Computer Networks

Lecture 11: Network layer Part 3

Based on slides from D. Choffnes Northeastern U. and P. Gill from StonyBrook University

Revised Autumn 2015 by S. Laki

Shortest Path Routing

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. **Mininize the number of hops along the path**

Dijkstra's Shortest Path Algorithm

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 $Q \neq 0$ do
        $u \leftarrow$ ExtractMin(Q)
        $S \leftarrow S \cup \{u\}$
        for $u \in Adj[u]$ do
            Relax(u,v,w)
```

executed Θ(V) times

Θ(E) times in total

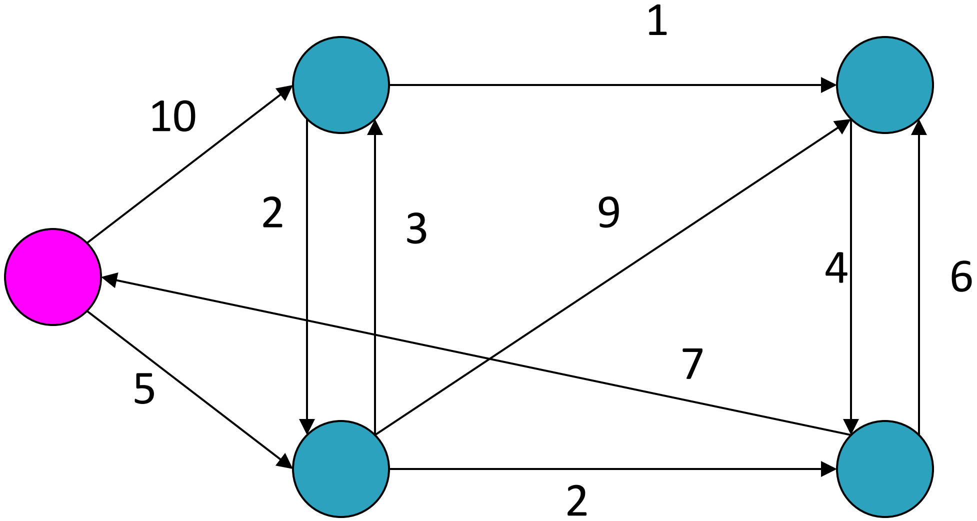
```
InitializeSingleSource(graph G, vertex  s)
    for $v \in V[G]$ do
        $d[v] \leftarrow \infty$
        $p[v] \leftarrow 0$
    $d[s] \leftarrow 0$
```

Θ(V)

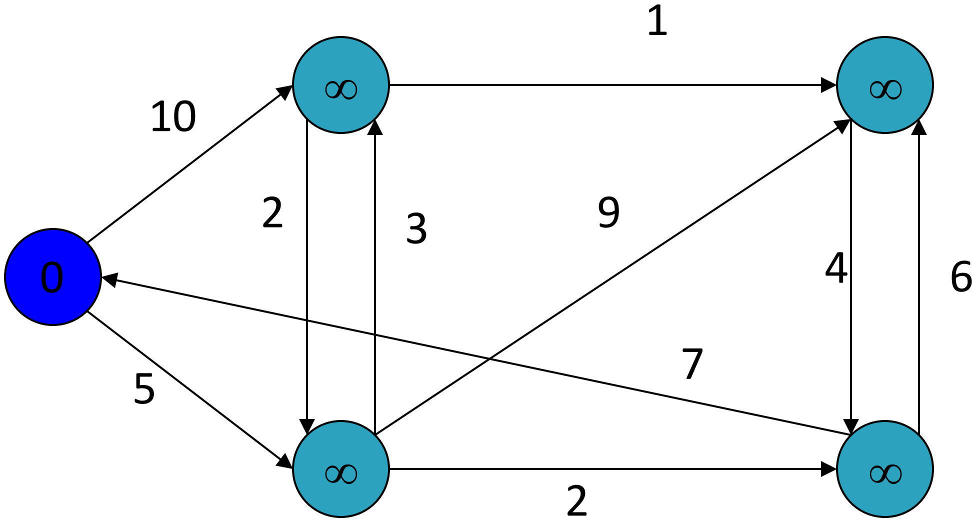
```
Relax(vertex  u, vertex v, weight w)
    if $d[v] > d[u] + w(u,v)$ then
        $d[v] \leftarrow d[u] + w(u,v)$
        $p[v] \leftarrow u$
```

Θ(1) ?

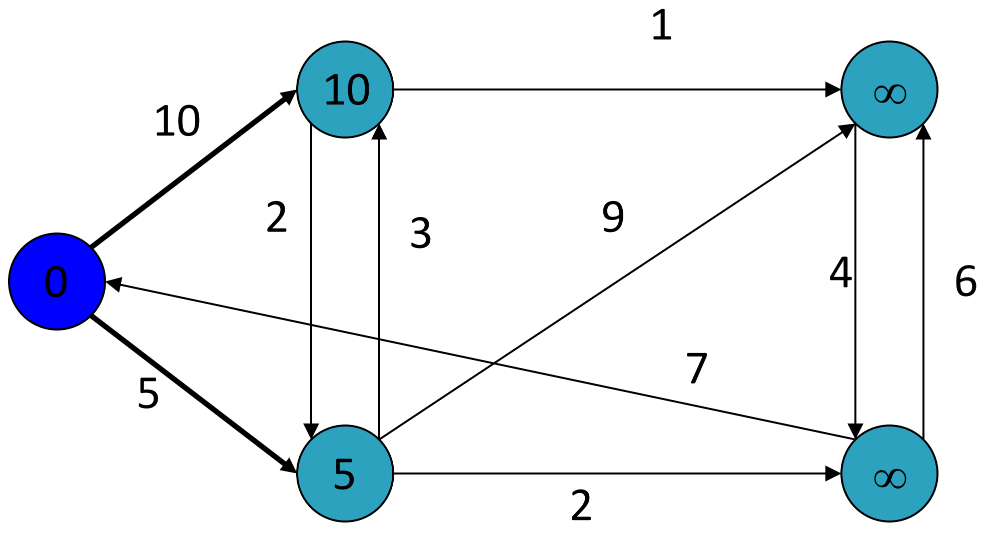
Dijkstra’s Algorithm - Example



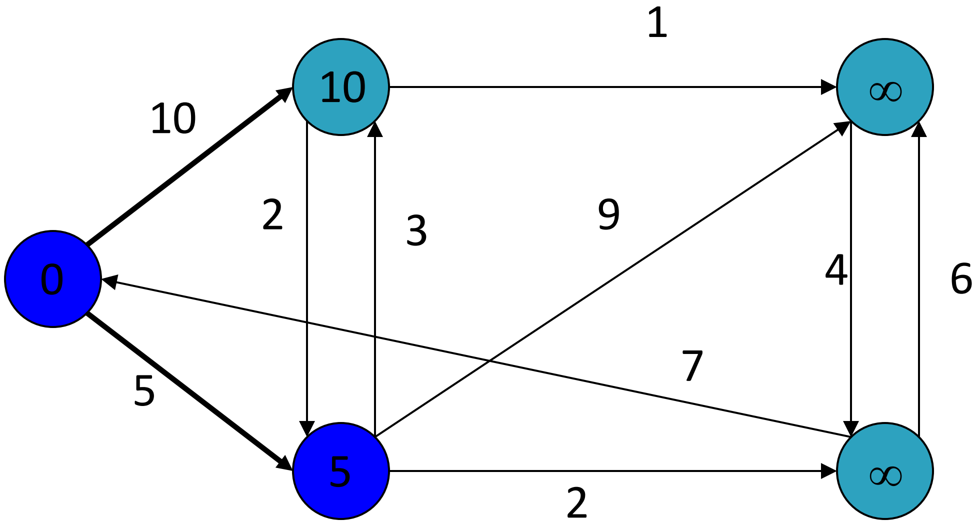
Dijkstra’s Algorithm - Example



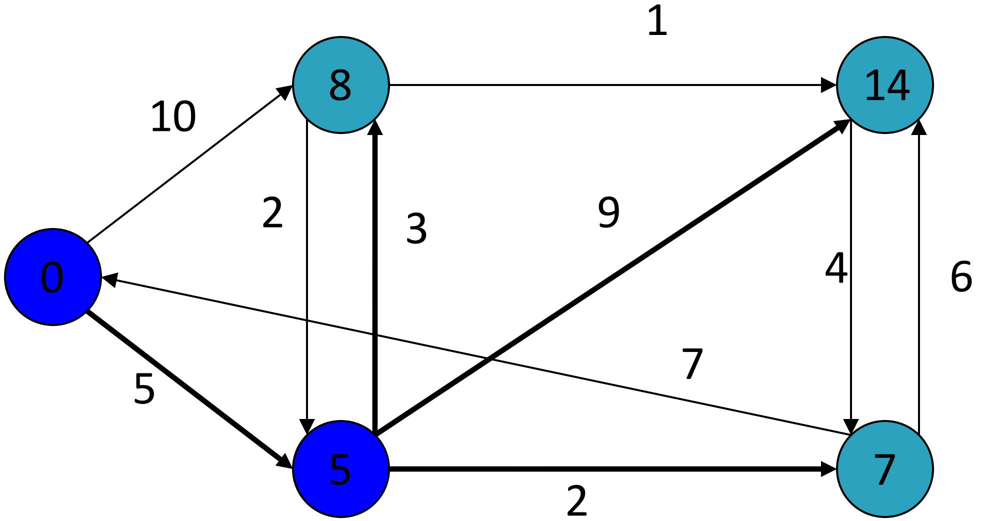
Dijkstra’s Algorithm - Example



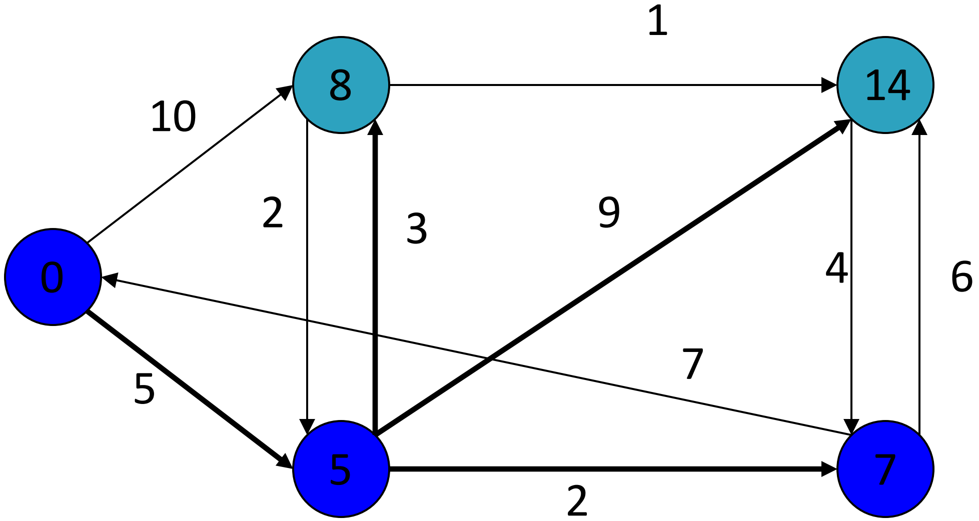
Dijkstra’s Algorithm - Example



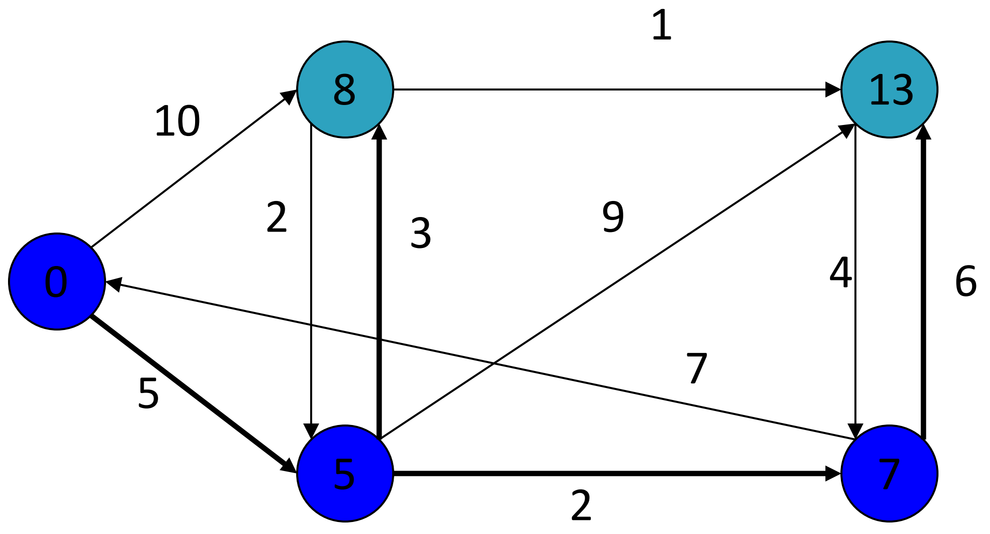
Dijkstra’s Algorithm - Example



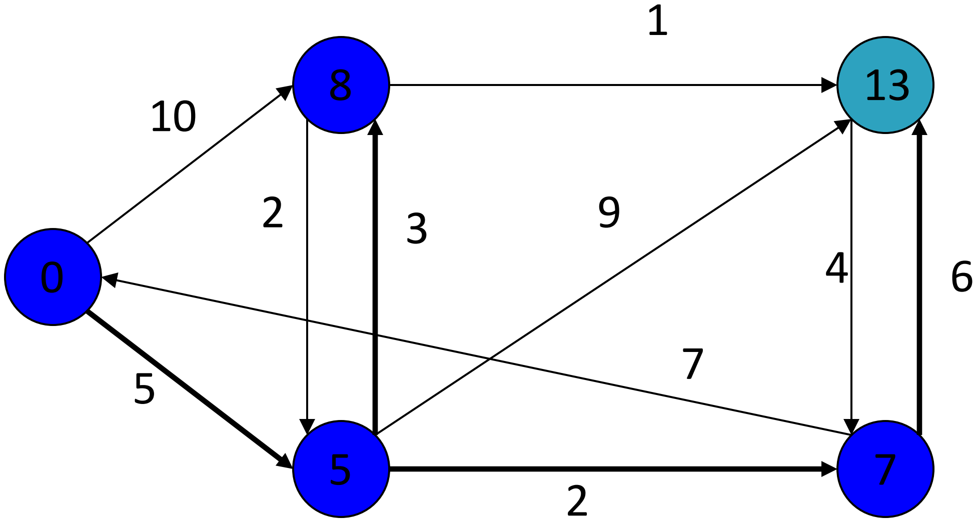
Dijkstra’s Algorithm - Example



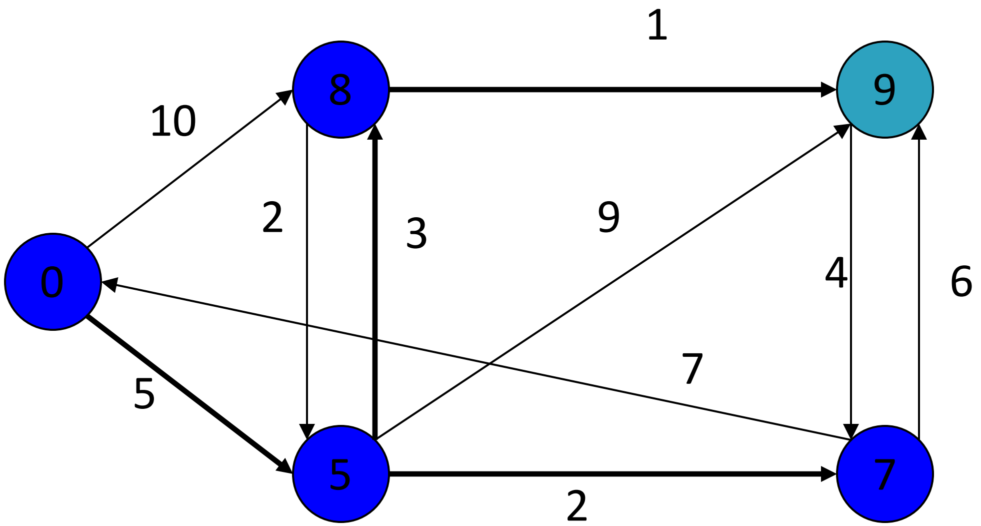
Dijkstra’s Algorithm - Example



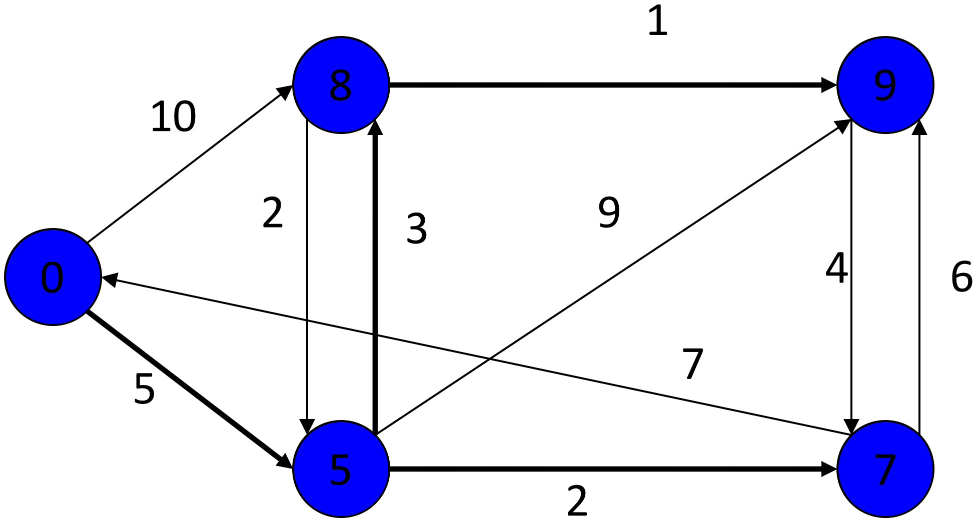
Dijkstra’s Algorithm - Example



Dijkstra’s Algorithm - Example



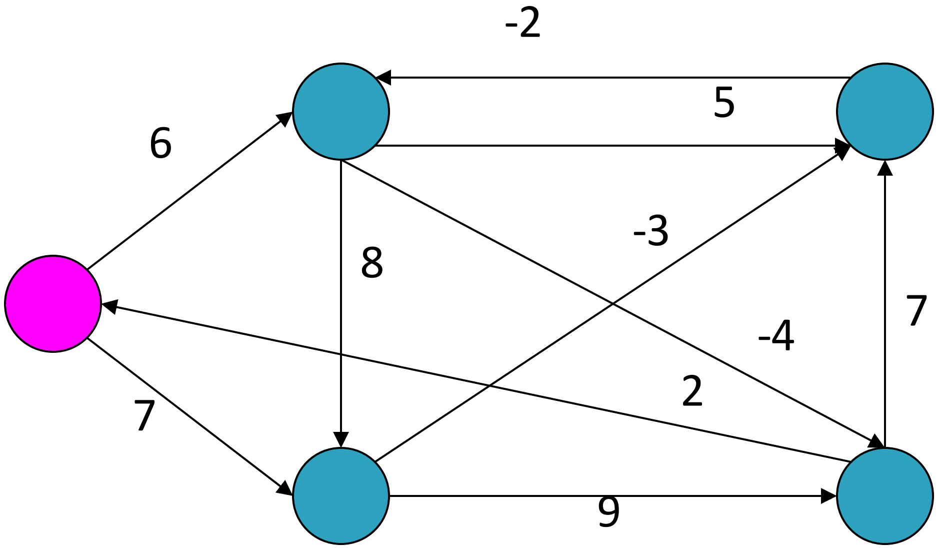
Dijkstra’s Algorithm - Example



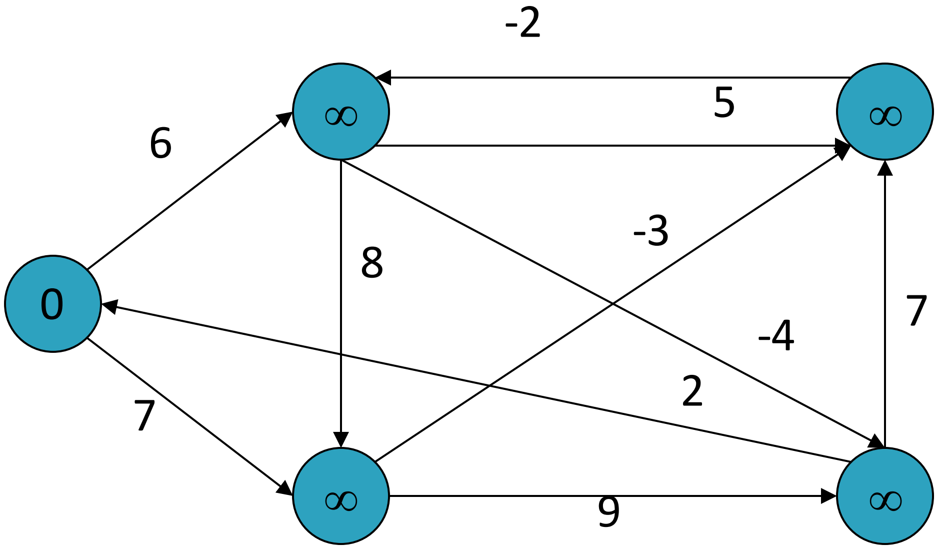
Bellman-Ford Algorithm

```
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
```

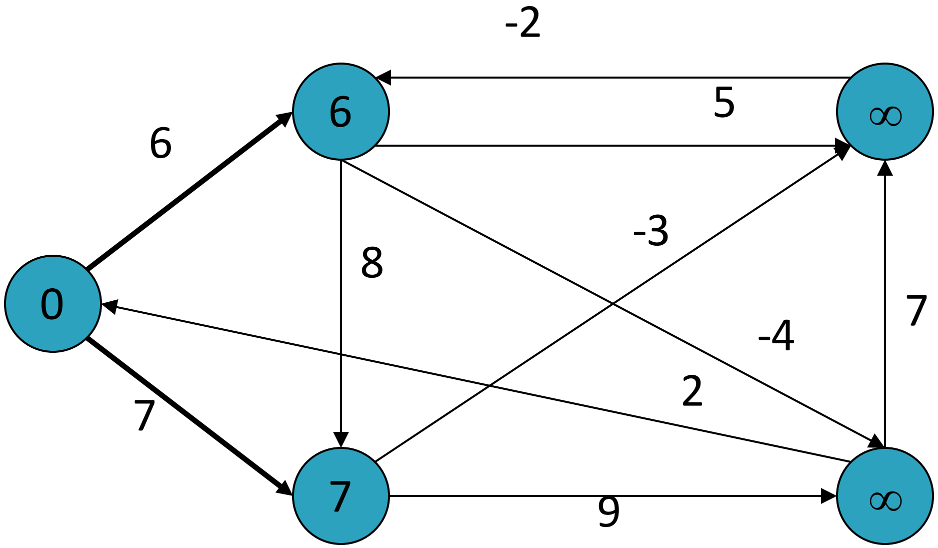
Bellman-Ford Algorithm - Example



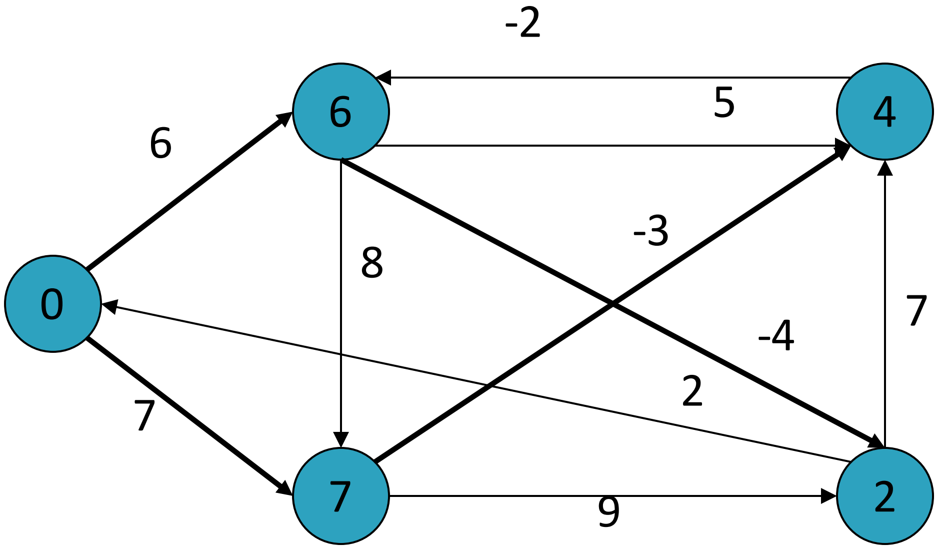
Bellman-Ford Algorithm - Example



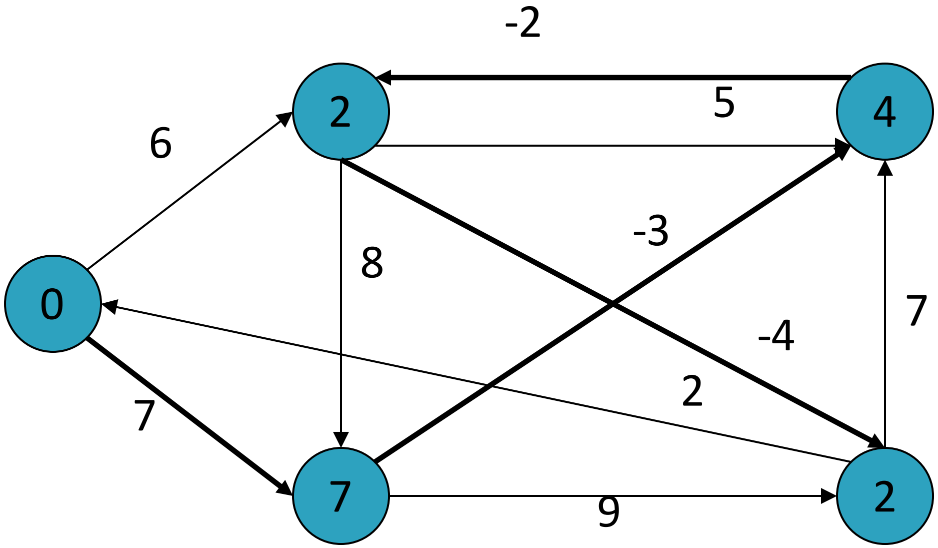
Bellman-Ford Algorithm - Example



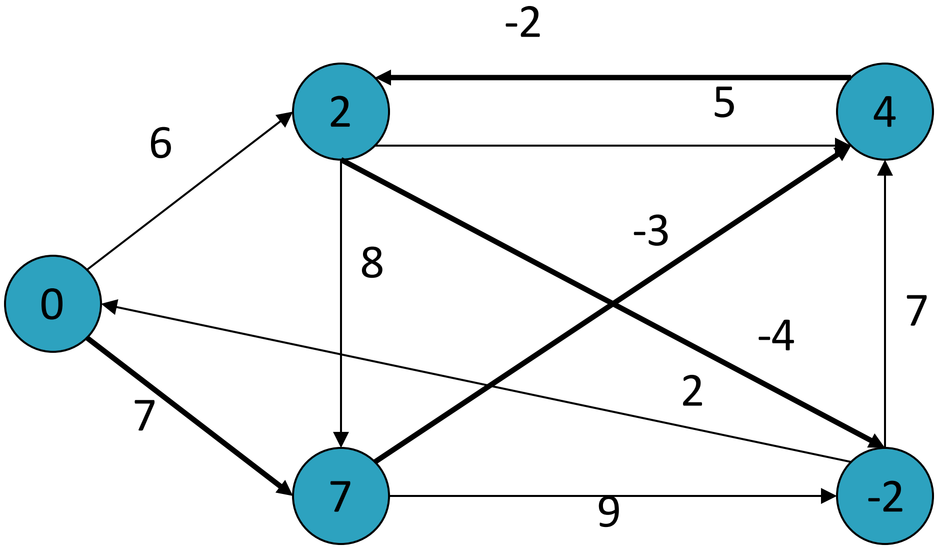
Bellman-Ford Algorithm - Example



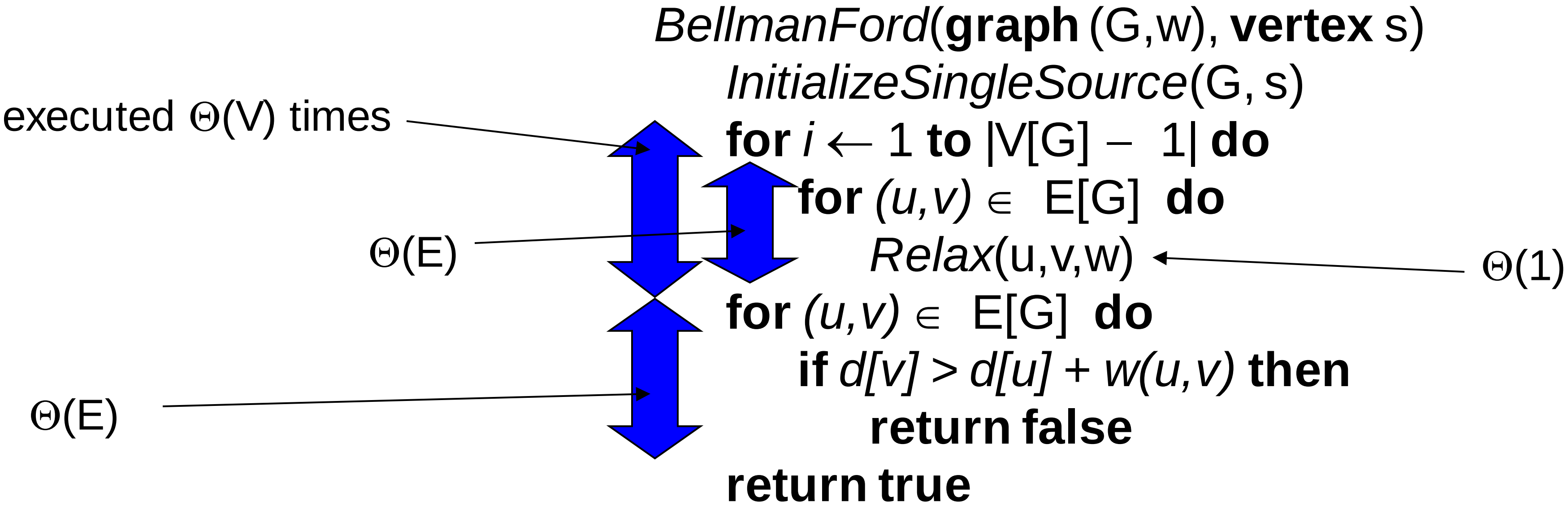
Bellman-Ford Algorithm - Example



Bellman-Ford Algorithm - Example



Bellman-Ford Algorithm - Complexity



```
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
```

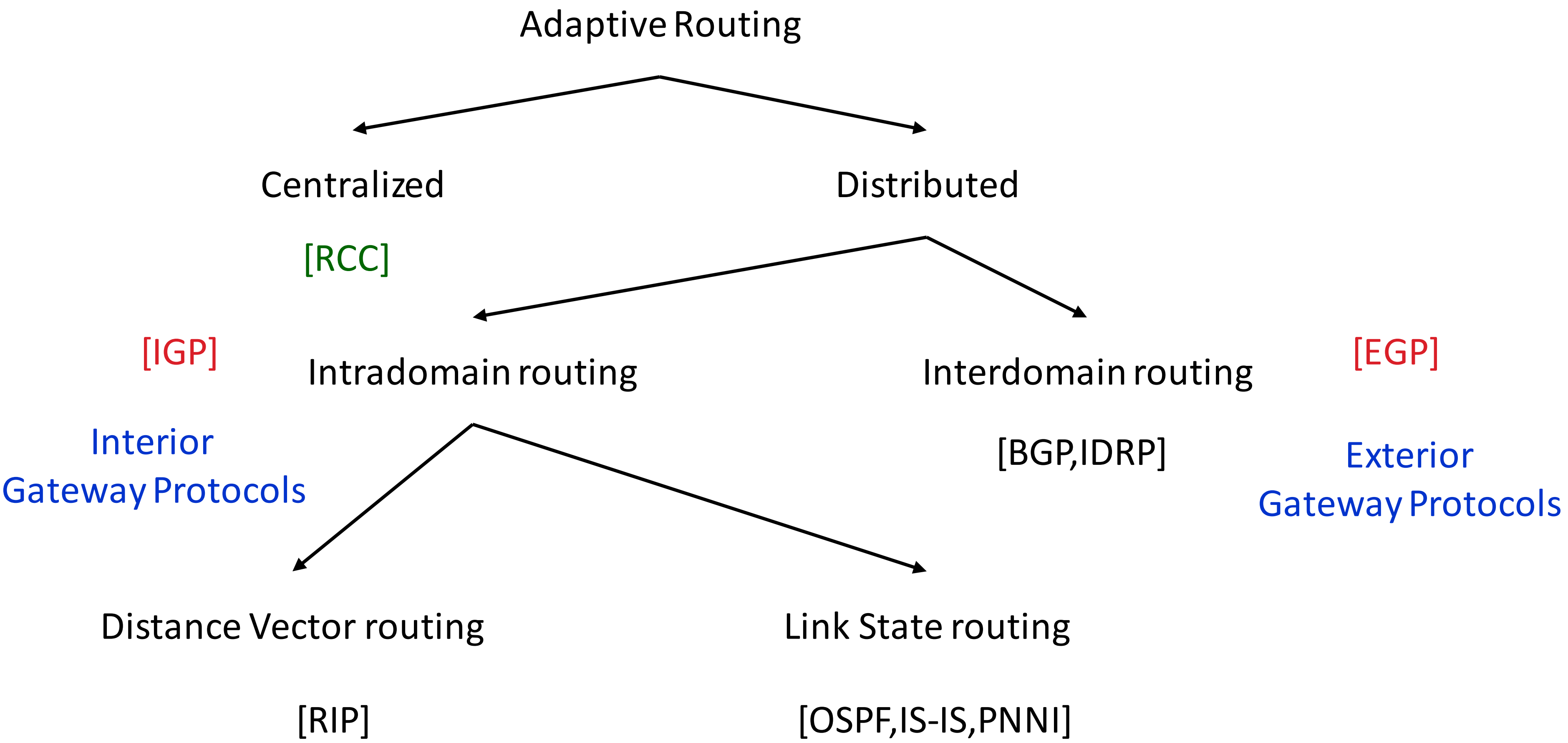
executed $\Theta(V)$ times

$\Theta(E)$

$\Theta(1)$

$\Theta(E)$

Internetwork Routing [Halsall]



Adaptive Routing

Centralized

[RCC]

Distributed

[IGP]

Intradomain routing

Interior Gateway Protocols

Interdomain routing

[EGP]

[BGP,IDRP]

Exterior Gateway Protocols

Distance Vector routing

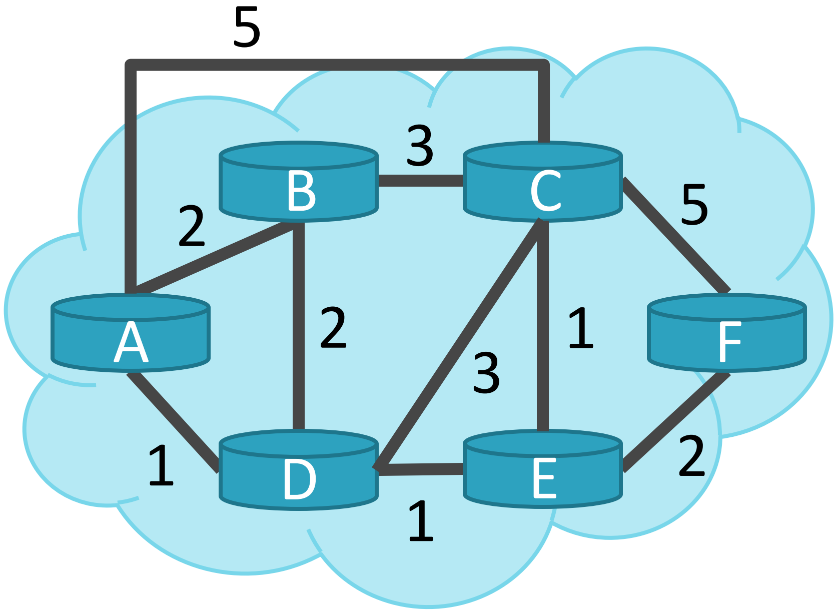
[RIP]

Link State routing

[OSPF,IS-IS,PNNI]

Routing Problems

- 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

- 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

Outline

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

Distance Vector Routing

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

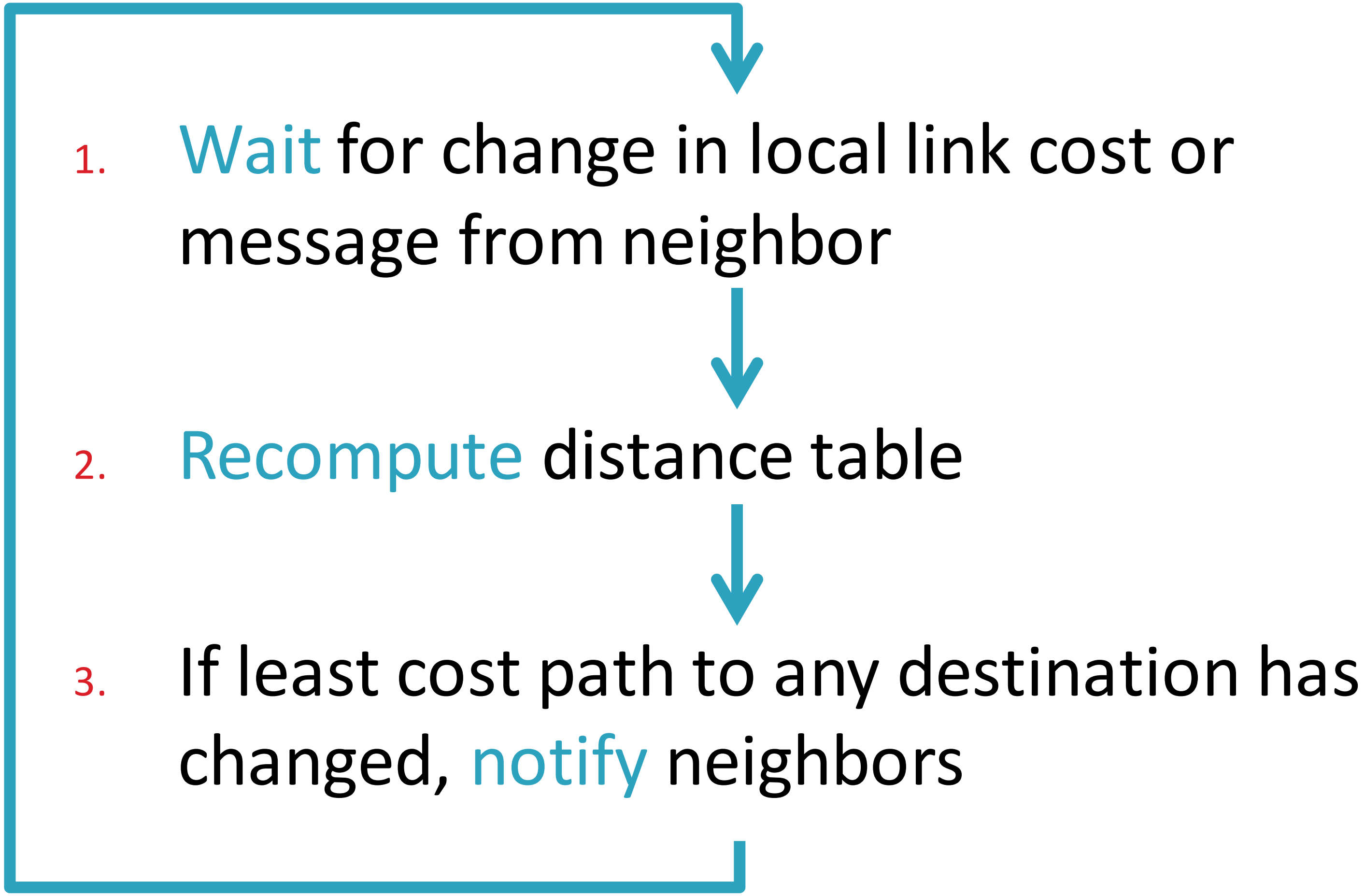
DV Table
at Node C

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

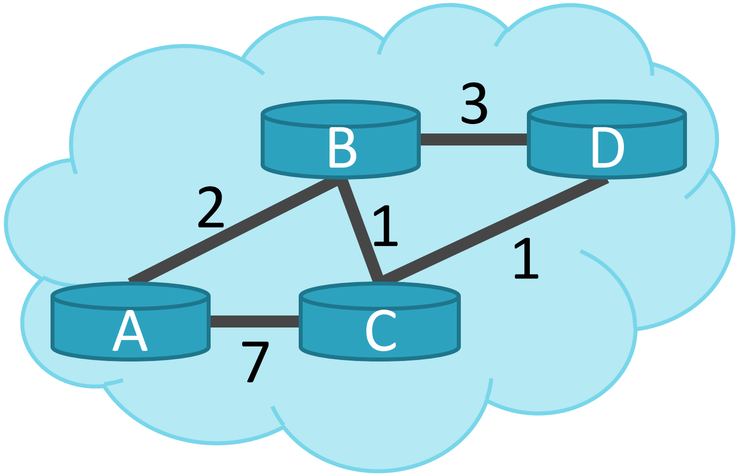
- Routing Information Protocol (RIP)

Distance Vector Routing Algorithm



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



Node A

Dest.	Cost	Next
B	2	B
C	7	C
D	∞	

Node B

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

```
Initialization:
  for all neighbors $V$ do
    if $V$ adjacent to $A$
      $D(A, V) = c(A,V)$;
  else
      $D(A, V) = \infty$;
…
```

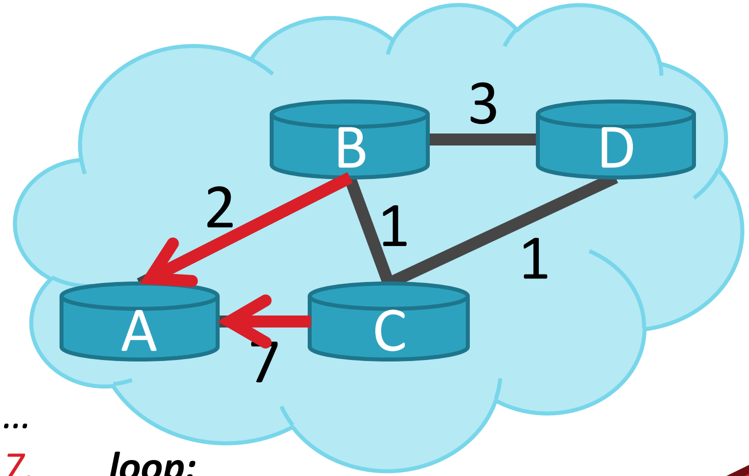
Node C

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

Node D

Dest.	Cost	Next
A	∞	
B	3	B
C	1	C

Distance Vector: 1st Iteration



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

```
...
loop:
...
  else if (update D(V, Y) rec
    for all de
      if (dest
        D(A,Y)
      else
        D(A, Y) =
                m
                D(A, V) + D(V, Y));
  if (there is a new min. for dest. Y)
    send D(A, Y) to all neighbors
forever
```

e D

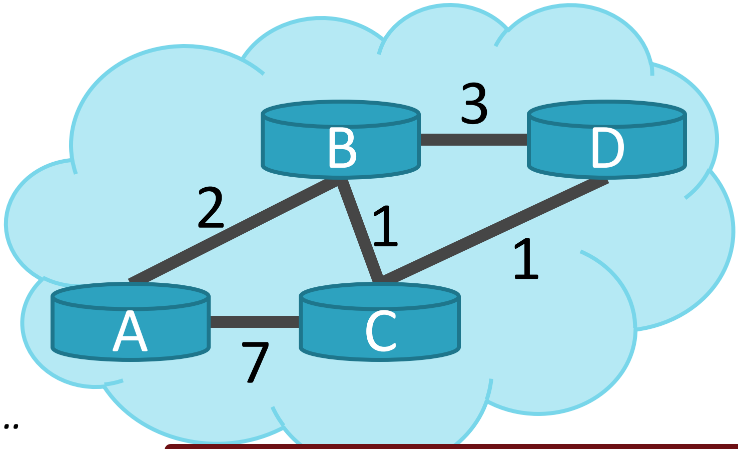
			Next
3	B	4	B

D(A,C) = min(D(A,C), D(A,B)+D(B,C))
= min(7, 2 + 1) = 3

D(A,D) = min(D(A,D), D((C,D))
= min(∞

D(A,D) = min(D(A,D), D(A,B)+D(B
= min(8, 2 + 3) = 5

Distance Vector: End of 3rd Iteration



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

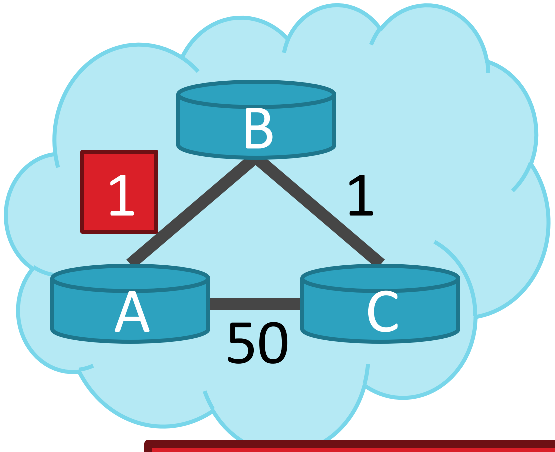
```
…
loop
…
else
  for
    if

    else
      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
```

- Nothing changes, algorithm terminates
- Until something changes…

Dest.	Cost	Next
A	3	B
B	1	B
D	1	D

Dest.	Cost	Next
A	4	C
B	2	C
C	1	C

```
loop:
  wait (link cost update or update message)
  if (c(A,V) changes by d)
    for all destinations Y through V do
      D(A,Y) = D(A,Y) + d
  else if (update D(V, Y) received from V)
    for all destinations Y do
      if (destination Y through V)
        D(A,Y) = D(A,V) + D(V, Y);
      else
        D(A, Y) = min(D(A, Y), D(A, V) + D(V, Y));
```



Link Co
Algori

Good news travels fast

Algorithm
erminates

Node B

D	C	N
A	4	A
C	1	B

	C	N
A	1	A
C	1	B

D	C	N
A	1	A
C	1	B

D	C	N
A	1	A
C	1	B

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

D	C	N
A	2	B
B	1	B

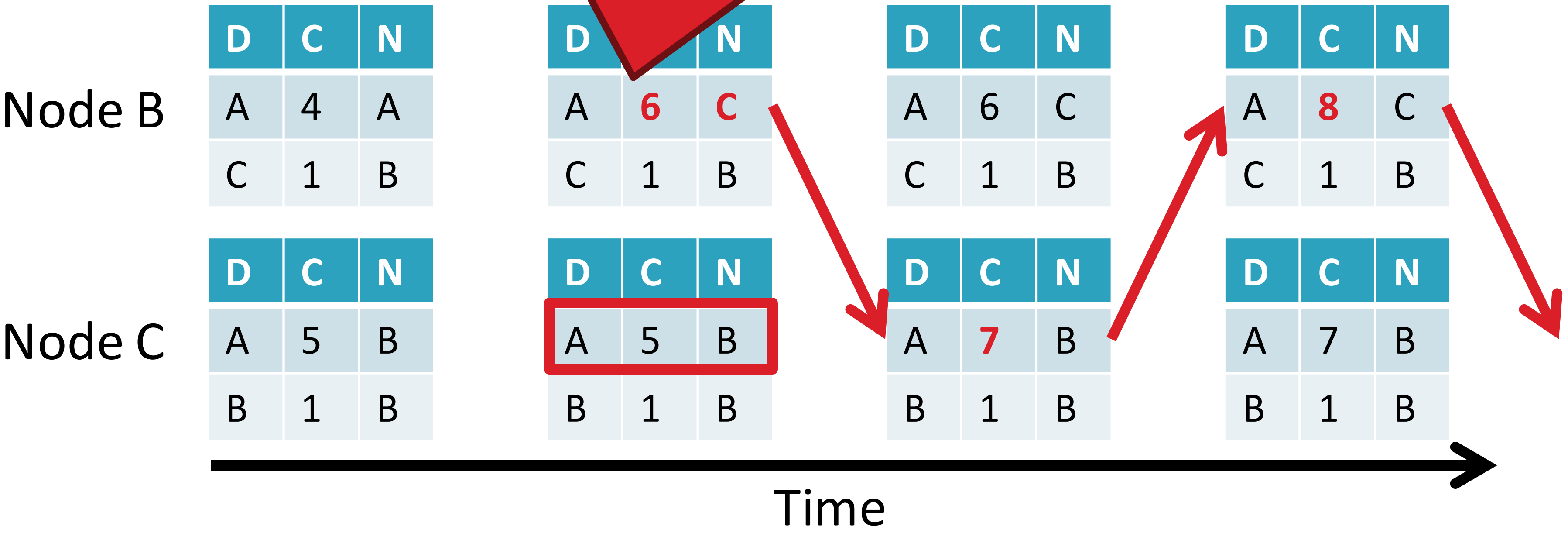
D	C	N
A	2	B
B	1	B

Time

Count to Infinity Problem

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

Bad news travels



Node B

D	C	N
A	4	A
C	1	B

D		N
A	6	C
C	1	B

D	C	N
A	6	C
C	1	B

D	C	N
A	8	C
C	1	B

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

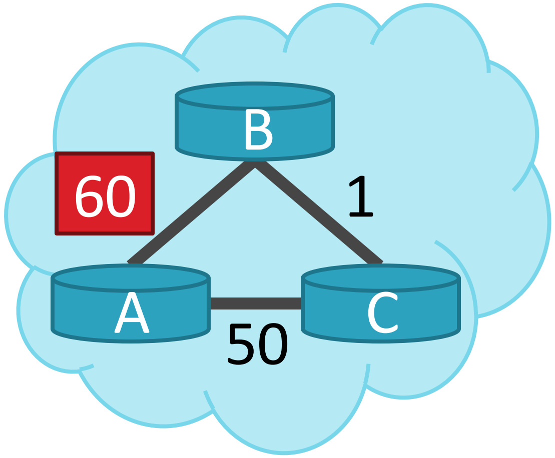
D	C	N
A	7	B
B	1	B

D	C	N
A	7	B
B	1	B

Time

Poisoned Reverse

- If C routes through B to get to A
  - C tells B that D(C, A) = ∞
  - Thus, B won’t route to A via C



Node B

D	C	N
A	4	A
C	1	B

D	C	N
A	**60**	A
C	1	B

D	C	N
A	60	A
C	1	B

D	C	N
A	**51**	**C**
C	1	B

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

D	C	N
A	**50**	**A**
B	1	B

D	C	N
A	50	A
B	1	B

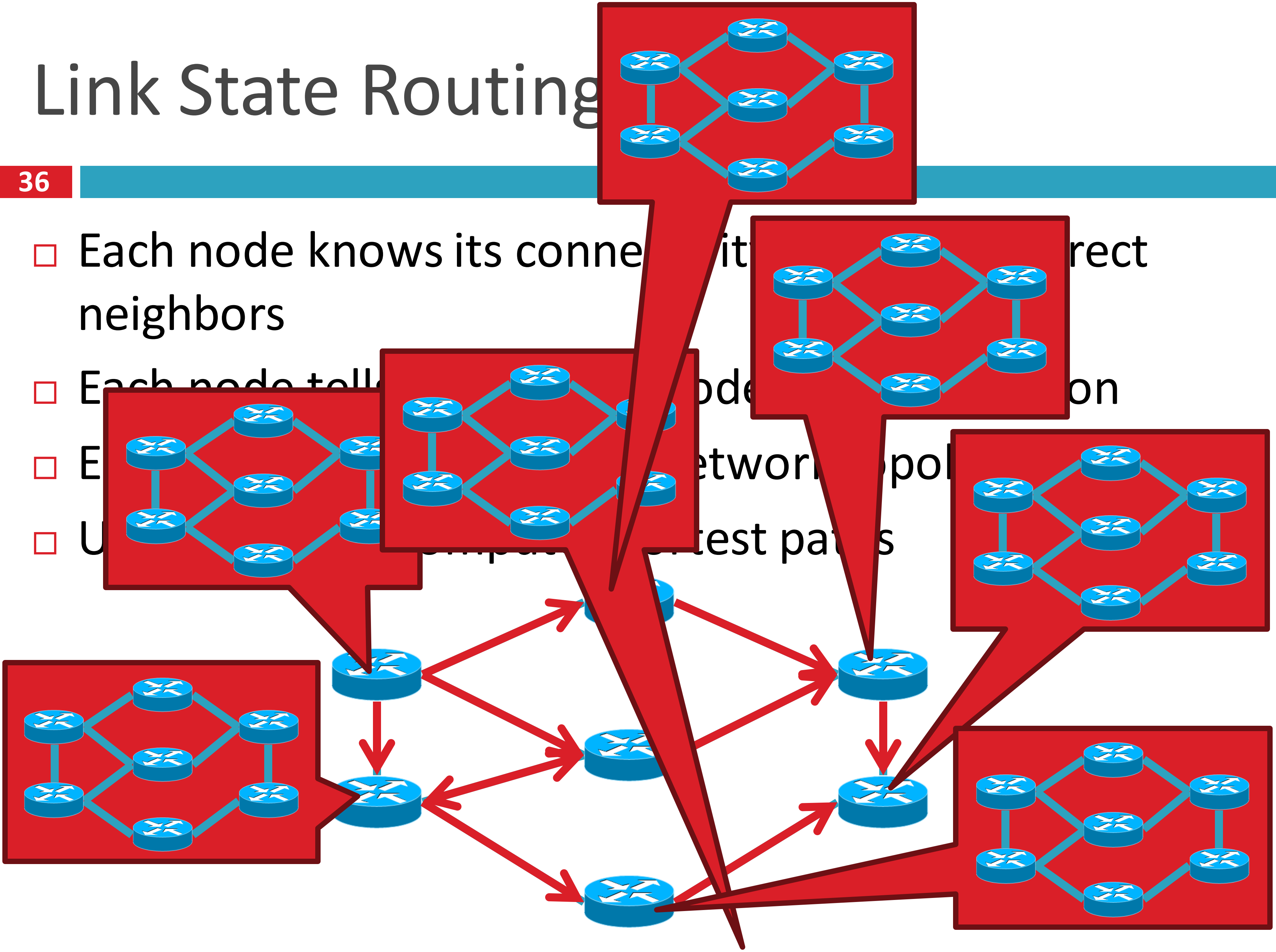
Time

Outline

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

Link State Routing

- □ Each node knows its conne it rect neighbors
- □ Each node tell ode on
- □ E etwor opol
- □ U omput test pat s



Flooding Details

- 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

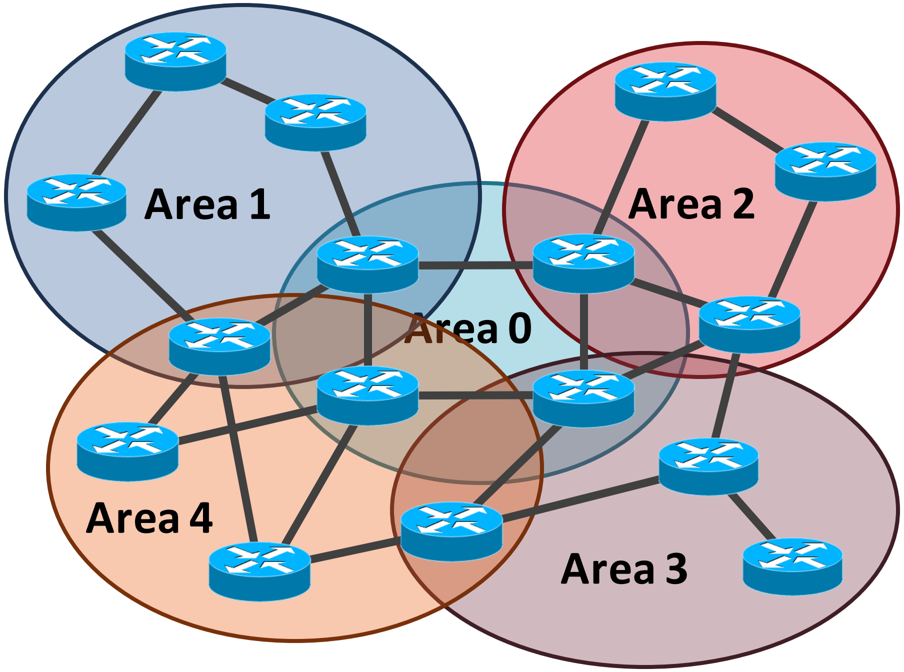
- Two different implementations of link-state routing

OSPF	IS-IS
□ Favored by companies, datacenters	□ Favored by ISPs
□ More optional features	□ Less “chatty” Less network overhead Supports more devices
□ Built on top of IPv4 LSAs are sent via IPv4 OSPFv3 needed for IPv6	□ Not tied to IP Works with IPv4 or IPv6

Different Organizational Structure

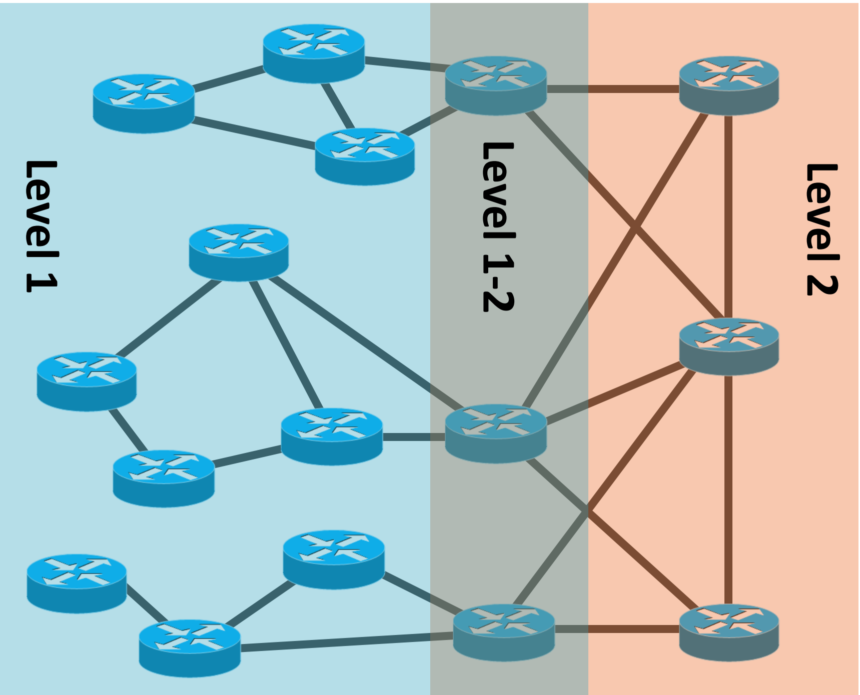
OSPF

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



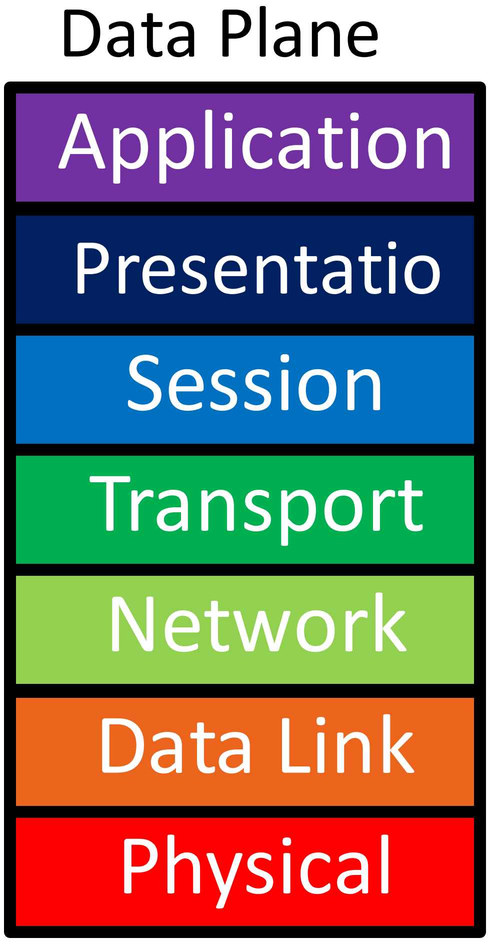
IS-IS

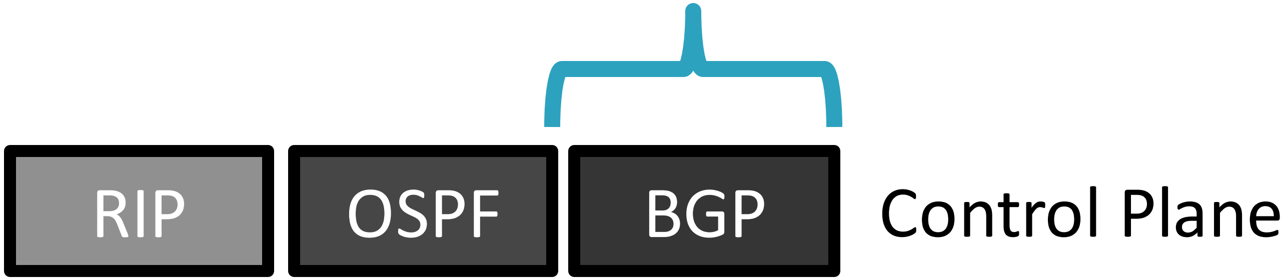
- Organized as a 2-level hierarchy
- Level 2 is the backbone



Network Layer, Control Plane

- □ Function:
  - Set up routes between networks
- □ Key challenges:
  - Implementing provider policies
  - Creating stable paths

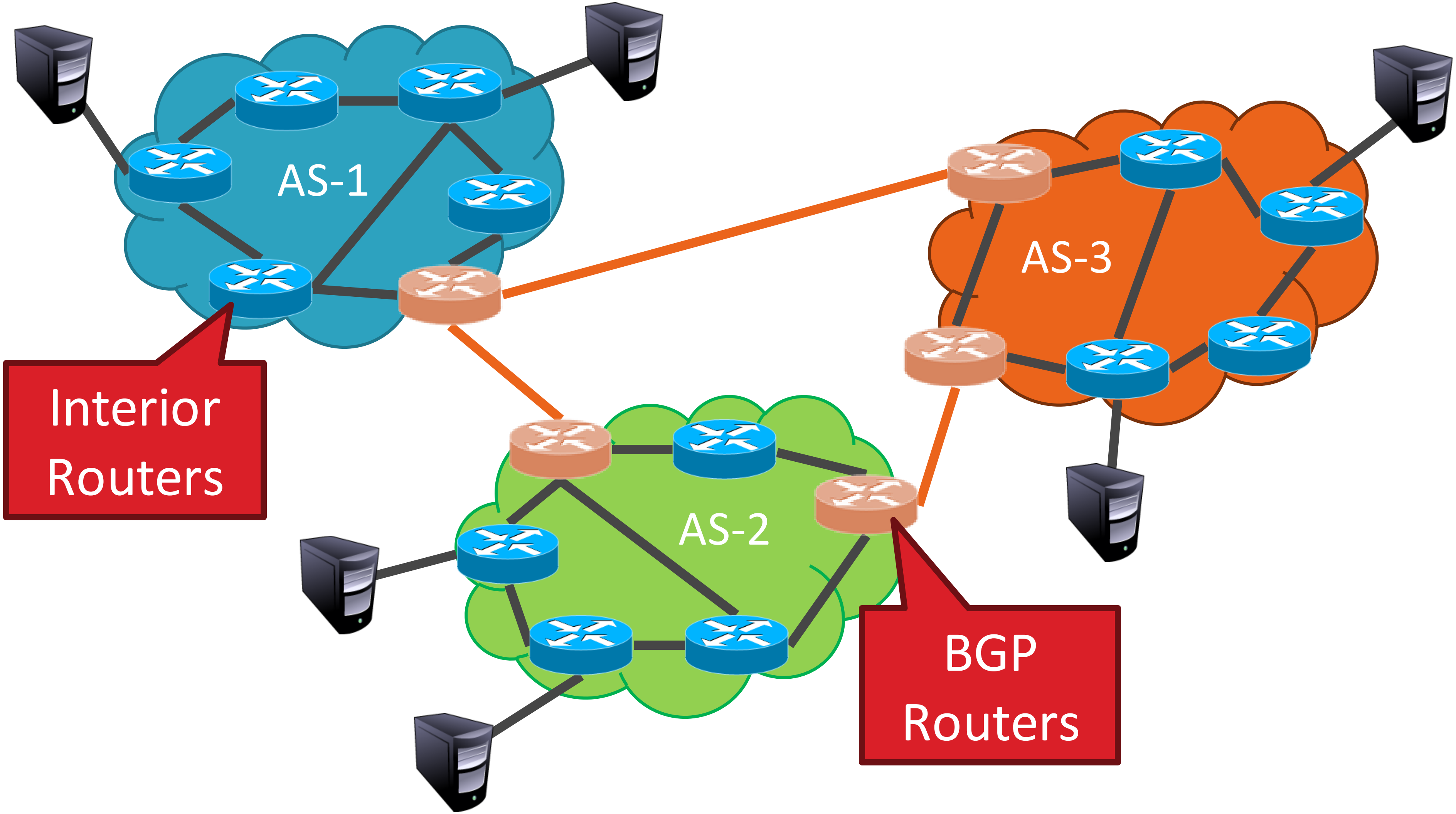




Outline

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

ASs, Revisited



Interior Routers

BGP Routers

AS Numbers

- 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

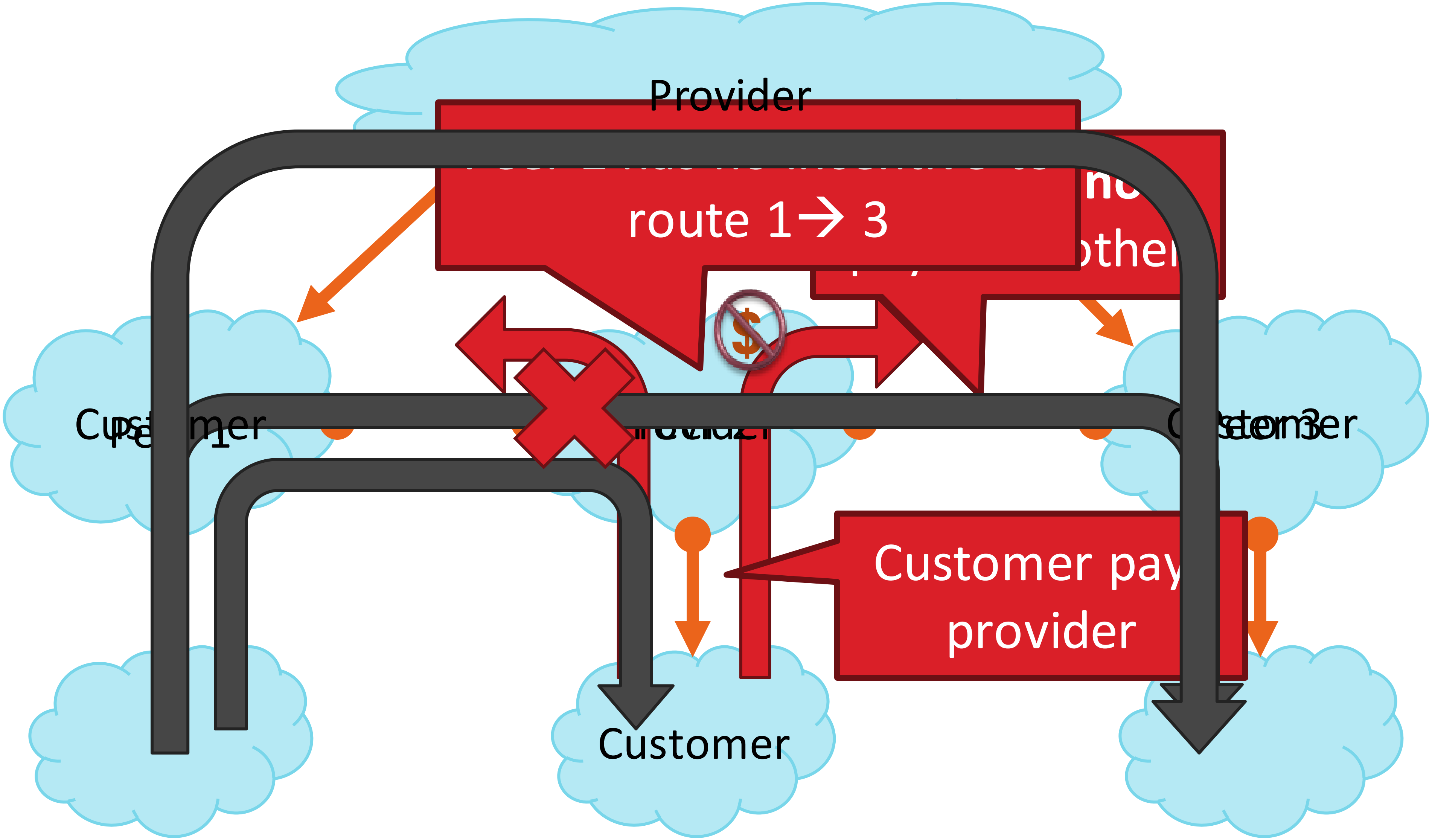
Inter-Domain Routing

- 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

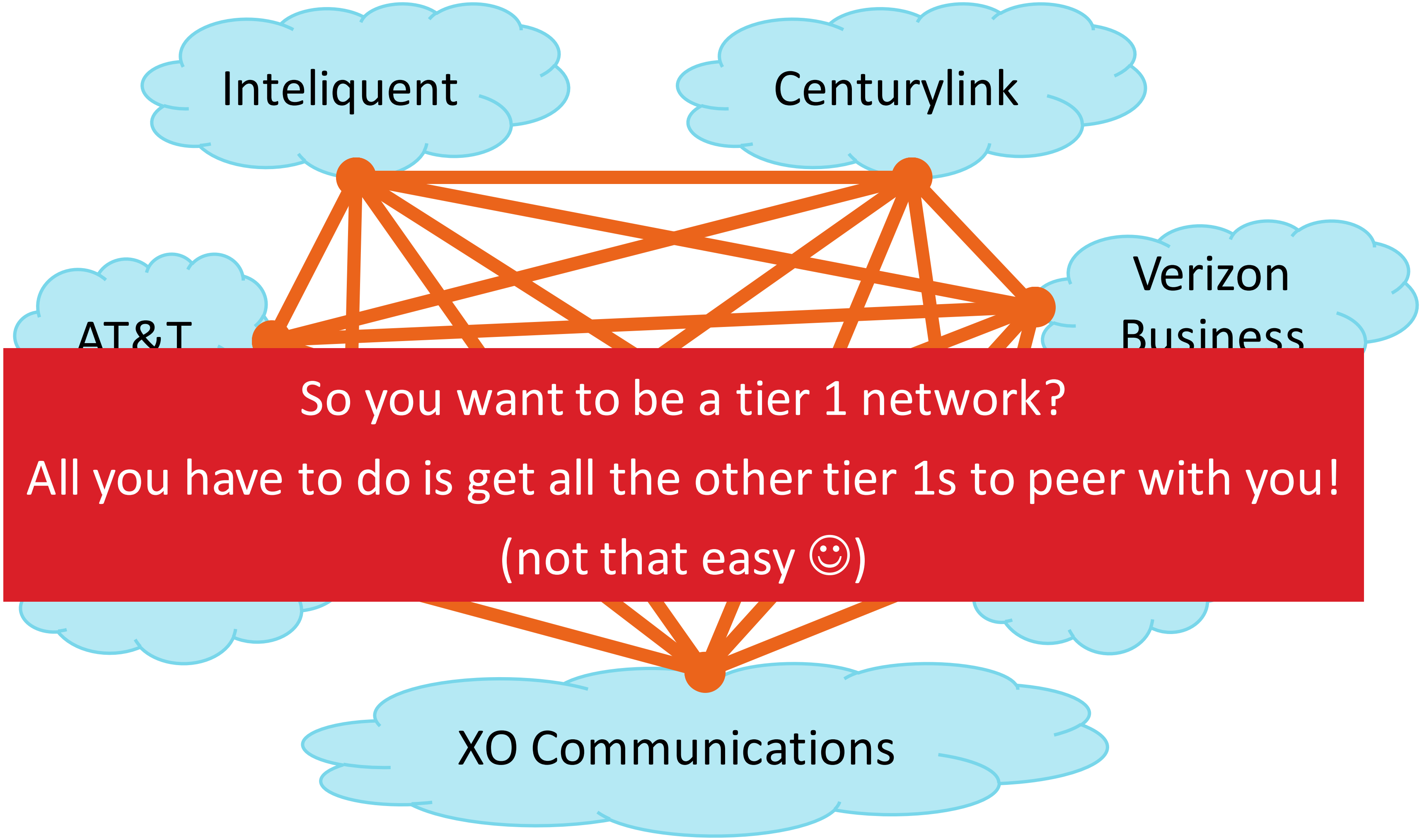
- Border Gateway Protocol
  - De facto inter-domain protocol of the Internet
  - Policy based routing protocol
  - Uses a Bellman-Ford path vector protocol
- Relatively simple protocol, but…
  - Complex, manual configuration
  - Entire world sees advertisements
    - Errors can screw up traffic globally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

BGP Relationships



Customer pay
provider

Tier-1 ISP Peering

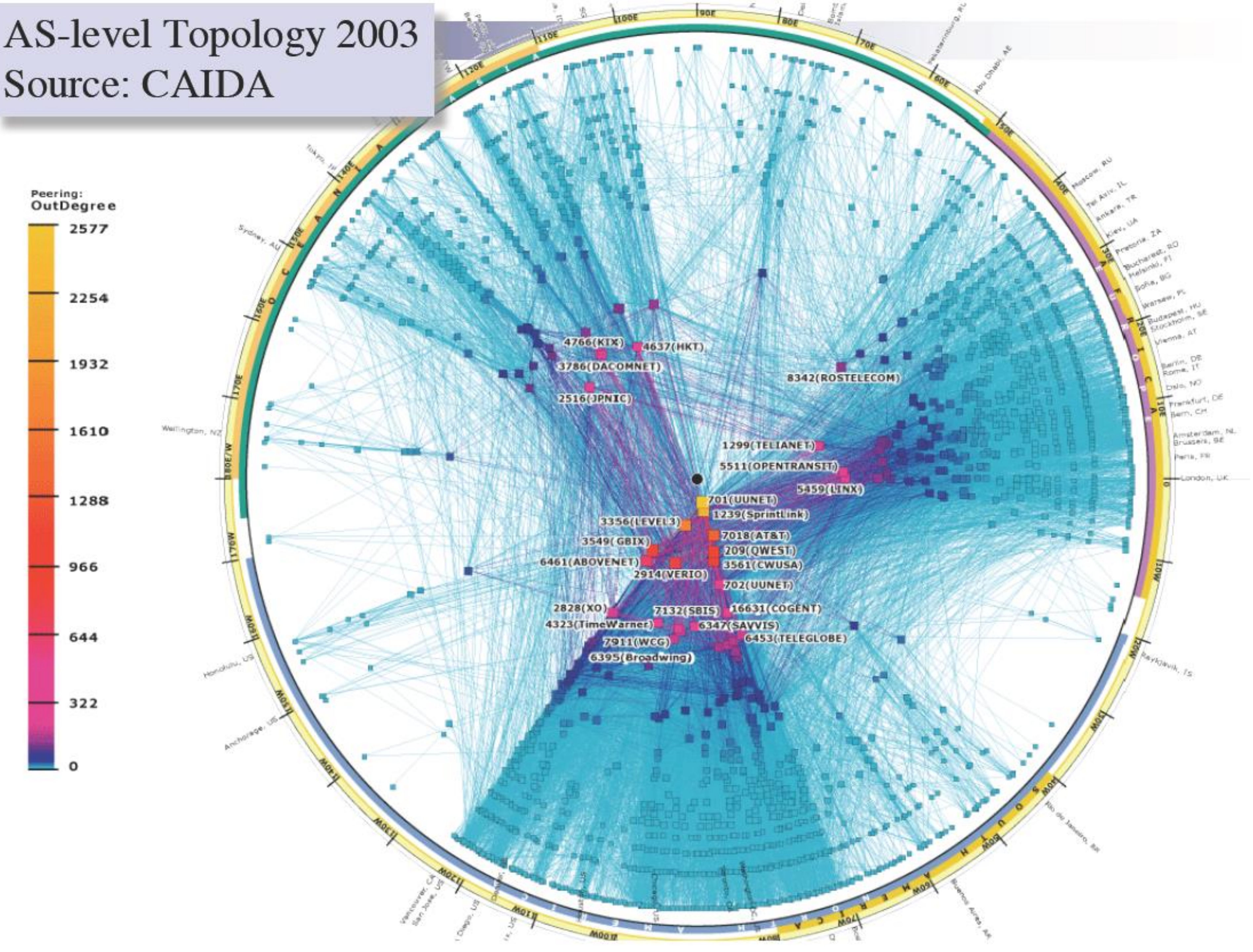


So you want to be a tier 1 network?

All you have to do is get all the other tier 1s to peer with you!

(not that easy ☺)

AS-level Topology 2003
Source: CAIDA



Peering Wars

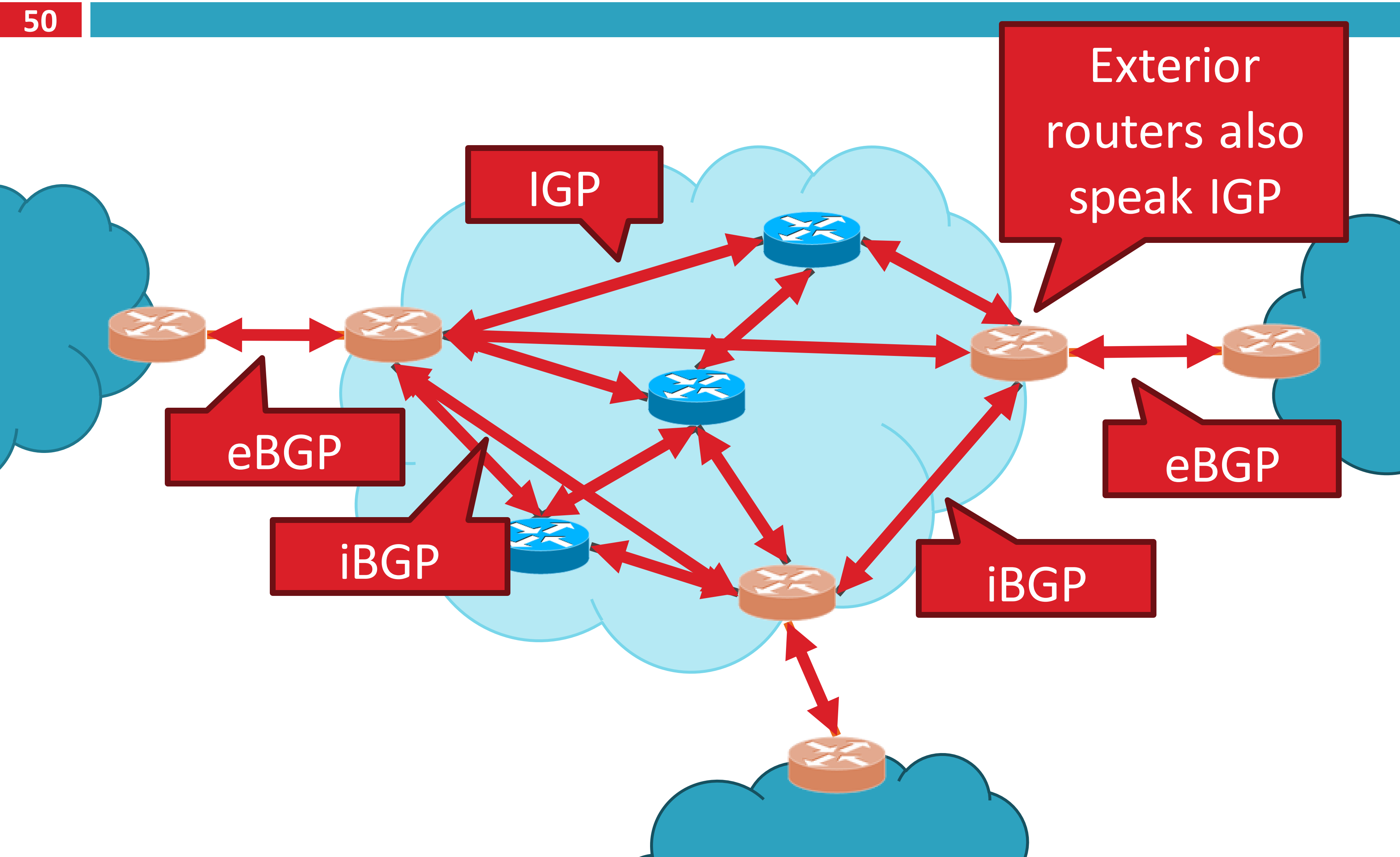
Peer	Don’t Peer
□ Reduce upstream costs	□ You would rather have
□ I p	
□ M c I	

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

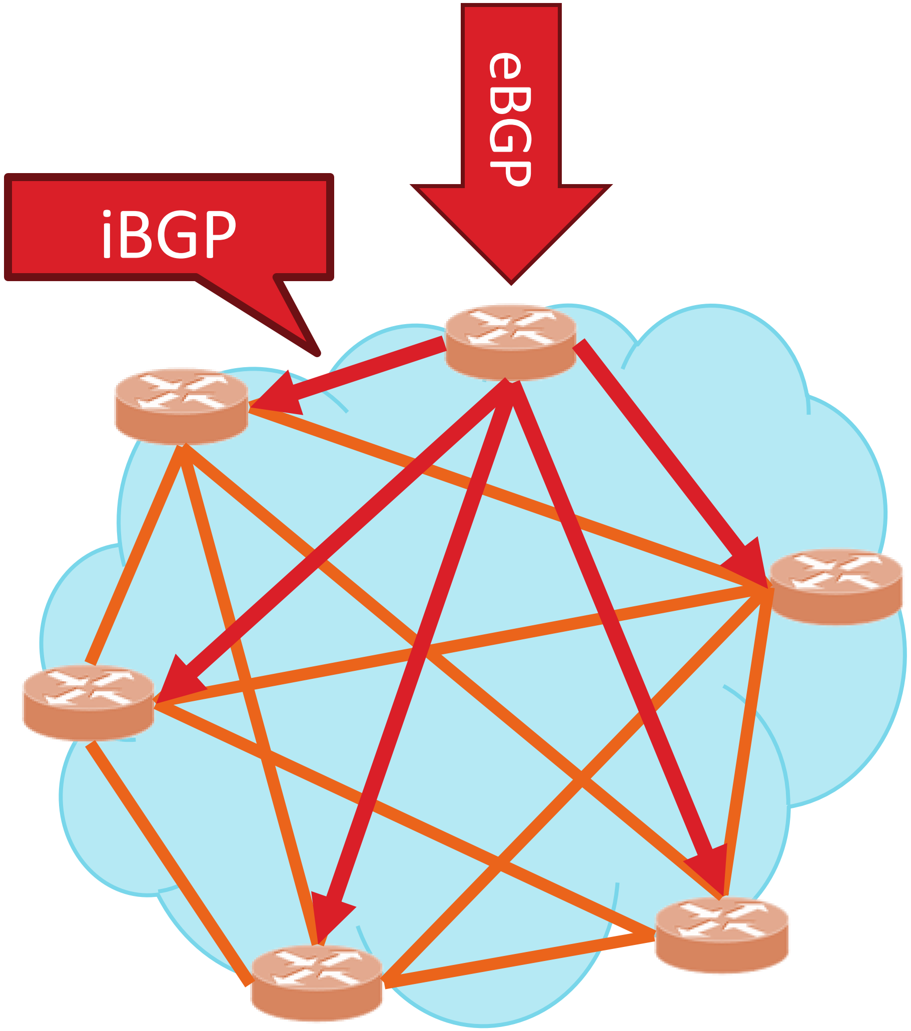
Example: If you are a customer of my peer why should I peer with you? You should pay me too!

Incentive to keep relationships private!

Two Types of BGP Neighbors



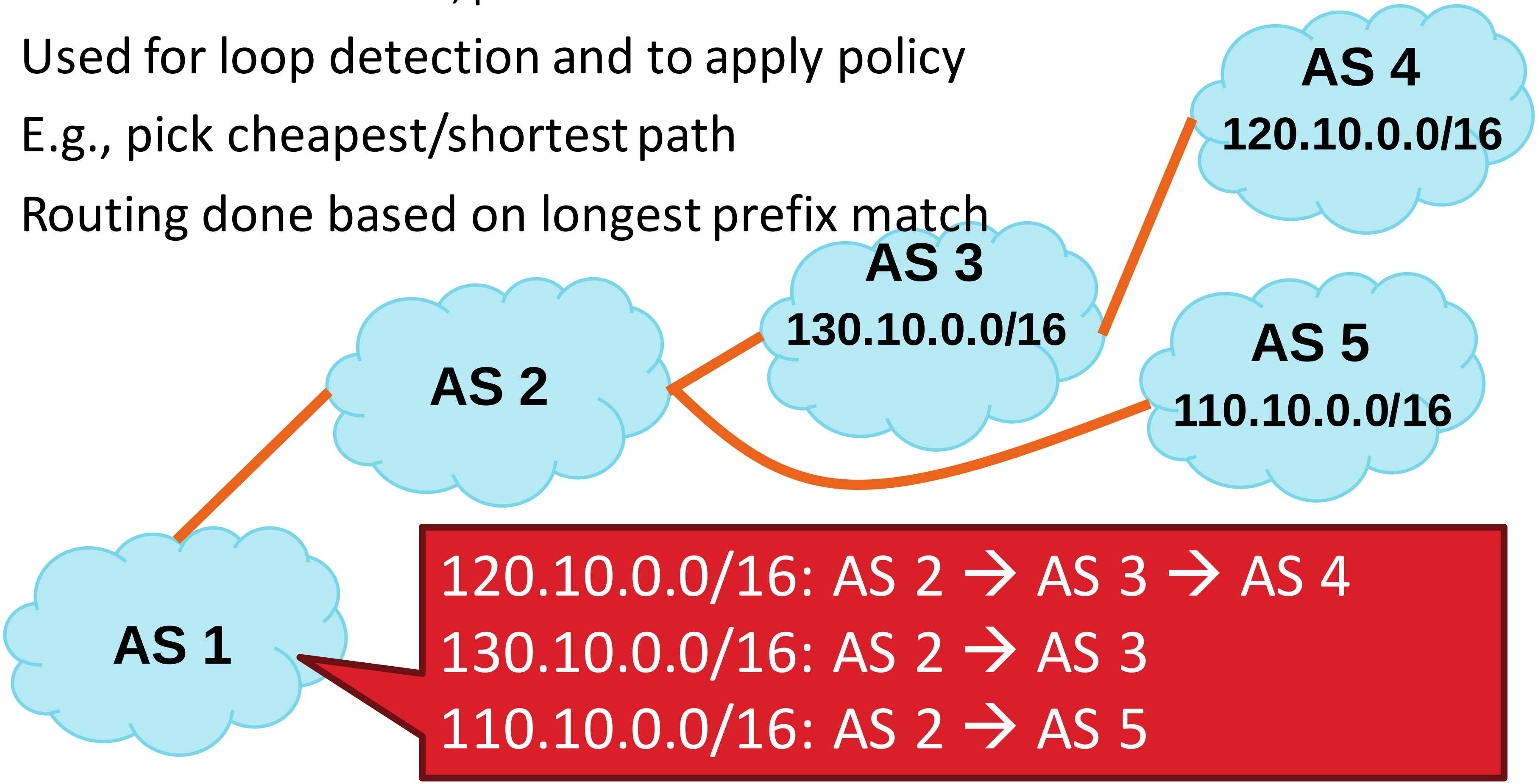
Full iBGP Meshes



- □ 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

- 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



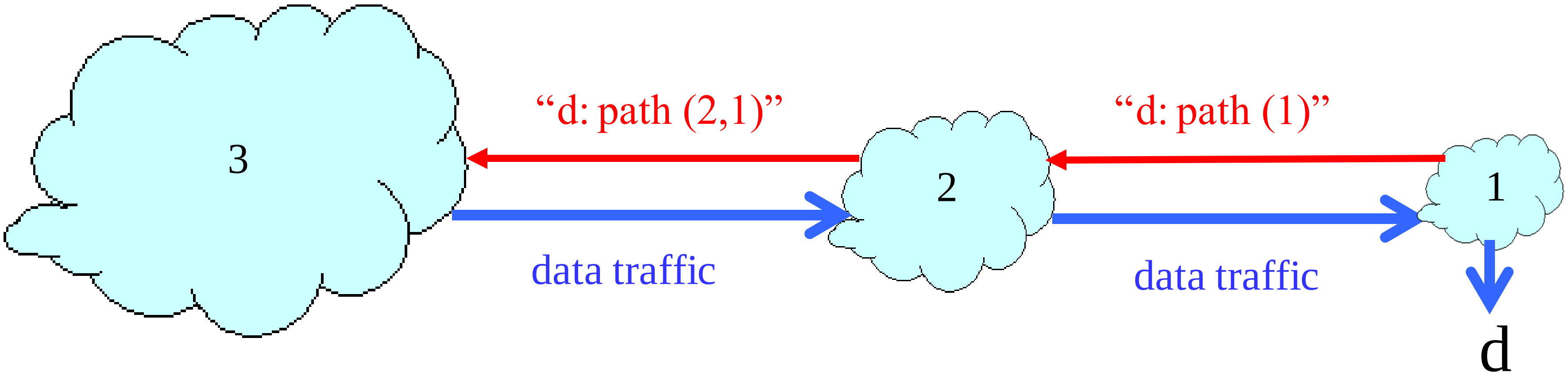
120.10.0.0/16: AS 2 → AS 3 → AS 4

130.10.0.0/16: AS 2 → AS 3

110.10.0.0/16: AS 2 → AS 5

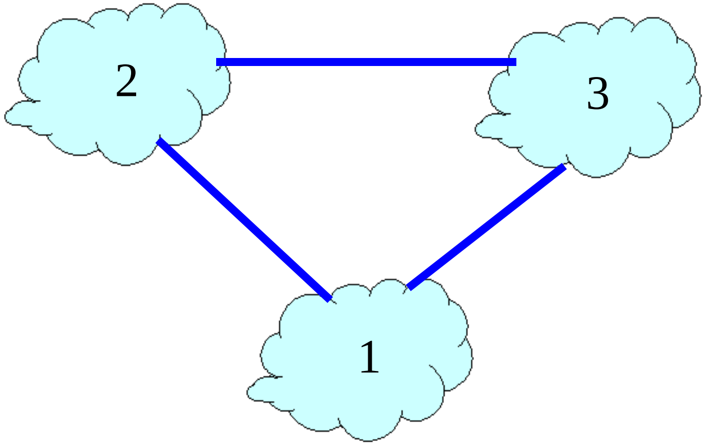
Path-Vector Routing

- 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

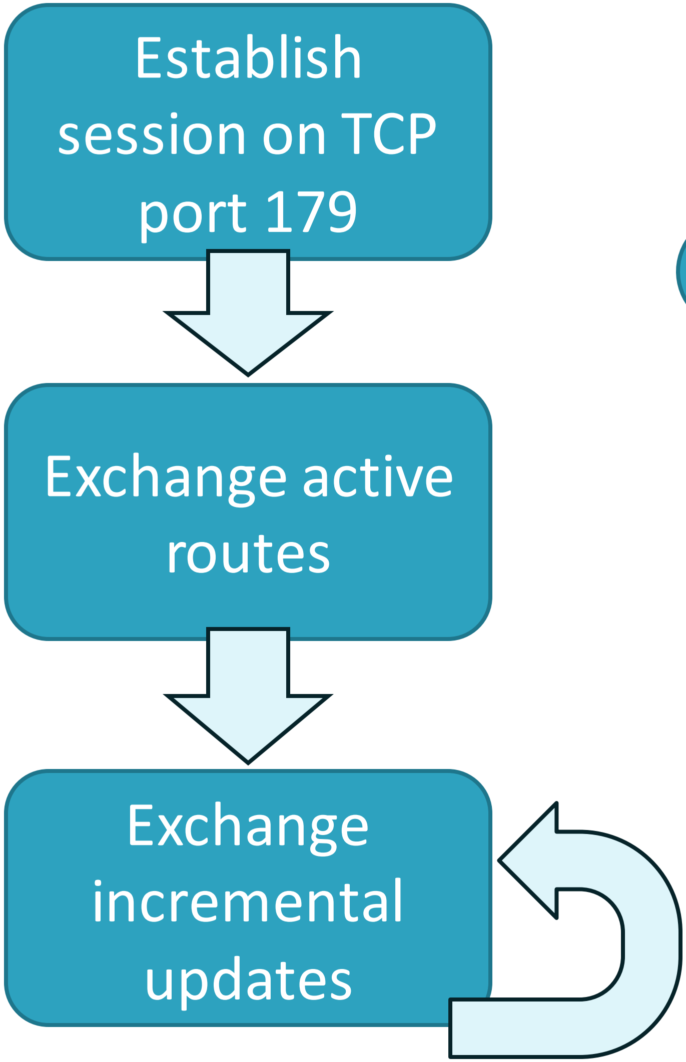


Flexible Policies

- 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”



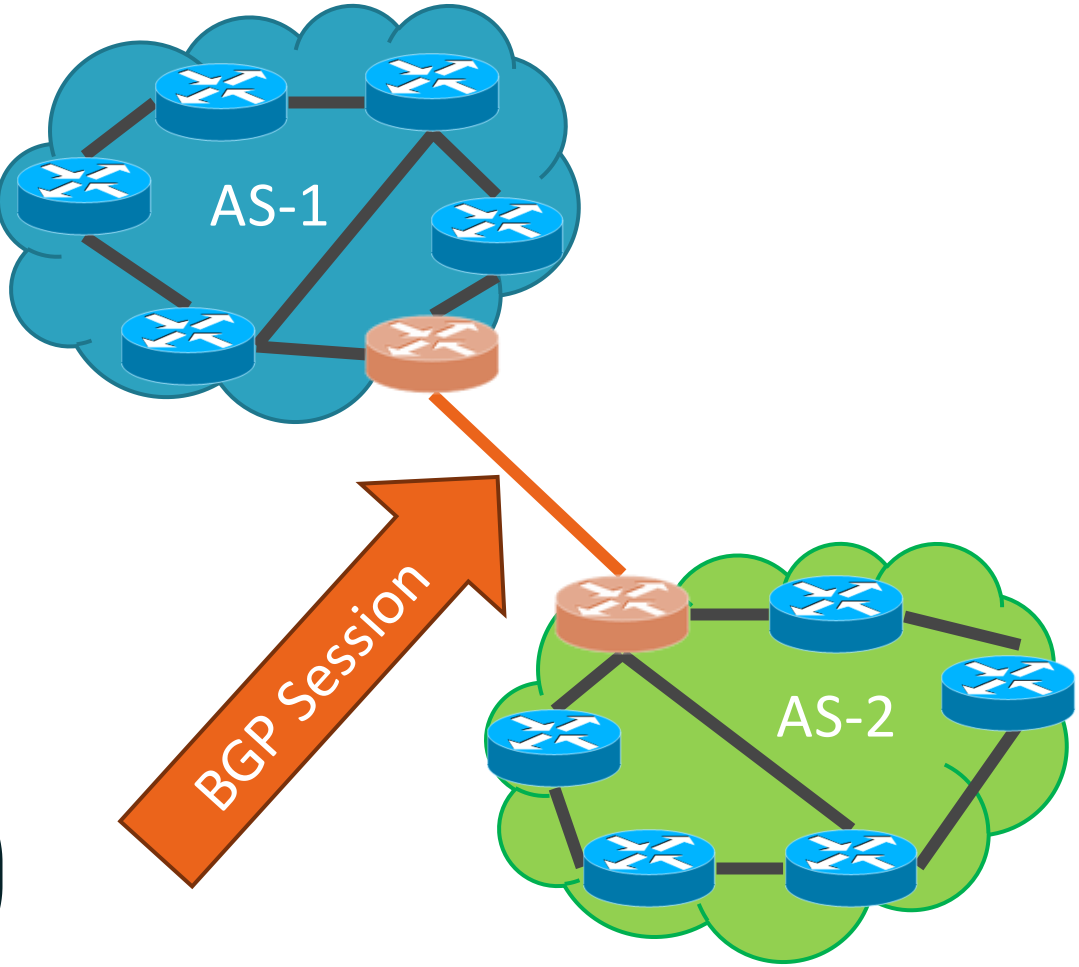
BGP Operations (Simplified)



Establish session on TCP port 179

Exchange active routes

Exchange incremental updates



Four Types of BGP Messages

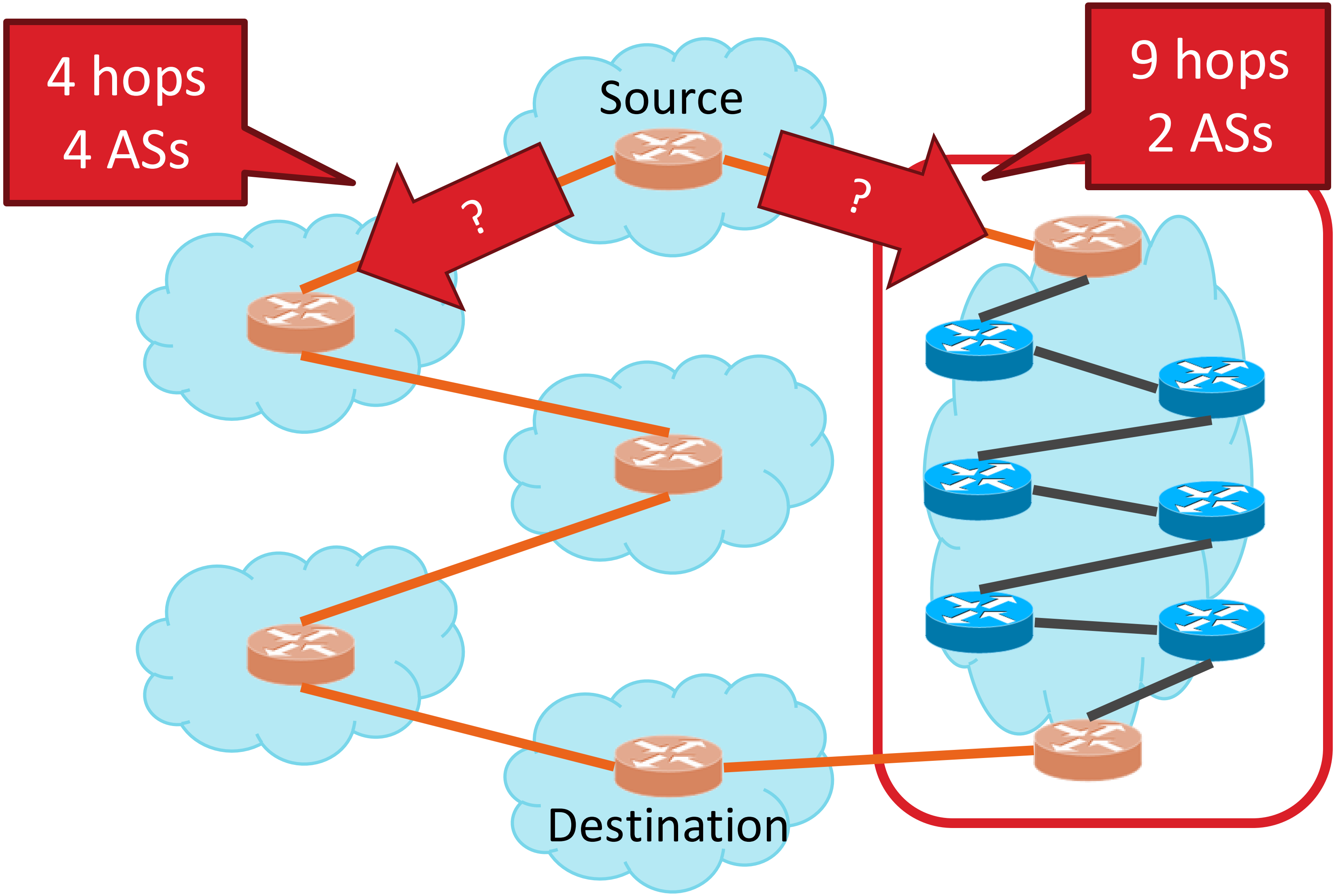
- Open: Establish a peering session.
- Keep Alive: Handshake at regular intervals.
- Notification: Shuts down a peering session.
- Update: Announce new routes or withdraw previously announced routes.

announcement = IP prefix + attributes values

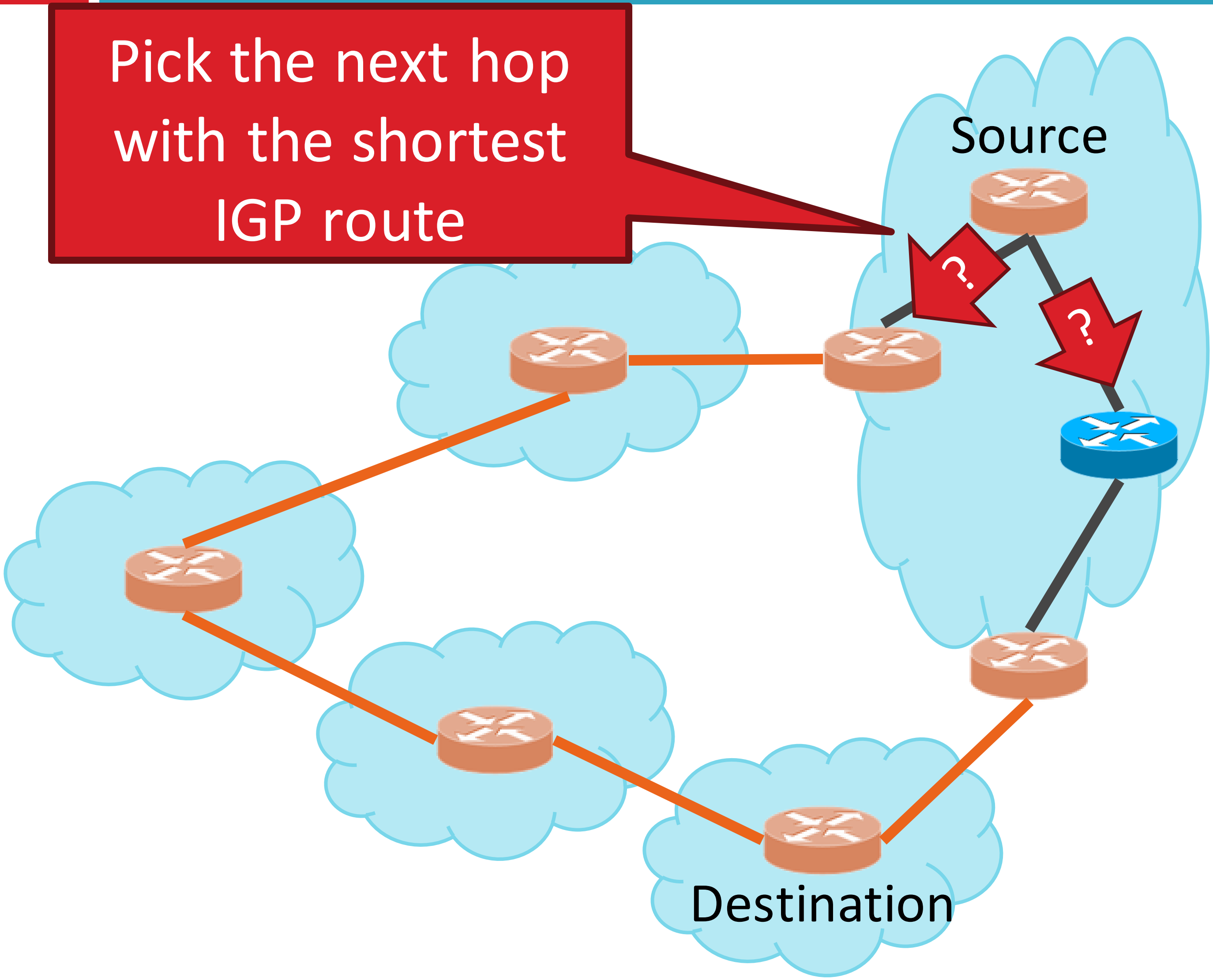
BGP Attributes

- 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 != Shortest Path

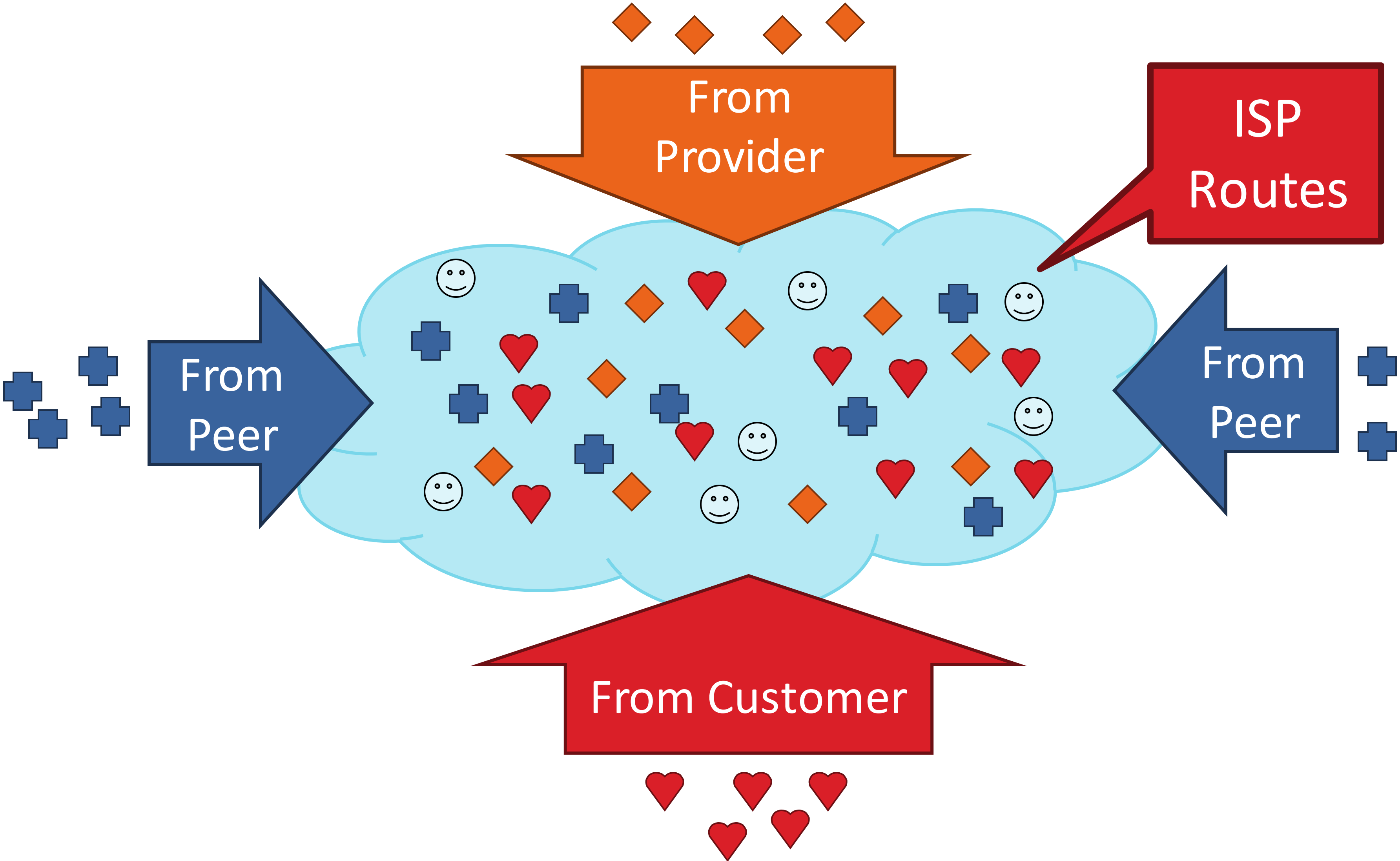


Hot Potato Routing

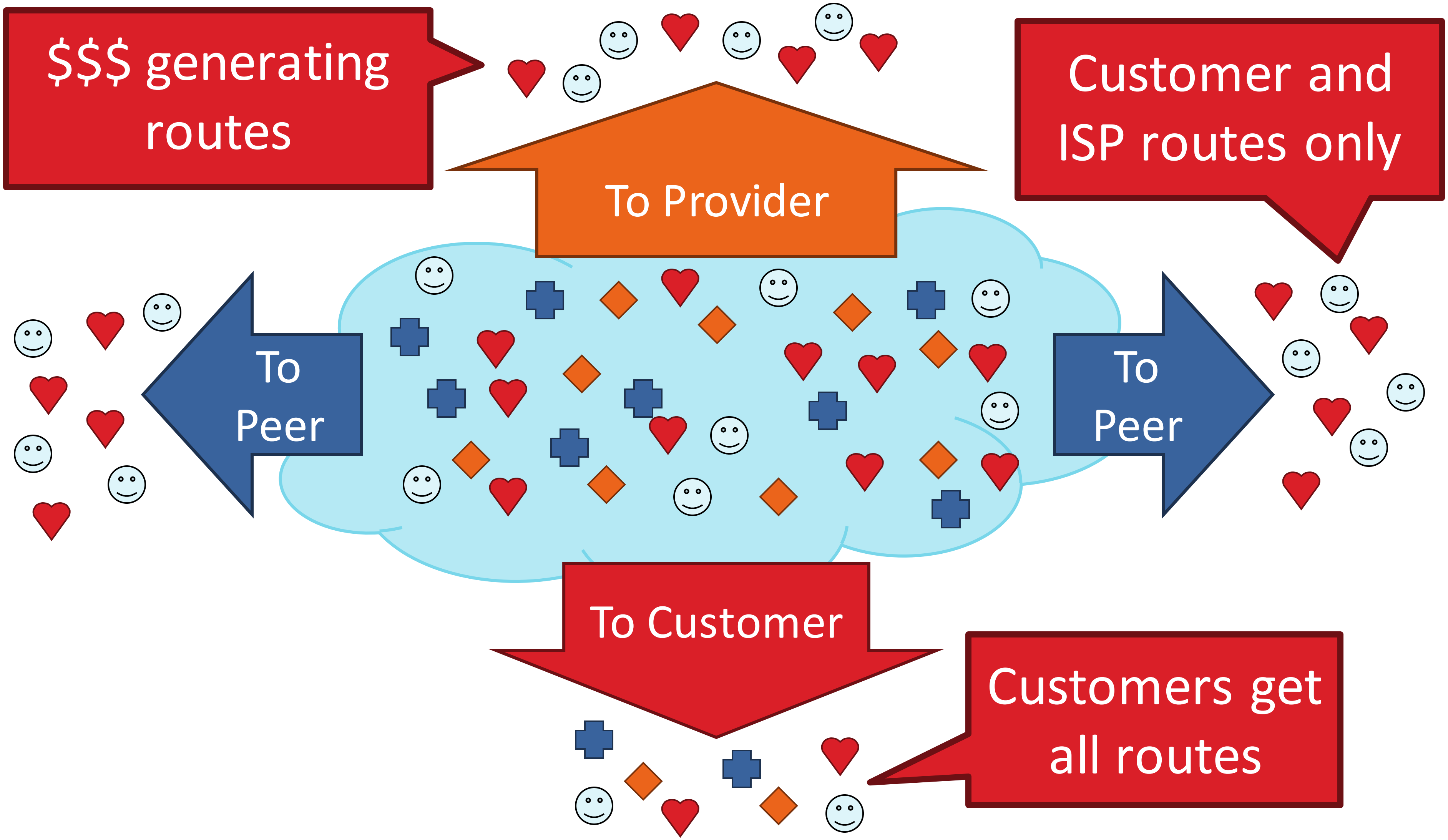


Pick the next hop with the shortest IGP route

Importing Routes



Exporting Routes



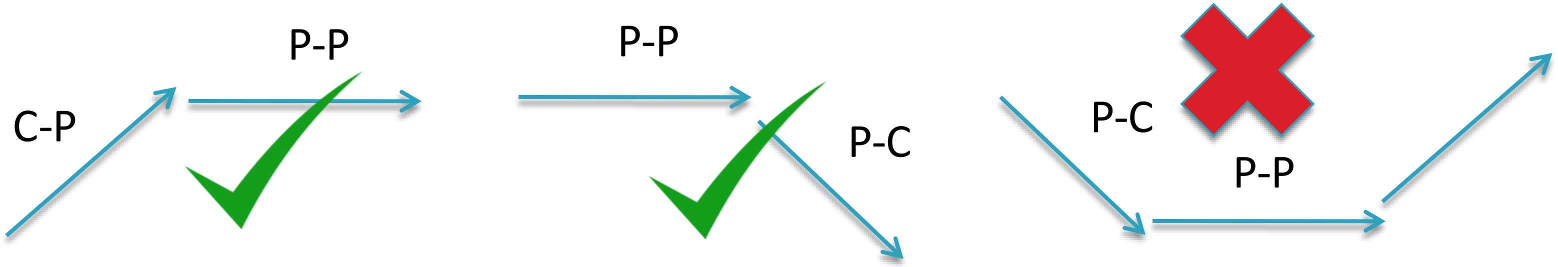
$$$ generating routes

Customer and ISP routes only

Customers get all routes

Modeling BGP

- 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

- 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

- 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?