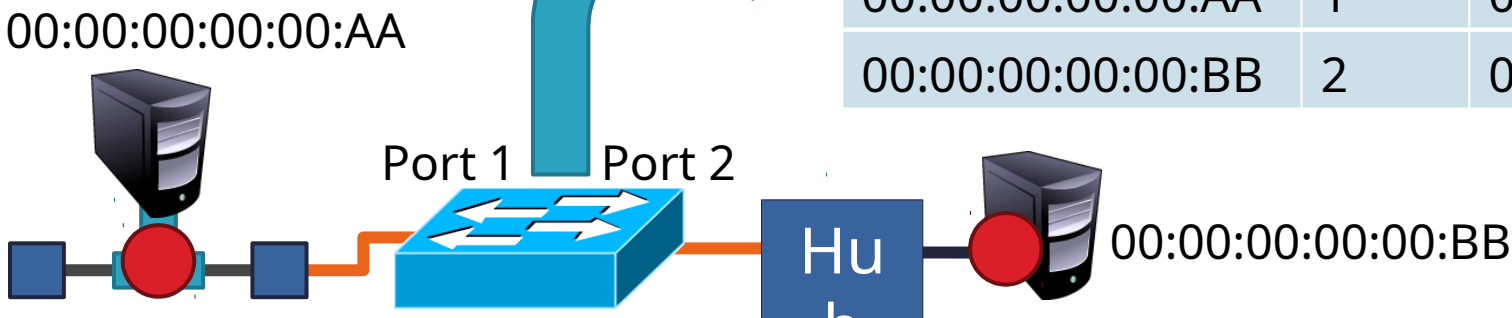
- ❑ Manual configuration is possible, but...
  - Time consuming
  - Error Prone
  - Not adaptable (hosts may get added or removed)

- ❑ Instead, learn addresses using heuristic

Delete old entries after a timeout

- Look at the **source** of frames that arrive on each port

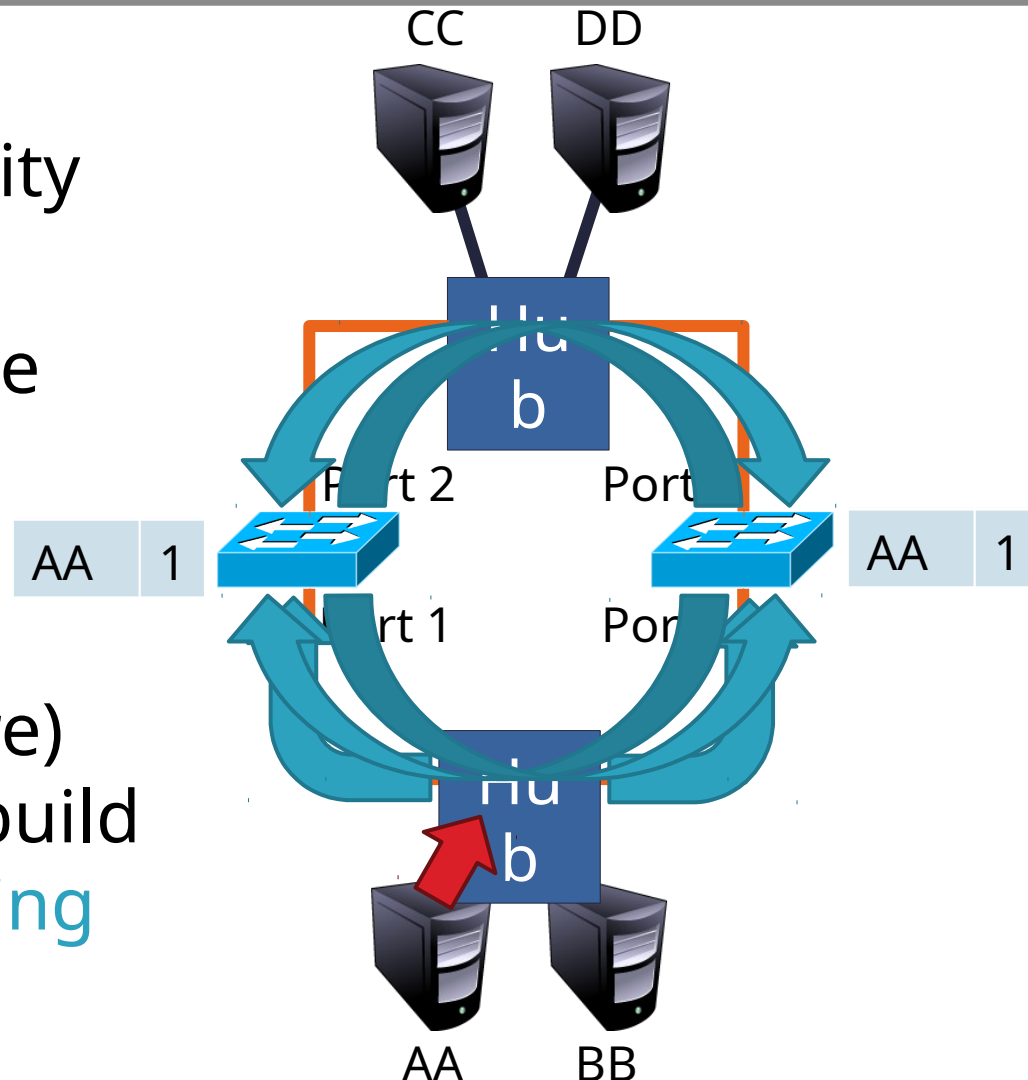
MAC Address	Port	Age
00:00:00:00:00:AA	1	0 minutes
00:00:00:00:00:BB	2	0 minutes



# The Danger of Loops

8

- ❑ <Src=AA, Dest=DD>
- ❑ This continues to infinity
  - How do we stop this?
- ❑ Remove loops from the topology
  - Without physically unplugging cables
- ❑ 802.1 (LAN architecture) uses an algorithm to build and maintain a **spanning tree** for routing

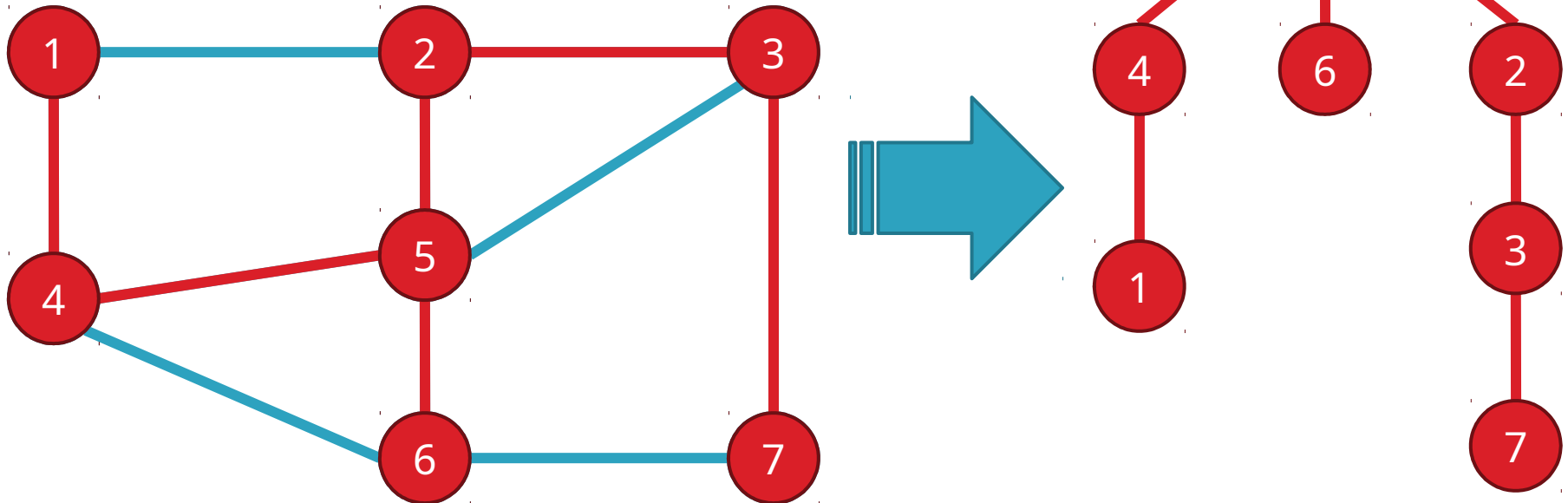




# Spanning Tree Definition

9

- A subset of edges in a graph that:
  - Span all nodes
  - Do not create any cycles
- This structure is a tree



# 802.1 Spanning Tree Approach

10

1. Elect a bridge to be the root of the tree
  2. Every bridge finds shortest path to the root
  3. Union of these paths becomes the spanning tree
- 
- Bridges exchange Configuration Bridge Protocol Data Units (BPDUs) to build the tree
    - Used to elect the root bridge
    - Calculate shortest paths
    - Locate the next hop closest to the root, and its port
    - Select ports to be included in the spanning trees

# Determining the Root

11

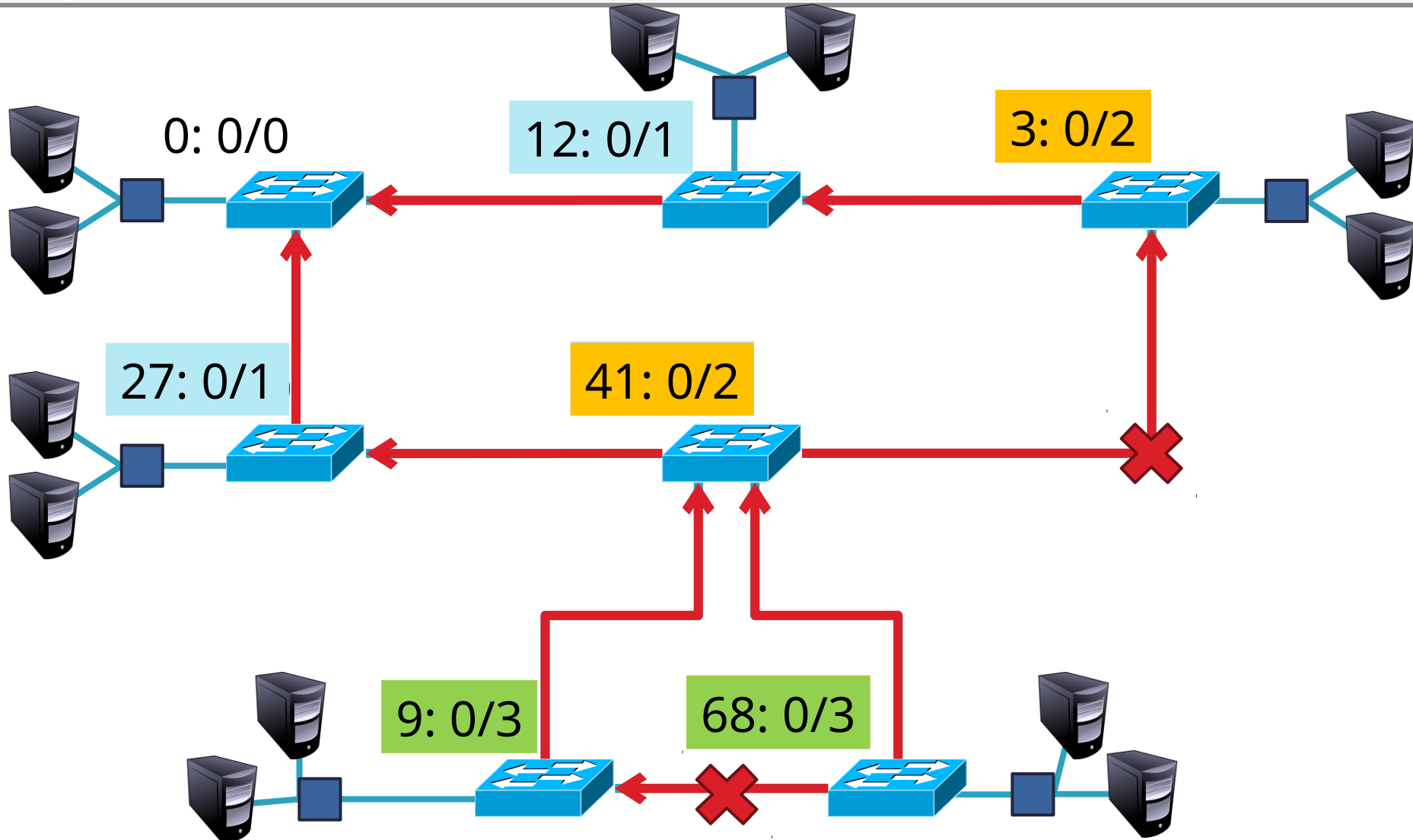
- Initially, all hosts assume they are the root
- Bridges broadcast BPDUs:

Bridge ID	Root ID	Path Cost to Root
-----------	---------	-------------------

- Based on received BPDUs, each switch chooses:
  - A new root (smallest known Root ID)
  - A new root port (what interface goes towards the root)
  - A new designated bridge (who is the next hop to root)

# Spanning Tree Construction

12



# Bridges vs. Switches

13

- ❑ Bridges make it possible to increase LAN capacity
  - Reduces the amount of broadcast packets
  - No loops
- ❑ Switch is a special case of a bridge
  - Each port is connected to a **single** host
    - Either a client machine
    - Or another switch
  - Links are full duplex
  - Simplified hardware: no need for CSMA/CD!
  - Can have different speeds on each port

# Switching the Internet

14

- Capabilities of switches:
  - Network-wide routing based on MAC addresses
  - Learn routes to new hosts automatically
  - Resolve loops
- Could the whole Internet be one switching domain?

NO

# Limitations of MAC Routing

15

- ❑ Inefficient
  - Flooding packets to locate unknown hosts
- ❑ Poor Performance
  - Spanning tree does not balance load
  - Hot spots
- ❑ Extremely Poor Scalability
  - Every switch needs every MAC address on the Internet in its routing table!
- ❑ IP addresses these problems (next ...)

# Computer Networks

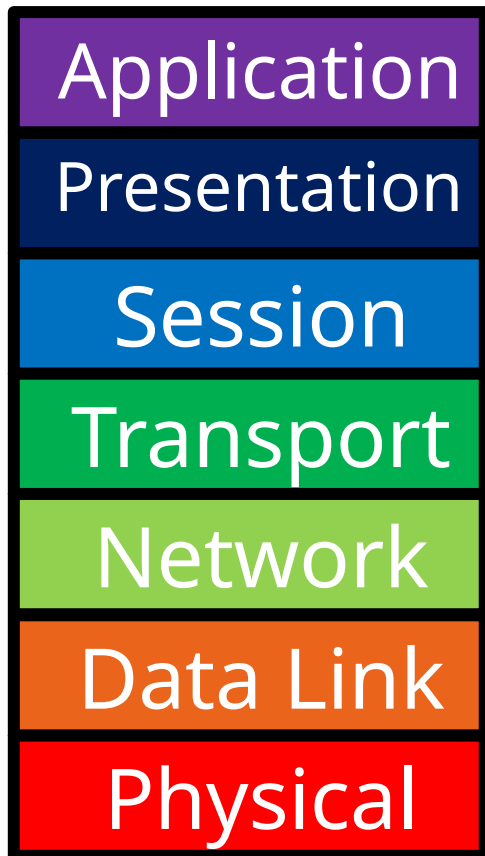
## Lecture 8: Network Layer

Based on slides from D. Choffnes Northeastern U. and P. Gill from StonyBrook University  
Revised Autumn 2015 by S. Laki



# Network Layer

17

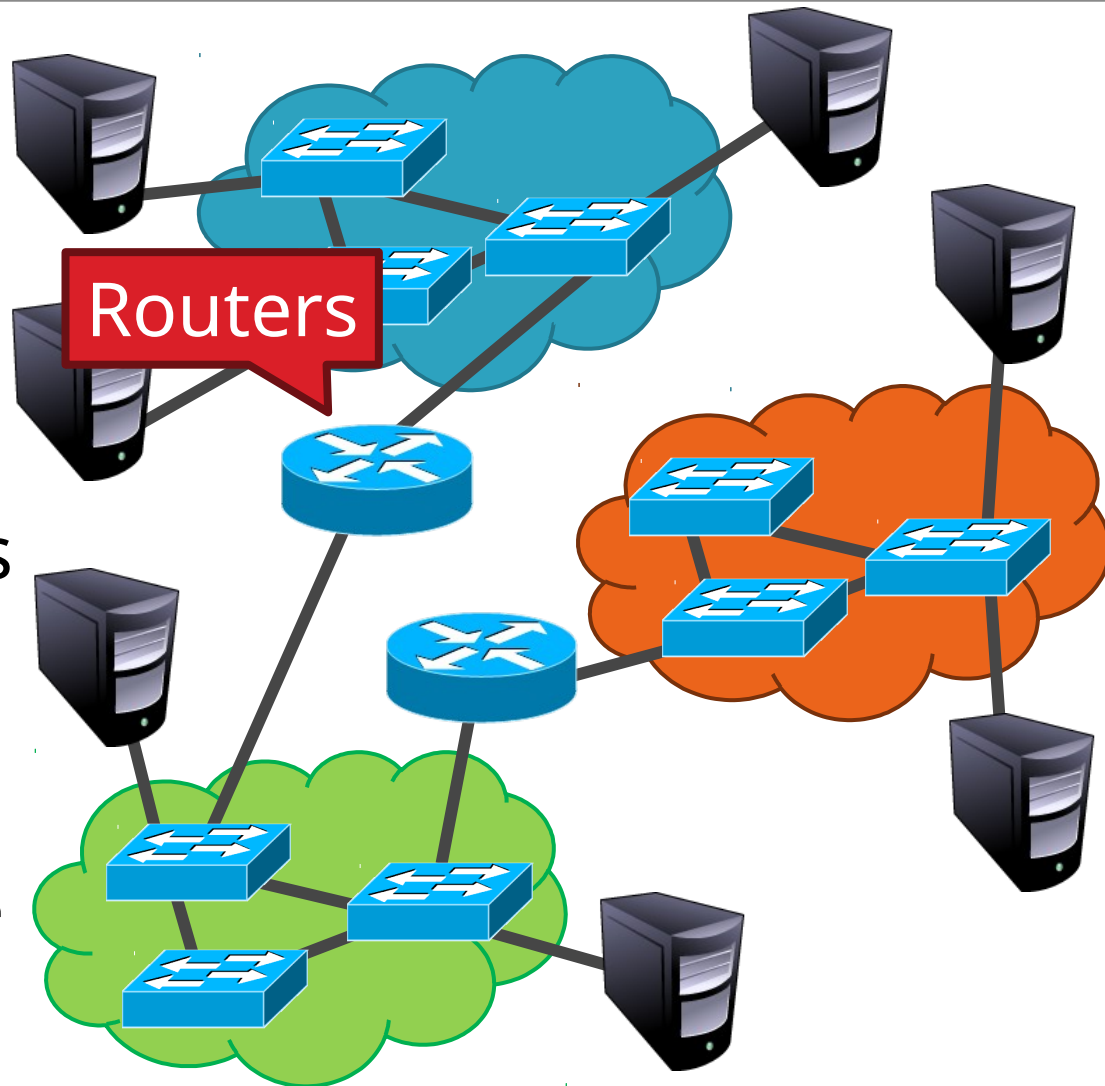


- Function:
  - Route packets end-to-end on a network, through multiple hops
- Key challenge:
  - How to represent addresses
  - How to route packets
    - Scalability
    - Convergence

# Routers, Revisited

18

- How to connect multiple LANs?
- LANs may be incompatible
  - Ethernet, Wifi, etc...
- Connected networks form an **internetwork**
  - The Internet is the best known example



## Internet Service Model

- Best-effort (i.e. things may break)
- Store-and-forward datagram network

Lowest common denominator

work)

### □ Service Model

- What gets sent?
- How fast will it go?
- What happens if there are failures?
- Must deal with **heterogeneity**
  - Remember, every network is different

- ❑ Addressing
  - ❑ Class-based
  - ❑ CIDR
- ❑ IPv4 Protocol Details
  - ❑ Packed Header
  - ❑ Fragmentation
- ❑ IPv6

# Possible Addressing Schemes

21

## □ Flat

- e.g. each host is identified by a 48-bit MAC address
- Router needs an entry for every host in the world
  - Too big
  - Too hard to maintain (hosts come and go all the time)
  - Too slow (more later)

## □ Hierarchy

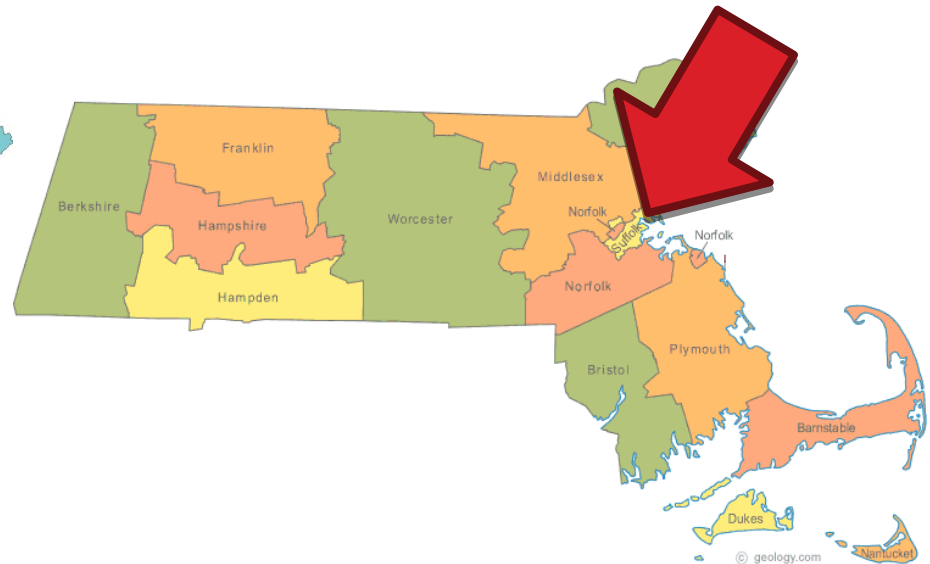
- Addresses broken down into segments
- Each segment has a different level of specificity

# Example: Telephone Numbers

22

1-617-373- 3278

Very General



Northeastern University

West Village G  
Room 254

Updates are Local

Very Specific

# IP Addressing

24

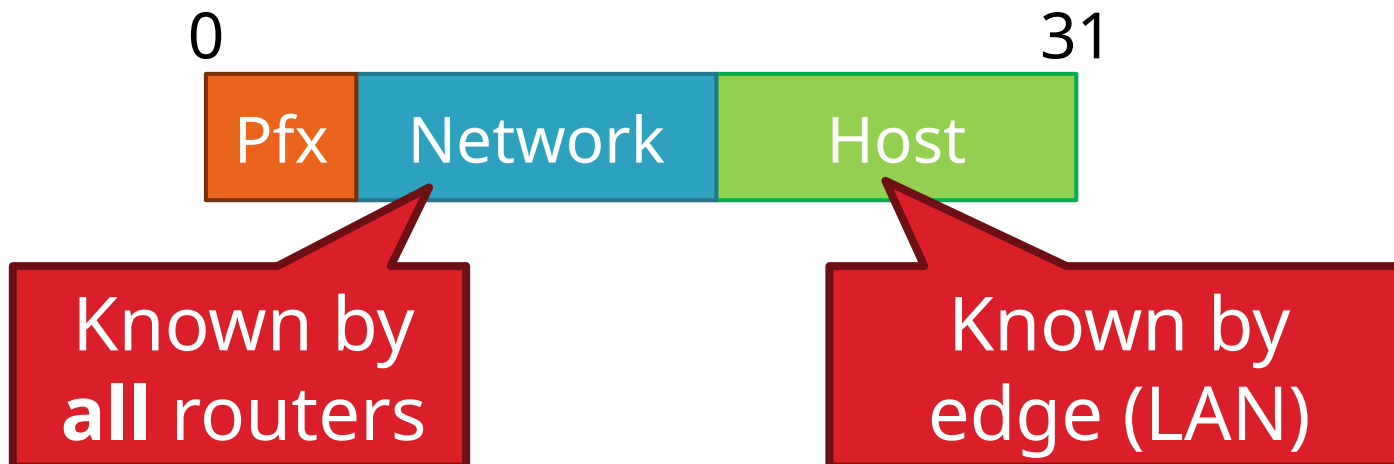
- IPv4: 32-bit addresses
  - Usually written in dotted notation, e.g. 192.168.21.76
  - Each number is a byte

	0	8	16	24	31
Decimal	192	168	21	76	
Hex	C0	A8	15	4C	
Binary	11000000	10101000	00010101	01001100	

# IP Addressing and Forwarding

25

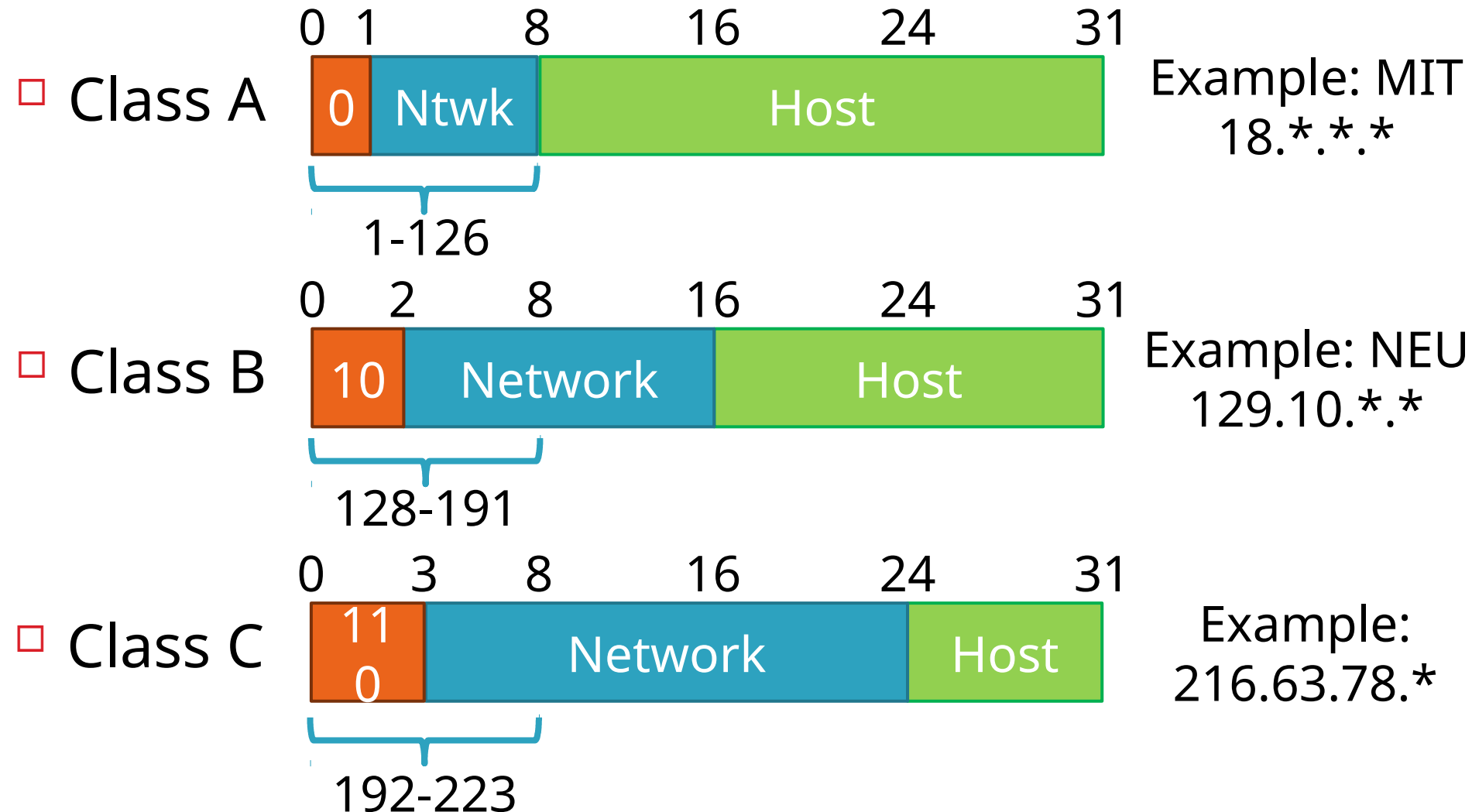
- Routing Table Requirements
  - For every possible IP, give the next hop
  - But for 32-bit addresses,  $2^{32}$  possibilities!
- Hierarchical address scheme
  - Separate the address into a network and a host





# Classes of IP Addresses

26



# How Do You Get IPs?

27

- IP address ranges controlled by IANA

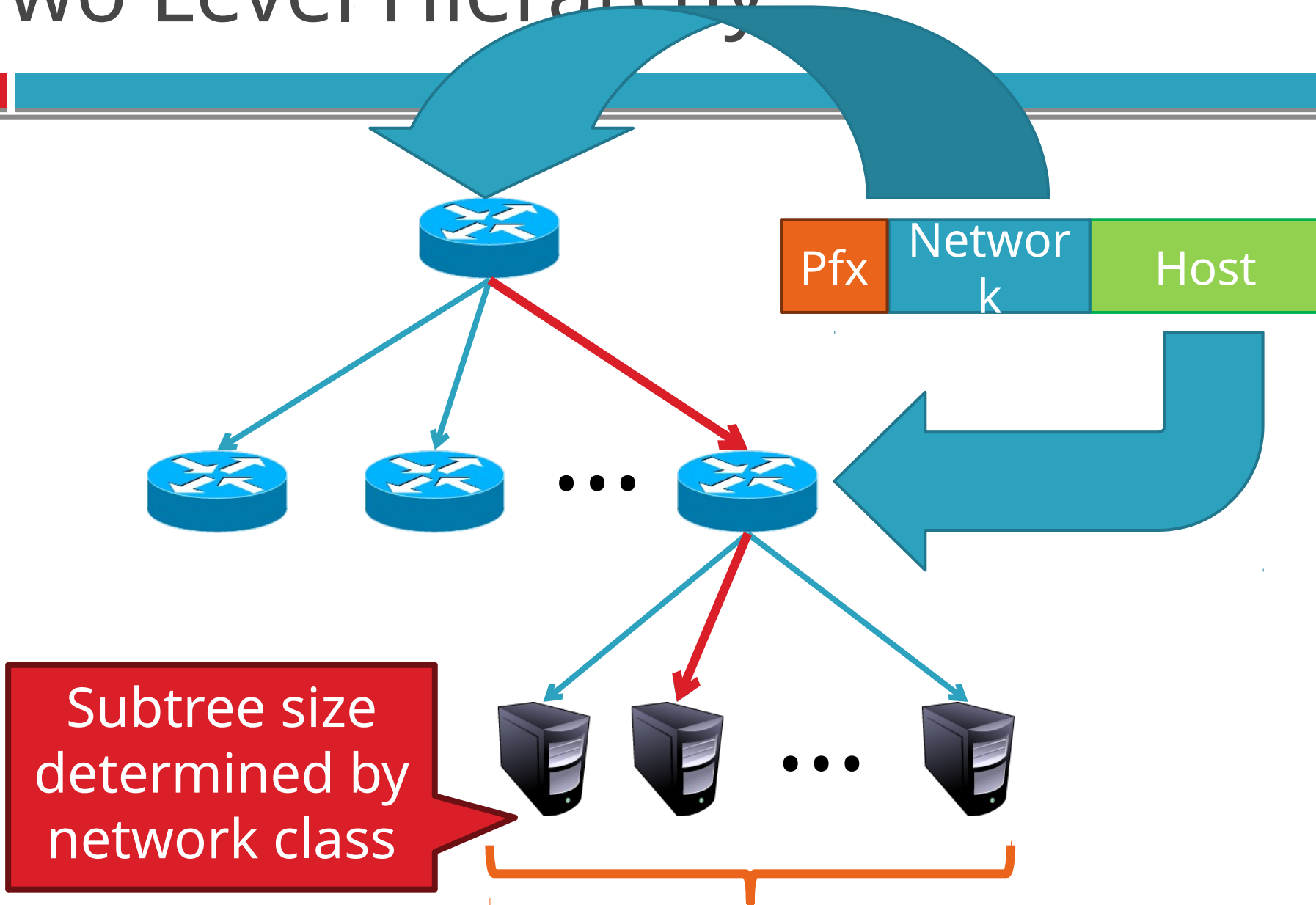


Internet Assigned Numbers Authority

- Internet Assigned Number Authority
  - Roots go back to 1972, ARPANET, UCLA
  - Today, part of ICANN
- IANA grants IPs to regional authorities
    - ARIN (American Registry of Internet Numbers) may grant you a range of IPs
    - You may then advertise routes to your new IP range
    - There are now secondary markets, auctions, ...

# Two Level Hierarchy

28



# Class Sizes

29

Way too big

Class	Prefix Bits	Network Bits	Number of Classes	Hosts per Class
A	1	7	$2^7 - 2 = 126$ (0 and 127 are reserved)	$2^{24} - 2 = 16,777,214$ (All 0 and all 1 are reserved)
B	2	14	$2^{14} - 2 = 16,382$ (0 and 1 are reserved)	$2^{16} - 2 = 65,534$ (All 0 and all 1 are reserved)
C	3	21	$2^{21} = 2,097,152$ (0 and 1 are reserved)	$2^8 - 2 = 254$ (All 0 and all 1 are reserved)
D	4	28	$2^{28} = 268,435,456$ (0 and 1 are reserved)	$2^0 = 1$ (All 0 and all 1 are reserved)

Too many network IDs

Too small to be useful



# Subnet Example

31

## □ Extract network:

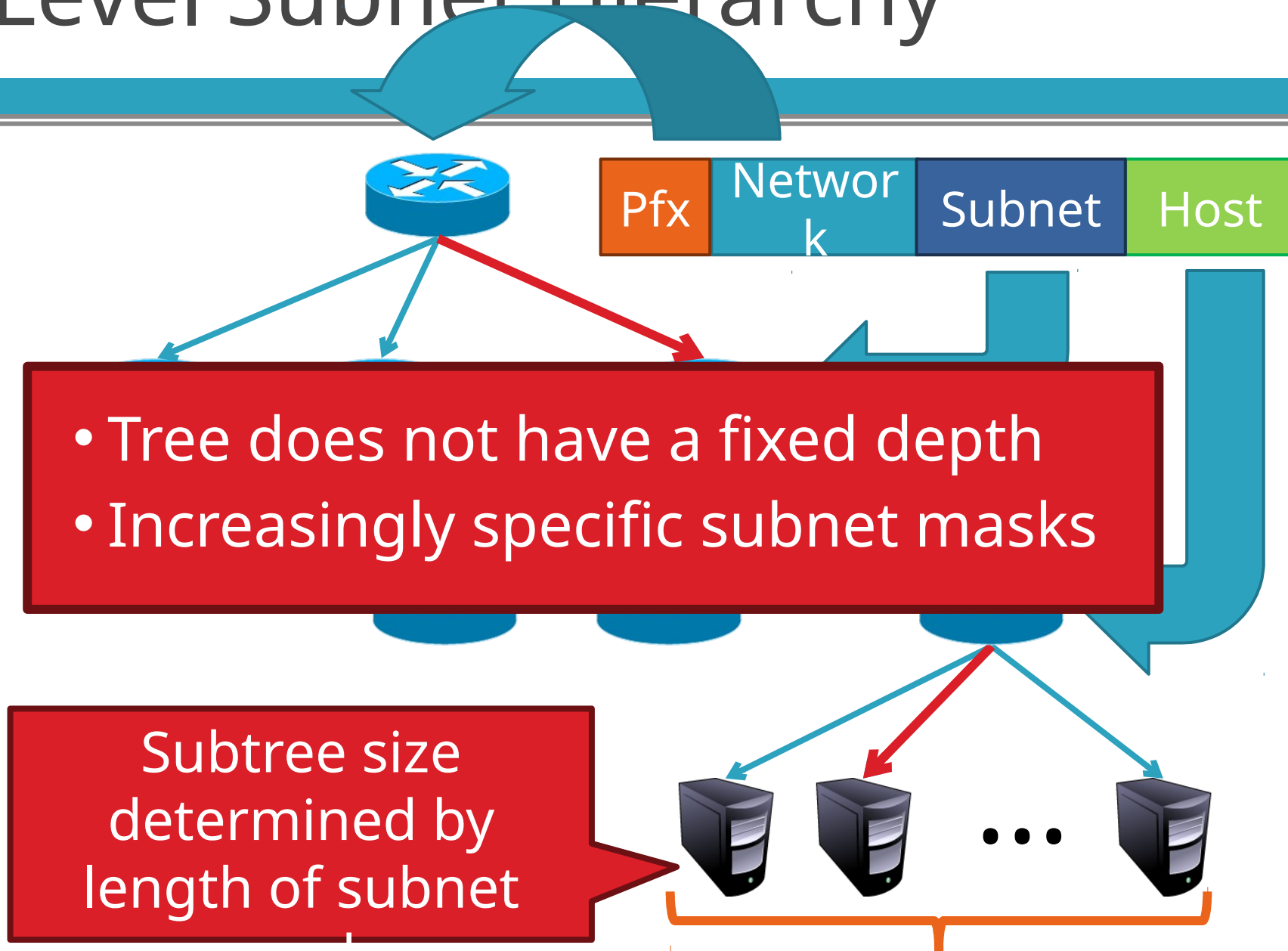
IP Address:	10110101 11011101 01010100 01110010
Subnet Mask:	& 11111111 11111111 11000000 00000000
Result:	10110101 11011101 01000000 00000000

## □ Extract host:

IP Address:	10110101 11011101 01010100 01110010
Subnet Mask:	& ~(11111111 11111111 11000000 00000000)
Result:	00000000 00000000 00010100 01110010

# N-Level Subnet Hierarchy

32



# Example Routing Table

33

Address Pattern	Subnet Mask	Destination Router
0.0.0.0	0.0.0.0	Router 4
18.0.0.0	255.0.0.0	Router 2
128.42.0.0	255.255.0.0	Router 3
128.42.128.0	255.255.128.0	Router 5
128.42.222.0	255.255.255.0	Router 1

□ Question: 128.42.222.198 matches all four rows

- Which router do we forward to?

□ Longest prefix matching

- Use the row with the longest number of 1's in the mask
- This is the **most specific match**



# Subnetting Revisited

34

- Question: does subnetting solve all the problems of class-based routing?

NO

- Classes are still too coarse
  - Class A can be subnetted, but only 126 available
  - Class C is too small
  - Class B is nice, but there are only 16,398 available
- Routing tables are still too big
  - 2.1 million entries per router

# Classless Inter Domain Routing

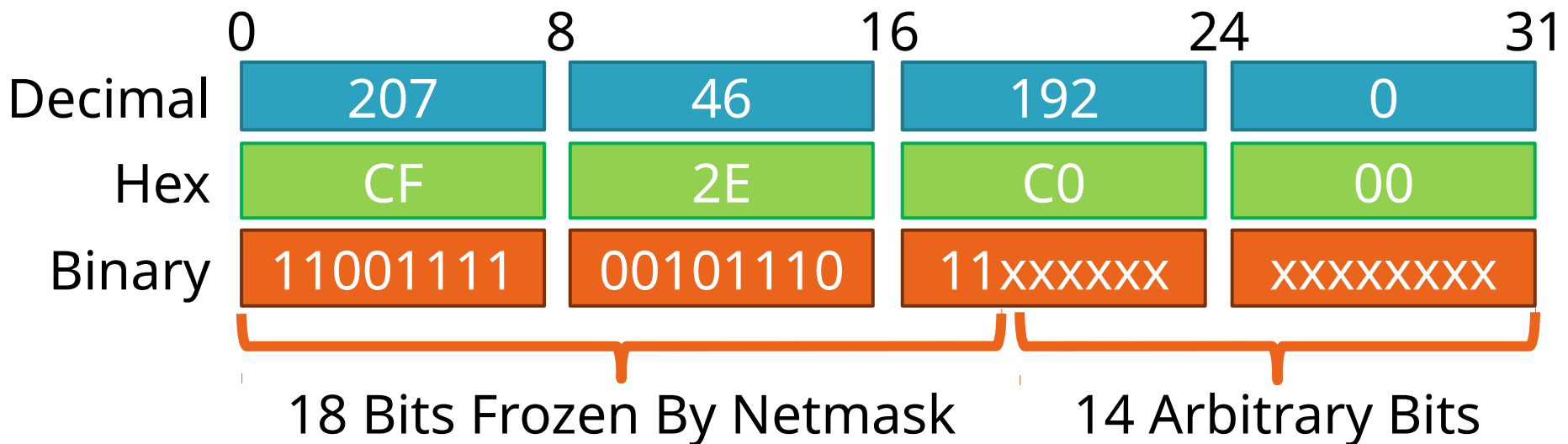
35

- CIDR, pronounced 'cider'
- Key ideas:
  - Get rid of IP classes
  - Use bitmasks for all levels of routing
  - **Aggregation** to minimize FIB (forwarding information base)
- Arbitrary split between network and host
  - Specified as a bitmask or prefix length
  - Example: Stony Brook
    - 130.245.0.0 with **netmask** 255.255.0.0
    - 130.245.0.0 / 16

# Aggregation with CIDR

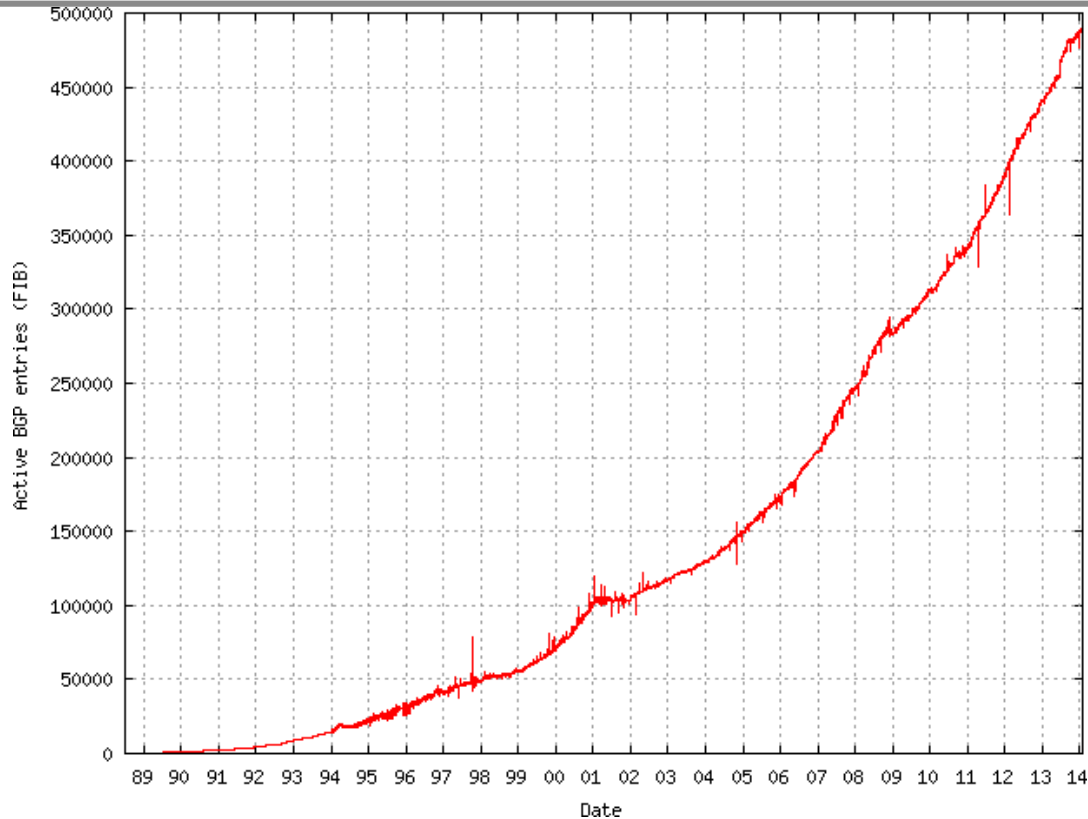
36

- ❑ Original use: aggregating class C ranges
- ❑ One organization given contiguous class C ranges
  - Example: Microsoft, 207.46.192.\* – 207.46.255.\*
  - Represents  $2^6 = 64$  class C ranges
  - Specified as CIDR address 207.46.192.0/18



# Size of CIDR Routing Tables

37



- From [www.cidr-report.org](http://www.cidr-report.org)
- CIDR has kept IP routing table sizes in check
  - Currently ~500,000 entries for a complete IP routing table
  - Only required by backbone routers

# We had a special day in summer 2014!

38

- 512K day – August 12, 2014
- Default threshold size for IPv4 route data in older Cisco routers □ 512K routes
  - Some routers failed over to slower memory
    - RAM vs. CAM (content addressable memory)
  - Some routes dropped
- Cisco issues update in May anticipating this issue
  - Reallocated some IPv6 space for IPv4 routes
- Part of the cause
  - Growth in emerging markets
- <http://cacm.acm.org/news/178293-internet-routing-failures-bring-architecture-changes-back-to-the-table/fulltext>

# Takeaways

39

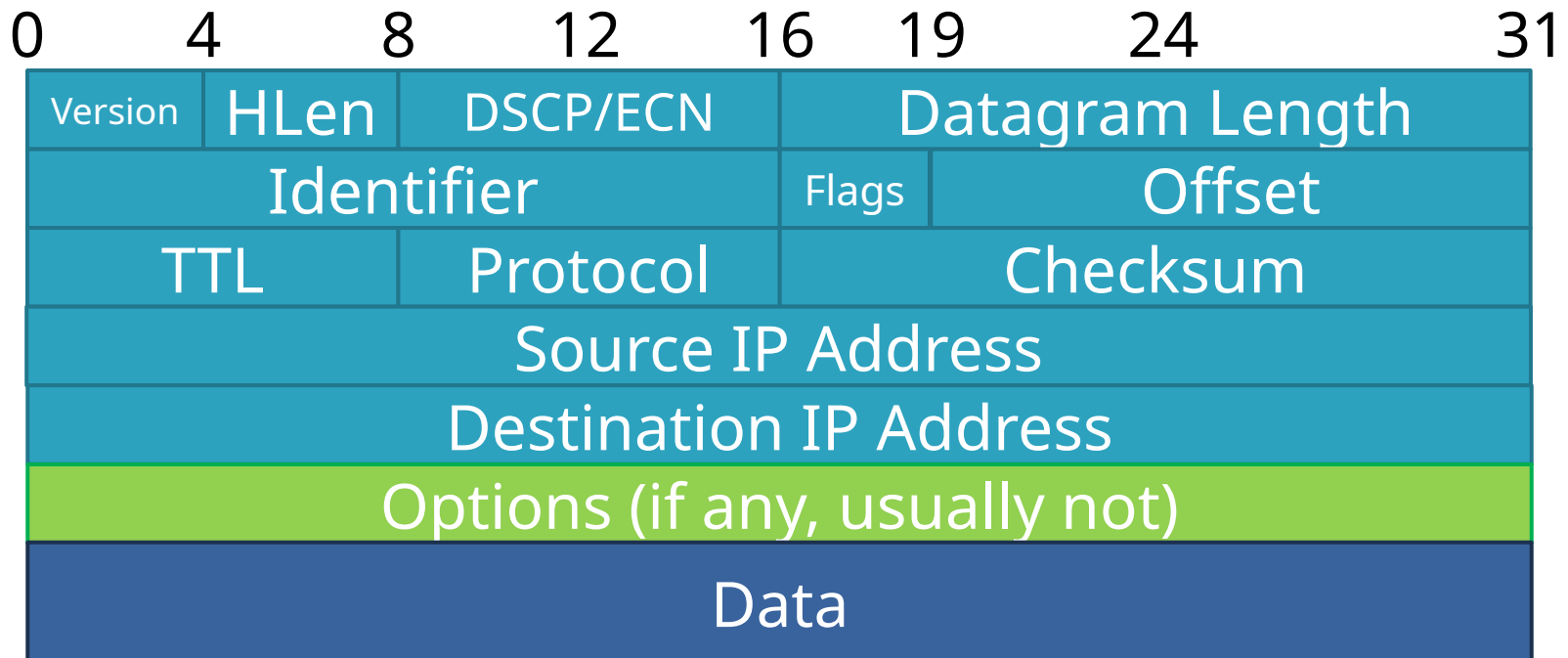
- ❑ Hierarchical addressing is critical for scalability
  - Not all routers need all information
  - Limited number of routers need to know about changes
- ❑ Non-uniform hierarchy useful for heterogeneous networks
  - Class-based addressing is too coarse
  - CIDR improves scalability and granularity
- ❑ Implementation challenges
  - Longest prefix matching is more difficult than schemes with no ambiguity

- ❑ Addressing
  - ❑ Class-based
  - ❑ CIDR
- ❑ IPv4 Protocol Details
  - ❑ Packed Header
  - ❑ Fragmentation
- ❑ IPv6

# IP Datagrams

41

- IP Datagrams are like a letter
  - Totally self-contained
  - Include all necessary addressing information
  - No advanced setup of connections or circuits

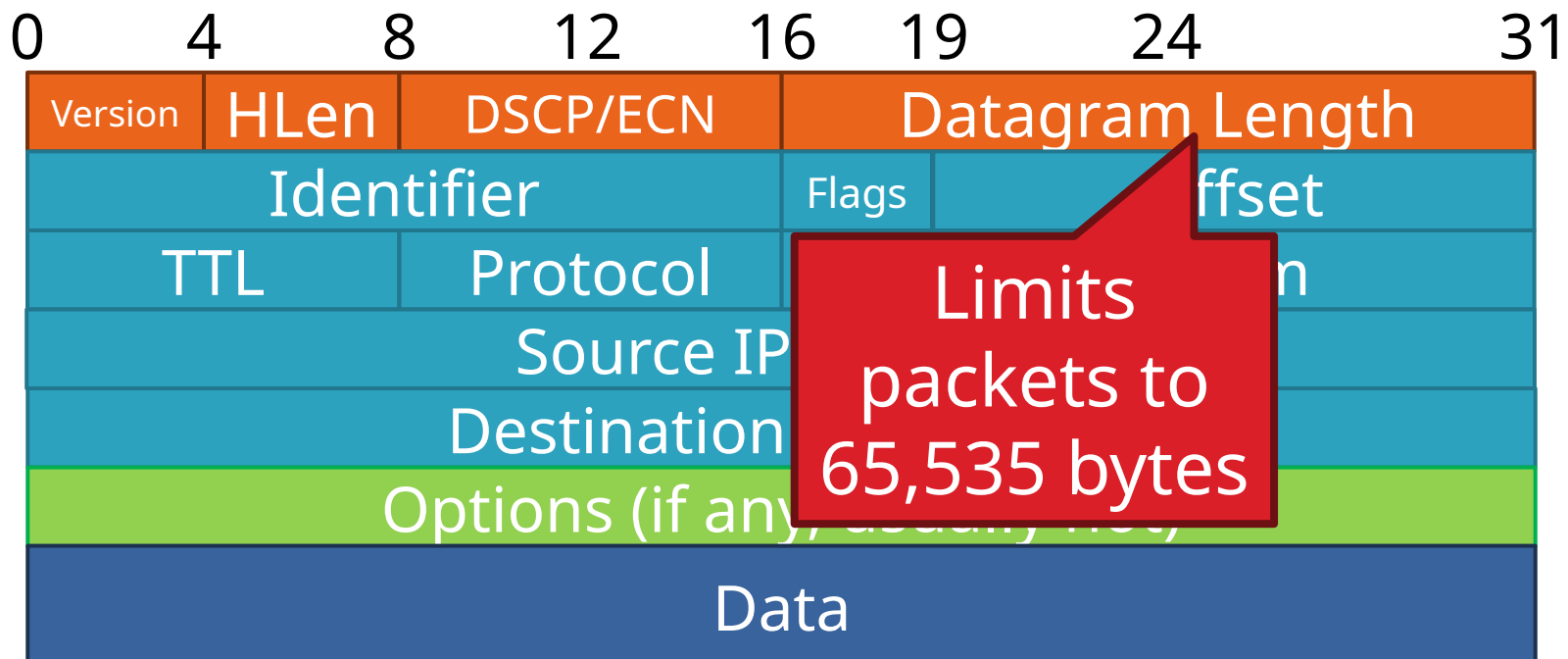




# IP Header Fields: Word 1

42

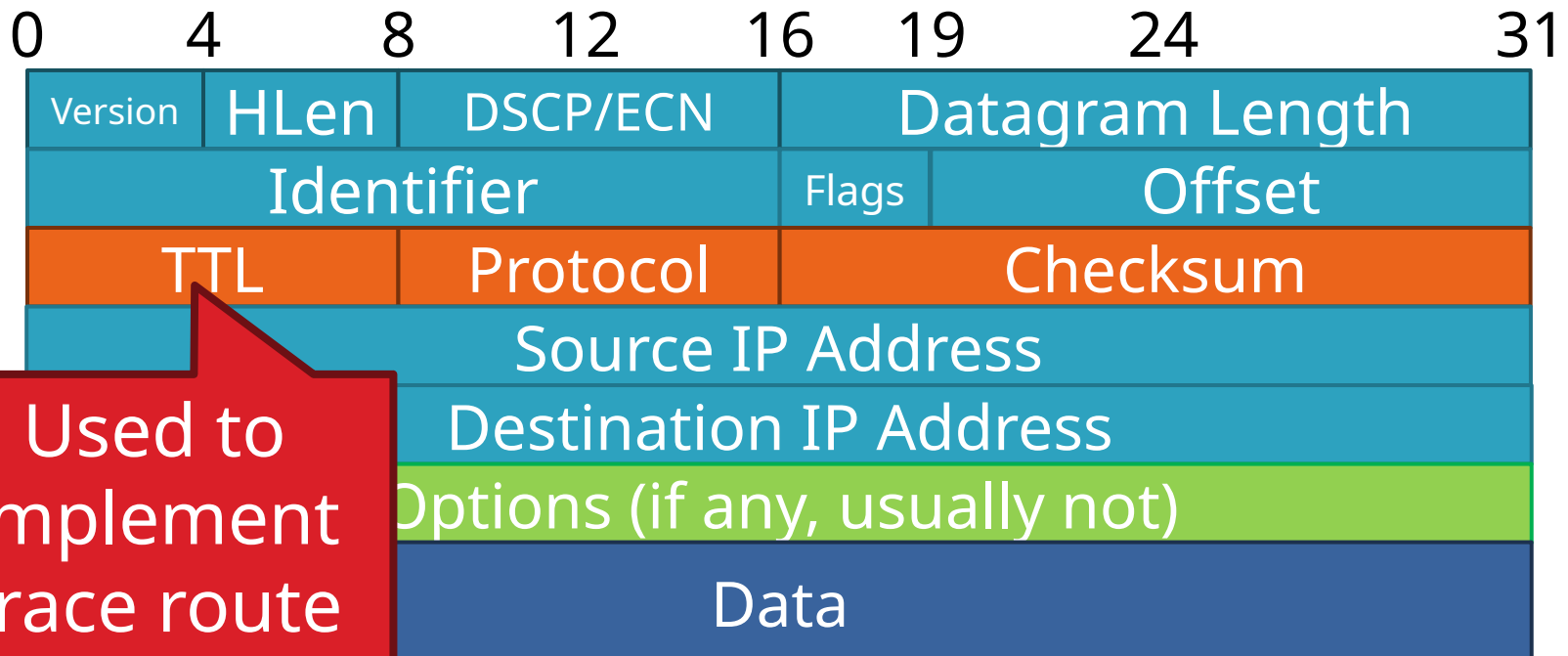
- Version: 4 for IPv4
- Header Length: Number of 32-bit words (usually 5)
- Type of Service: Priority information (unused)
- Datagram Length: Length of header + data in bytes



# IP Header Fields: Word 3

43

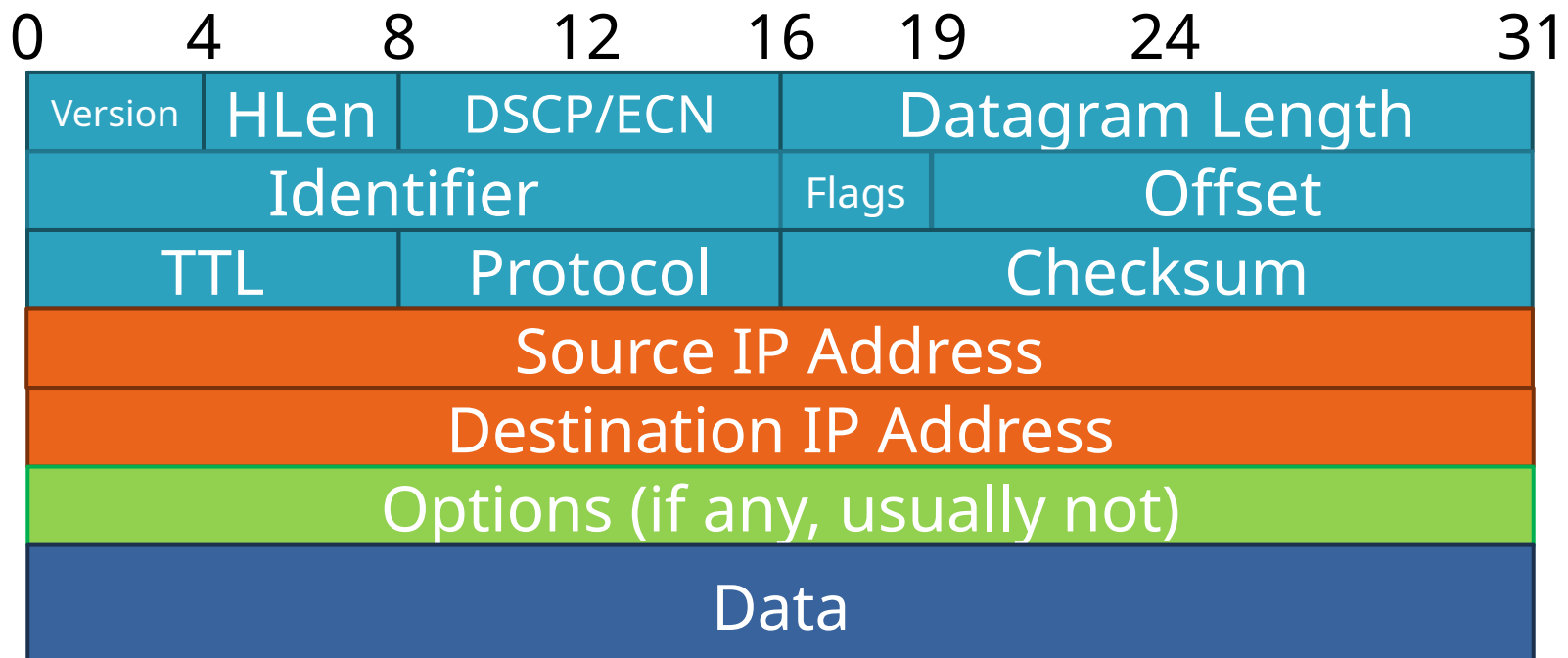
- ❑ Time to Live: decremented by each router
  - Used to kill looping packets
- ❑ Protocol: ID of encapsulated protocol
  - 6 = TCP, 17 = UDP
- ❑ Checksum



# IP Header Fields: Word 4 and 5

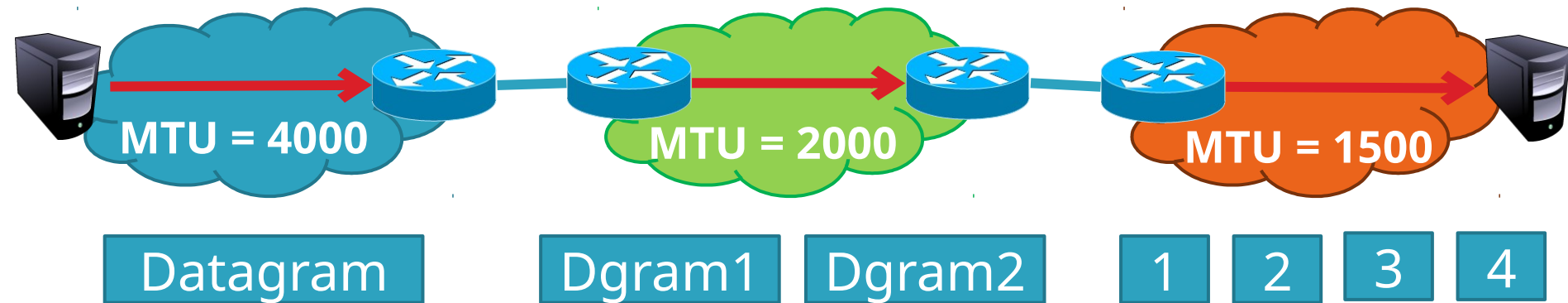
44

- Source and destination address
  - In theory, must be globally unique
  - In practice, this is often violated



# Problem: Fragmentation

45

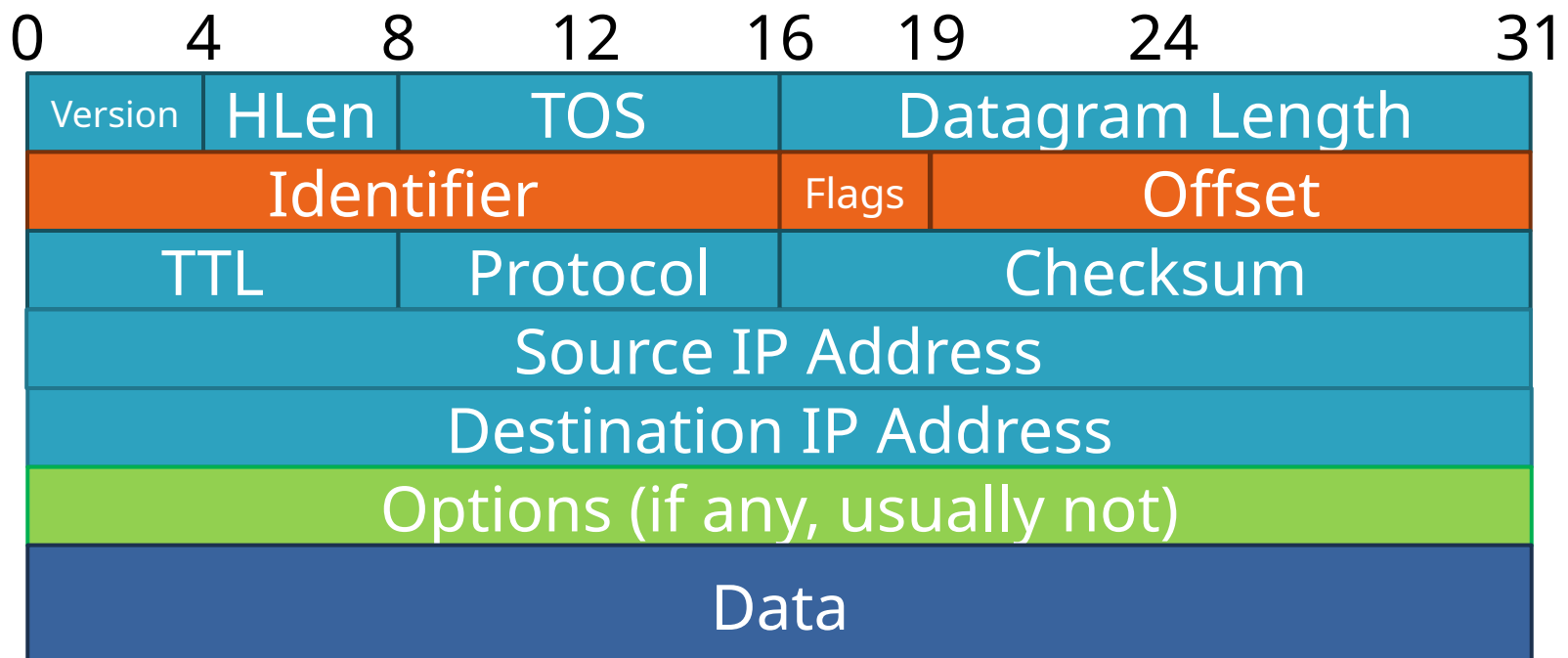


- Problem: each network has its own MTU
  - DARPA principles: networks allowed to be heterogeneous
  - Minimum MTU may not be known for a given path
- IP Solution: fragmentation
  - Split datagrams into pieces when MTU is reduced
  - Reassemble original datagram at the receiver

# IP Header Fields: Word 2

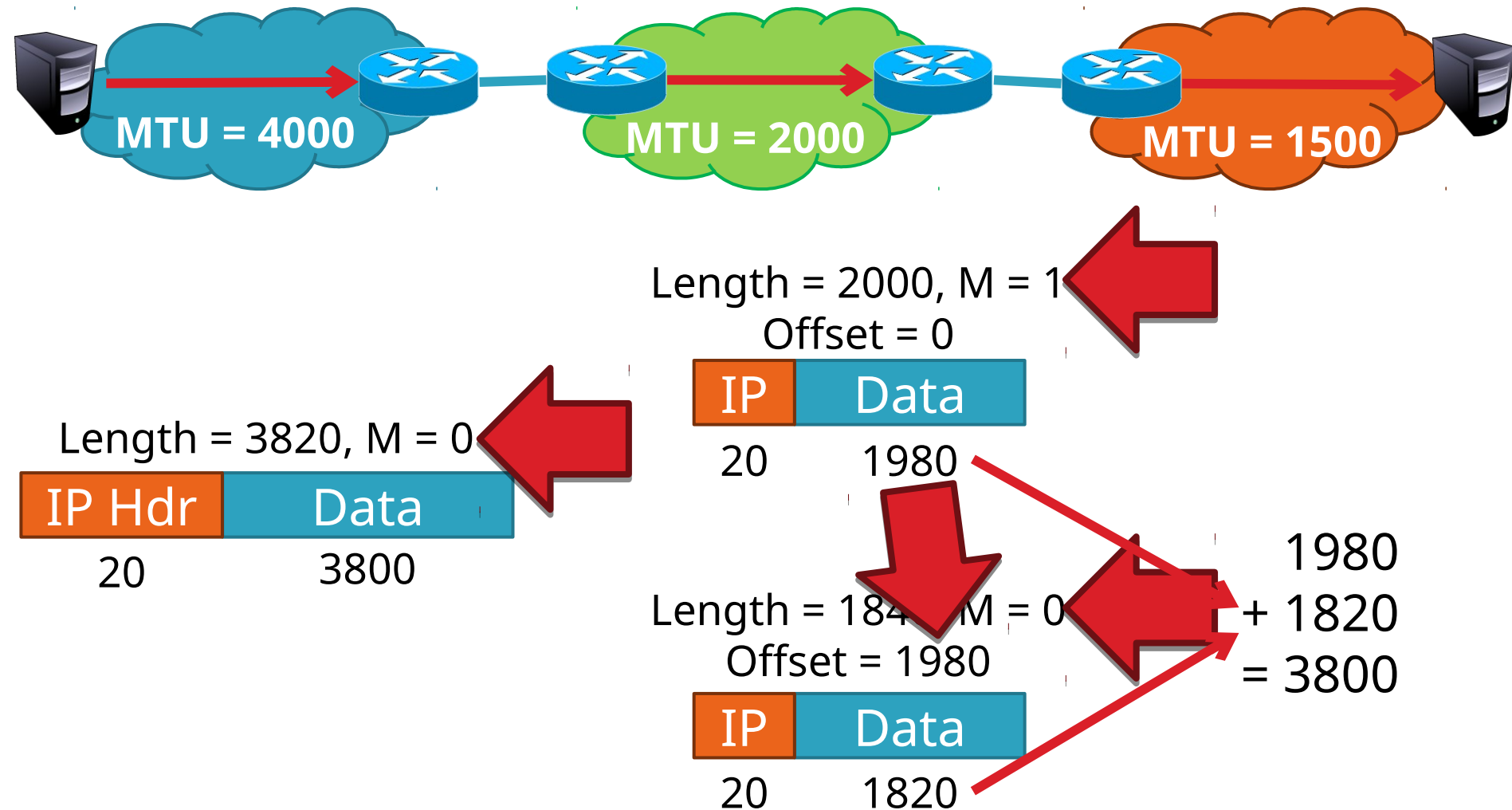
46

- Identifier: a unique number for the original datagram
- Flags: M flag, i.e. this is the last fragment
- Offset: byte position of the first byte in the fragment
  - Divided by 8



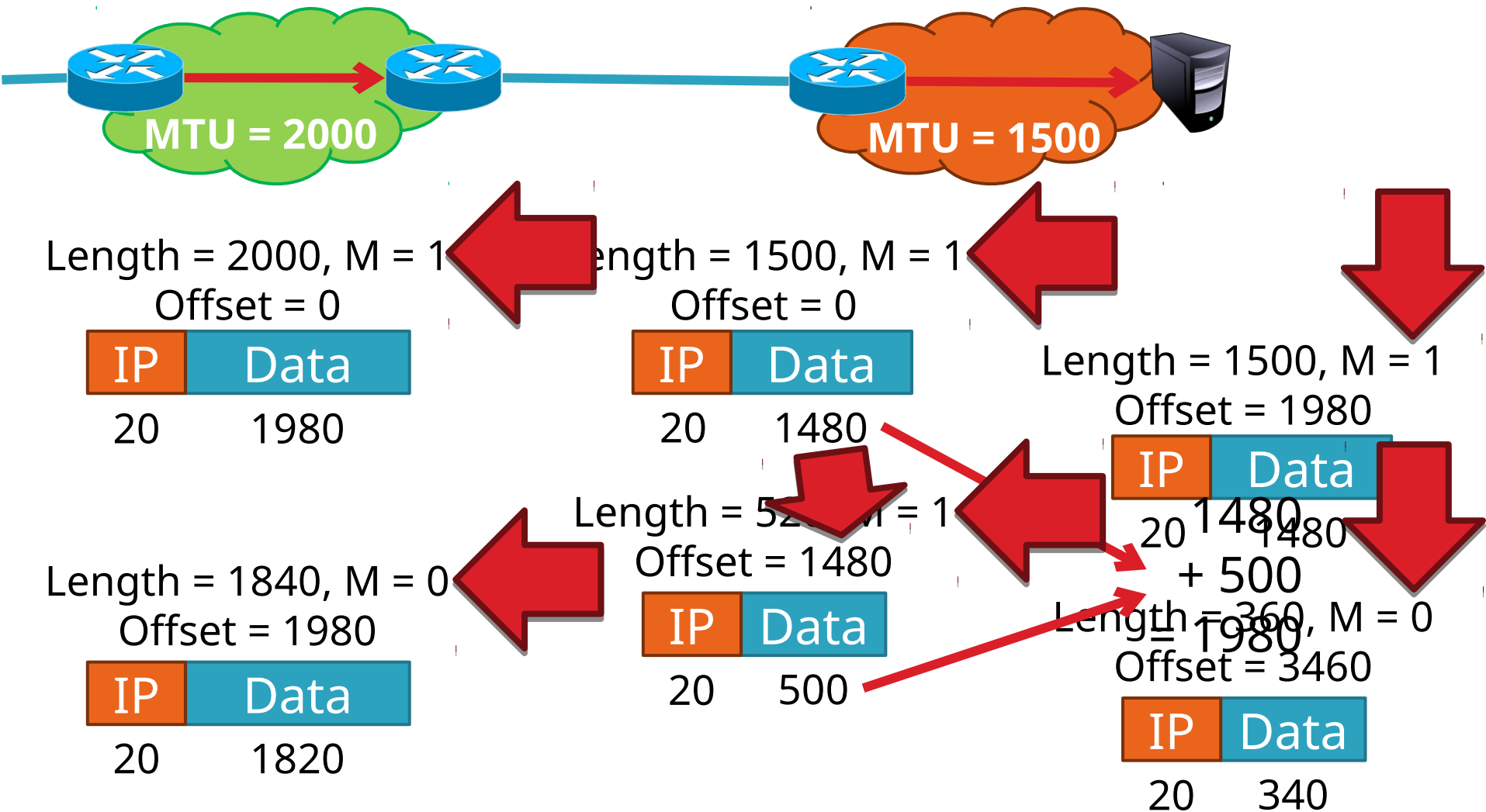
# Fragmentation Example

47



# Fragmentation Example

48



# IP Fragment Reassembly

49

Length = 1500, M = 1, Offset = 0

IP	Data
20	1480

Length = 520, M = 1, Offset = 1480

IP	Data
20	500

Length = 1500, M = 1, Offset = 1980

IP	Data
20	1480

Length = 360, M = 0, Offset = 3460

IP	Data
20	340

- Performed at destination
- M = 0 fragment gives us total data size
  - $360 - 20 + 3460 = 3800$
- Challenges:
  - Out-of-order fragments
  - Duplicate fragments
  - Missing fragments
- Basically, memory management



# Fragmentation Concepts

50

- Highlights many key Internet characteristics
  - Decentralized and heterogeneous
    - Each network may choose its own MTU
  - Connectionless datagram protocol
    - Each fragment contains full routing information
    - Fragments can travel independently, on different paths
  - Best effort network
    - Routers/receiver may silently drop fragments
    - No requirement to alert the sender
  - Most work is done at the endpoints
    - i.e. reassembly

# Fragmentation in Reality

51

- Fragmentation is expensive
  - Memory and CPU overhead for datagram reconstruction
  - Want to avoid fragmentation if possible
- MTU discovery protocol
  - Send a packet with “don’t fragment” bit set
  - Keep decreasing message length until one arrives
  - May get “can’t fragment” error from a router, which will explicitly state the supported MTU
- Router handling of fragments
  - Fast, specialized hardware handles the common case
  - Dedicated, general purpose CPU just for handling fragments

- ❑ Addressing
  - ❑ Class-based
  - ❑ CIDR
- ❑ IPv4 Protocol Details
  - ❑ Packed Header
  - ❑ Fragmentation
- ❑ IPv6

# The IPv4 Address Space Crisis

53

- Problem: the IPv4 address space is too small
  - $2^{32} = 4,294,967,296$  possible addresses
  - Less than one IP per person
- Parts of the world have already run out of addresses
  - IANA assigned the last /8 block of addresses in 2011

Region	Regional Internet Registry (RIR)	Exhaustion Date
Asia/Pacific	APNIC	April 19, 2011
Europe/Middle East	RIPE	September 14, 2012
North America	ARIN	13 Jan 2015 (Projected)
South America	LACNIC	13 Jan 2015 (Projected)
Africa	AFRINIC	17 Jan 2022(Projected)

# IPv6

54

- IPv6, first introduced in 1998(!)
  - 128-bit addresses
  - $4.8 * 10^{28}$  addresses per person
- Address format
  - 8 groups of 16-bit values, separated by ':'
  - Leading zeroes in each group may be omitted
  - Groups of zeroes can be omitted using '::'

2001:0db8:0000:0000:0000:ff00:0042:8329

2001:0db8:0:0:0:ff00:42:8329

2001:0db8::ff00:42:8329

# IPv6 Trivia

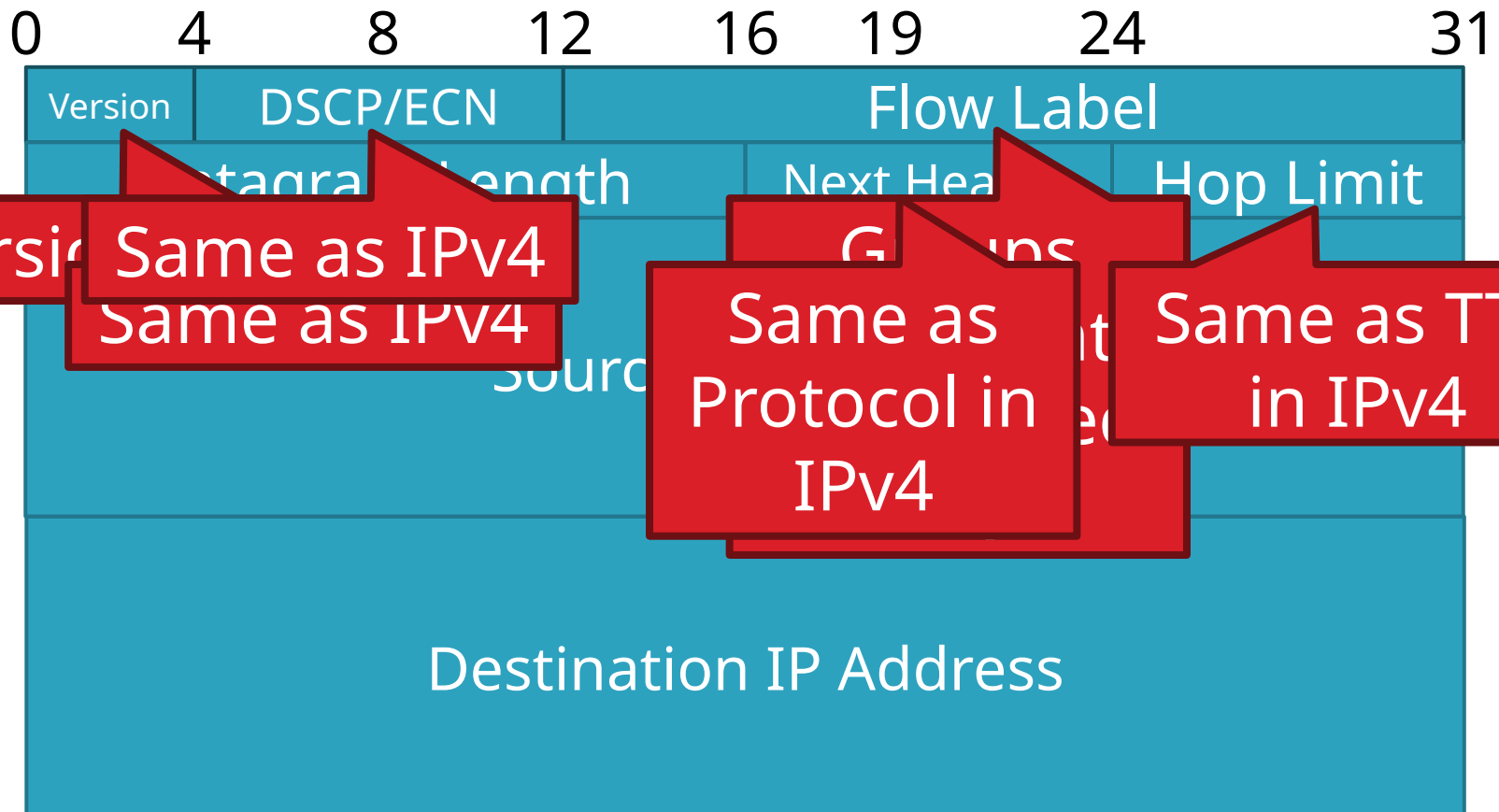
55

- Who knows the IP for localhost?
  - 127.0.0.1
  
- What is localhost in IPv6?
  - ::1

# IPv6 Header

56

- Double the size of IPv4 (320 bits vs. 160 bits)



# Differences from IPv4 Header

57

- Several header fields are missing in IPv6
  - Header length – rolled into Next Header field
  - Checksum – was useless, so why keep it
  - Identifier, Flags, Offset
    - IPv6 routers do not support fragmentation
    - Hosts are expected to use path MTU discovery
- Reflects changing Internet priorities
  - Today's networks are more homogeneous
  - Instead, routing cost and complexity dominate



# Performance Improvements

58

- ❑ No checksums to verify
- ❑ No need for routers to handle fragmentation
- ❑ Simplified routing table design
  - Address space is huge
  - No need for CIDR (but need for aggregation)
  - Standard subnet size is  $2^{64}$  addresses
- ❑ Simplified auto-configuration
  - Neighbor Discovery Protocol
  - Used by hosts to determine network ID
  - Host ID can be random!

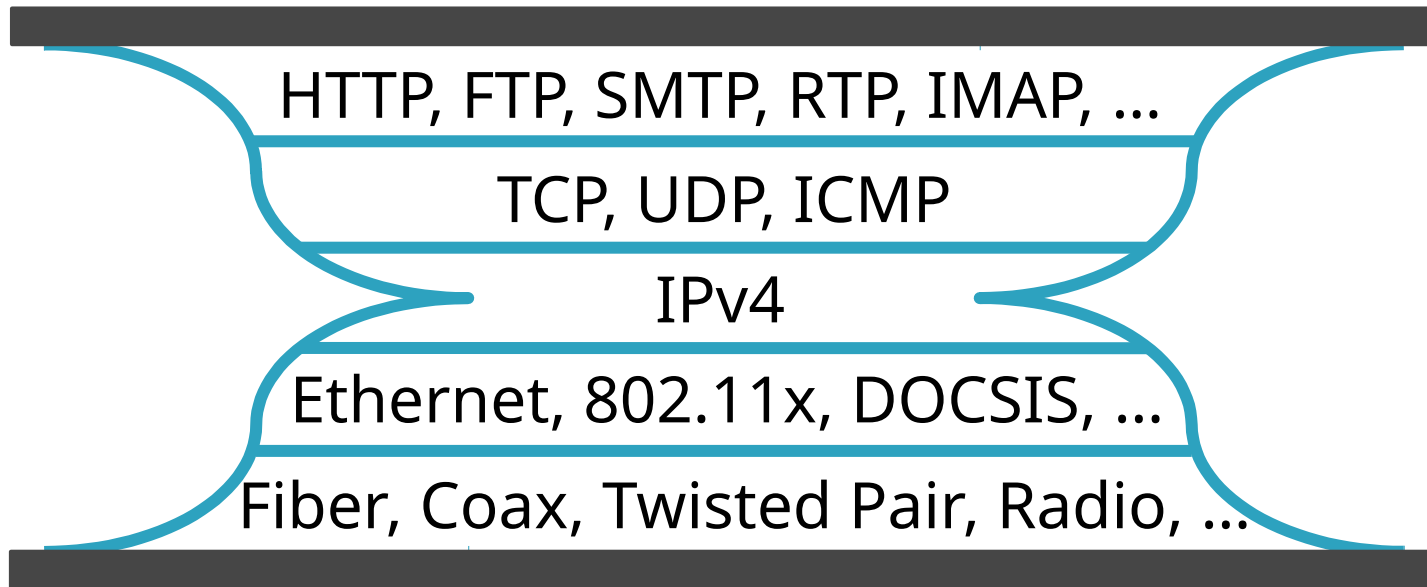
# Additional IPv6 Features

59

- Source Routing
  - Host specifies the route to wants packet to take
- Mobile IP
  - Hosts can take their IP with them to other networks
  - Use source routing to direct packets
- Privacy Extensions
  - Randomly generate host identifiers
  - Make it difficult to associate one IP to a host
- Jumbograms
  - Support for 4Gb datagrams

# Deployment Challenges

60



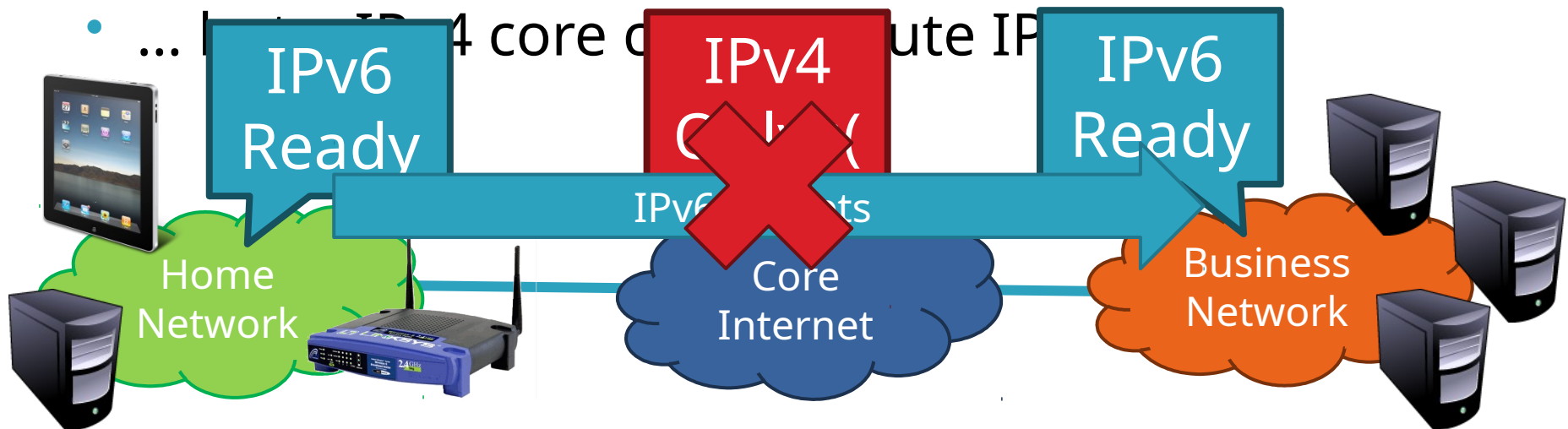
- Switching to IPv6 is a whole-Internet upgrade
  - All routers, all hosts
  - ICMPv6, DHCPv6, DNSv6
- 2013: 0.94% of Google traffic was IPv6, 2.5% today

# Transitioning to IPv6

61

## □ How do we ease the transition from IPv4 to IPv6?

- Today, most network edges are IPv6 ready
  - Windows/OSX/iOS/Android all support IPv6
  - Your wireless access point probably supports IPv6
- The Internet core is hard to upgrade



# Transition Technologies

62

- How do you route IPv6 packets over an IPv4 Internet?
- Transition Technologies
  - Use **tunnels** to **encapsulate** and route IPv6 packets over the IPv4 Internet
  - Several different implementations
    - 6to4
    - IPv6 Rapid Deployment (6rd)
    - Teredo
    - ... etc.

# Network Layer, Control Plane

63

## Data Plane

Application

Presentation

Session

Transport

Network

Data Link

Physical

## Function:

- Set up routes within a single network

## Key challenges:

- Distributing and updating routes
- Convergence time
- Avoiding loops



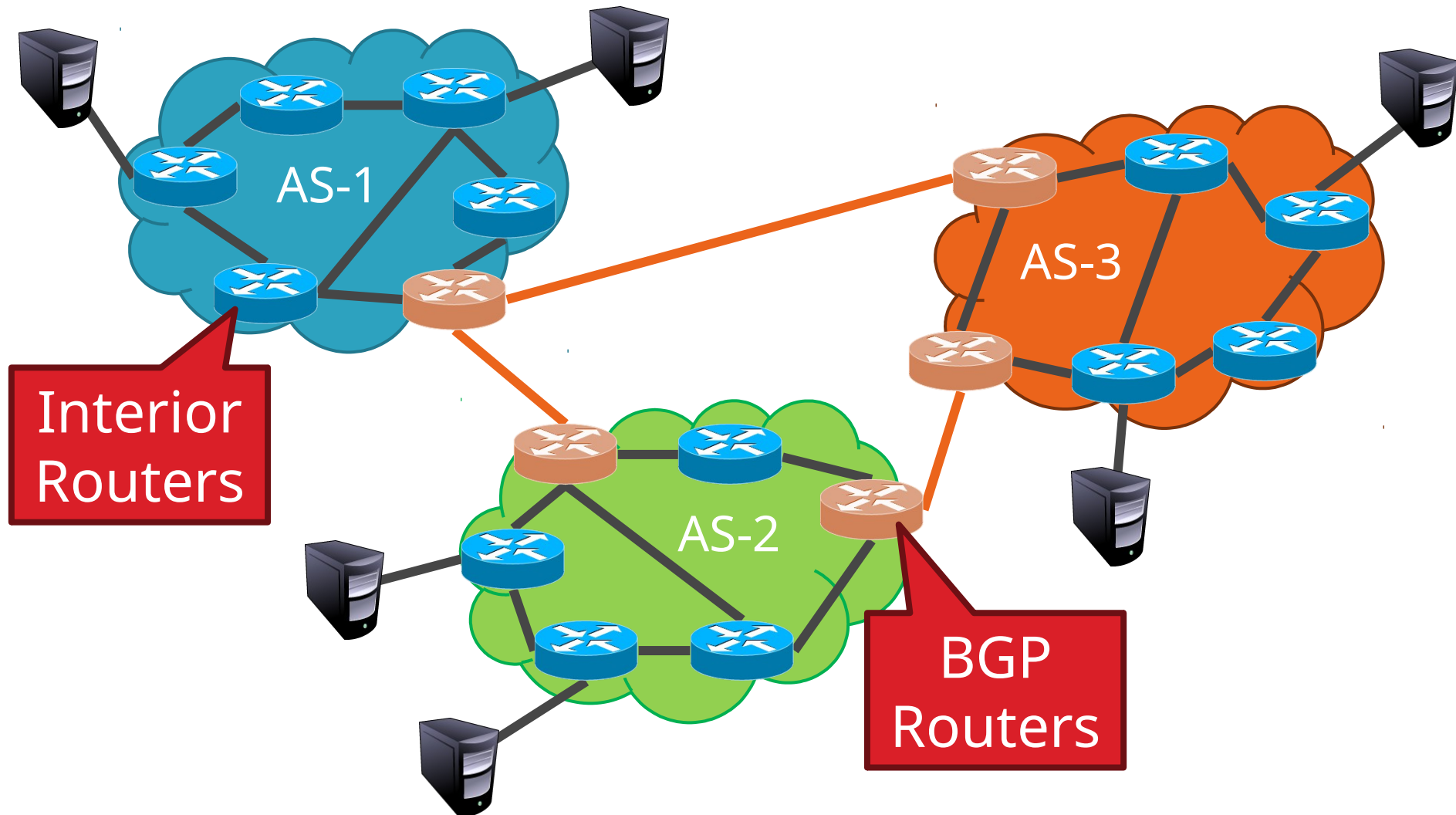
# Internet Routing

64

- Internet organized as a **two** level hierarchy
- First level – autonomous systems (AS's)
  - AS – region of network under a single administrative domain
  - Examples: Comcast, AT&T, Verizon, Sprint, etc.
- AS's use **intra-domain** routing protocols internally
  - Distance Vector, e.g., Routing Information Protocol (RIP)
  - Link State, e.g., Open Shortest Path First (OSPF)
- Connections between AS's use **inter-domain** routing protocols
  - Border Gateway Routing (BGP)
  - De facto standard today, BGP-4

# AS Example

65





# Why Do We Need ASs?

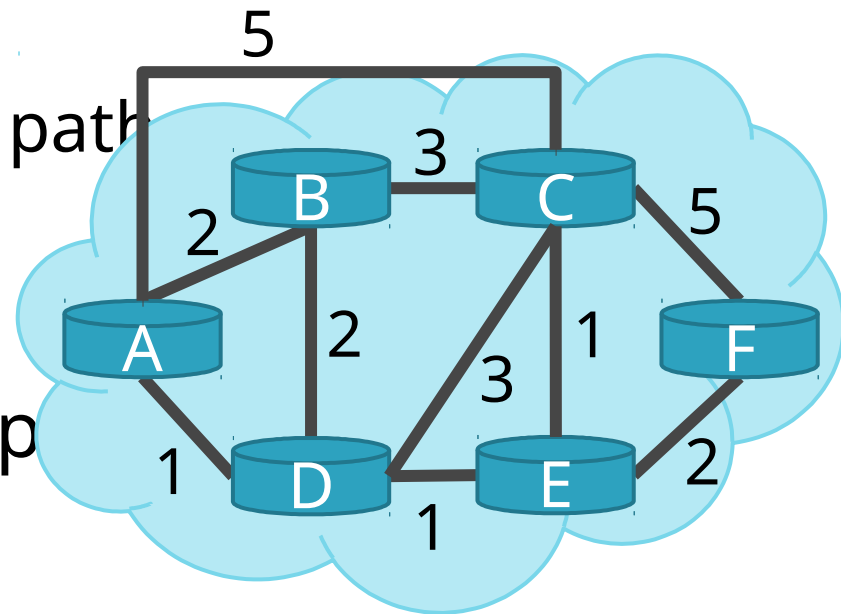
66

- ❑ Routing algorithms are not efficient enough to execute on the entire Internet topology
- ❑ Different routing algorithms
- ❑ Allow network autonomy
  - Easier to compute routes
  - Greater flexibility
  - More autonomy/independence
- ❑ Allows organizations to choose how to route across each other (BGP)

# Routing on a Graph

67

- Goal: determine a “good” path through the network from source to destination
- What is a good path?
  - Usually means the shortest path
  - Load balanced
  - Lowest \$\$\$ cost
- Network modeled as a graph
  - Routers □ nodes
  - Link □ edges
    - Edge cost: delay, congestion level, etc.



# Shortest Path Routing

68

1. Bellman-Ford Algorithm [Distance Vector]
2. Dijkstra's Algorithm [Link State]

*What does it mean to be the shortest (or optimal) route?*

- a. **Minimize mean packet delay**
- b. **Maximize the network throughput**
- c. **Minimize the number of hops along the path**

# Dijkstra's Shortest Path Algorithm

69

Initially mark all nodes (except source) with infinite distance.

working node = source node

Sink node = destination node

While the working node is not equal to the sink

1. Mark the working node as permanent.

2. Examine all adjacent nodes in turn

If the sum of label on working node plus distance from working node to adjacent node is less than current labeled distance on the adjacent node, this implies a shorter path. Relabel the distance on the adjacent node and label it with the node from which the probe was made.

3. Examine all tentative nodes (not just adjacent nodes) and mark the node with the smallest labeled value as permanent. This node becomes the new working node.

Reconstruct the path backwards from sink to source.

# Dijkstra's Algorithm

**Dijkstra(graph (G,w), vertex s)**

*InitializeSingleSource(G, s)*

$S \leftarrow \emptyset$

$Q \leftarrow V[G]$

**while**  $Q \neq \emptyset$  **do**

$u \leftarrow \text{ExtractMin}(Q)$

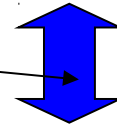
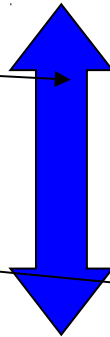
$S \leftarrow S \cup \{u\}$

**for**  $u \in \text{Adj}[u]$  **do**

*Relax(u,v,w)*

executed  $\Theta(V)$  times

$\Theta(E)$  times in total



*InitializeSingleSource(graph G, vertex s)*

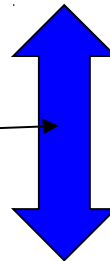
**for**  $v \in V[G]$  **do**

$d[v] \leftarrow \infty$

$p[v] \leftarrow 0$

$d[s] \leftarrow 0$

$\Theta(V)$



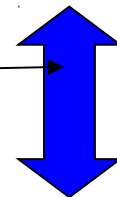
*Relax(vertex u, vertex v, weight w)*

**if**  $d[v] > d[u] + w(u,v)$  **then**

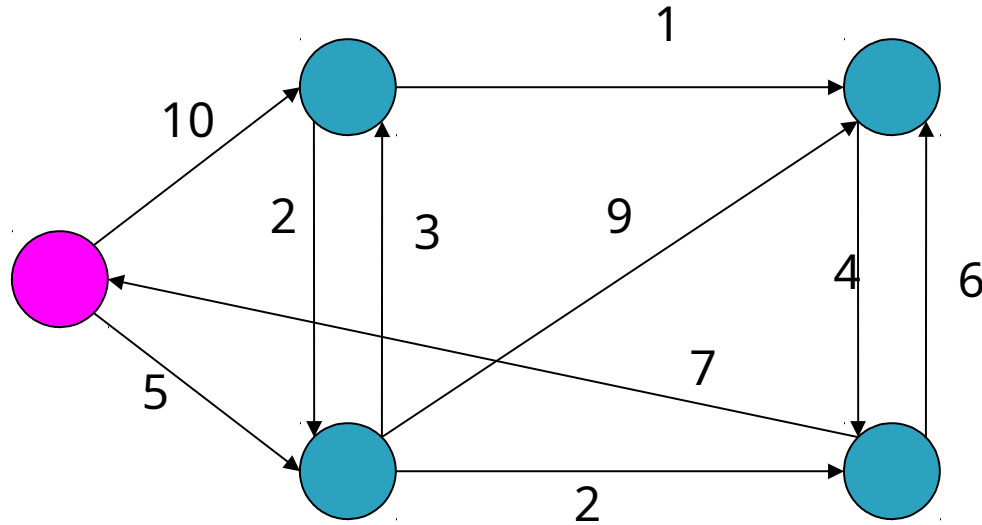
$d[v] \leftarrow d[u] + w(u,v)$

$p[v] \leftarrow u$

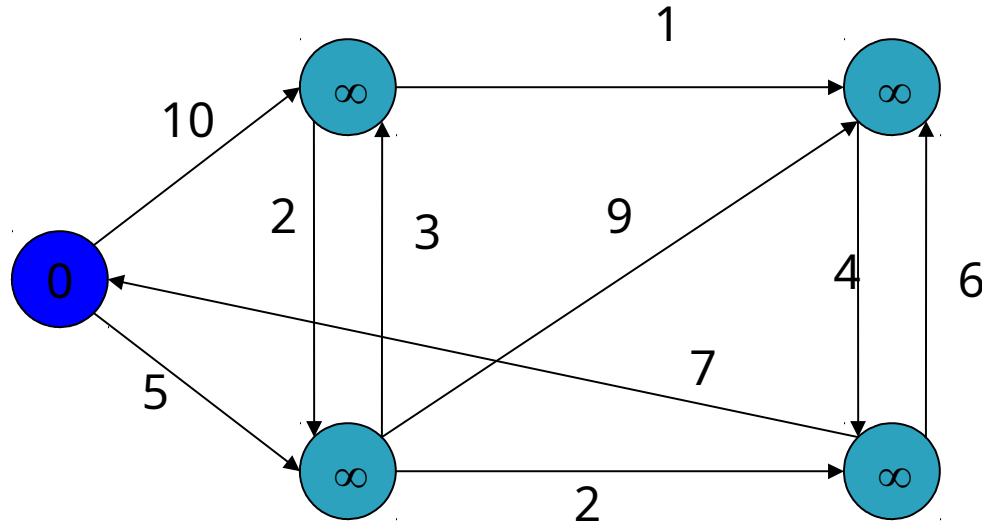
$\Theta(1)$  ?



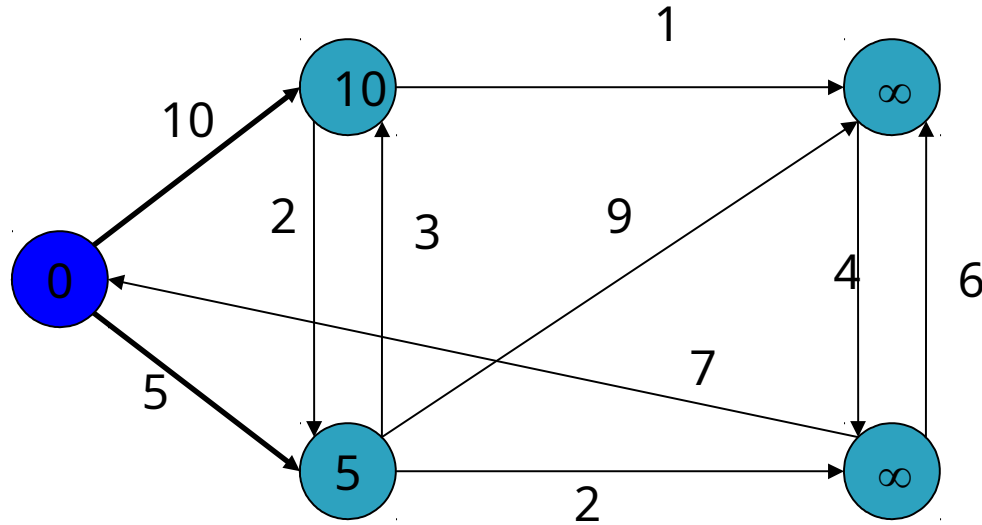
# Dijkstra's Algorithm - Example



# Dijkstra's Algorithm - Example

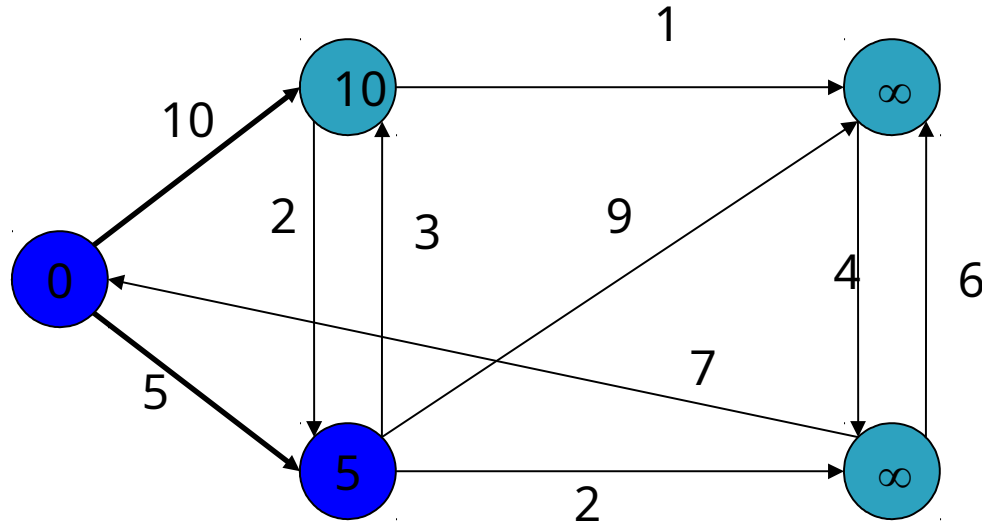


# Dijkstra's Algorithm - Example

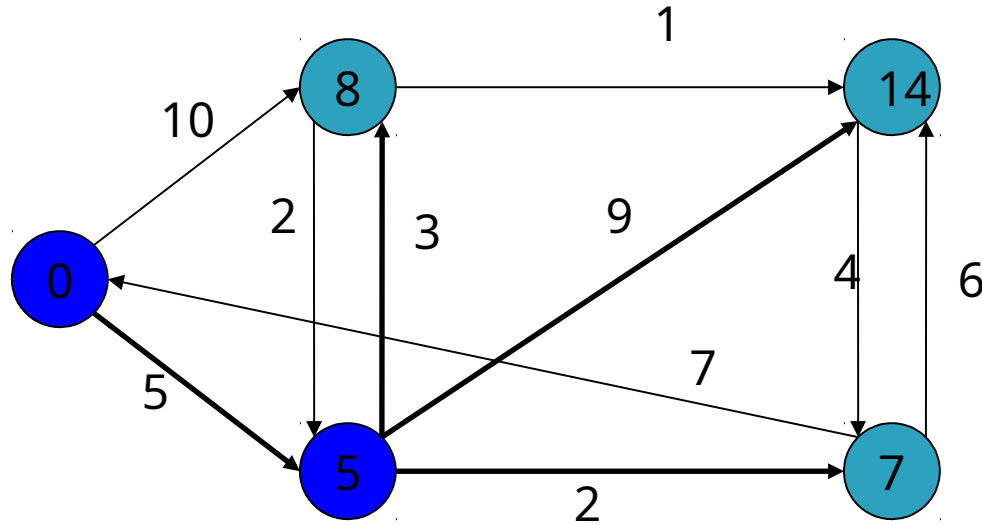




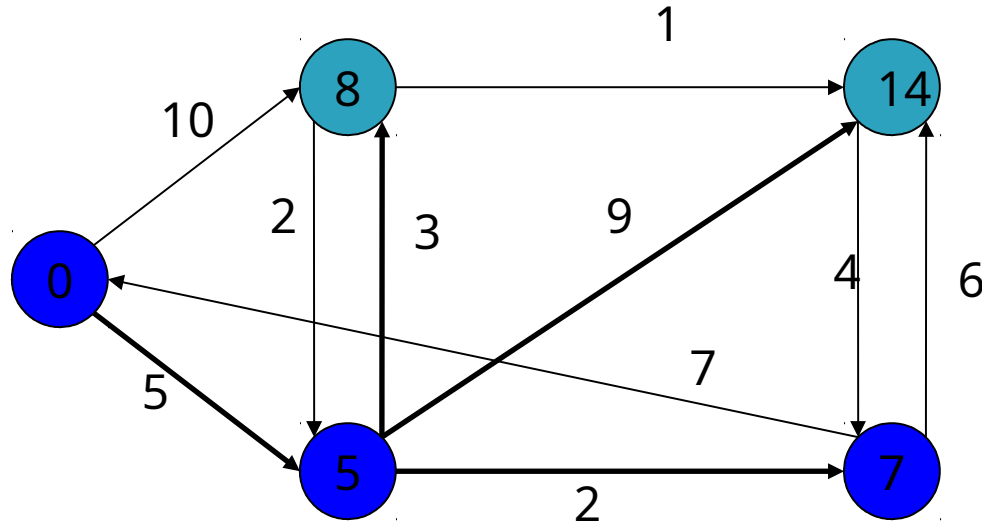
# Dijkstra's Algorithm - Example



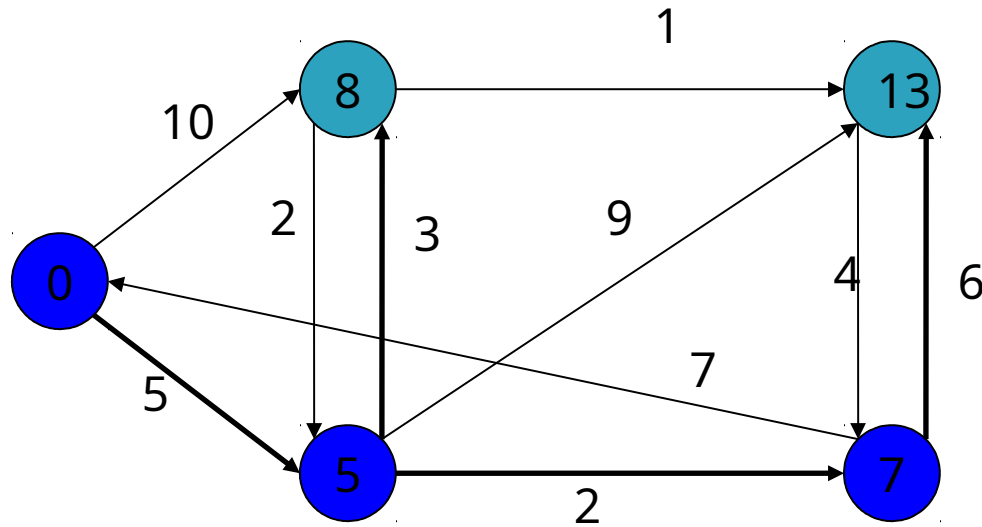
# Dijkstra's Algorithm - Example



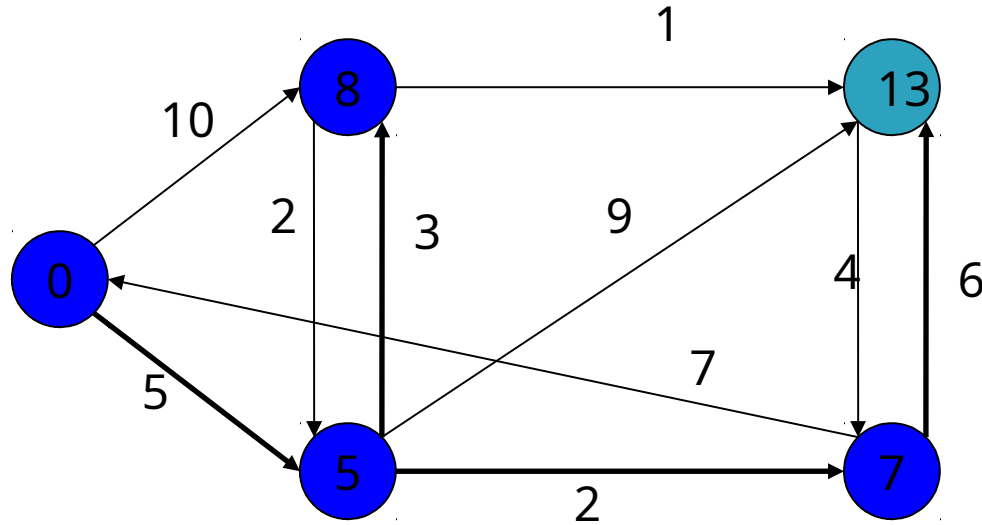
# Dijkstra's Algorithm - Example



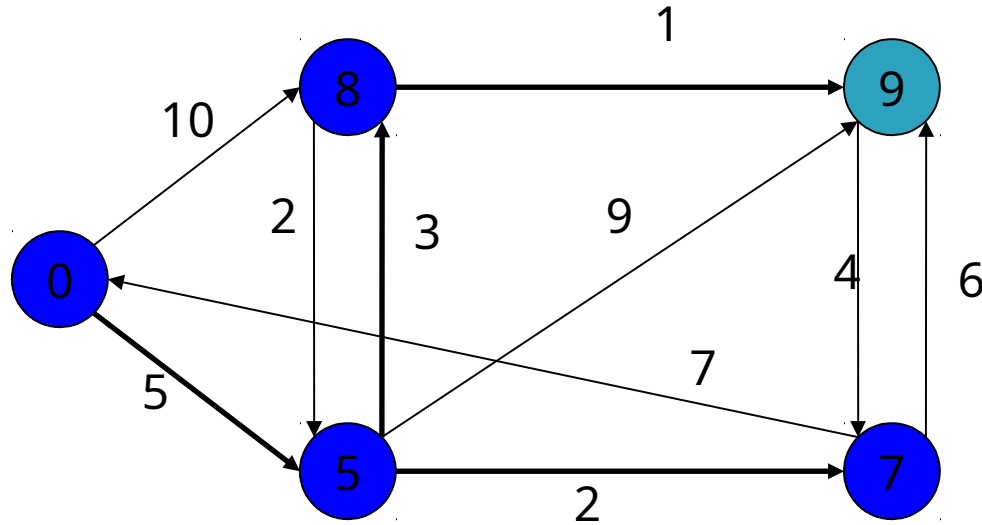
# Dijkstra's Algorithm - Example



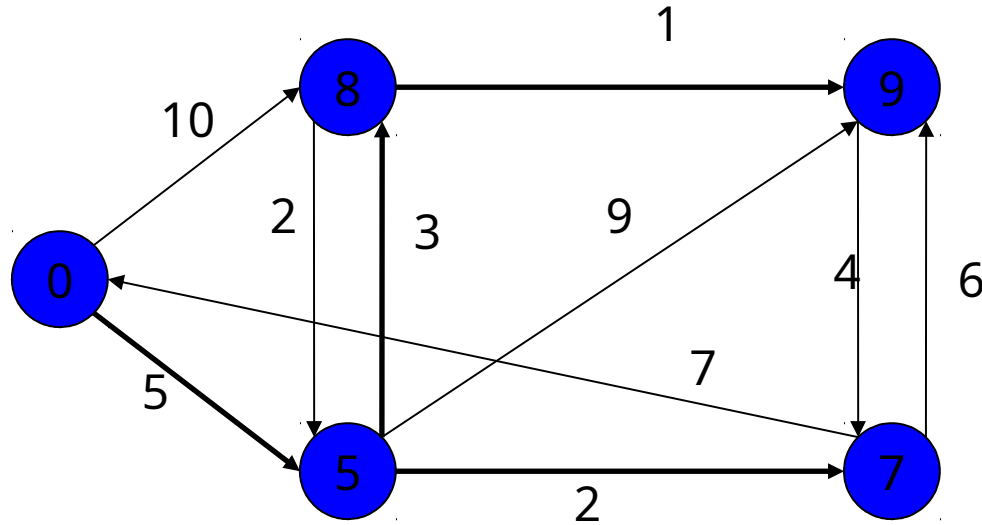
# Dijkstra's Algorithm - Example



# Dijkstra's Algorithm - Example



# Dijkstra's Algorithm - Example

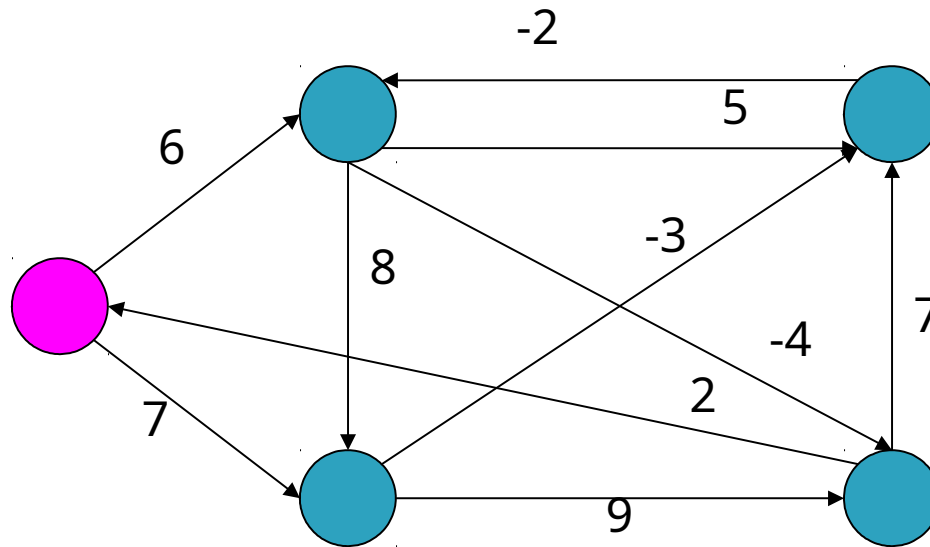


# Bellman-Ford Algorithm

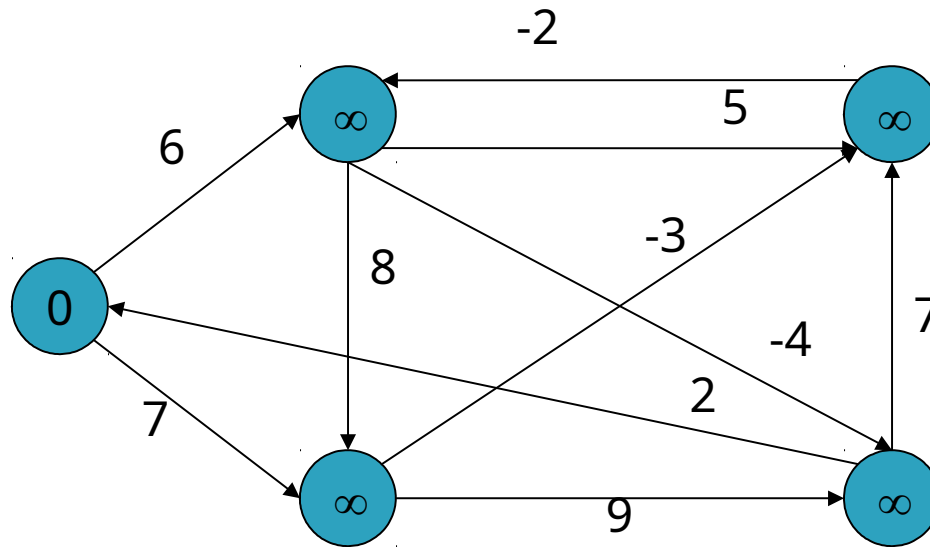
```
BellmanFord(graph (G,w), vertex s)  
  InitializeSingleSource(G, s)  
  for  $i \leftarrow 1$  to  $|V[G] - 1|$  do  
    for  $(u,v) \in E[G]$  do  
      Relax(u,v,w)  
  for  $(u,v) \in E[G]$  do  
    if  $d[v] > d[u] + w(u,v)$  then  
      return false  
  return true
```



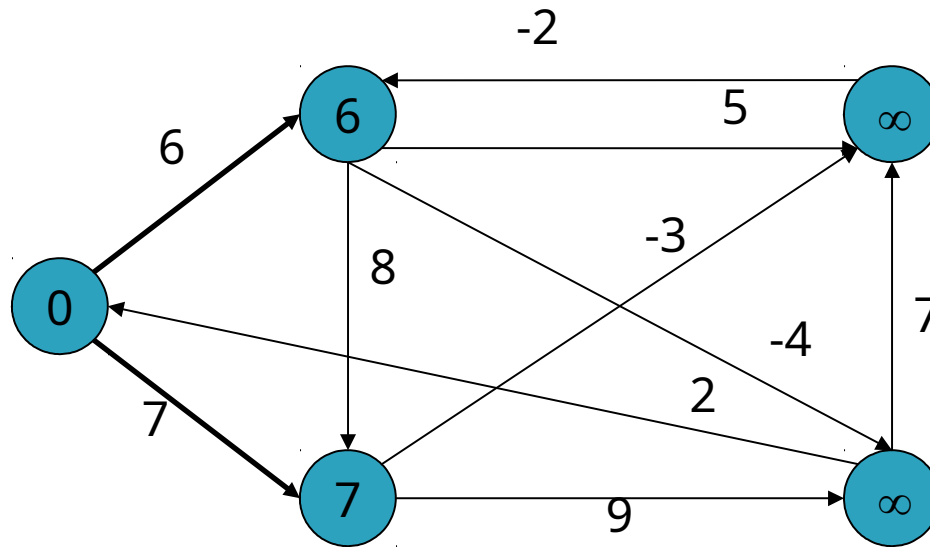
# Bellman-Ford Algorithm - Example



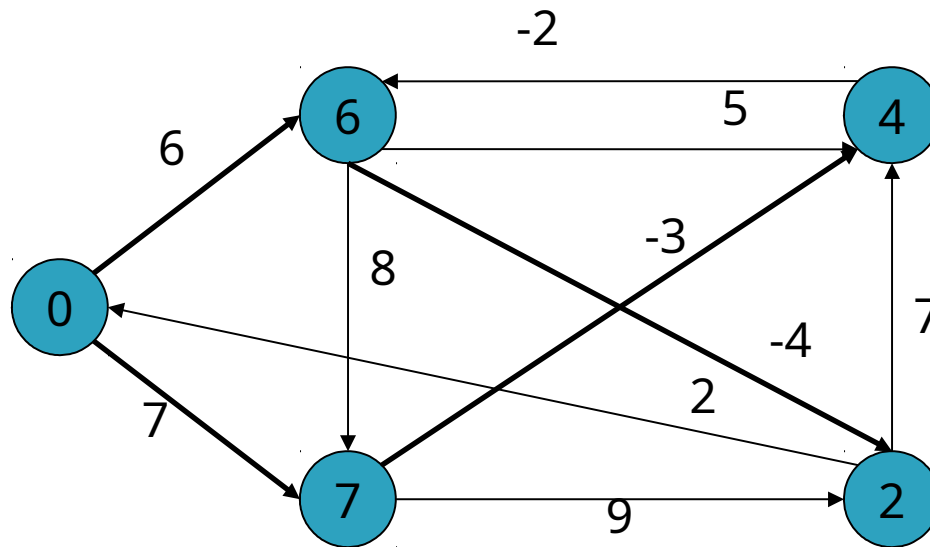
# Bellman-Ford Algorithm - Example



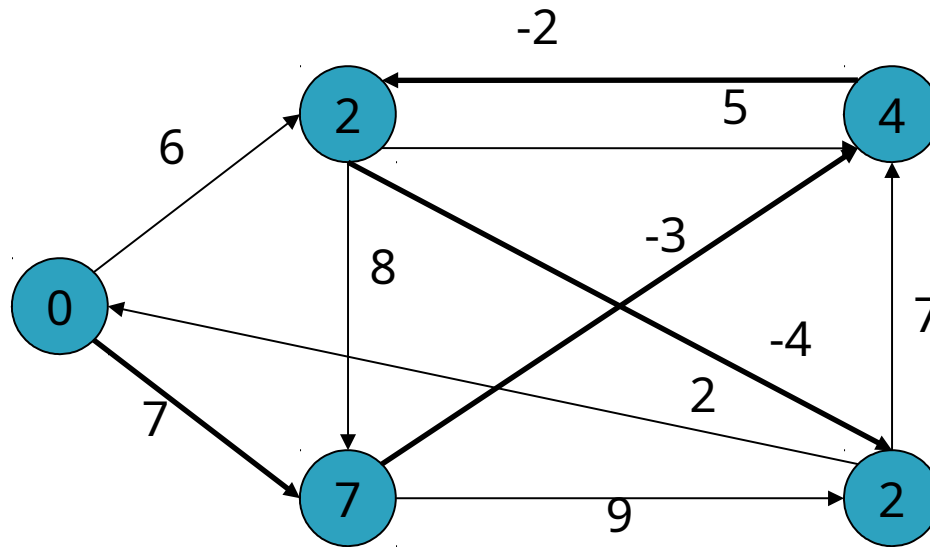
# Bellman-Ford Algorithm - Example



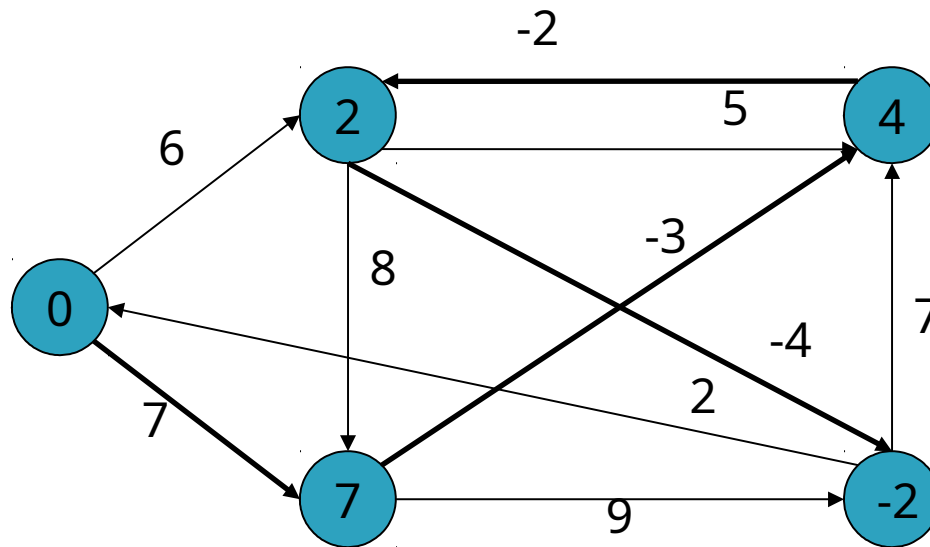
# Bellman-Ford Algorithm - Example



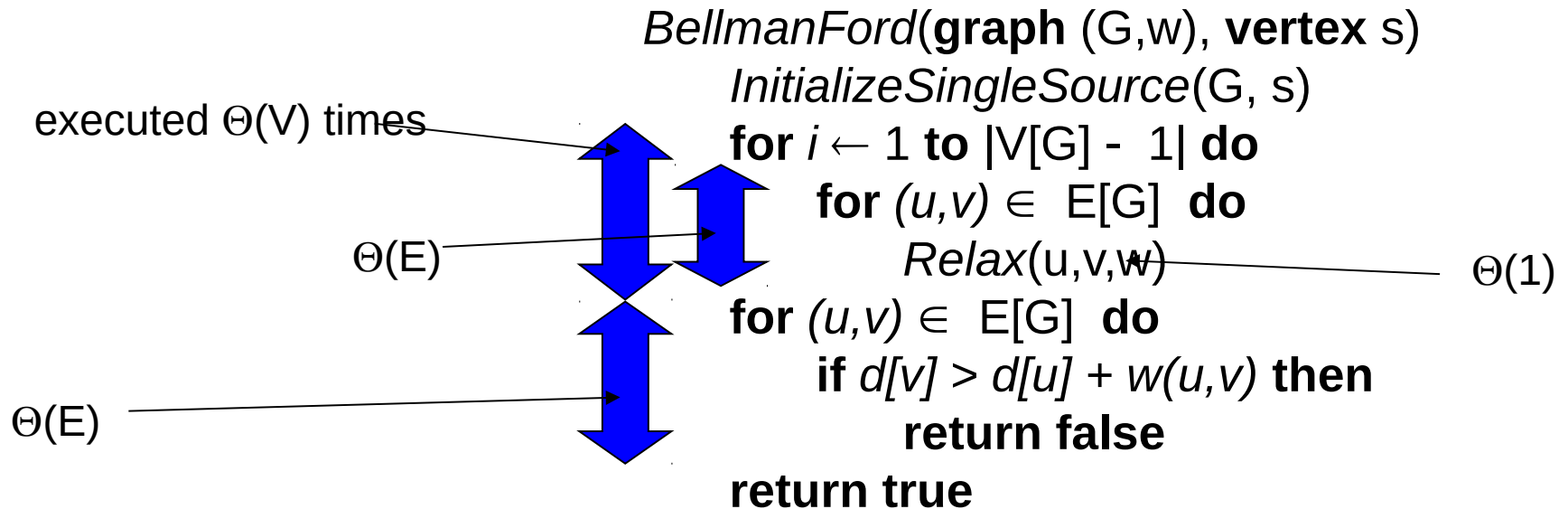
# Bellman-Ford Algorithm - Example



# Bellman-Ford Algorithm - Example

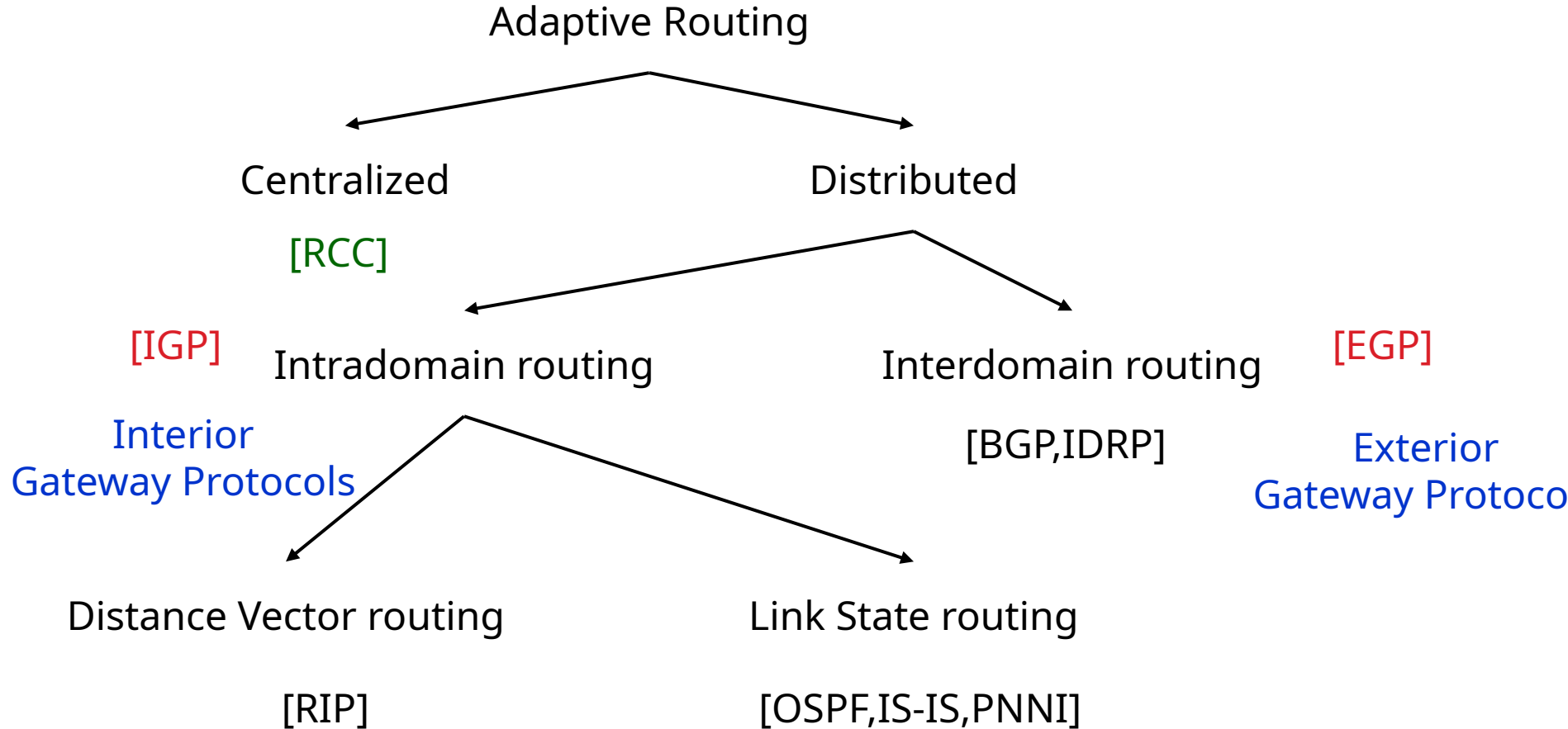


# Bellman-Ford Algorithm - Complexity



# Internetwork Routing [Halsall]

89

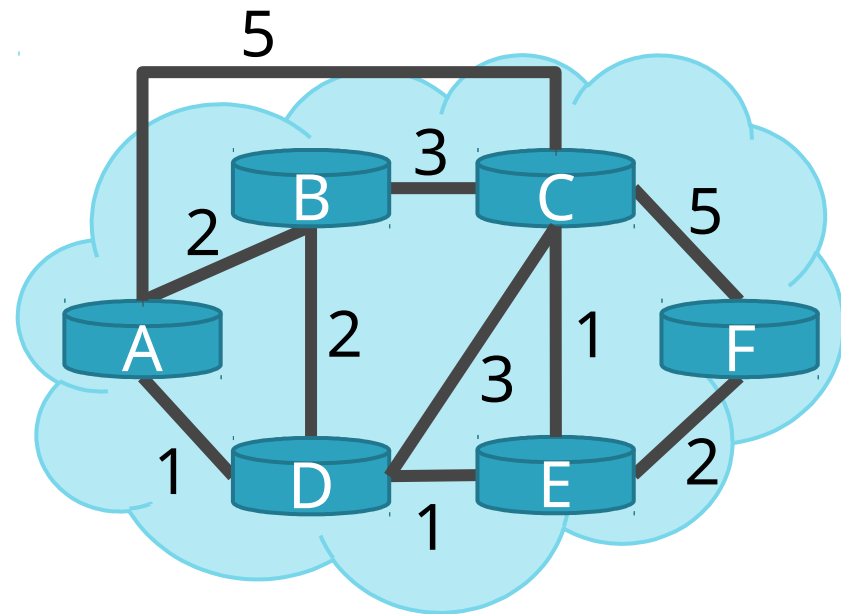




# Routing Problems

90

- Assume
  - A network with  $N$  nodes
  - Each node only knows
    - Its immediate neighbors
    - The cost to reach each neighbor
- How does each node learn the shortest path to every other node?



# Intra-domain Routing Protocols

91

## □ Distance vector

- Routing Information Protocol (RIP), based on Bellman-Ford
- Routers periodically exchange reachability information with neighbors

## □ Link state

- Open Shortest Path First (OSPF), based on Dijkstra
- Each network periodically **floods** immediate reachability information to all other routers
- Per router local computation to determine full routes

- ❑ Distance Vector Routing
  - ❑ RIP
- ❑ Link State Routing
  - ❑ OSPF
  - ❑ IS-IS

# Distance Vector Routing

93

- What is a distance vector?
  - Current best known cost to reach a destination
- Idea: exchange vectors among neighbors to learn about lowest cost paths

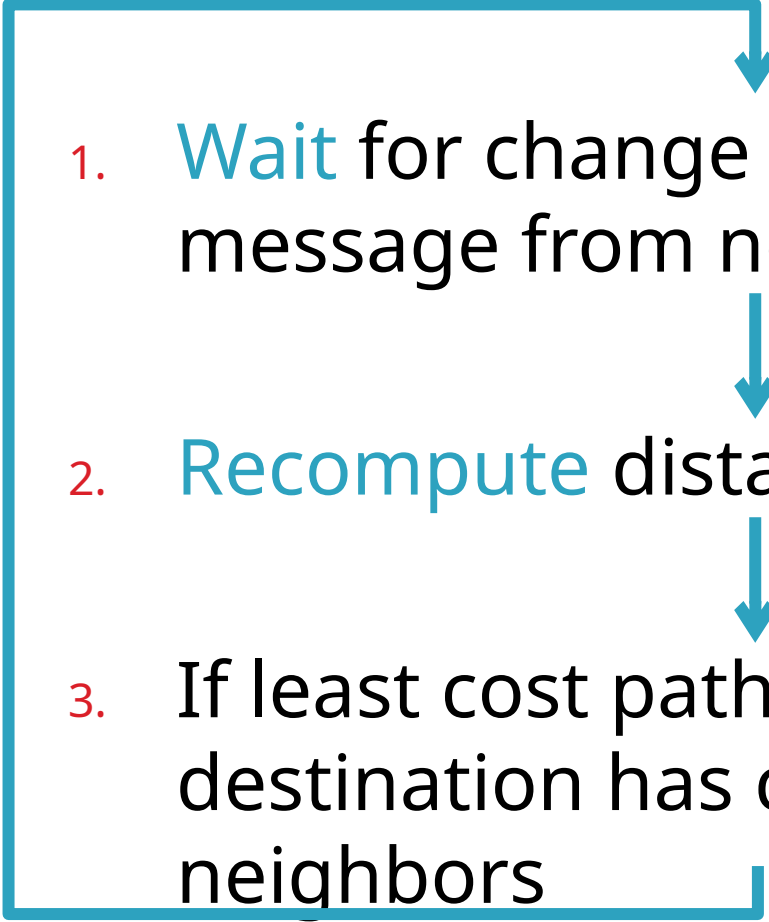
DV Table  
at Node C

Destination	Cost
A	7
B	1
D	2
E	5
F	1

- No entry for C
- Initially, only has info for immediate neighbors
  - Other destinations cost =  $\infty$
- Routing Information Protocol (RIP) eventually, vector is

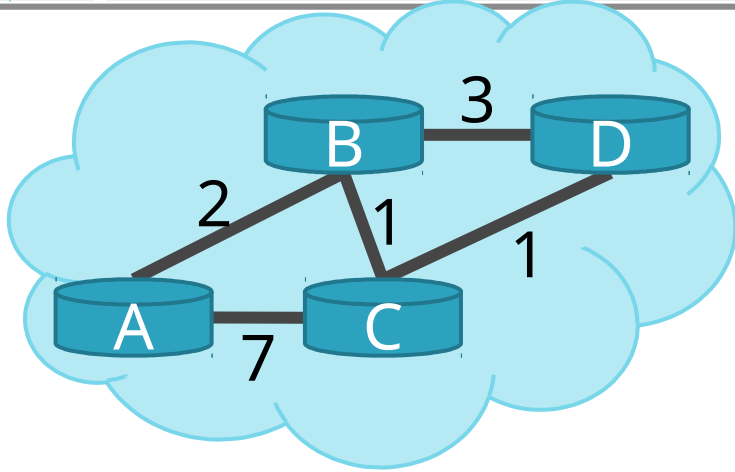
# Distance Vector Routing Algorithm

94

- 
- ```
graph TD; A[ ] --> B[1. Wait for change in local link cost or message from neighbor]; B --> C[2. Recompute distance table]; C --> D[3. If least cost path to any destination has changed, notify neighbors]; D --> A;
```
1. **Wait** for change in local link cost or message from neighbor
  2. **Recompute** distance table
  3. If least cost path to any destination has changed, **notify** neighbors

# Distance Vector Initialization

95



Node A

| Dest. | Cost     | Next |
|-------|----------|------|
| B     | 2        | B    |
| C     | 7        | C    |
| D     | $\infty$ |      |

Node B

| Dest. | Cost | Next |
|-------|------|------|
| A     | 2    | A    |
| C     | 1    | C    |
| D     | 3    | D    |

Node C

| Dest. | Cost | Next |
|-------|------|------|
| A     | 7    | A    |
| B     | 1    | B    |
| D     | 1    | D    |

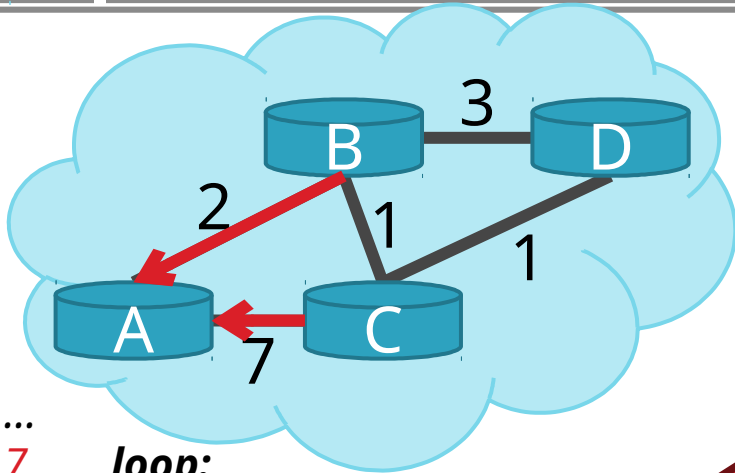
Node D

| Dest. | Cost     | Next |
|-------|----------|------|
| A     | $\infty$ |      |
| B     | 3        | B    |
| C     | 1        | C    |

1. **Initialization:**
2.   **for all** neighbors  $V$  **do**
3.     **if**  $V$  adjacent to  $A$
4.        $D(A, V) = c(A, V)$ ;
5.   **else**
6.      $D(A, V) = \infty$ ;
- ...

# Distance Vector: 1<sup>st</sup> Iteration

96



Node A

| Dest. | Cost | Next |
|-------|------|------|
| B     | 2    | B    |
| C     | 3    | B    |
| D     | 5    | B    |

Node B

| Dest. | Cost | Next |
|-------|------|------|
| A     | 2    | A    |
| C     | 1    | C    |
| D     | 2    | C    |



```

...
7.  loop:
...
12. else if (update D(V, Y) received from neighbor V)
13.   for all destinations Y
14.     if (destination Y is not V)
15.       D(A, Y) = min(D(A, Y), D(A, V) + D(V, Y))
16.     else
17.       D(A, Y) = D(A, V)
18.   if (there is a new min. for dest. Y)
19.     send D(A, Y) to all neighbors
20. forever
    
```

$$D(A, C) = \min(D(A, C), D(A, B) + D(B, C))$$

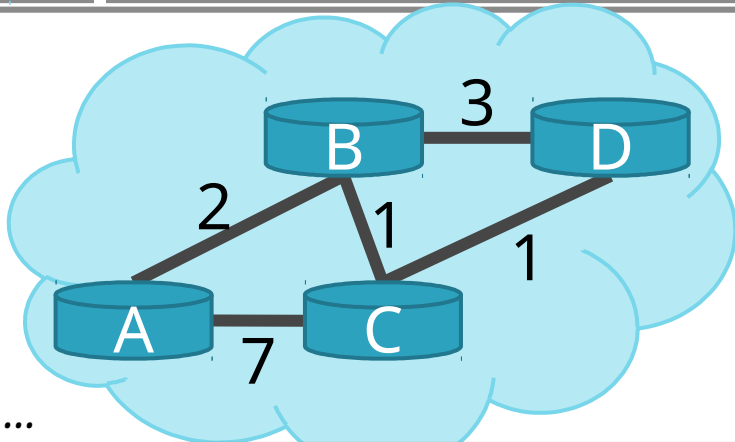
$$D(A, D) = \min(D(A, D), D(A, B) + D(B, D))$$

$$= \min(8, 3 + 3) = 5$$

| Dest. | Cost | Next |
|-------|------|------|
| B     | 1    | B    |
| D     | 1    | D    |

# Distance Vector: End of 3<sup>rd</sup> Iteration

97



Node A

| Dest. | Cost | Next |
|-------|------|------|
| B     | 2    | B    |
| C     | 3    | B    |
| D     | 4    | B    |

Node B

| Dest. | Cost | Next |
|-------|------|------|
| A     | 2    | A    |
| C     | 1    | C    |
| D     | 2    | C    |

- Nothing changes, algorithm terminates
- Until something changes...

| Dest. | Cost | Next | Dest. | Cost | Next |
|-------|------|------|-------|------|------|
| A     | 3    | B    | A     | 4    | C    |
| B     | 1    | B    | B     | 2    | C    |
| D     | 1    | D    | C     | 1    | C    |

$D(A, Y) =$

$\min(D(A, Y),$   
 $D(A, V) + D(V, Y));$

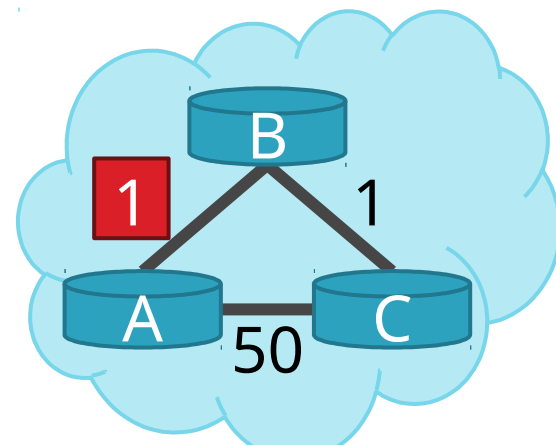
**if** (there is a new min. for dest. Y)

**send**  $D(A, Y)$  to all neighbors

**forever**



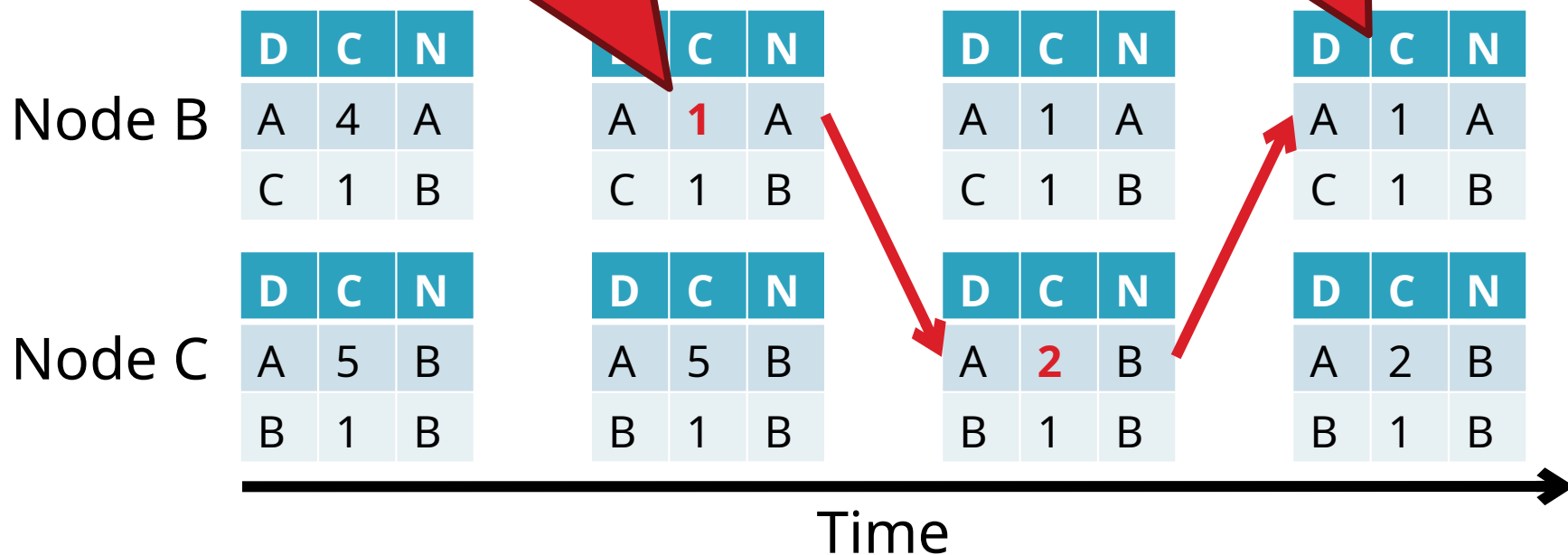
7. **loop:**
8. **wait** (link cost update or update message)
9. **if** ( $c(A, V)$  changes by  $d$ )
10. **for all** destinations  $Y$  through  $V$  **do**
11.  $D(A, Y) = D(A, Y) + d$
12. **else if** (update  $D(V, Y)$  received from  $V$ )
13. **for all** destinations  $Y$  **do**
14. **if** (destination  $Y$  through  $V$ )
15.  $D(A, Y) = D(A, V) + D(V, Y);$
16. **else**
17.  $D(A, Y) = \min(D(A, Y), D(A, V) + D(V, Y));$



Link Cost  
Algorithm

Good news travels fast

Algorithm  
terminates

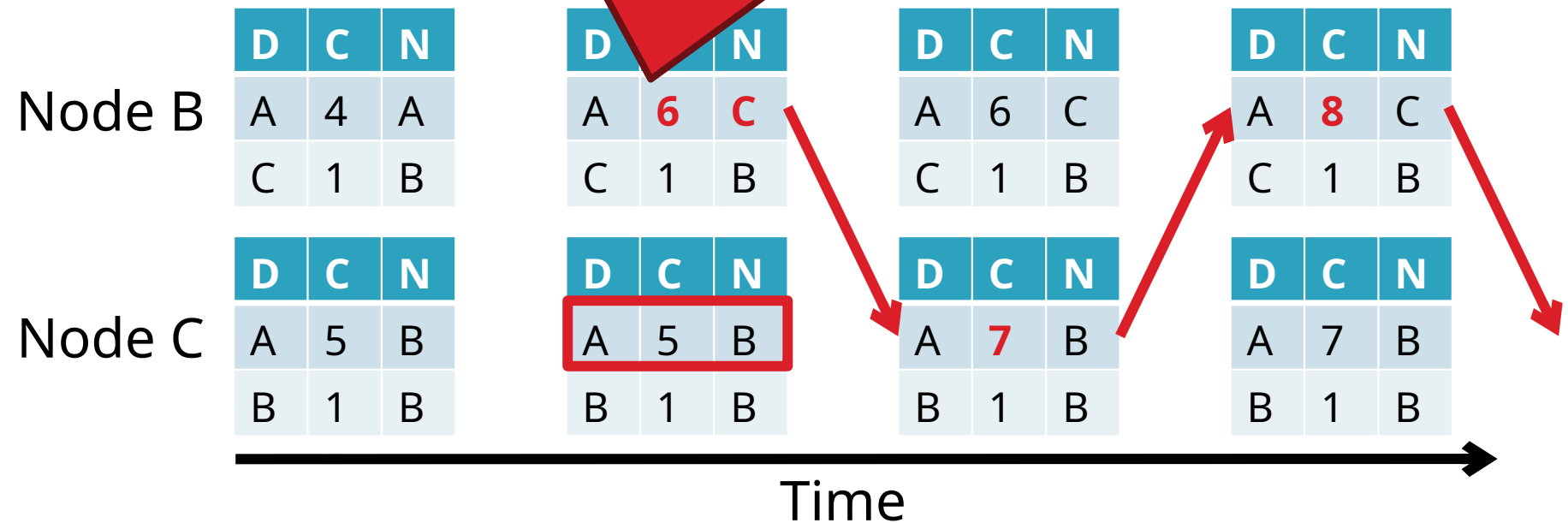
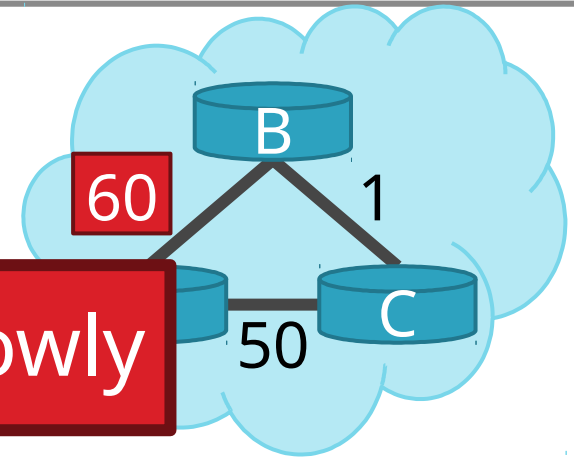


# Count to Infinity Problem

99

- Node B knows  $D(C, A) = 5$
- However, B does not know the path:  $C \rightarrow B \rightarrow A$
- Thus,  $D(B, A) = 4$

Bad news travels slowly



# Poisoned Reverse

100

- If C routes through B to get to A
  - C tells B that  $D(C, A) = \infty$
  - Thus, B won't route to A via C

