

Computer Networks

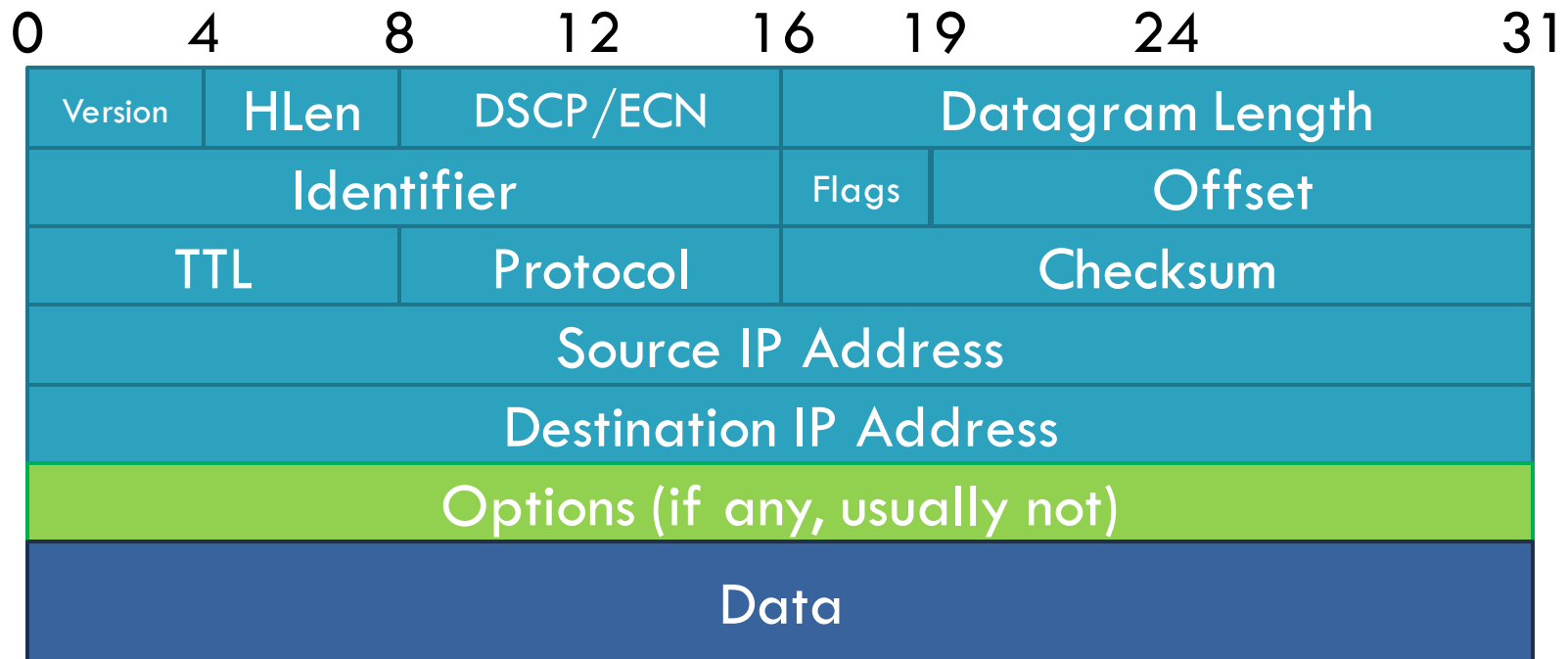
Lecture 10: Network layer Part 2

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

IP Datagrams

2

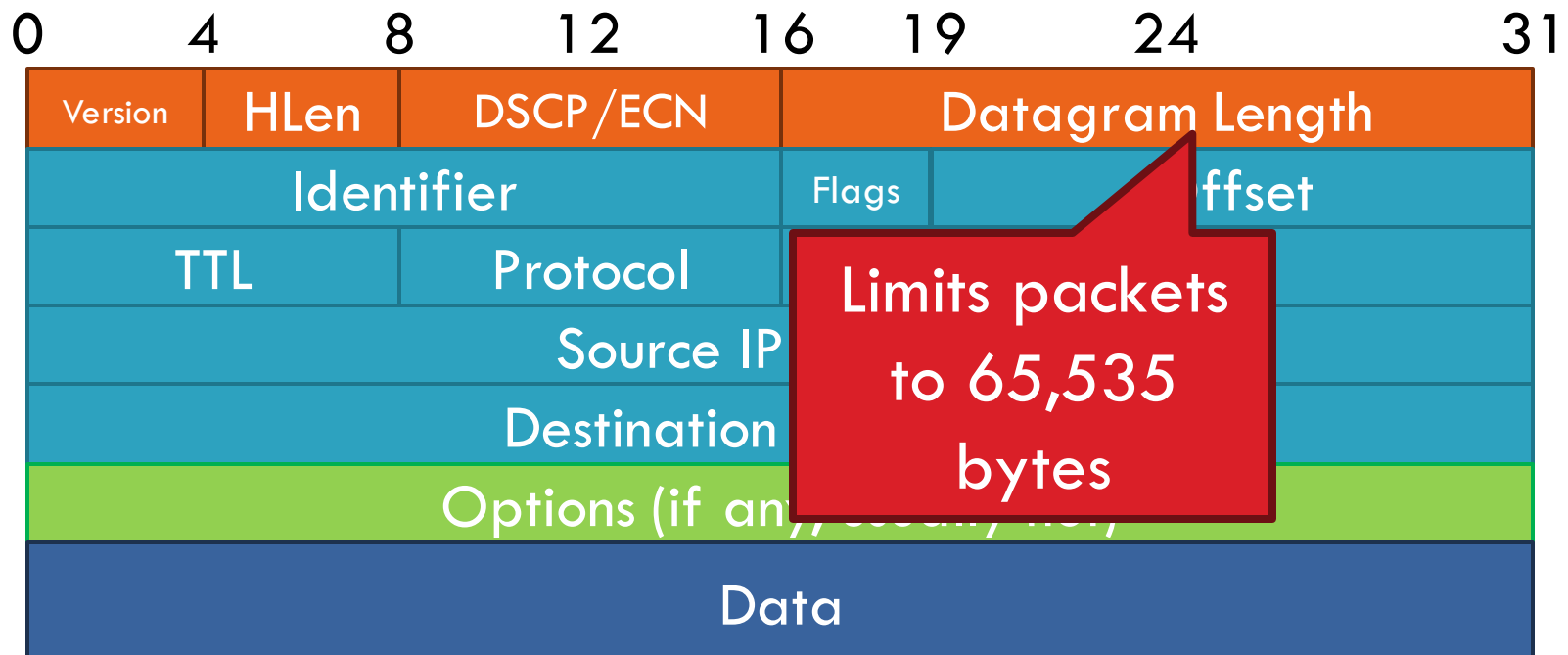
- IP Datagrams are like a letter
 - ▣ Totally self-contained
 - ▣ Include all necessary addressing information
 - ▣ No advanced setup of connections or circuits



IP Header Fields: Word 1

3

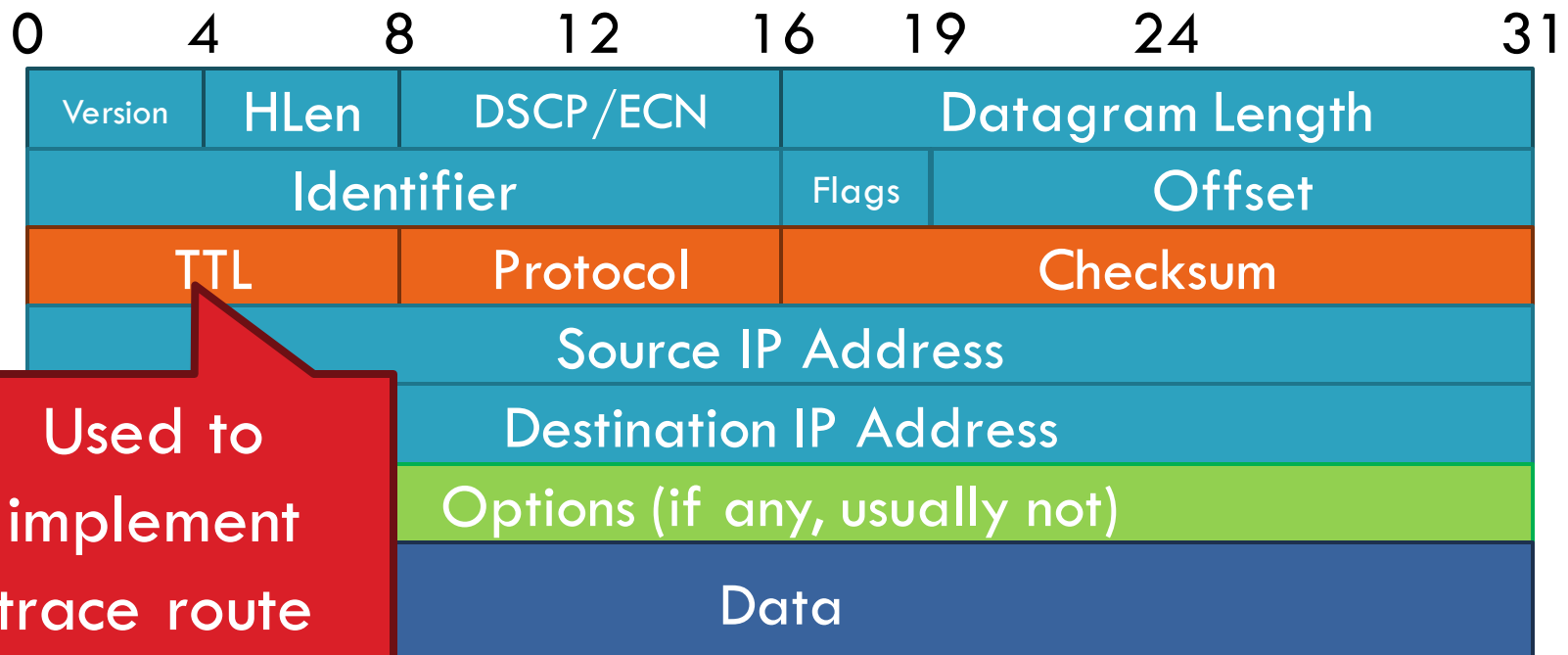
- Version: 4 for IPv4
- Header Length: Number of 32-bit words (usually 5)
- Type of Service: Priority information (unused)
- Datagram Length: Length of header + data in bytes



IP Header Fields: Word 3

4

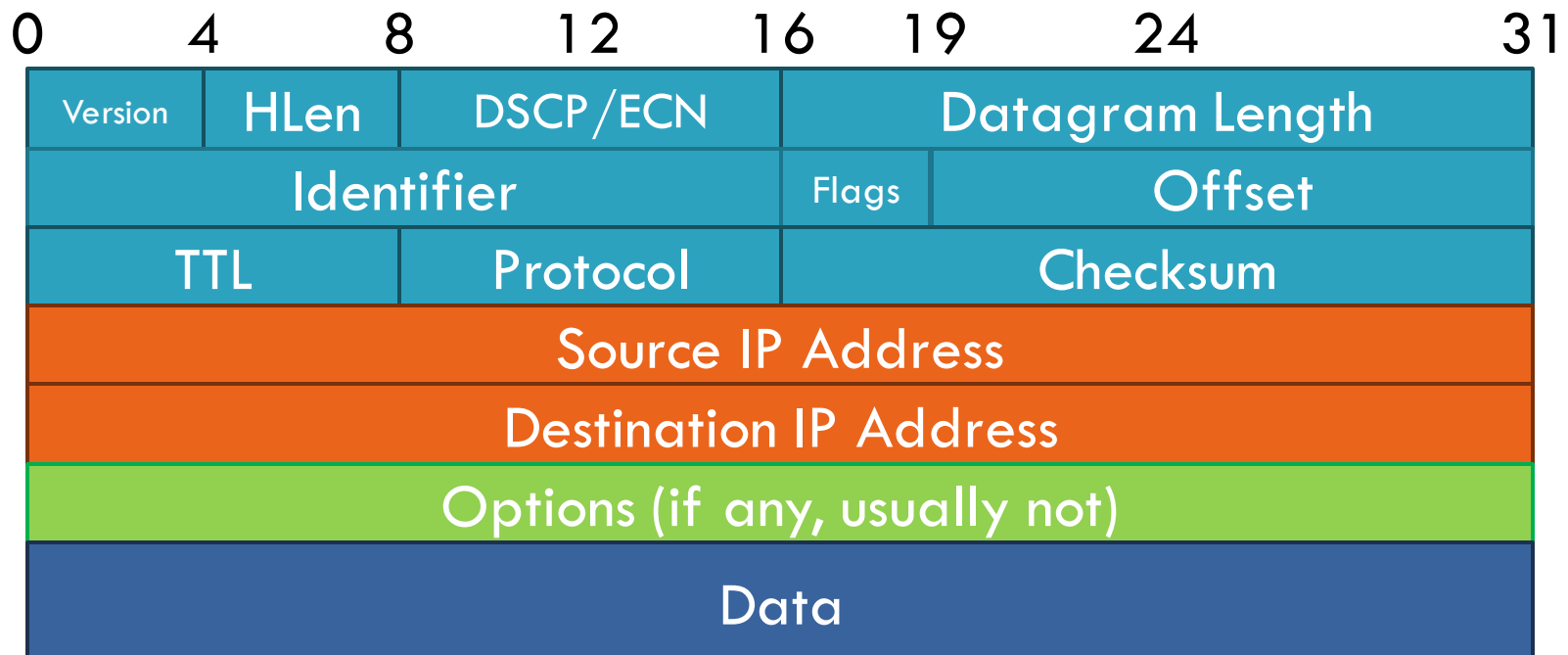
- Time to Live: decremented by each router
 - ▣ Used to kill looping packets
- Protocol: ID of encapsulated protocol
 - ▣ 6 = TCP, 17 = UDP
- Checksum



IP Header Fields: Word 4 and 5

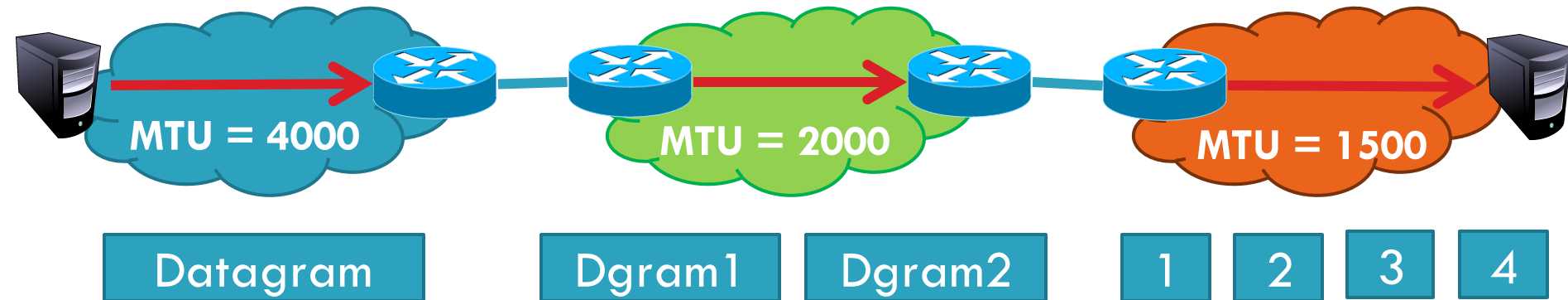
5

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



Problem: Fragmentation

6

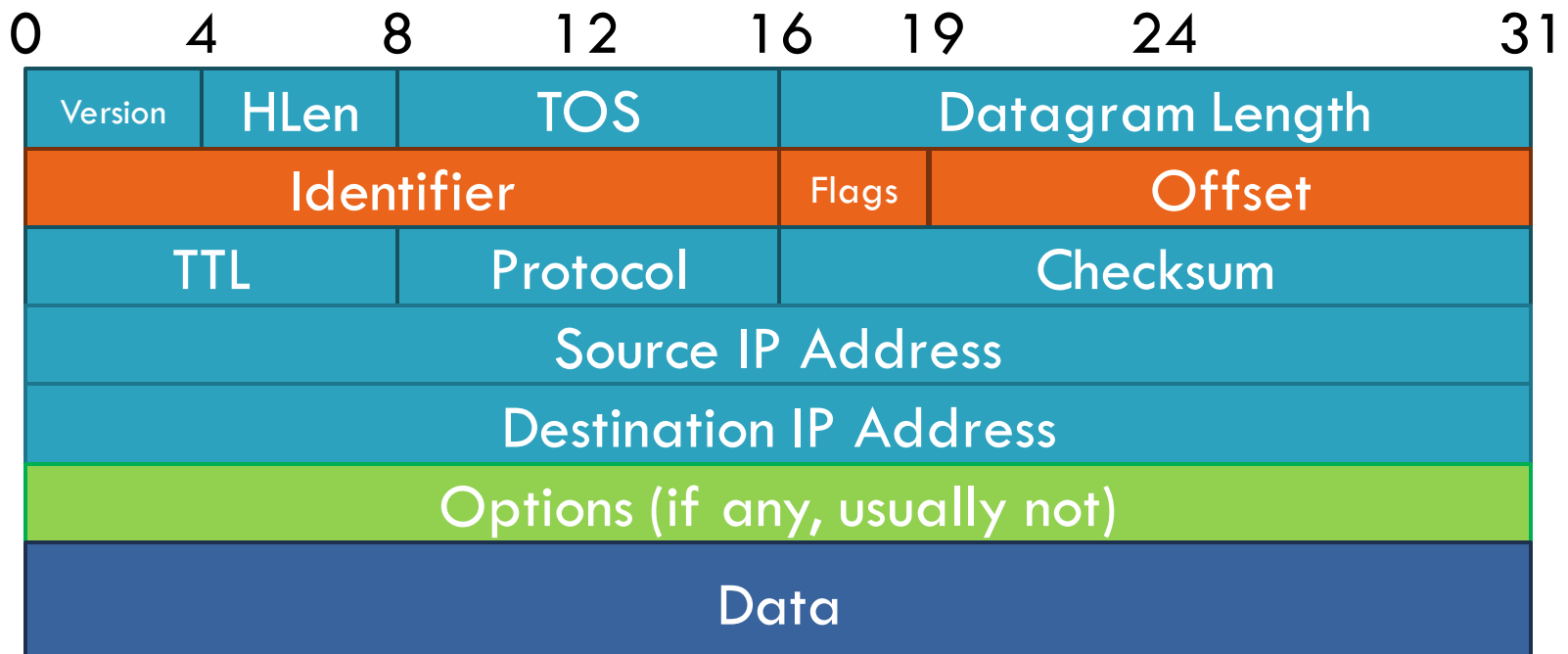


- ❑ Problem: each network has its own MTU
 - ▣ DARPA principles: networks allowed to be heterogeneous
 - ▣ Minimum MTU may not be known for a given path
- ❑ IP Solution: fragmentation
 - ▣ Split datagrams into pieces when MTU is reduced
 - ▣ Reassemble original datagram at the receiver

IP Header Fields: Word 2

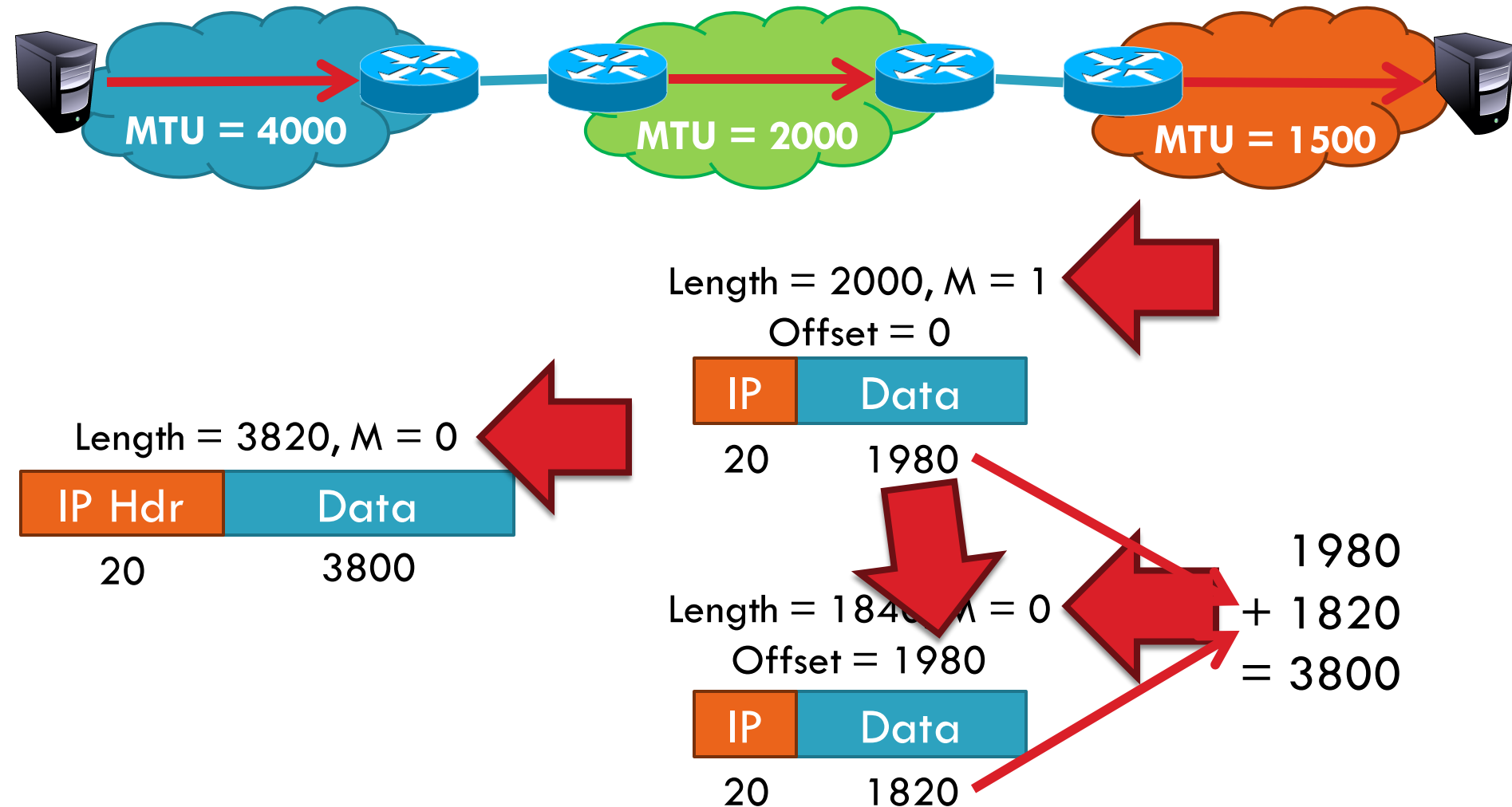
7

- ❑ 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



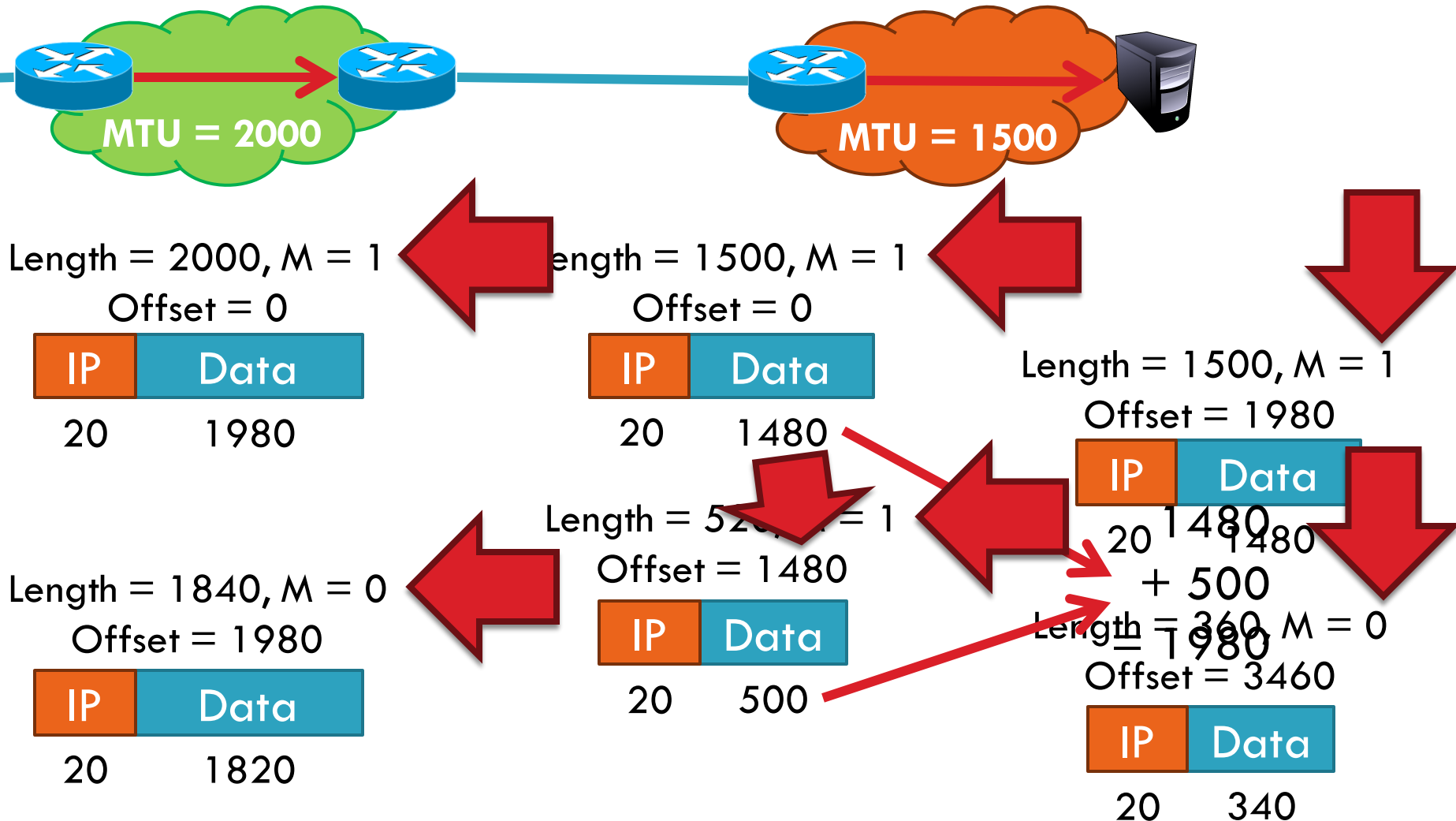
Fragmentation Example

8



Fragmentation Example

9



IP Fragment Reassembly

10

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 nightmare

Fragmentation Concepts

11

- 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

12

- ❑ 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

14

- ❑ 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

15

- ❑ 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

16

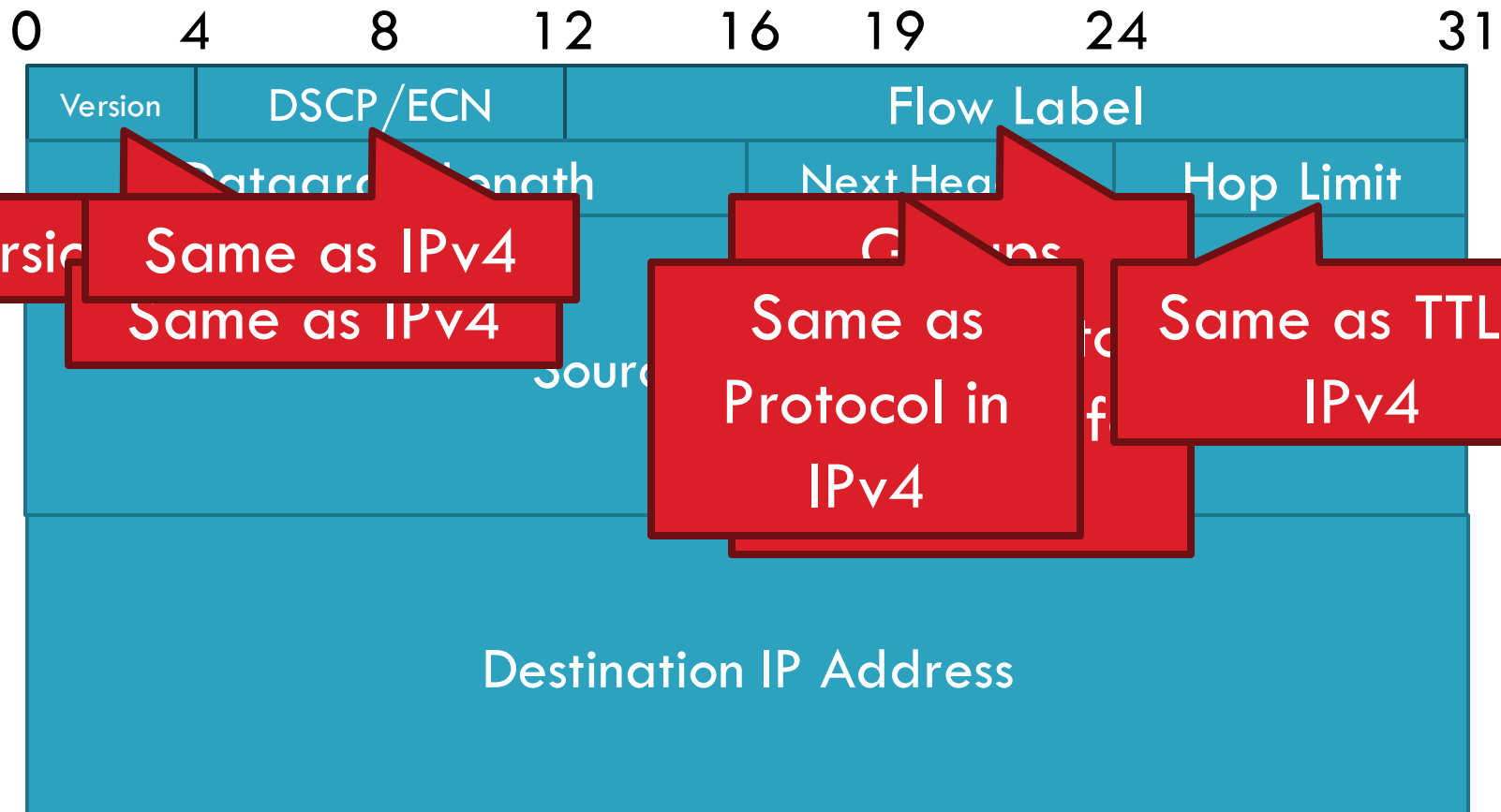
- ❑ Who knows the IP for localhost?
 - ▣ 127.0.0.1

- ❑ What is localhost in IPv6?
 - ▣ ::1

IPv6 Header

17

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



Differences from IPv4 Header

18

- ❑ 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

19

- ❑ 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!

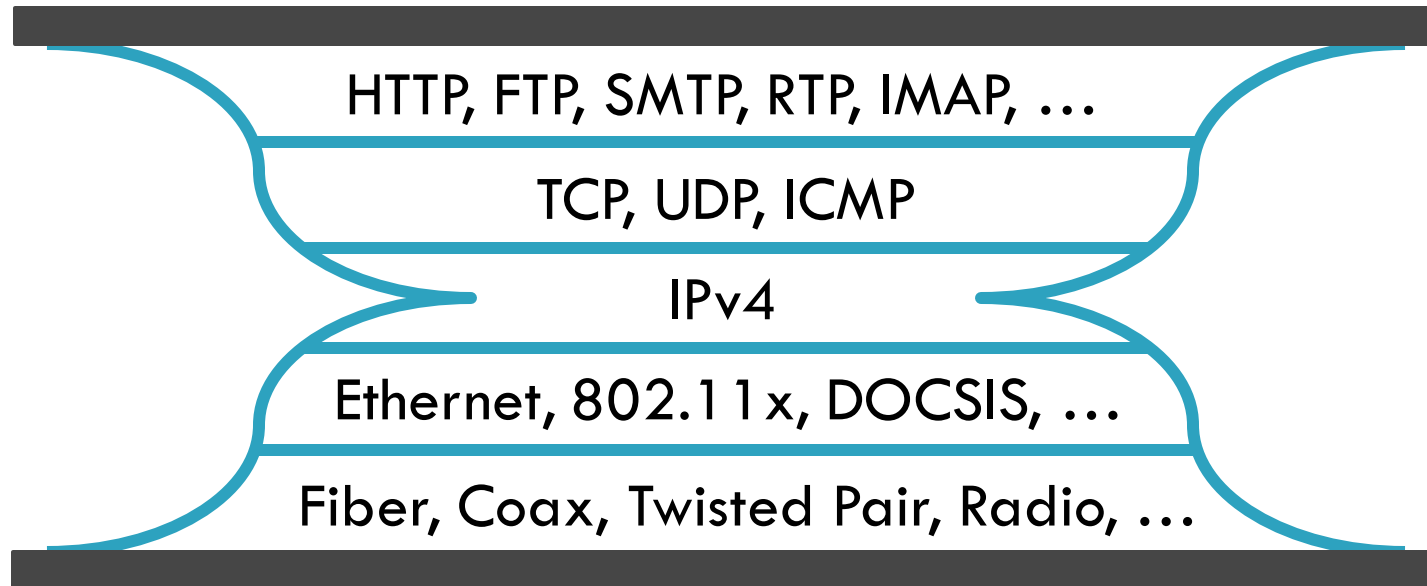
Additional IPv6 Features

20

- ❑ 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

21

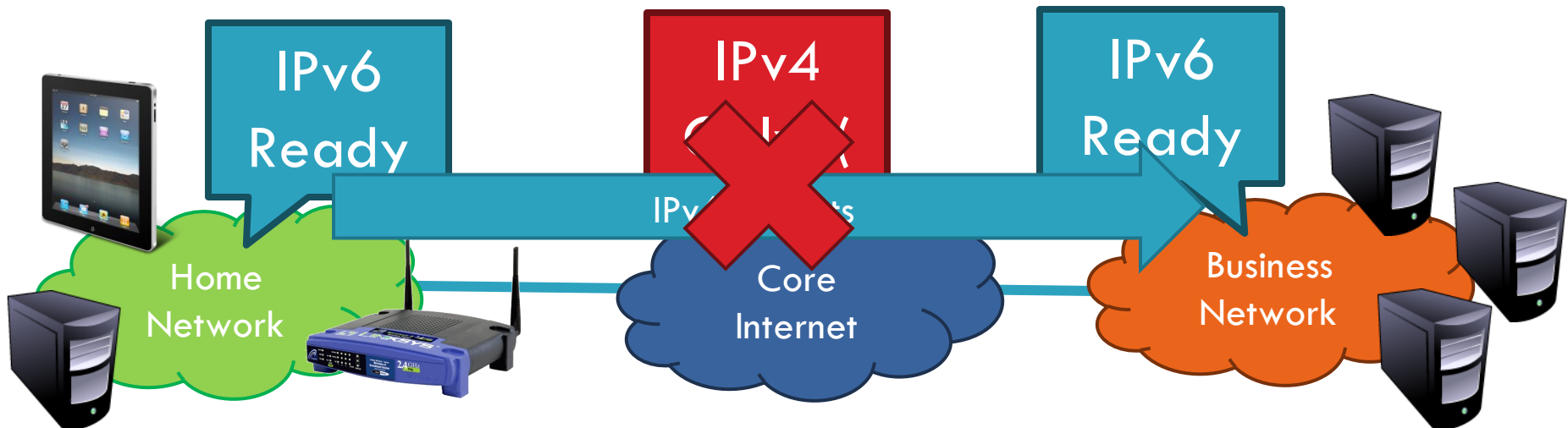


- ❑ Switching to IPv6 is a whole-Internet upgrade
 - ▣ All routers, all hosts
 - ▣ ICMPv6, DHCPv6, DNSv6
- ❑ 2013: 0.94% of Google traffic was IPv6, 2.5% today

Transitioning to IPv6

22

- ❑ 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
 - ▣ ... but a IPv4 core cannot route IPv6 traffic



Transition Technologies

23

- ❑ 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

24

Data Plane

Application

Presentation

Session

Transport

Network

Data Link

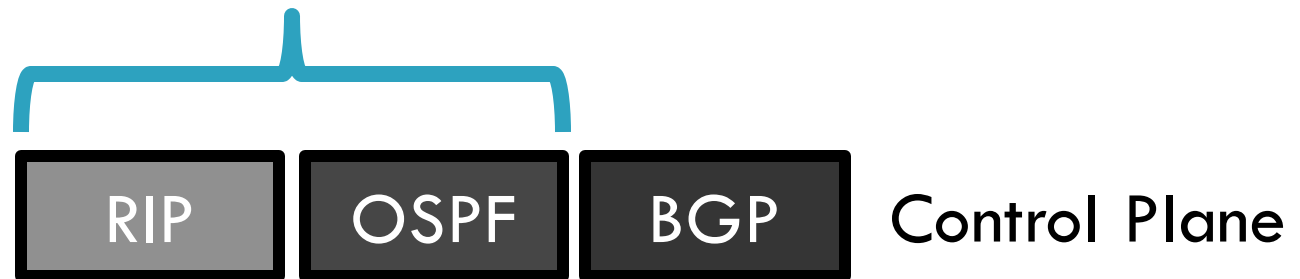
Physical

Function:

- Set up routes within a single network

Key challenges:

- Distributing and updating routes
- Convergence time
- Avoiding loops



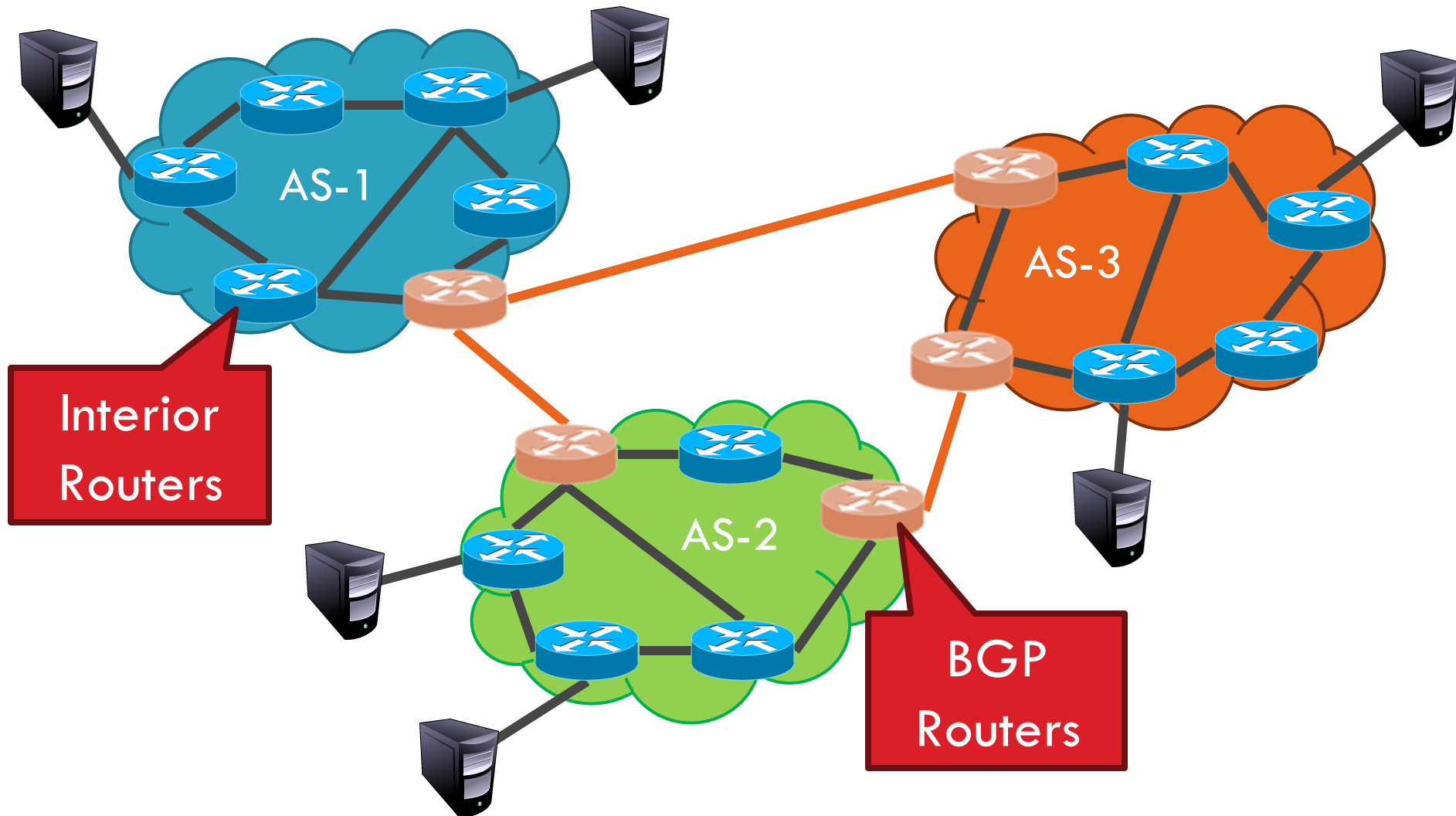
Internet Routing

25

- ❑ 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

26



Why Do We Need ASs?

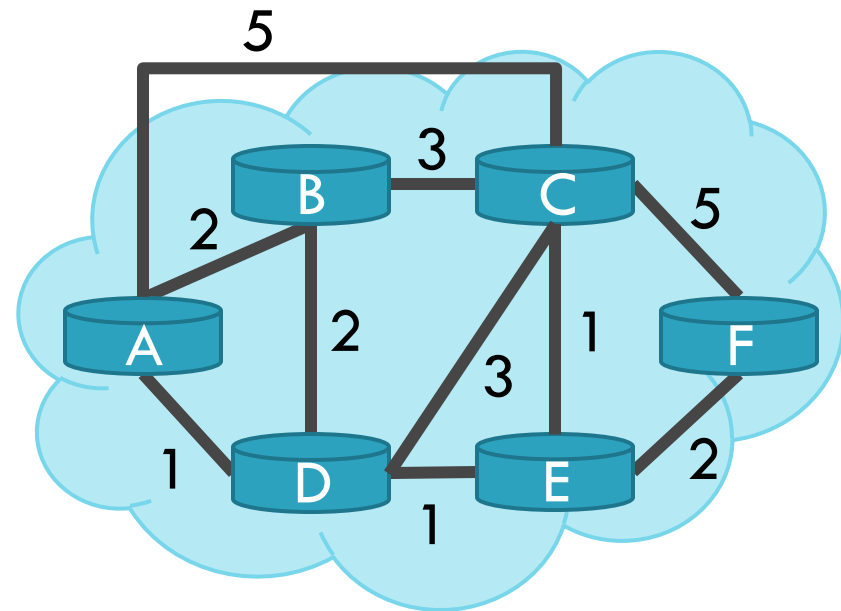
27

- ❑ Routing algorithms are not efficient enough to execute on the entire Internet topology
 - ❑ Different policies
 - ❑ Allow structure
 - ❑ Allow other (BGP)
- Easier to compute routes
 - Greater flexibility
 - More autonomy/independence
- as each

Routing on a Graph

28

- ❑ 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

29

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

30

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)

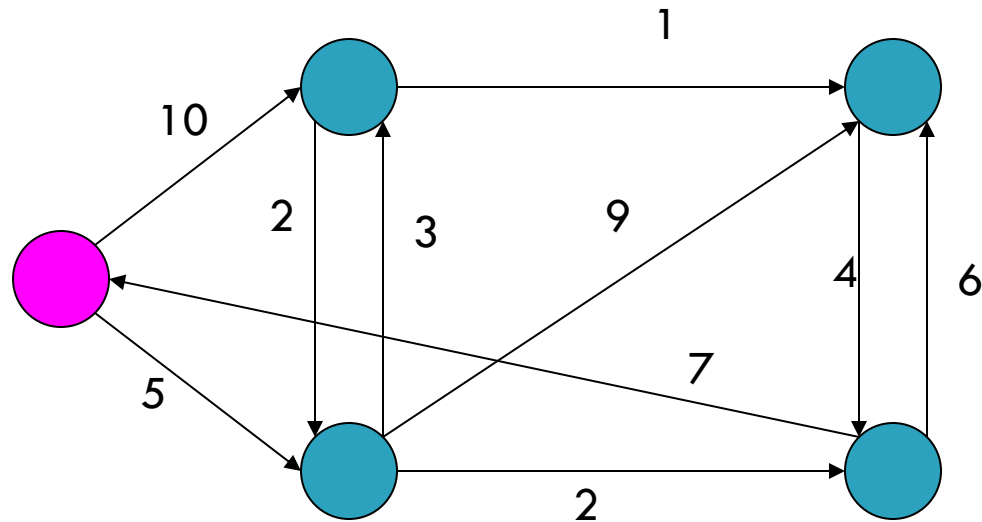
if $d[v] > d[u] + w(u,v)$ **then**

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

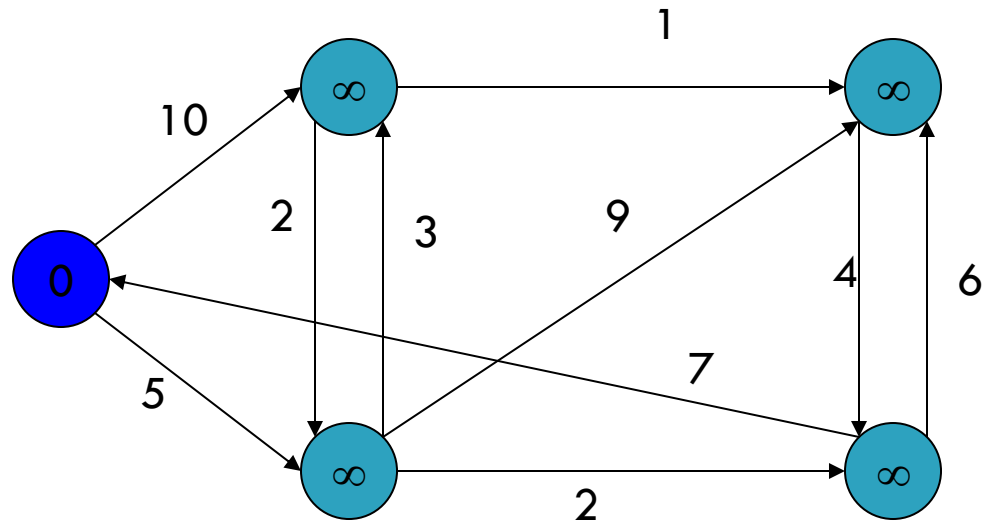
$p[v] \leftarrow u$

$\Theta(1)$?

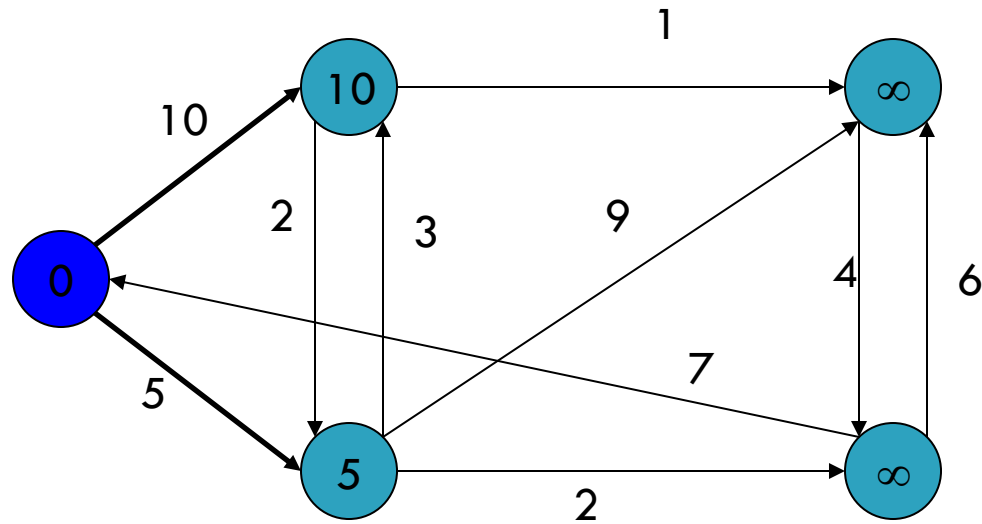
Dijkstra's Algorithm - Example



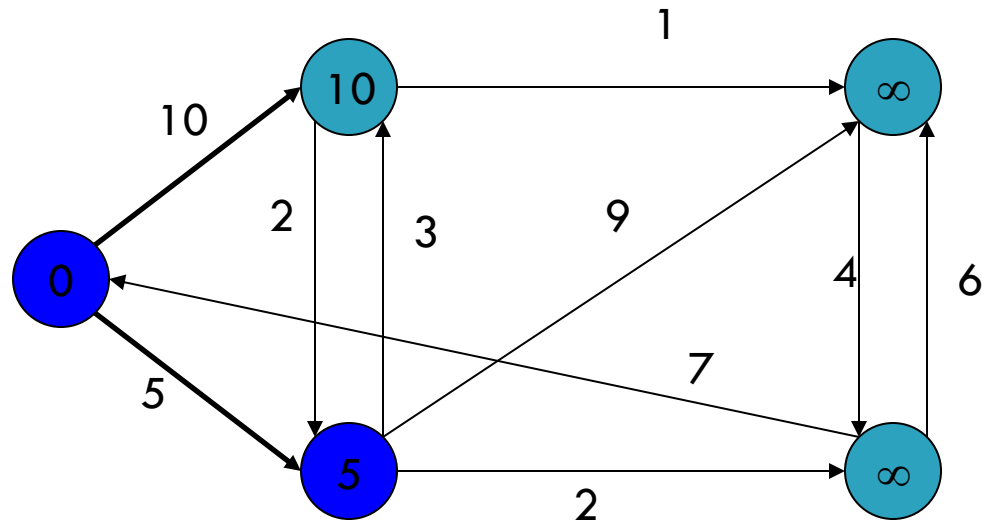
Dijkstra's Algorithm - Example



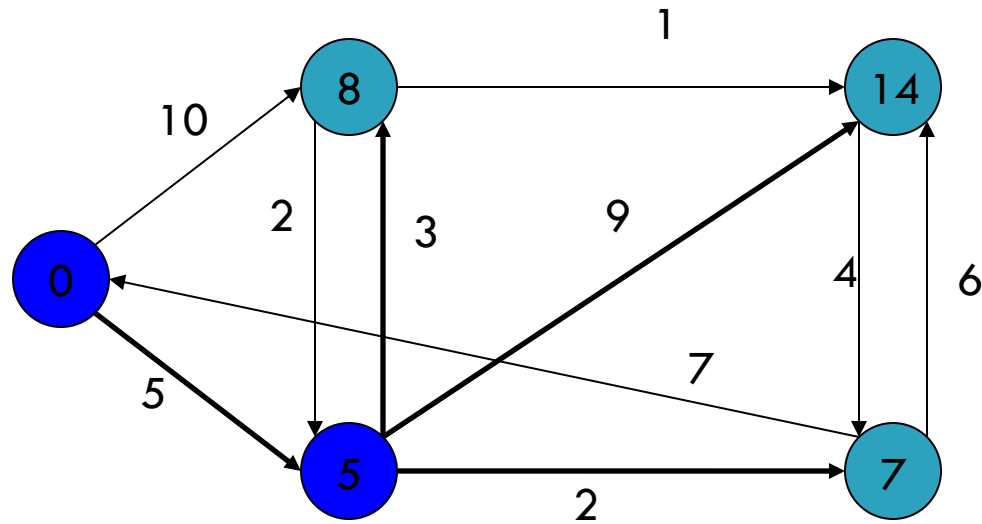
Dijkstra's Algorithm - Example



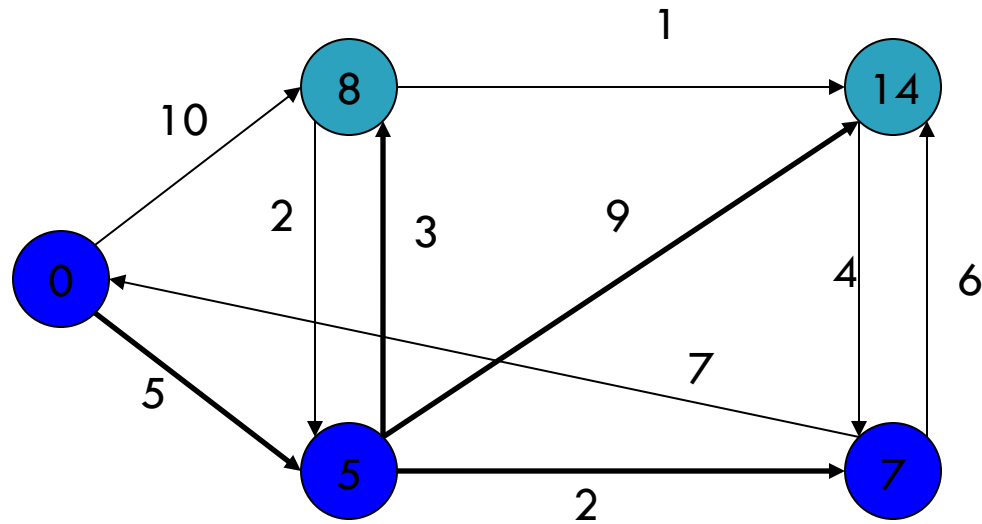
Dijkstra's Algorithm - Example



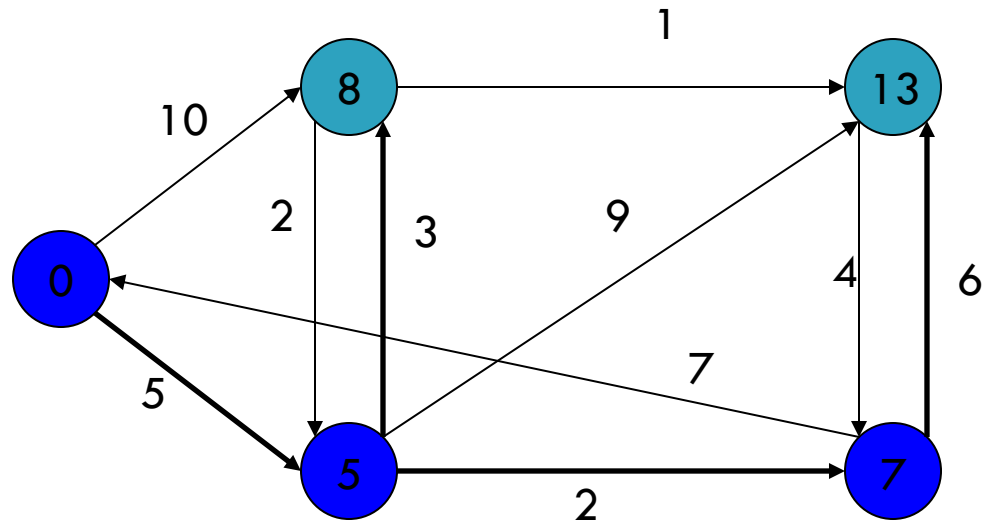
Dijkstra's Algorithm - Example



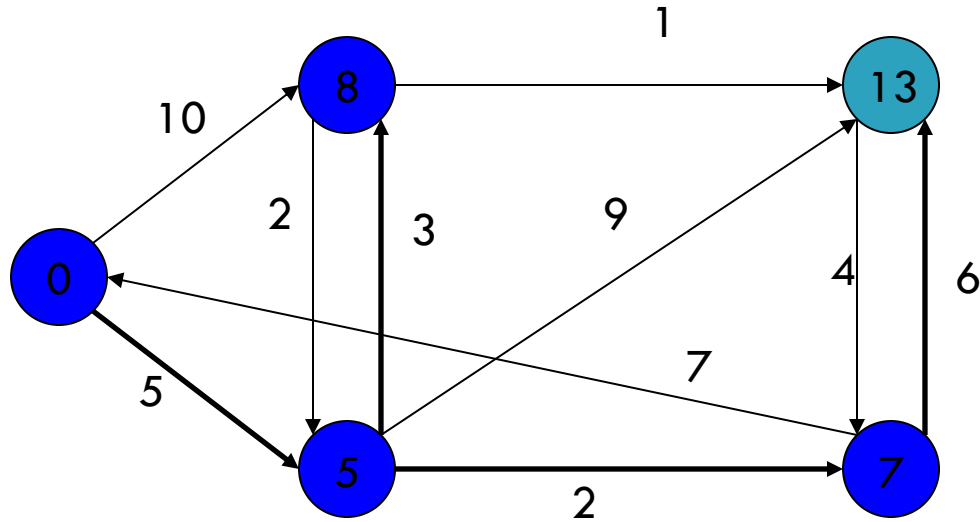
Dijkstra's Algorithm - Example



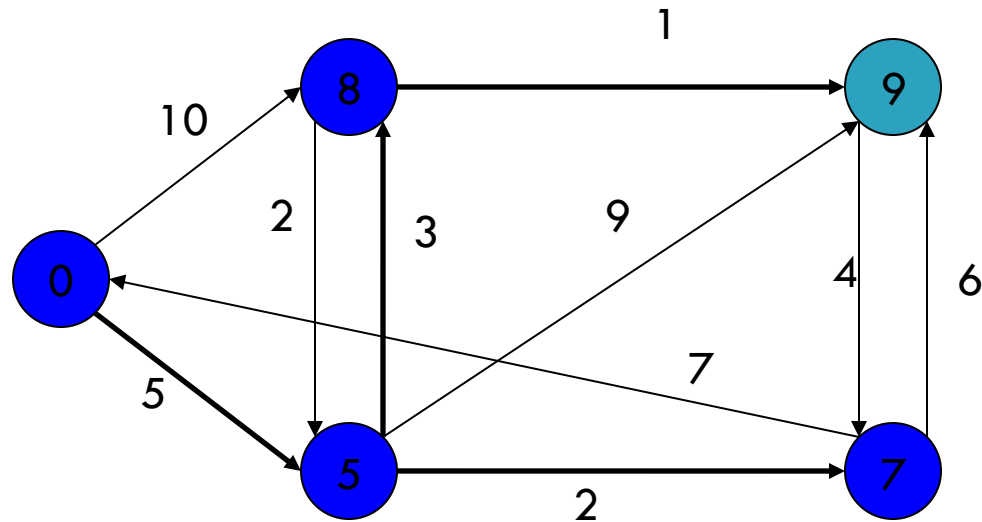
Dijkstra's Algorithm - Example



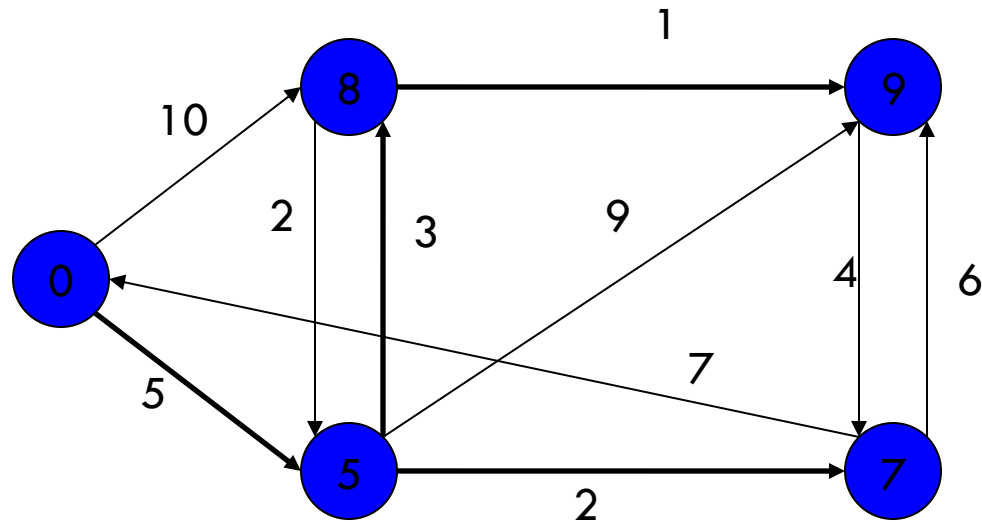
Dijkstra's Algorithm - Example



Dijkstra's Algorithm - Example



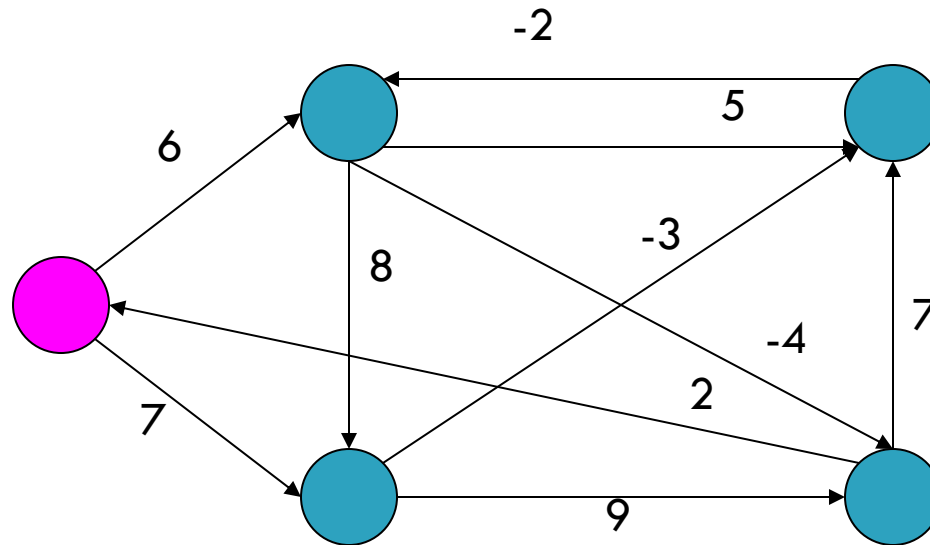
Dijkstra's Algorithm - Example



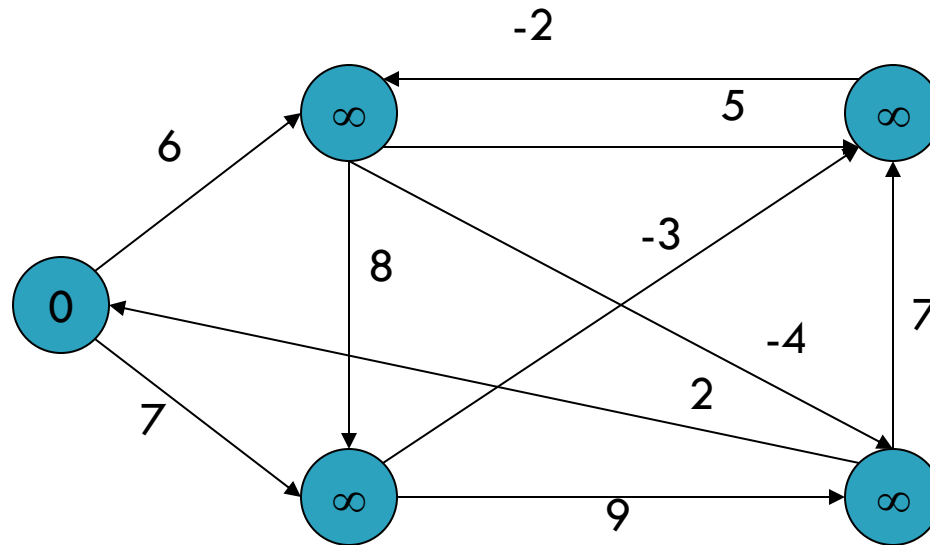
Bellman-Ford Algorithm

```
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
```

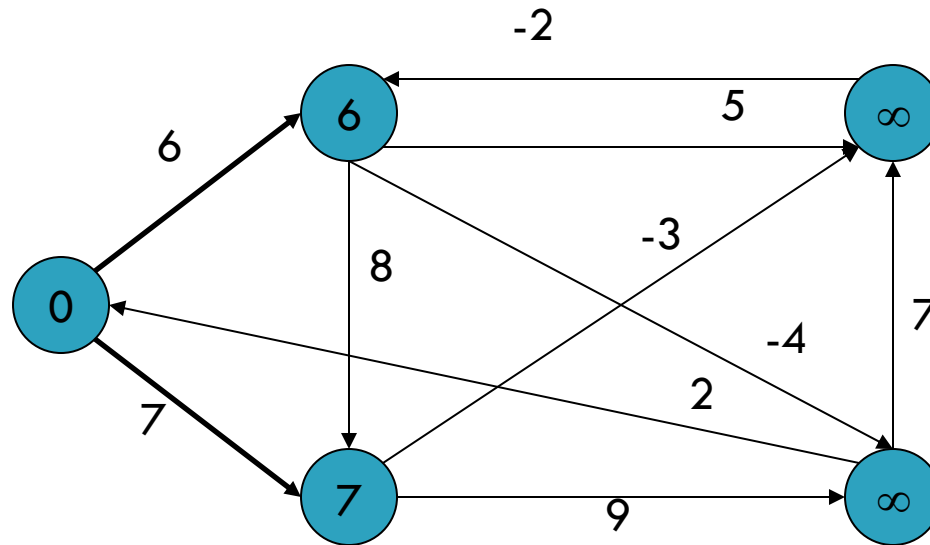
Bellman-Ford Algorithm - Example



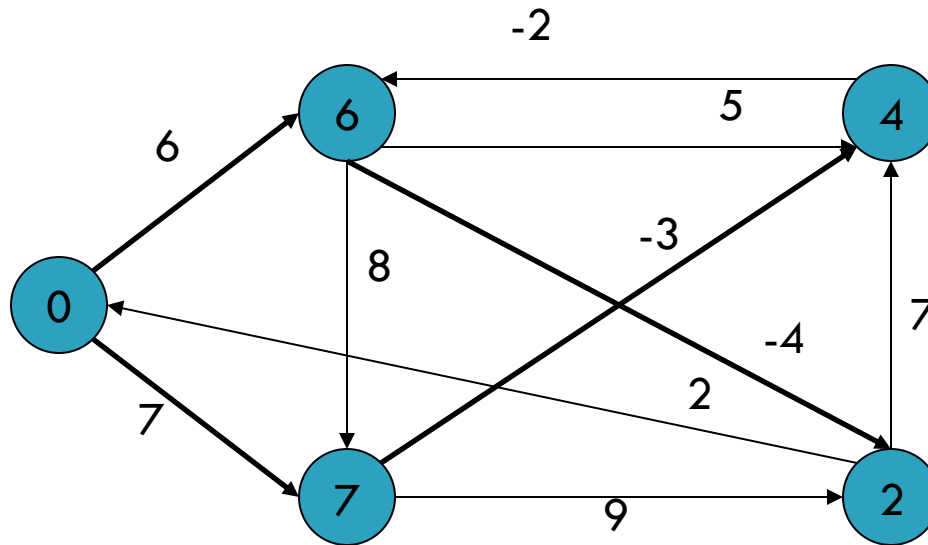
Bellman-Ford Algorithm - Example



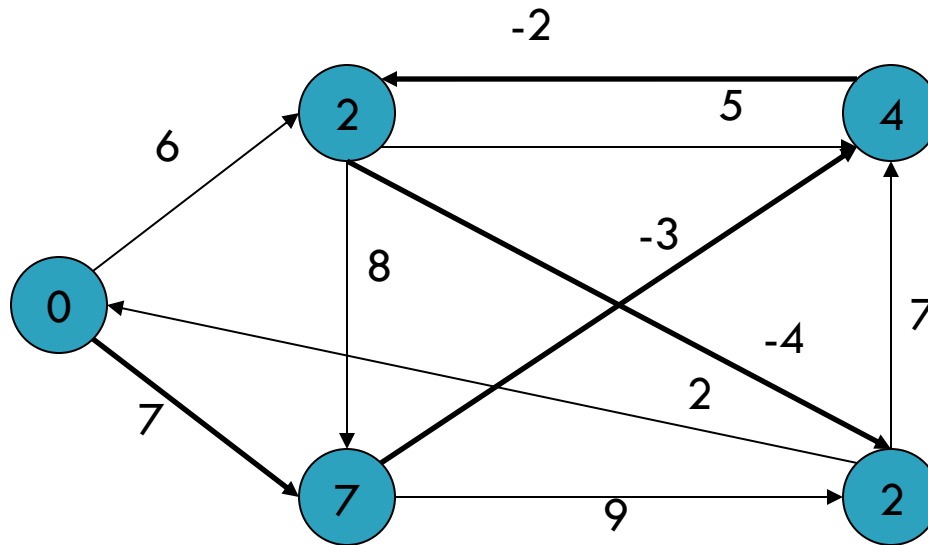
Bellman-Ford Algorithm - Example



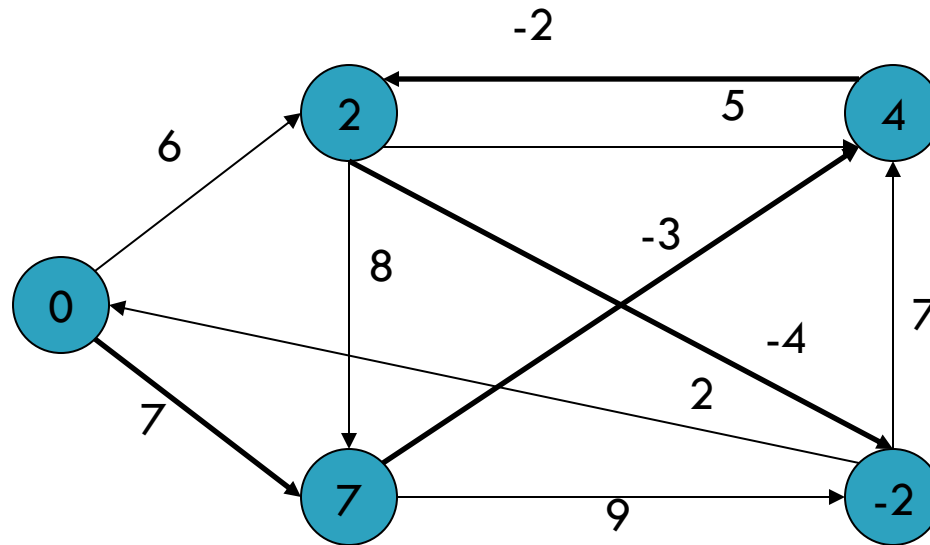
Bellman-Ford Algorithm - Example



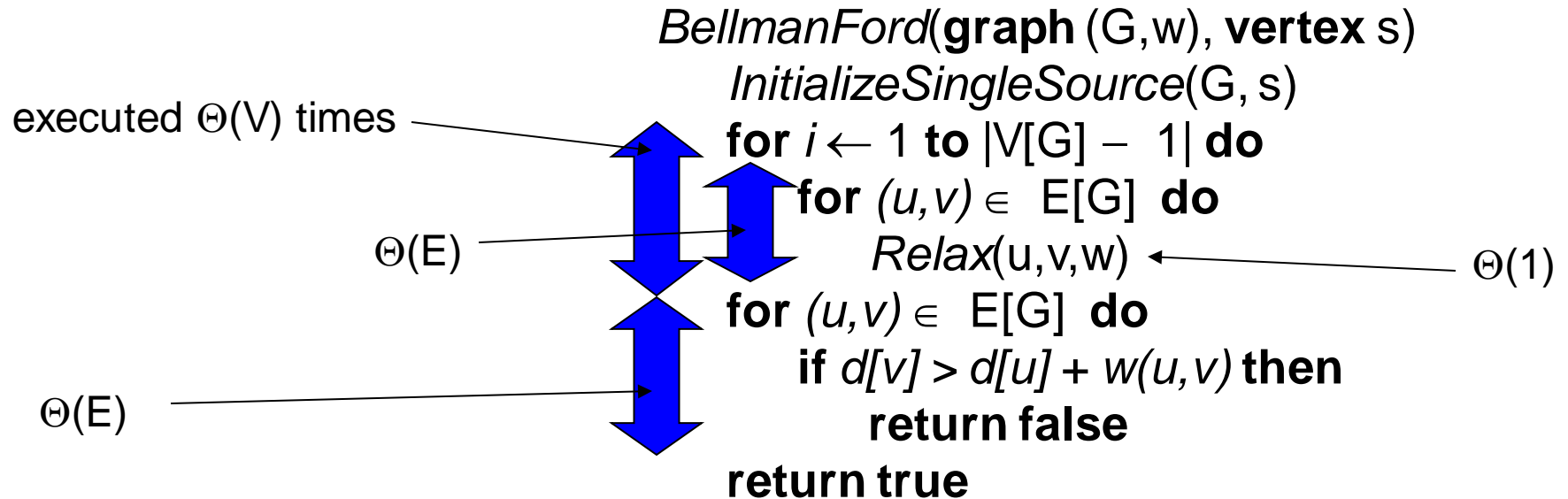
Bellman-Ford Algorithm - Example



Bellman-Ford Algorithm - Example



Bellman-Ford Algorithm - Complexity



Internetwork Routing [Halsall]

50

Adaptive Routing

Centralized

[RCC]

Distributed

[IGP]

Intradomain routing

Interdomain routing

[EGP]

Interior
Gateway Protocols

[BGP, IDRP]

Exterior
Gateway Protocols

Distance Vector routing

[RIP]

Link State routing

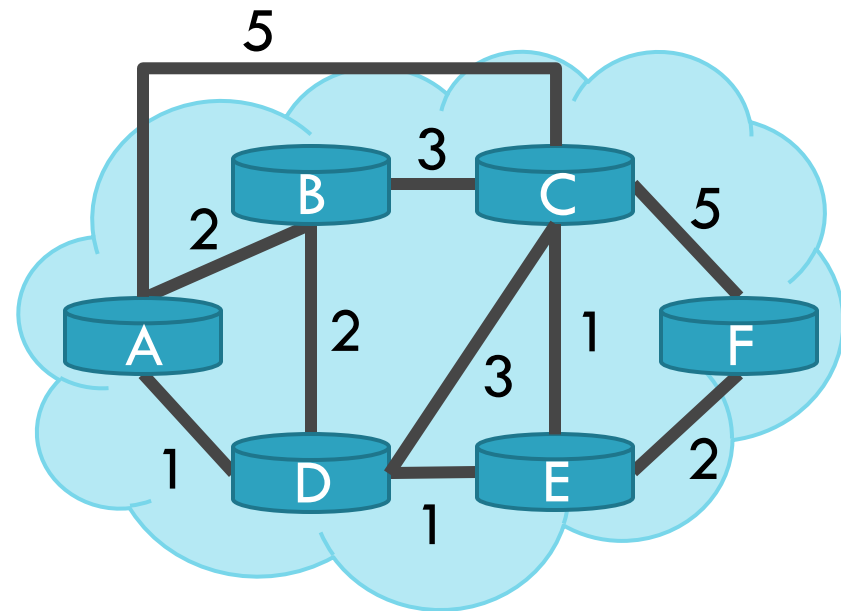
[OSPF, IS-IS, PNNI]

Networks: Routing

Routing Problems

51

- 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

52

- ❑ 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

54

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

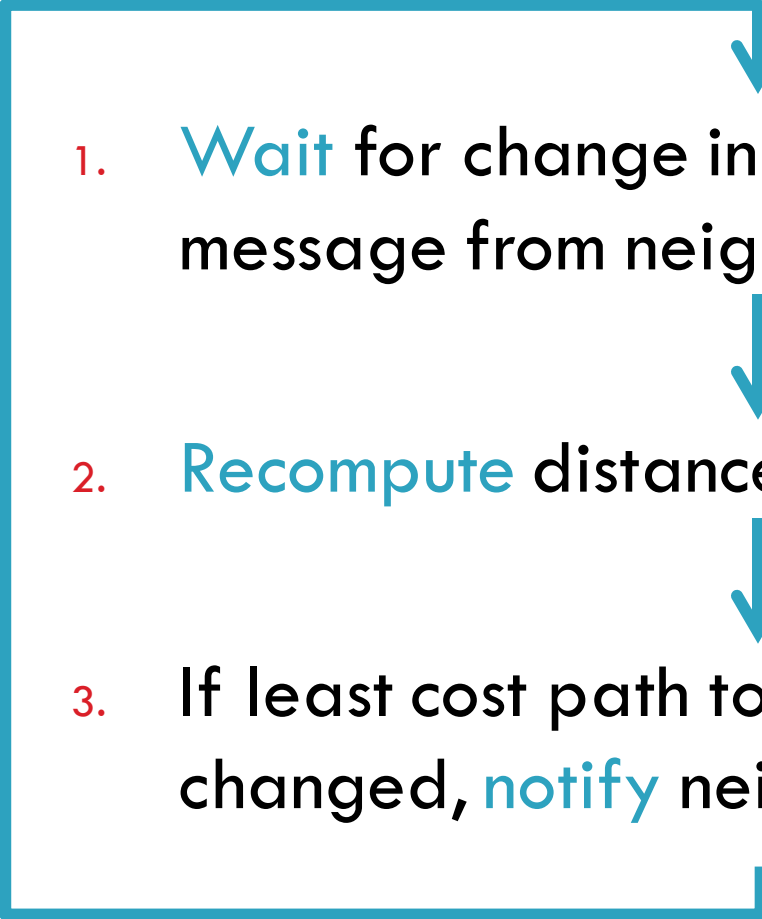
DV Table
at Node C

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 = ∞
 - Eventually, vector is filled
-
- Routing Information Protocol (RIP)

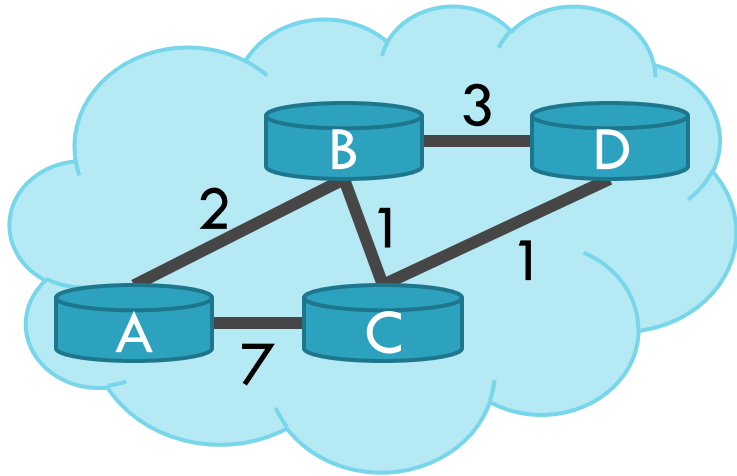
Distance Vector Routing Algorithm

55

- 
- ```
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

56



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

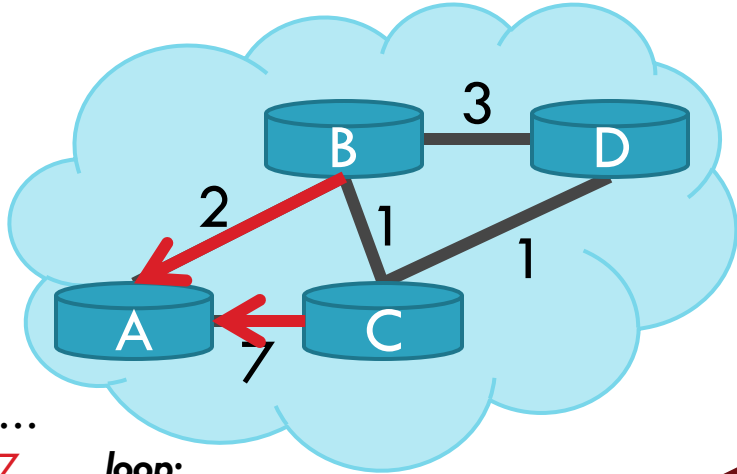
| Dest. | Cost     | Next |
|-------|----------|------|
| A     | $\infty$ |      |
| B     | 3        | B    |
| C     | 1        | C    |

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

57



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) re
13. for all d
14. if (des
15. D(A, Y)
16. else
17. D(A, Y) =
 min
 D(A, V) + D(V, Y));
18. if (there is a new min. for dest. Y)
19. send D(A, Y) to all neighbors
20. forever

```

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

$$D(A, C) = \min(7, 2 + 1) = 2$$

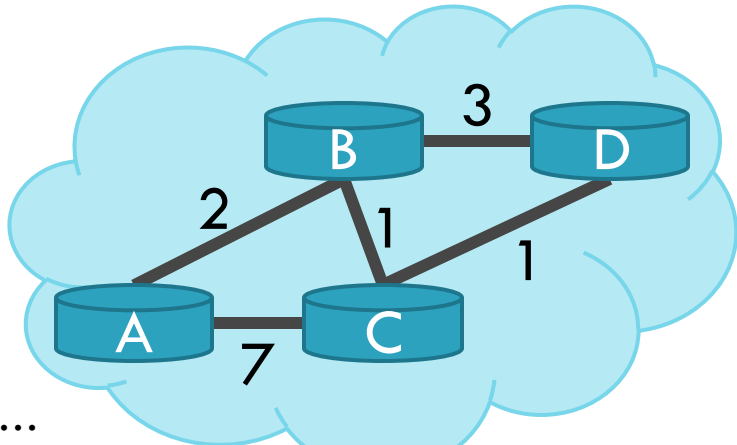
$$= \min(\infty, 3)$$

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

$$= \min(8, 2 + 3) = 5$$

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

58



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    |

```

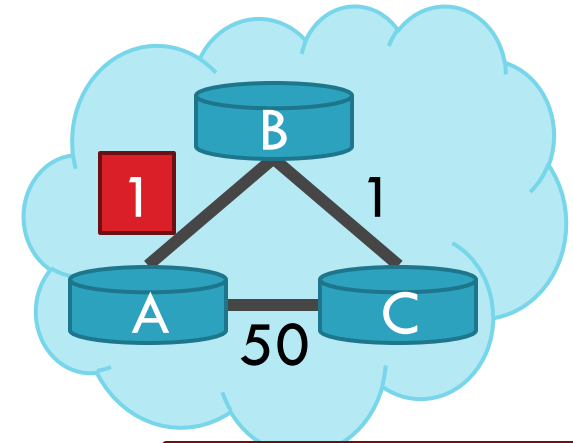
...
7. loop
...
12. else
13. for
14. if
15. if
16. else
17. D(A, Y) =
 min(D(A, Y),
 D(A, V) + D(V, Y));
18. if (there is a new min. for dest. Y)
19. send D(A, Y) to all neighbors
20. forever

```

```

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));

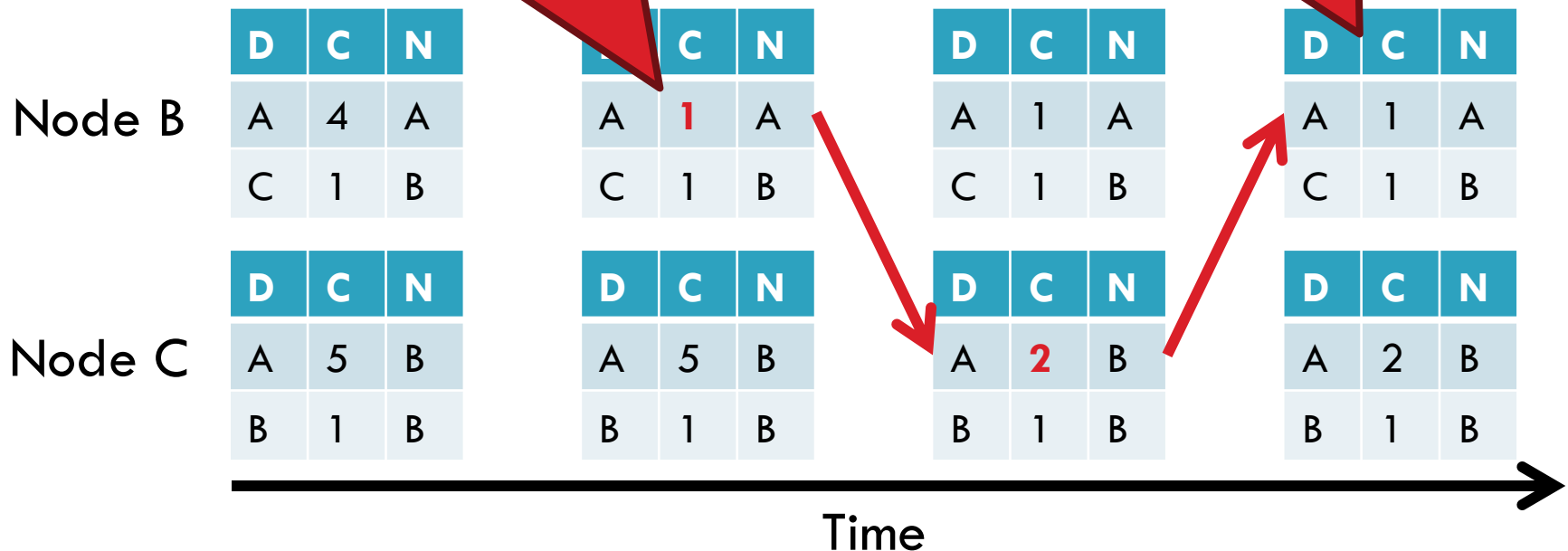
```



Link Cost  
Algorithm

Good news travels fast

Algorithm  
terminates

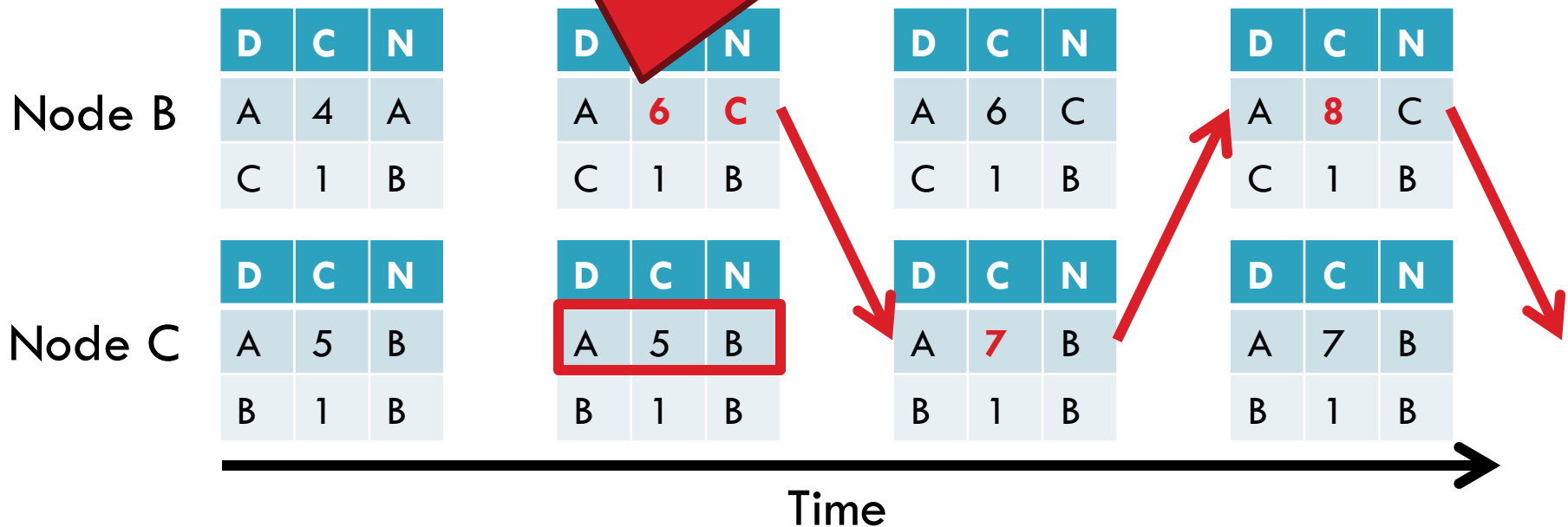


# Count to Infinity Problem

60

- Node B knows  $D(C, A) = 5$
- However, B does not know the path is  $C \rightarrow B \rightarrow A$
- Thus,  $D(B, A) = 6$  !

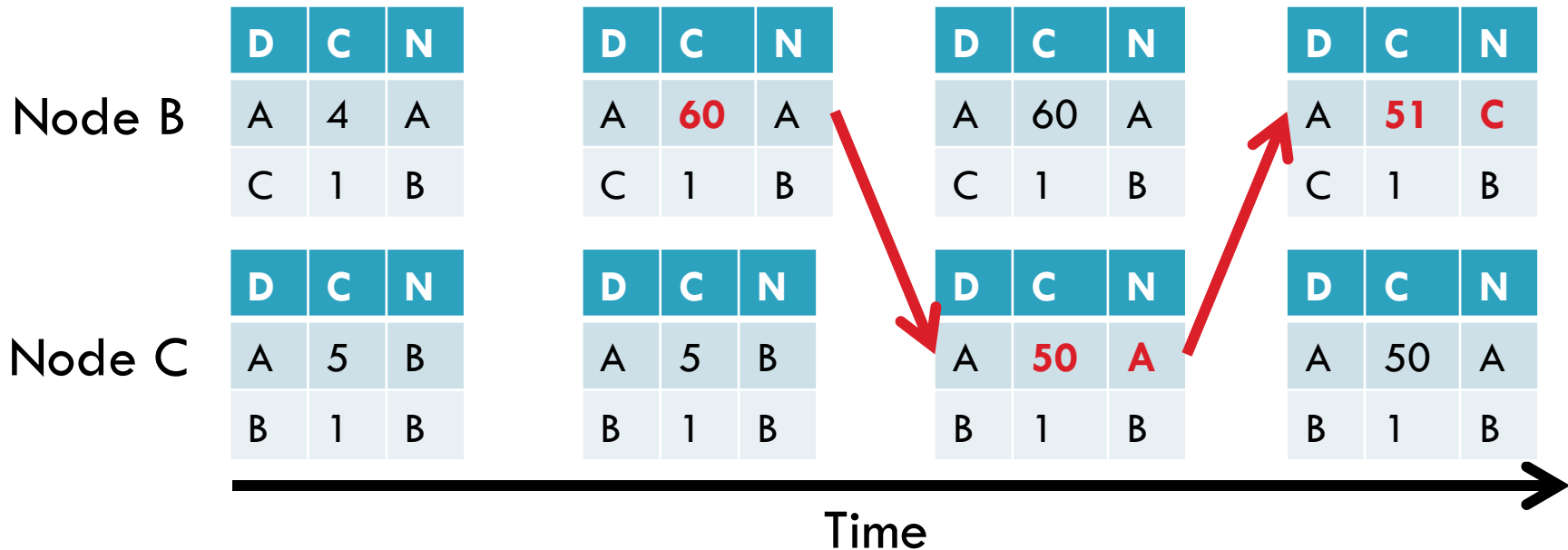
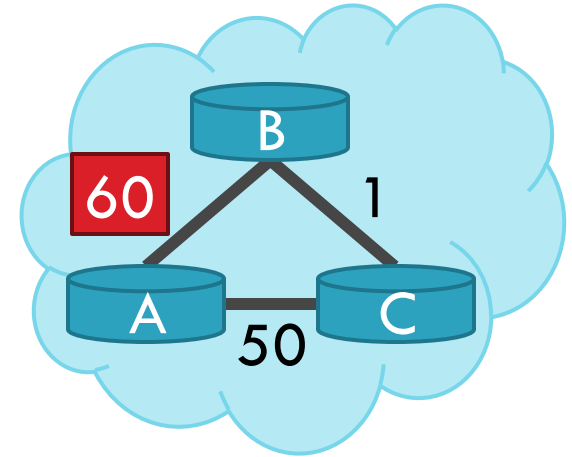
Bad news travels



# Poisoned Reverse

61

- 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

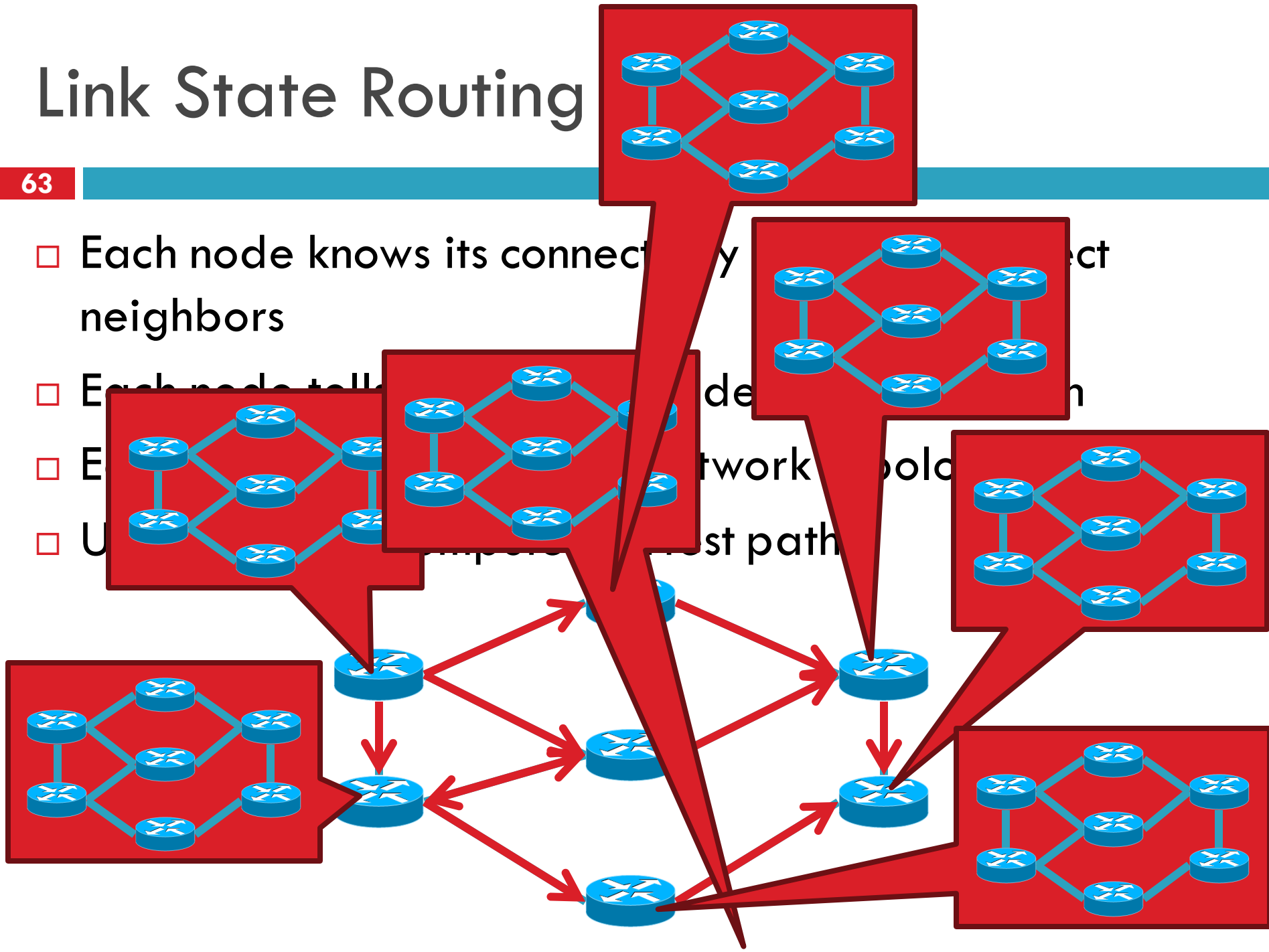


- ❑ Distance Vector Routing
  - ❑ RIP
- ❑ Link State Routing
  - ❑ OSPF
  - ❑ IS-IS

# Link State Routing

63

- Each node knows its connectivity to its neighbors
- Each node tells its neighbors about its own view of the network topology
- Each node calculates the shortest path to each destination



# Flooding Details

64

- ❑ Each node periodically generates Link State Packet
  - ❑ ID of node generating the LSP
  - ❑ List of direct neighbors and costs
  - ❑ Sequence number (64-bit, assumed to never wrap)
  - ❑ Time to live
- ❑ Flood is reliable (ack + retransmission)
- ❑ Sequence number “versions” each LSP
- ❑ Receivers flood LSPs to their own neighbors
  - ❑ Except whoever originated the LSP
- ❑ LSPs also generated when link states change



# OSPF vs. IS-IS

65

- Two different implementations of link-state routing

## OSPF

- Favored by companies, datacenters
- More optional features
- Built on top of IPv4
  - ▣ LSAs are sent via IPv4
  - ▣ OSPFv3 needed for IPv6

## IS-IS

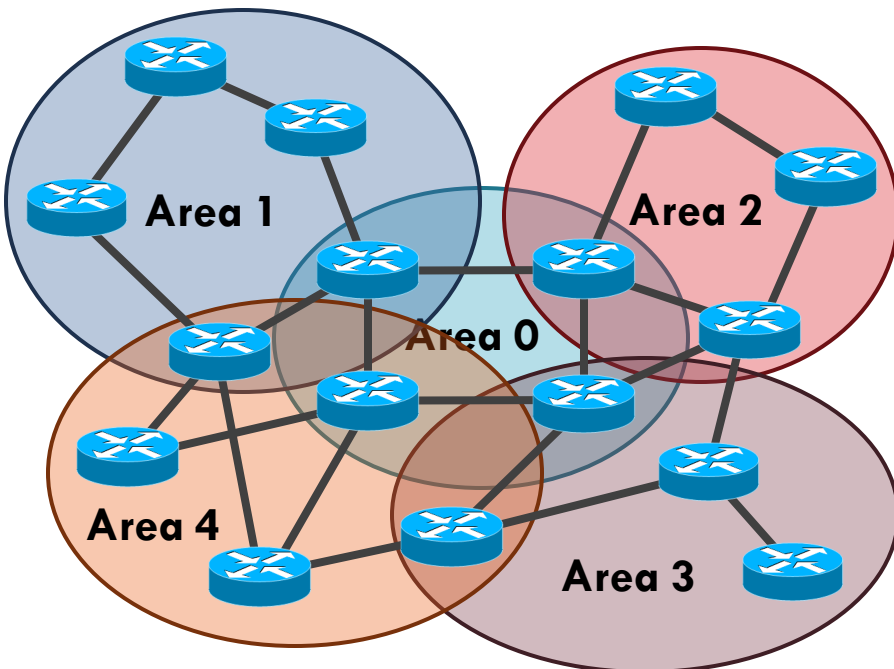
- Favored by ISPs
- Less “chatty”
  - ▣ Less network overhead
  - ▣ Supports more devices
- Not tied to IP
  - ▣ Works with IPv4 or IPv6

# Different Organizational Structure

66

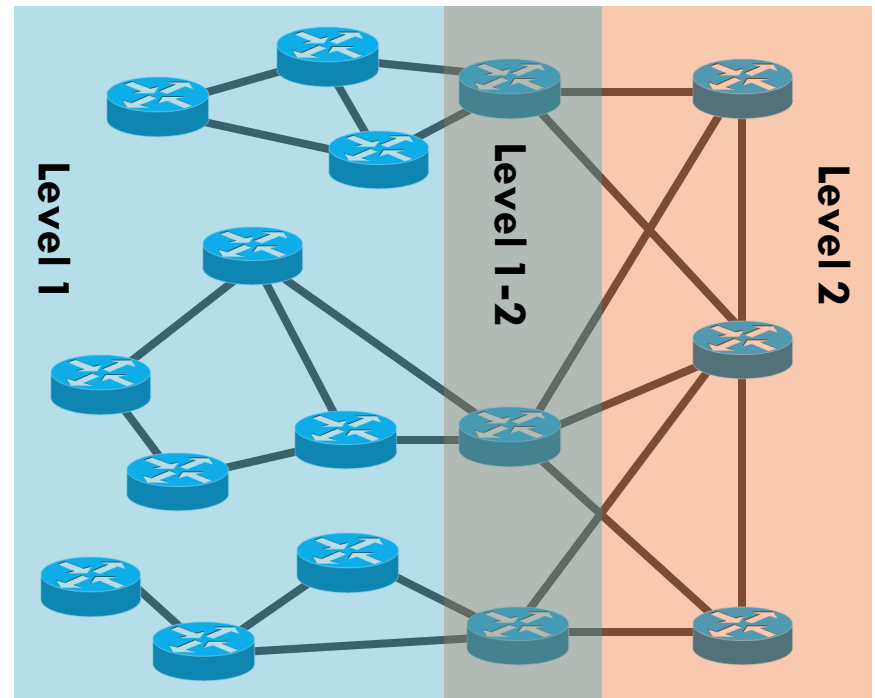
## OSPF

- ❑ Organized around overlapping areas
- ❑ Area 0 is the core network



## IS-IS

- ❑ Organized as a 2-level hierarchy
- ❑ Level 2 is the backbone



# Network Layer, Control Plane

67

Data Plane

Application

Presentation

Session

Transport

Network

Data Link

Physical

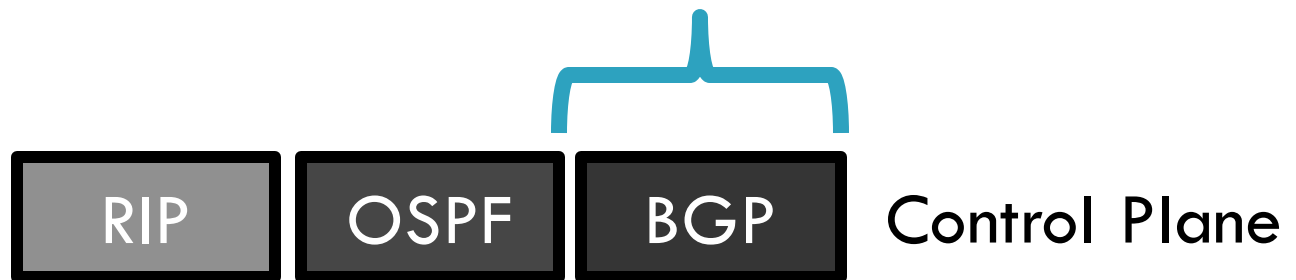
□ Function:

▣ Set up routes between networks

□ Key challenges:

▣ Implementing provider policies

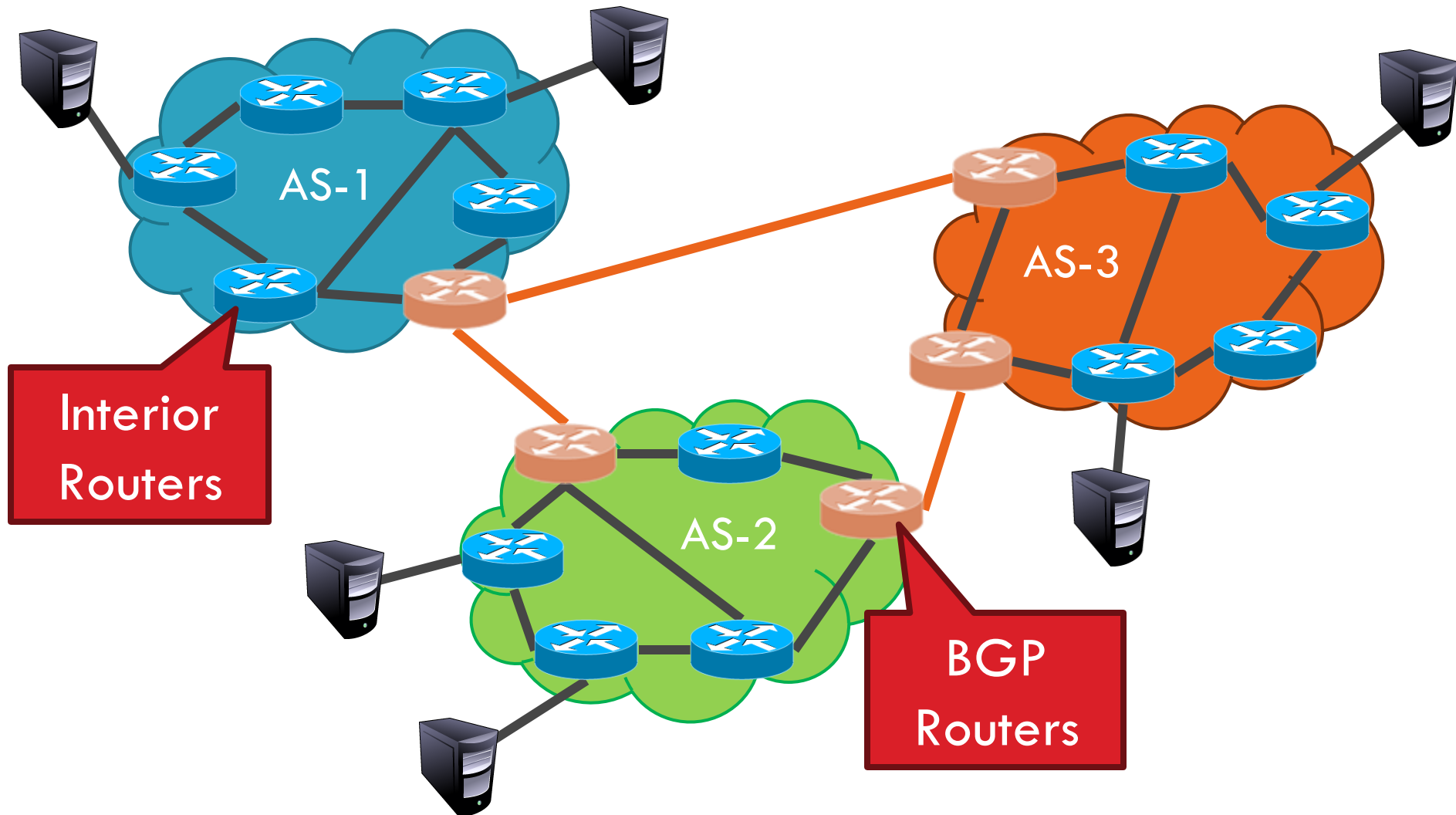
▣ Creating stable paths



- ❑ BGP Basics
- ❑ Stable Paths Problem
- ❑ BGP in the Real World
- ❑ Debugging BGP Path Problems

# ASs, Revisited

69



# AS Numbers

70

- ❑ Each AS identified by an ASN number
  - ▣ 16-bit values (latest protocol supports 32-bit ones)
  - ▣ 64512 – 65535 are reserved
- ❑ Currently, there are ~ 40000 ASNs
  - ▣ AT&T: 5074, 6341, 7018, ...
  - ▣ Sprint: 1239, 1240, 6211, 6242, ...
  - ▣ ELTE: 2012
  - ▣ Google 15169, 36561 (formerly YT), + others
  - ▣ Facebook 32934
  - ▣ North America ASs → <ftp://ftp.arin.net/info/asn.txt>

# Inter-Domain Routing

71

- ❑ Global connectivity is at stake!
  - ▣ Thus, all ASs must use the same protocol
  - ▣ Contrast with intra-domain routing
- ❑ What are the requirements?
  - ▣ Scalability
  - ▣ Flexibility in choosing routes
    - Cost
    - Routing around failures
- ❑ Question: link state or distance vector?
  - ▣ Trick question: BGP is a **path vector** protocol

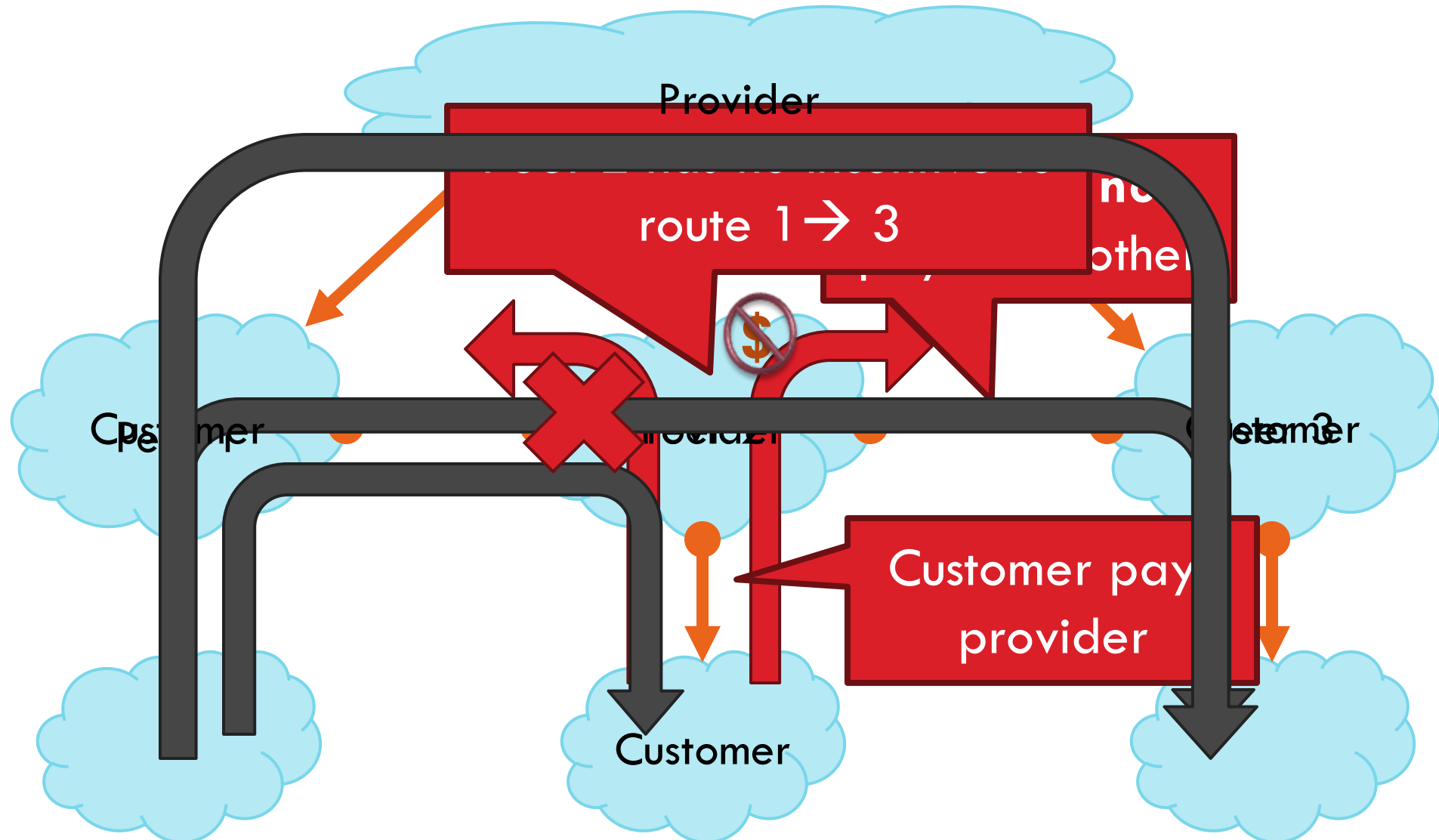
# BGP

72

- ❑ Border Gateway Protocol
  - ▣ De facto inter-domain protocol of the Internet
  - ▣ Policy based routing protocol
  - ▣ Uses a Bellman-Ford path vector protocol
- ❑ Relatively simple protocol, but...
  - ▣ Complex, manual configuration
  - ▣ Entire world sees advertisements
    - Errors can screw up traffic globally
  - ▣ Policies driven by economics
    - How much \$\$\$ does it cost to route along a given path?
    - Not by performance (e.g. shortest paths)

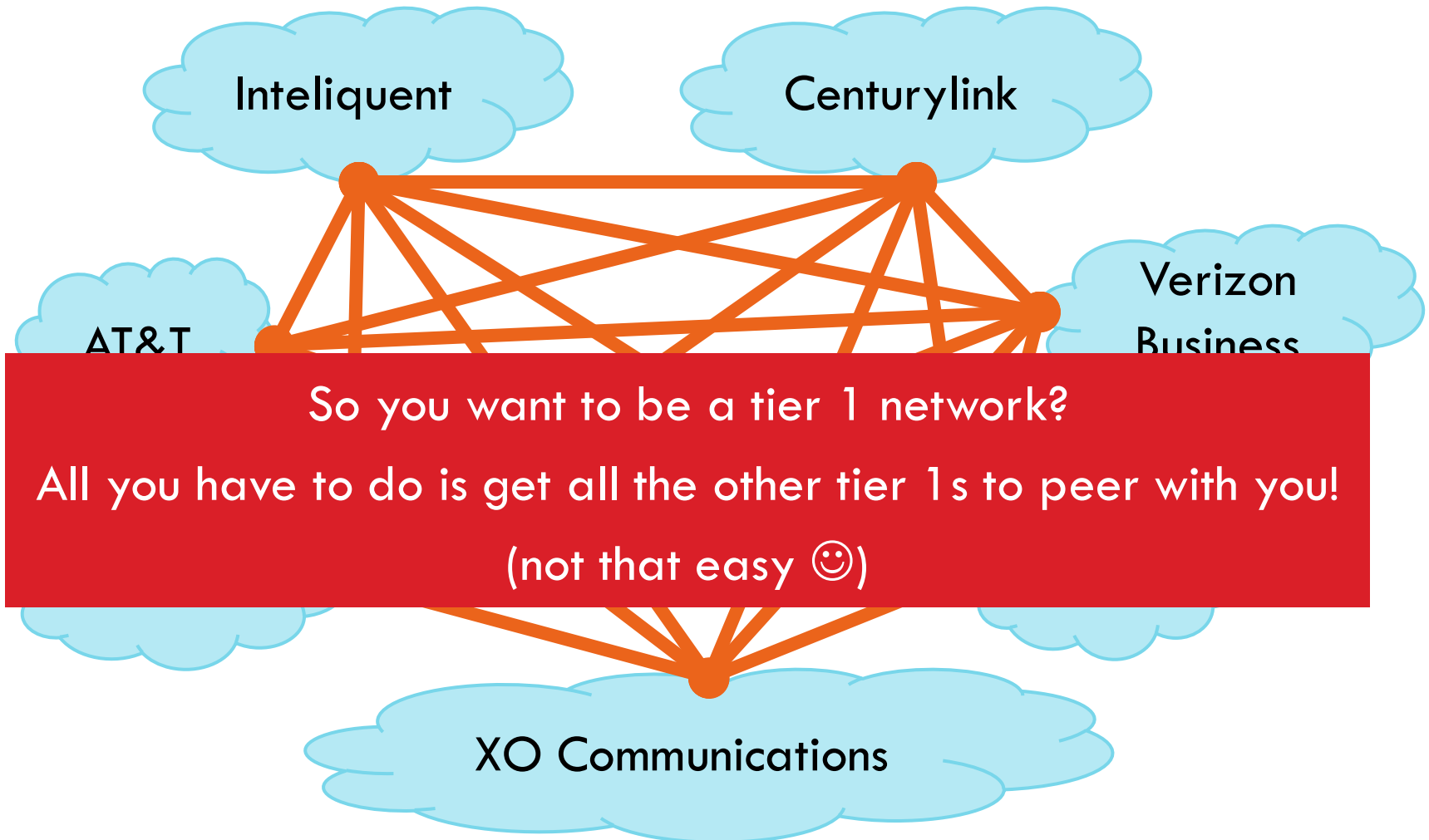


## 73



# Tier-1 ISP Peering

74



[illegible]

# Peering Wars

76

## Peer

- Reduce upstream costs

- Incentive to peer

- Peer-to-peer

- Asymmetric

- Cost

- Incentive

## Don't Peer

- You would rather have

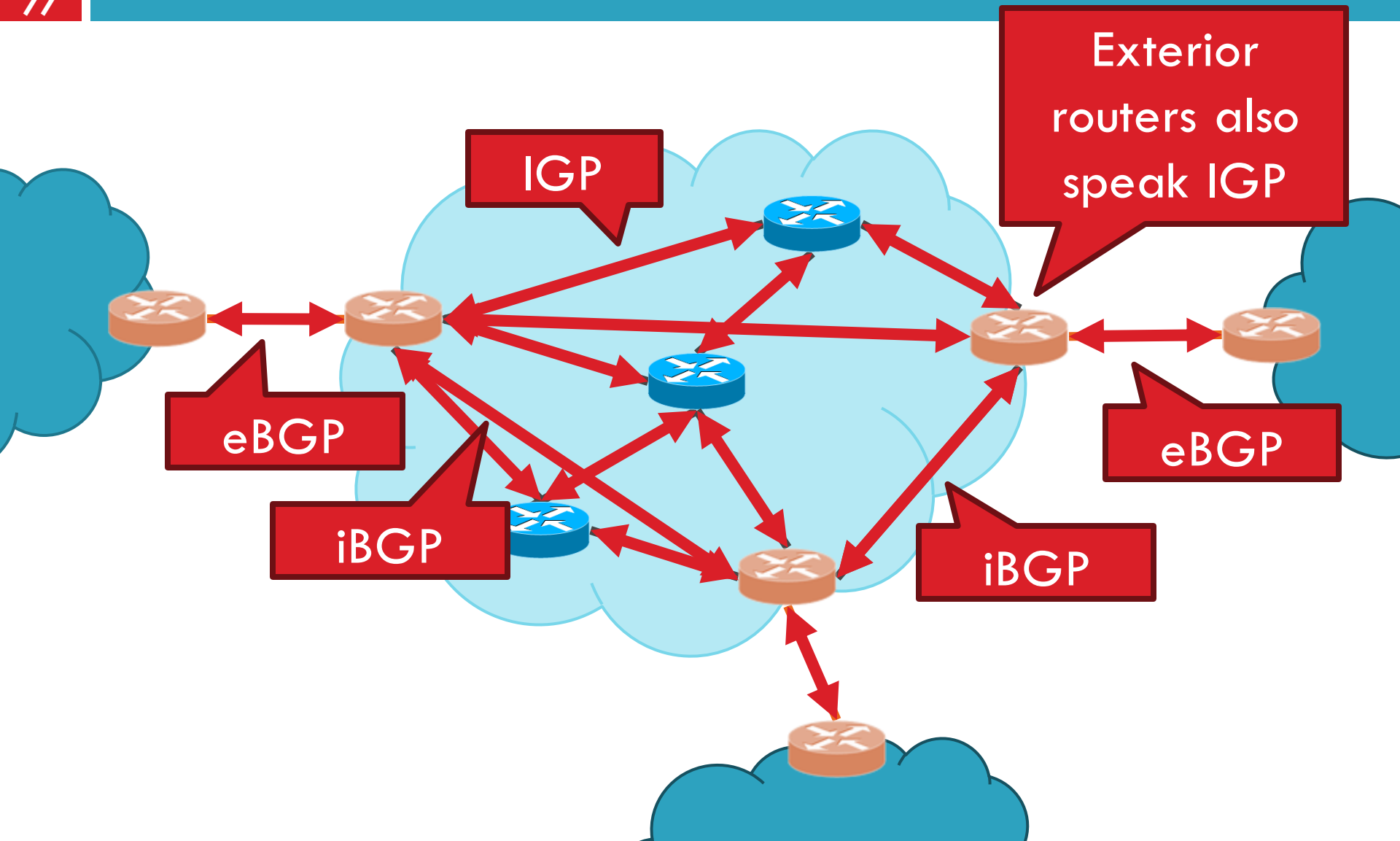
Peering struggles in the ISP world are extremely contentious  
agreements are usually confidential

Example: If you are a customer of my peer why should I peer  
with you? You should pay me too!

Incentive to keep relationships private!

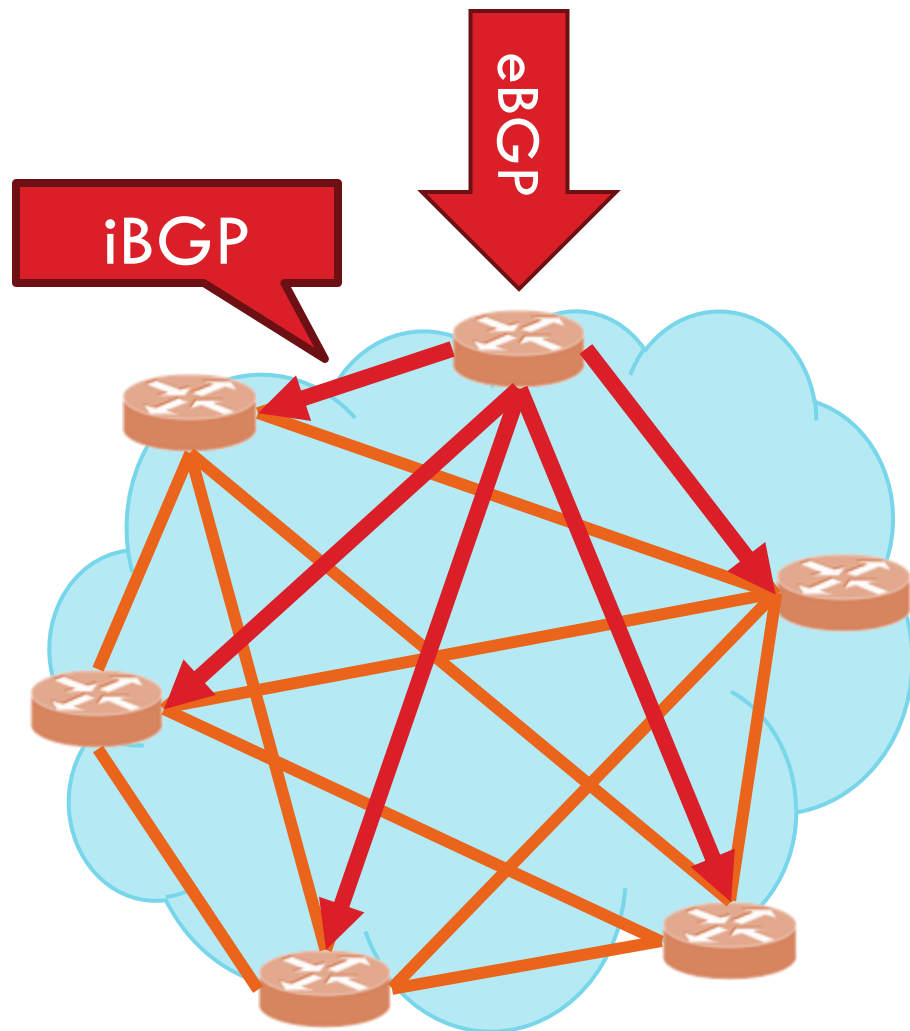
# Two Types of BGP Neighbors

77



# Full iBGP Meshes

78



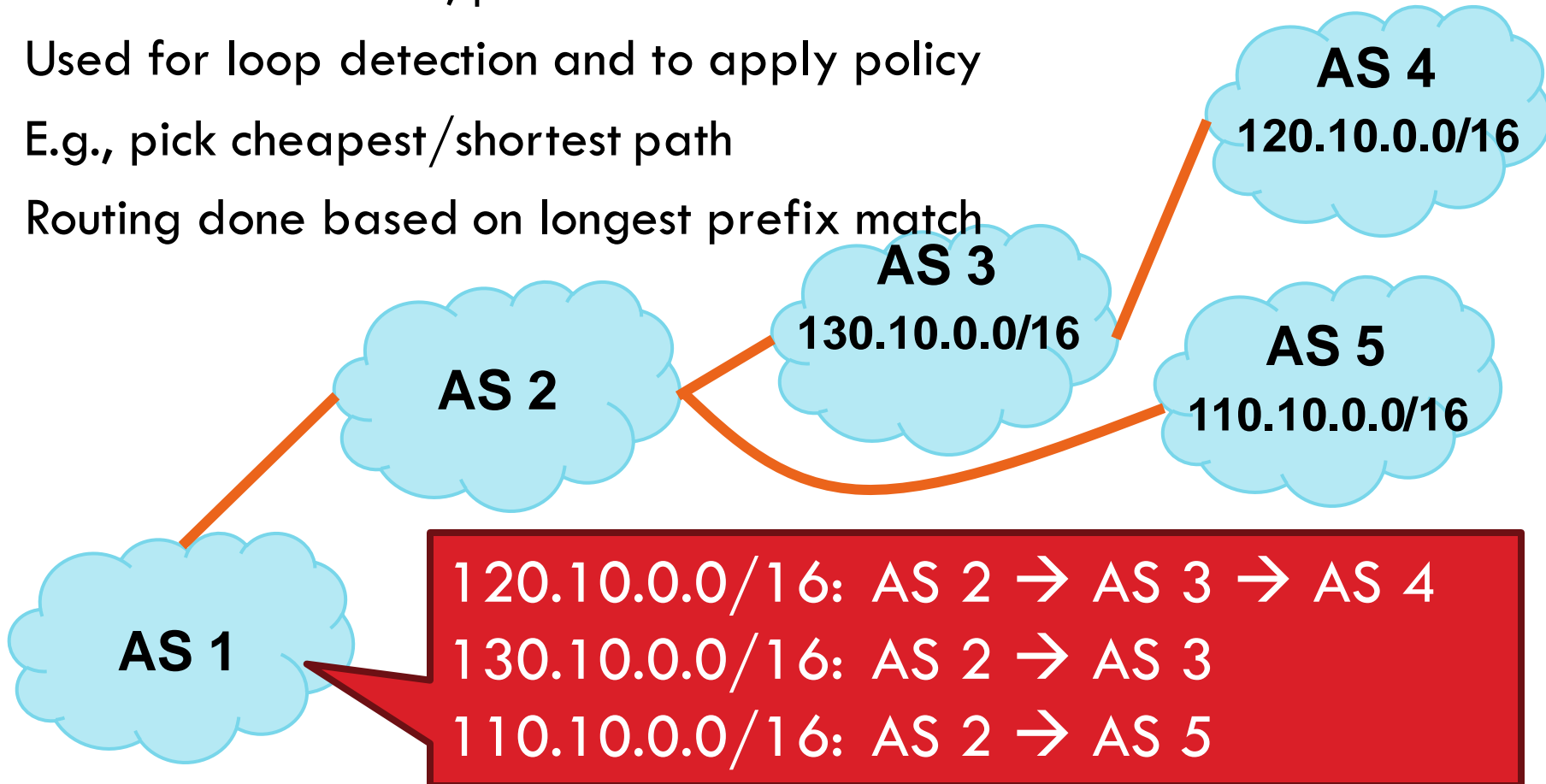
- ❑ Question: why do we need iBGP?
  - ▣ OSPF does not include BGP policy info
  - ▣ Prevents routing loops within the AS
- ❑ iBGP updates do not trigger announcements



# Path Vector Protocol

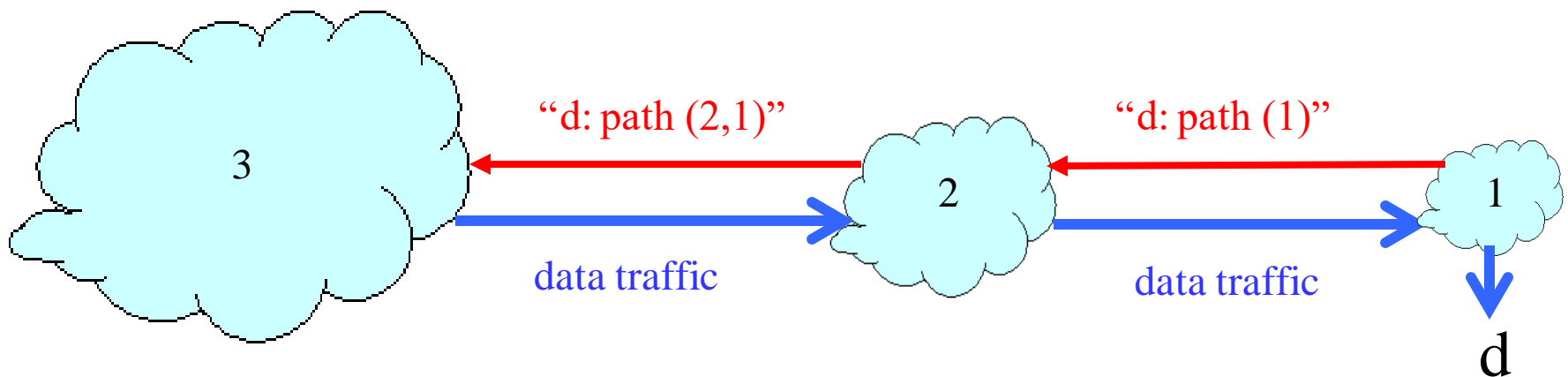
79

- AS-path: sequence of ASs a route traverses
  - ▣ Like distance vector, plus additional information
- Used for loop detection and to apply policy
- E.g., pick cheapest/shortest path
- Routing done based on longest prefix match



# Path-Vector Routing

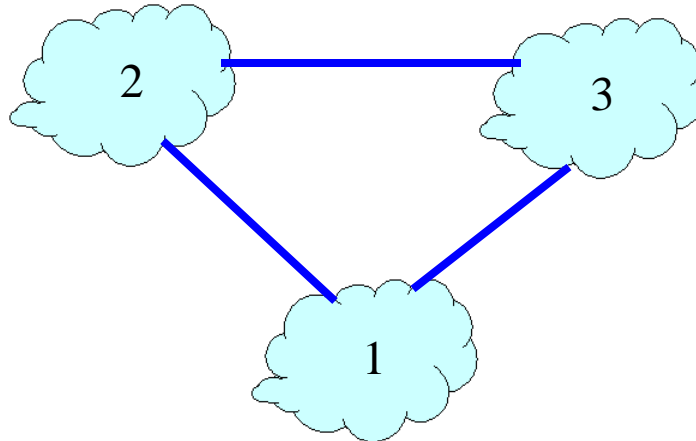
- ❑ Extension of distance-vector routing
  - ▣ Support flexible routing policies
  - ▣ Avoid count-to-infinity problem
- ❑ Key idea: advertise the entire path
  - ▣ Distance vector: send *distance metric* per dest  $d$
  - ▣ Path vector: send the *entire path* for each dest  $d$





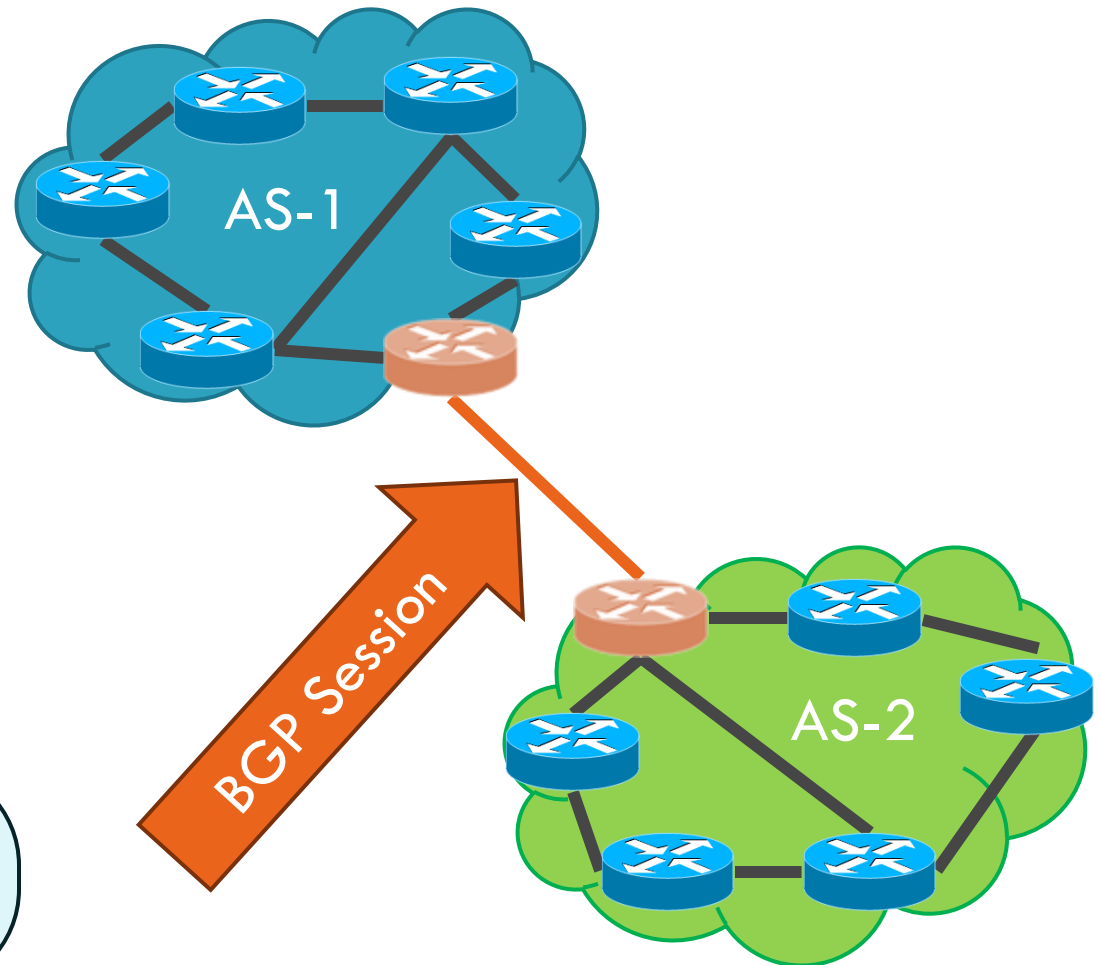
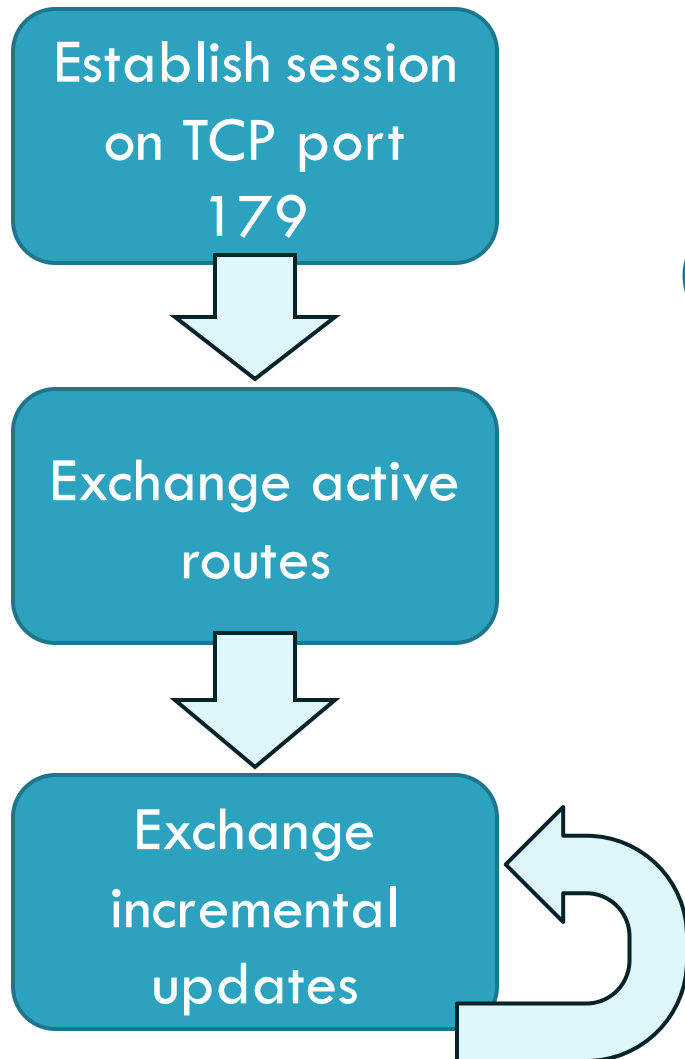
# Flexible Policies

- ❑ Each node can apply local policies
  - ▣ Path selection: Which path to use?
  - ▣ Path export: Which paths to advertise?
- ❑ Examples
  - ▣ Node 2 may prefer the path “2, 3, 1” over “2, 1”
  - ▣ Node 1 may not let node 3 hear the path “1, 2”



# BGP Operations (Simplified)

82



# Four Types of BGP Messages

83

- ❑ **Open**: Establish a peering session.
- ❑ **Keep Alive**: Handshake at regular intervals.
- ❑ **Notification**: Shuts down a peering session.
- ❑ **Update**: Announce new routes or withdraw previously announced routes.

announcement = IP prefix + attributes values

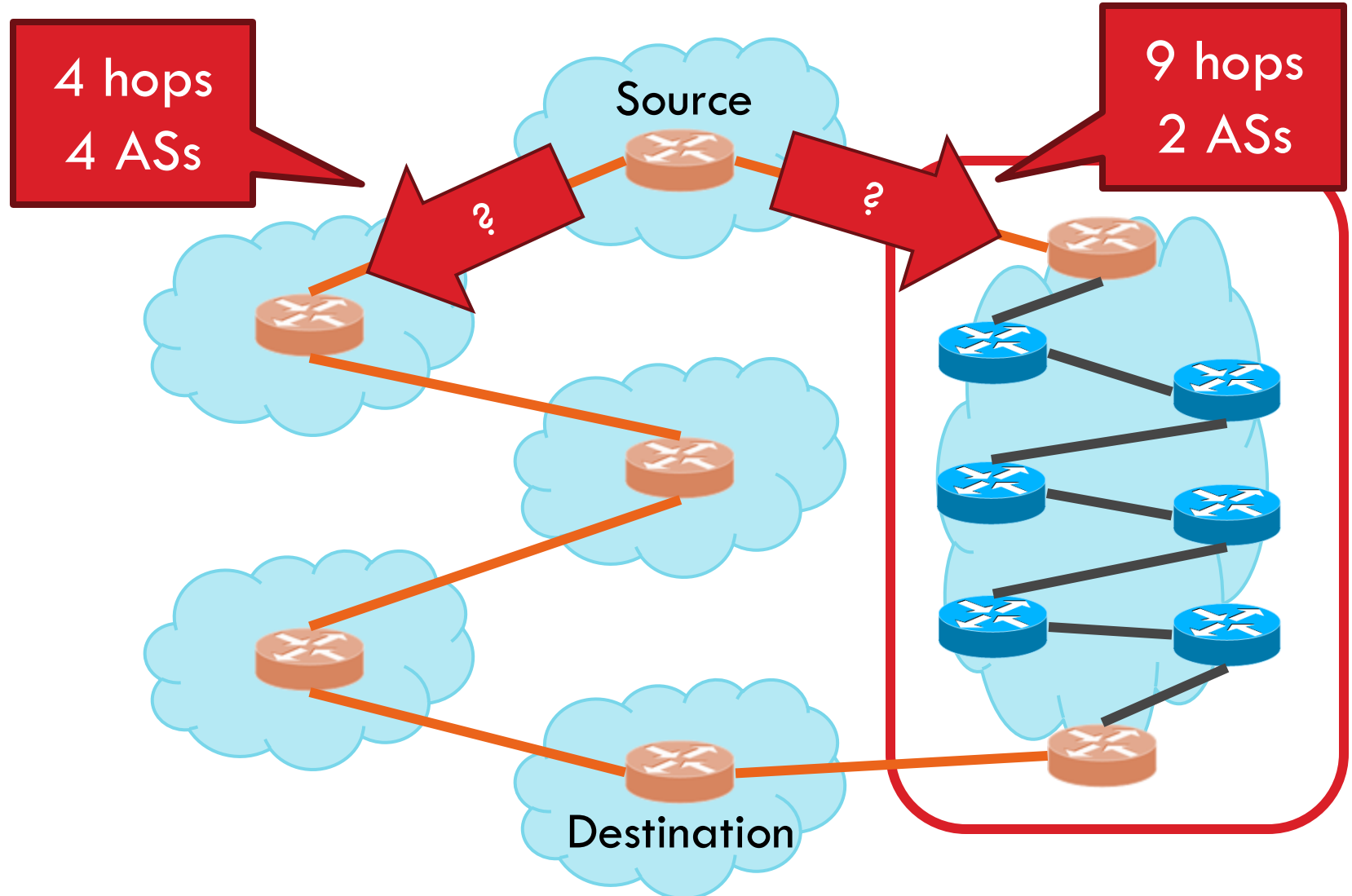
# BGP Attributes

84

- ❑ Attributes used to select “best” path
  - ❑ LocalPref
    - Local preference policy to choose most preferred route
    - Overrides default fewest AS behavior
  - ❑ Multi-exit Discriminator (MED)
    - Specifies path for external traffic destined for an internal network
    - Chooses peering point for your network
  - ❑ Import Rules
    - What route advertisements do I accept?
  - ❑ Export Rules
    - Which routes do I forward to whom?

# Shortest AS Path $\neq$ Shortest Path

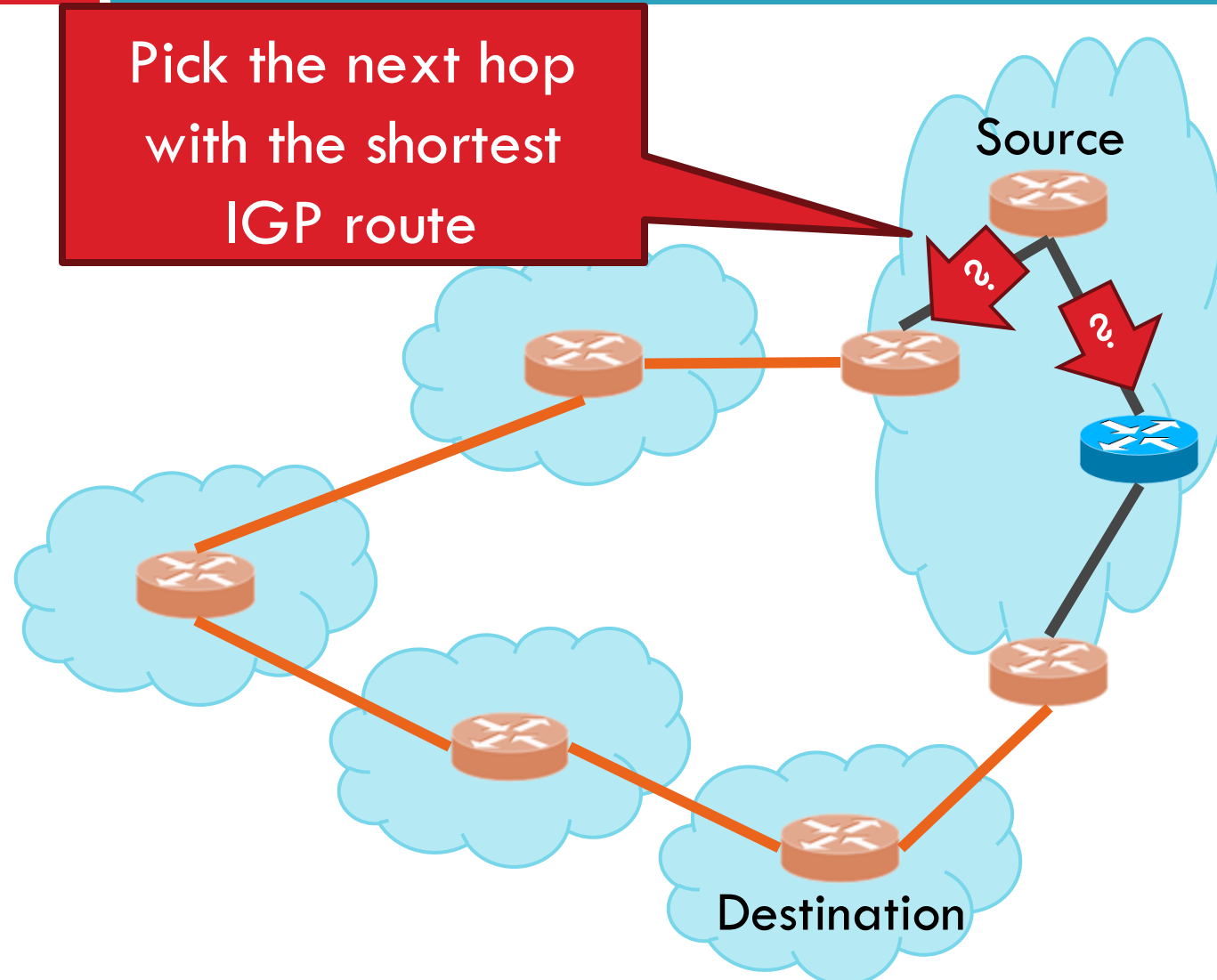
85



# Hot Potato Routing

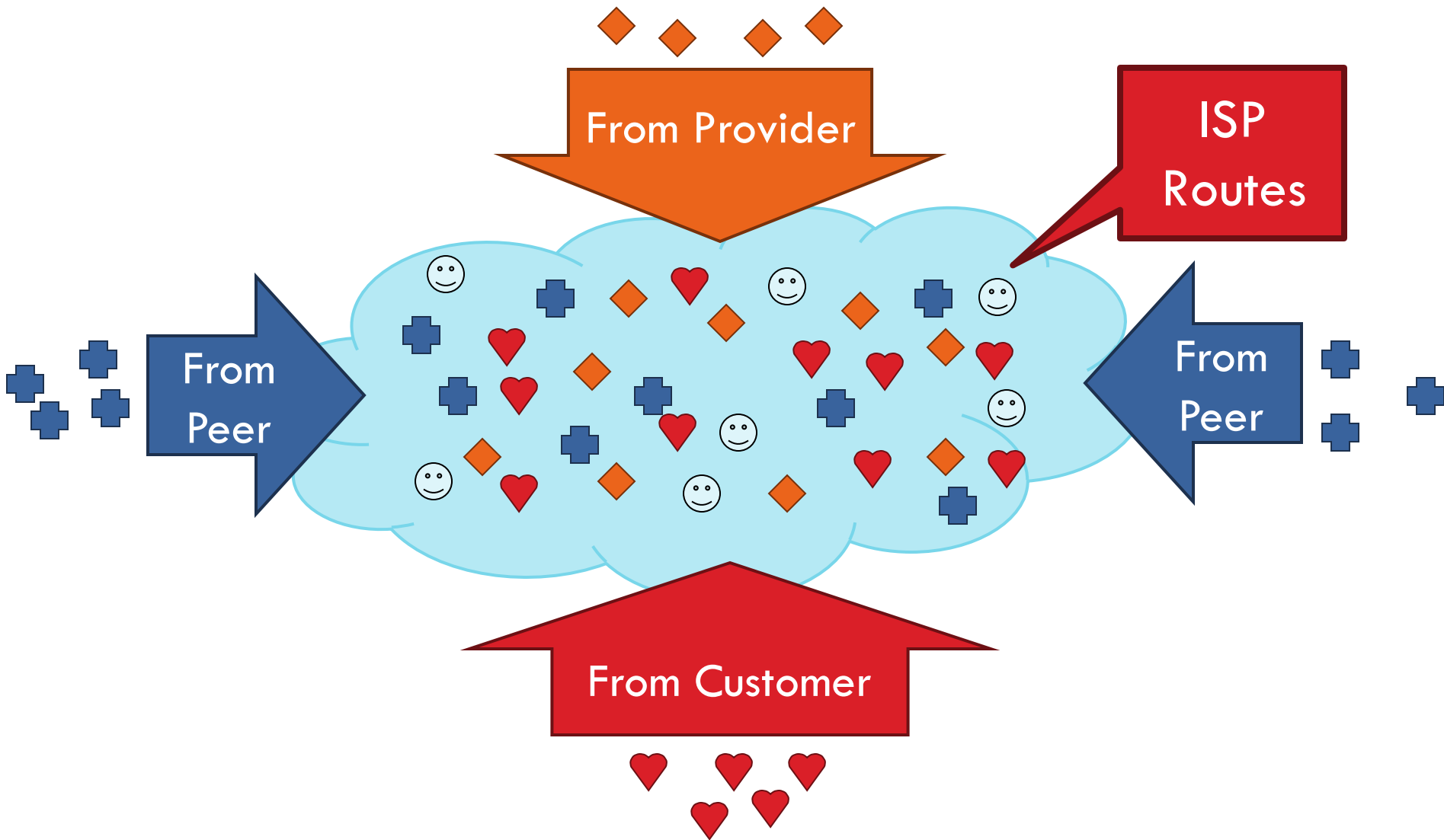
86

Pick the next hop  
with the shortest  
IGP route



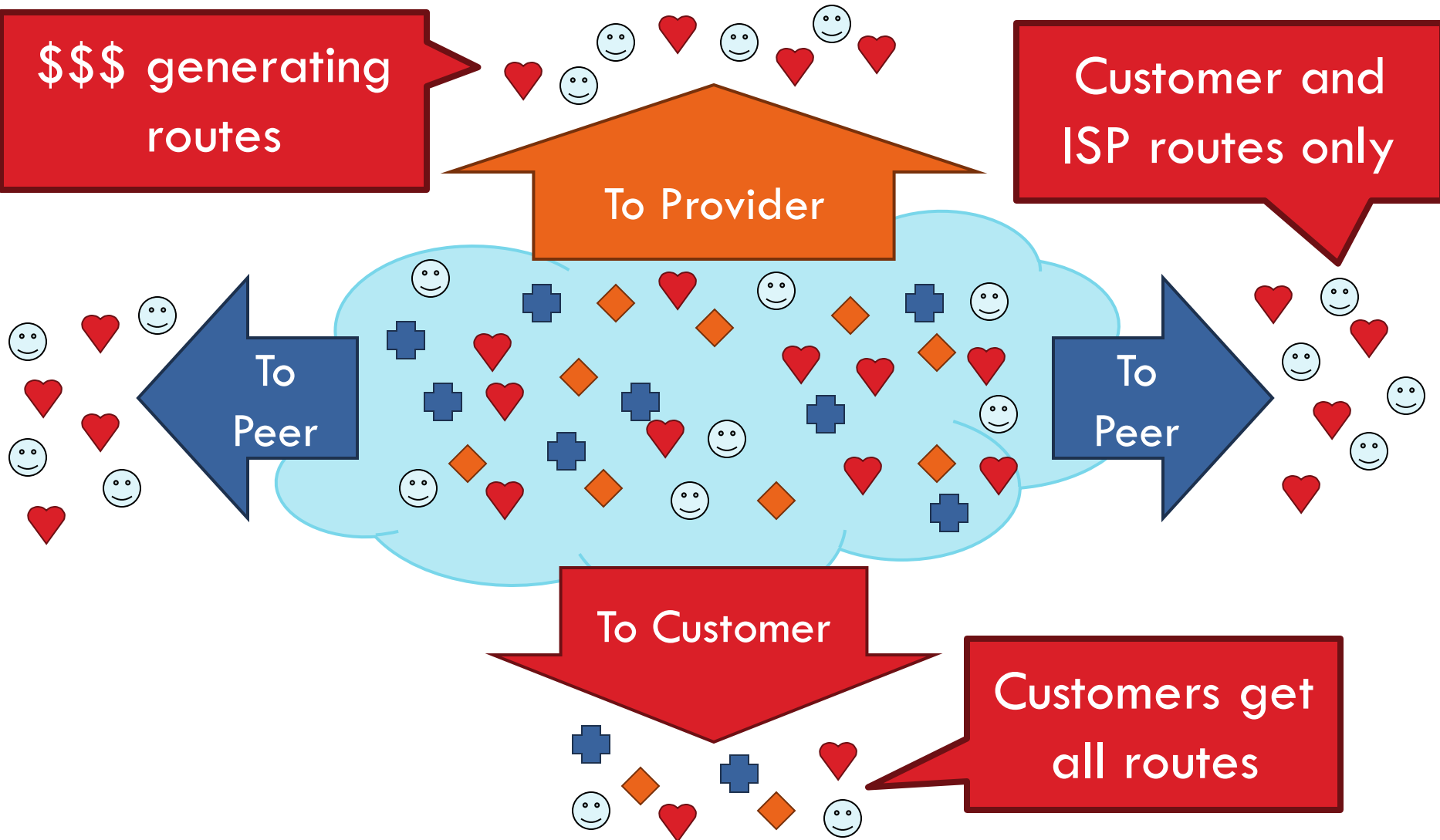
# Importing Routes

87



# Exporting Routes

88

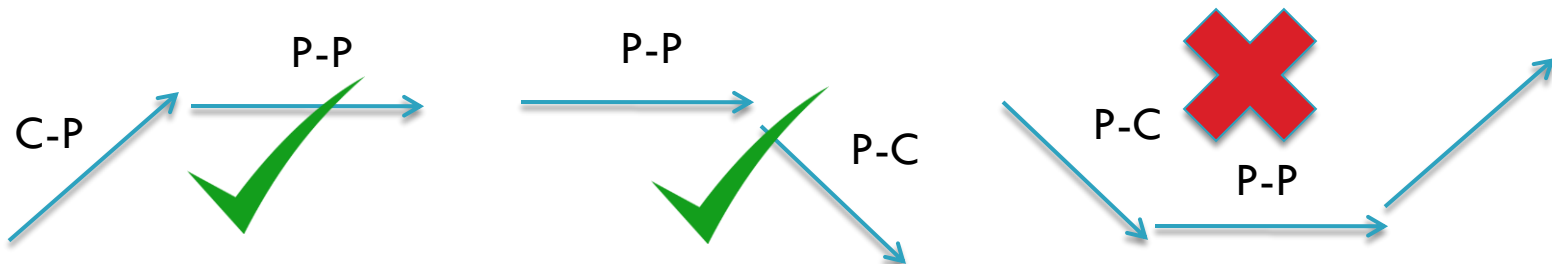




# Modeling BGP

89

- ❑ AS relationships
  - ▣ Customer/provider
  - ▣ Peer
  - ▣ Sibling, IXP
- ❑ Gao-Rexford model
  - ▣ AS prefers to use customer path, then peer, then provider
    - Follow the money!
  - ▣ Valley-free routing
  - ▣ Hierarchical view of routing (incorrect but frequently used)



# AS Relationships: It's Complicated

90

- ❑ GR Model is strictly hierarchical
  - ▣ Each AS pair has exactly one relationship
  - ▣ Each relationship is the same for all prefixes
- ❑ In practice it's much more complicated
  - ▣ Rise of widespread peering
  - ▣ Regional, per-prefix peerings
  - ▣ Tier-1's being shoved out by “hypergiants”
  - ▣ IXPs dominating traffic volume
- ❑ Modeling is very hard, very prone to error
  - ▣ Huge potential impact for understanding Internet behavior

# Other BGP Attributes

91

## □ AS\_SET

- ▣ Instead of a single AS appearing at a slot, it's a set of Ases

## □ Communities

- ▣ Arbitrary number that is used by neighbors for routing decisions
  - Export this route only in Europe
  - Do not export to your peers
- ▣ Usually stripped after first interdomain hop
- ▣ Why?

## □ Prepending

- ▣ Lengthening the route by adding multiple instances of ASN
- ▣ Why?