Computer Networks

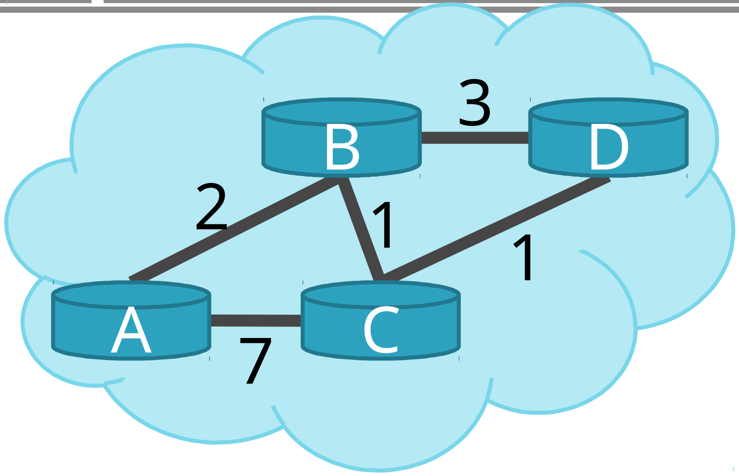
Lecture 8: Network layer Part III
Inter Domain Routing
(It's all about the Money)

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

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

```
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) = ∞;
…
```

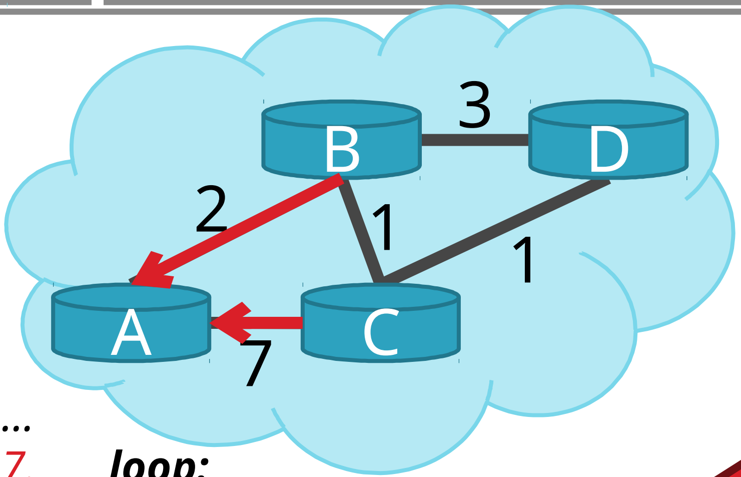
Node C

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



```
…
7.    loop:
…
12.   else if (update D(V, Y) re
13.     for all des
14.       if (destin
15.         D(A,Y) =
16.       else
17.         D(A, Y) =
                m
                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
```

Node A

Dest.	Cost	Next
B	2	B
C	3	B
[illegible]	5	B

Node B

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

[illegible]	3	B
B	1	B
D	1	D

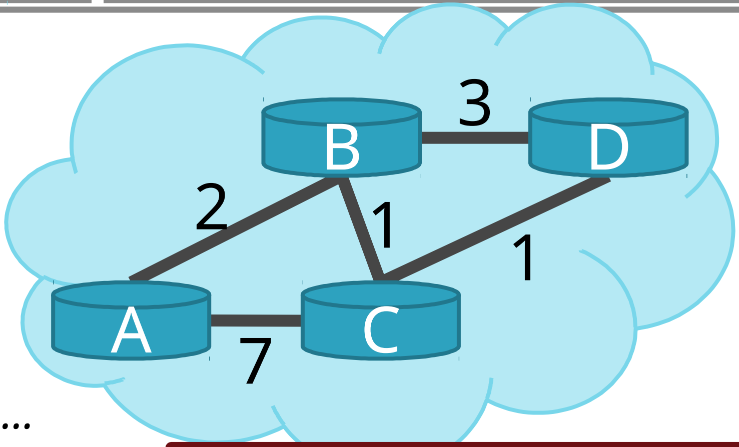
e D

		Next
[illegible]	4	B
B	3	B
C	1	C

D(A,C) = … (A,C), D(A,B)+D(B,C))

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

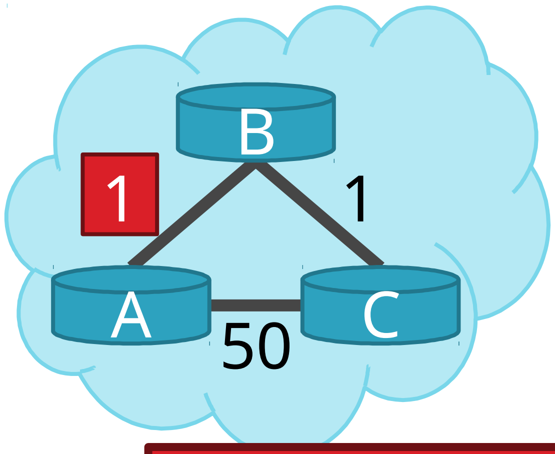
```
…
7.   loo
…
12.  els
13.    fo
14.
15.
16.        else
17.          D(A, Y) =
                  min(D(A, Y),
                  D(A, V) + D(V, Y));
18.  if (there is a new min. for dest. Y)
19.    send D(A, Y) to all neighbors
20.  forever
```

- 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 Cos
Algorit

Good news travels fast

lgorithm
erminates

Node B

D	C	N	D	C	N	D	C	N	D	C	N
A	4	A	A	1	A	A	1	A	A	1	A
C	1	B	C	1	B	C	1	B	C	1	B

Node C

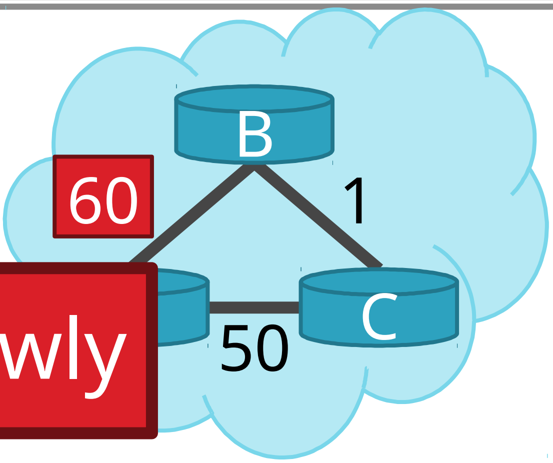
D	C	N	D	C	N	D	C	N	D	C	N
A	5	B	A	5	B	A	2	B	A	2	B
B	1	B	B	1	B	B	1	B	B	1	B

Time

Count to Infinity Problem

- Node B knows D(C, A) = 5
- However, B does not know the path i
- Thus, D(

Bad news travels slowly



Node B

D	C	N
A	4	A
C	1	B

D	C	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

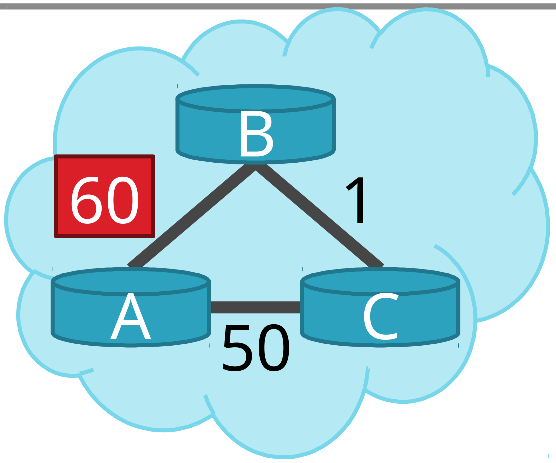
D	C	N
A	7	B
B	1	B

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

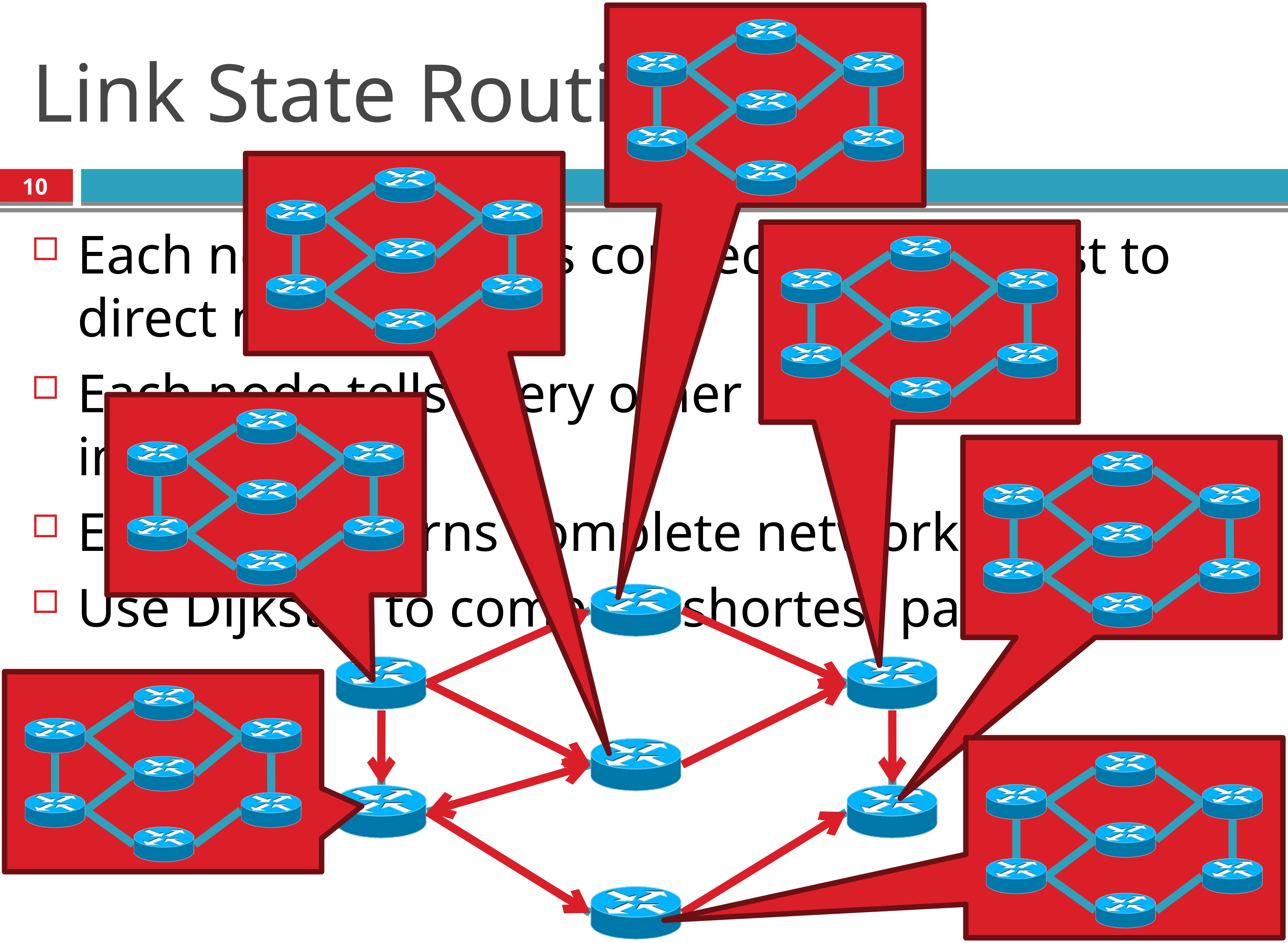
Time

Outline

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

Link State Routi

- Each n s co ec st to direct r
- Each node tells ery o er in
- E rns omplete net ork
- Use Dijks to com shortes pa

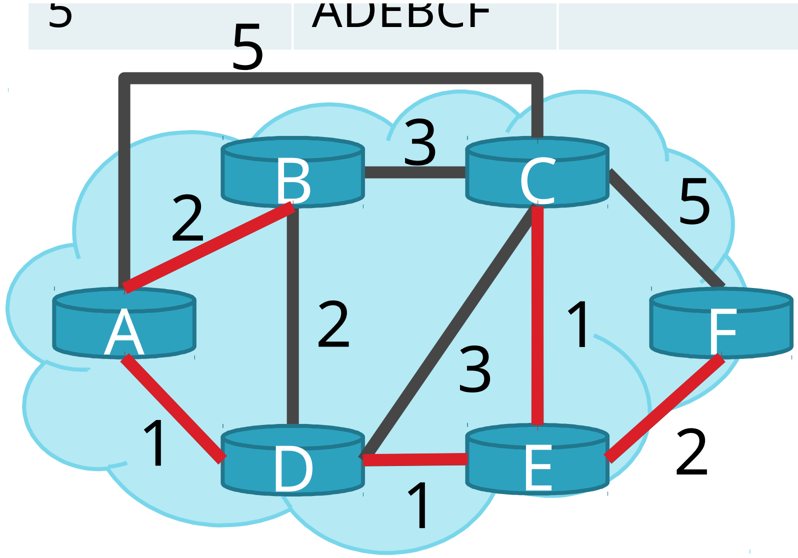


Flooding Details

- 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

Dijkstra’s Algorithm

Step	Start S	□B	□C	□D	□E	□F
0	A	2, A	5, A	1, A	∞	∞
5	ADEBCF					



```
…
8.  Loop
9.    find w not in S s.t. D(w) is a minimum;
10.   add w to S;
11.   update D(v) for all v adjacent to w and not in S:
12.     D(v) = min( D(v), D(w) + c(w,v) );
```

```
1.  Initialization:
2.    S = {A};
3.    for all nodes v
4.      if v adjacent to A
5.        then D(v) = c(A,v);
6.        else D(v) = ∞;
…
```

OSPF vs. IS-IS

- Two different implementations of link-state routing

OSPF

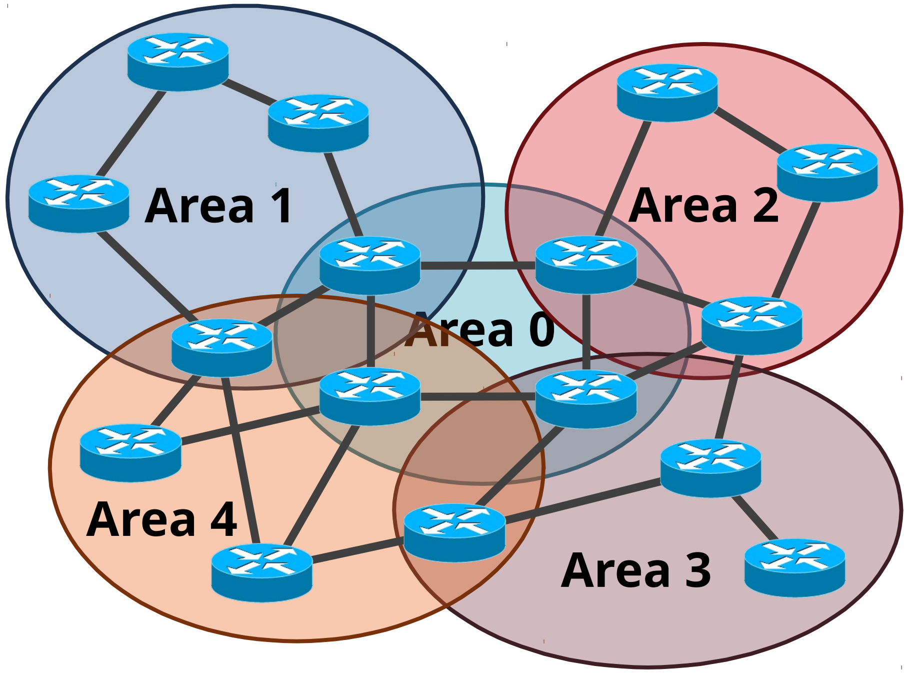
- Favored by companies, datacenters
- More optional features
- Built on top of IPv4
  - LSAs are sent via IPv4
  - OSPFv3 needed for IPv6

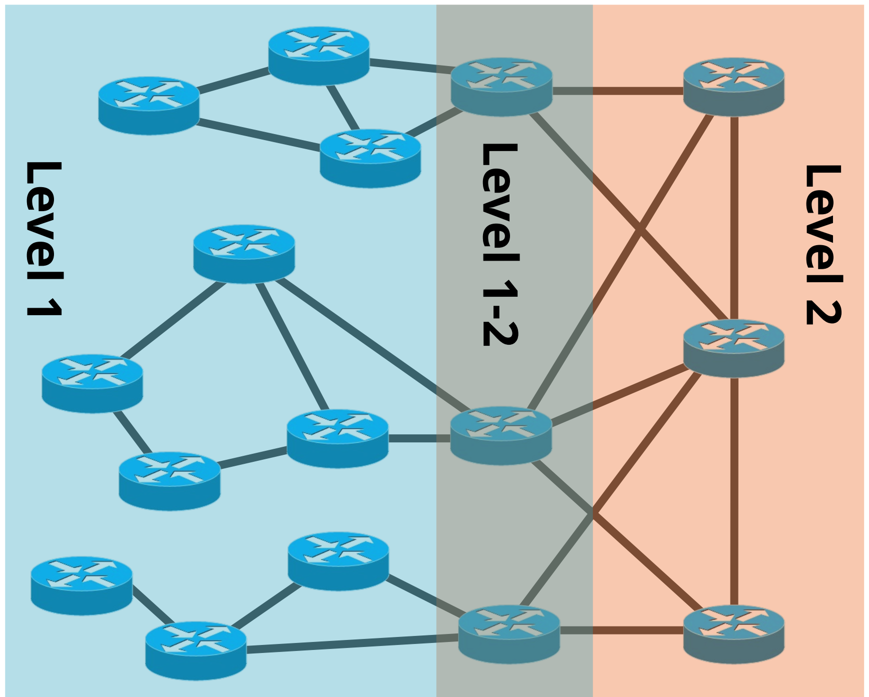
IS-IS

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

Different Organizational Structure

OSPF	IS-IS
Organized around overlapping areas	Organized as a 2-level hierarchy
Area 0 is the core network	Level 2 is the backbone





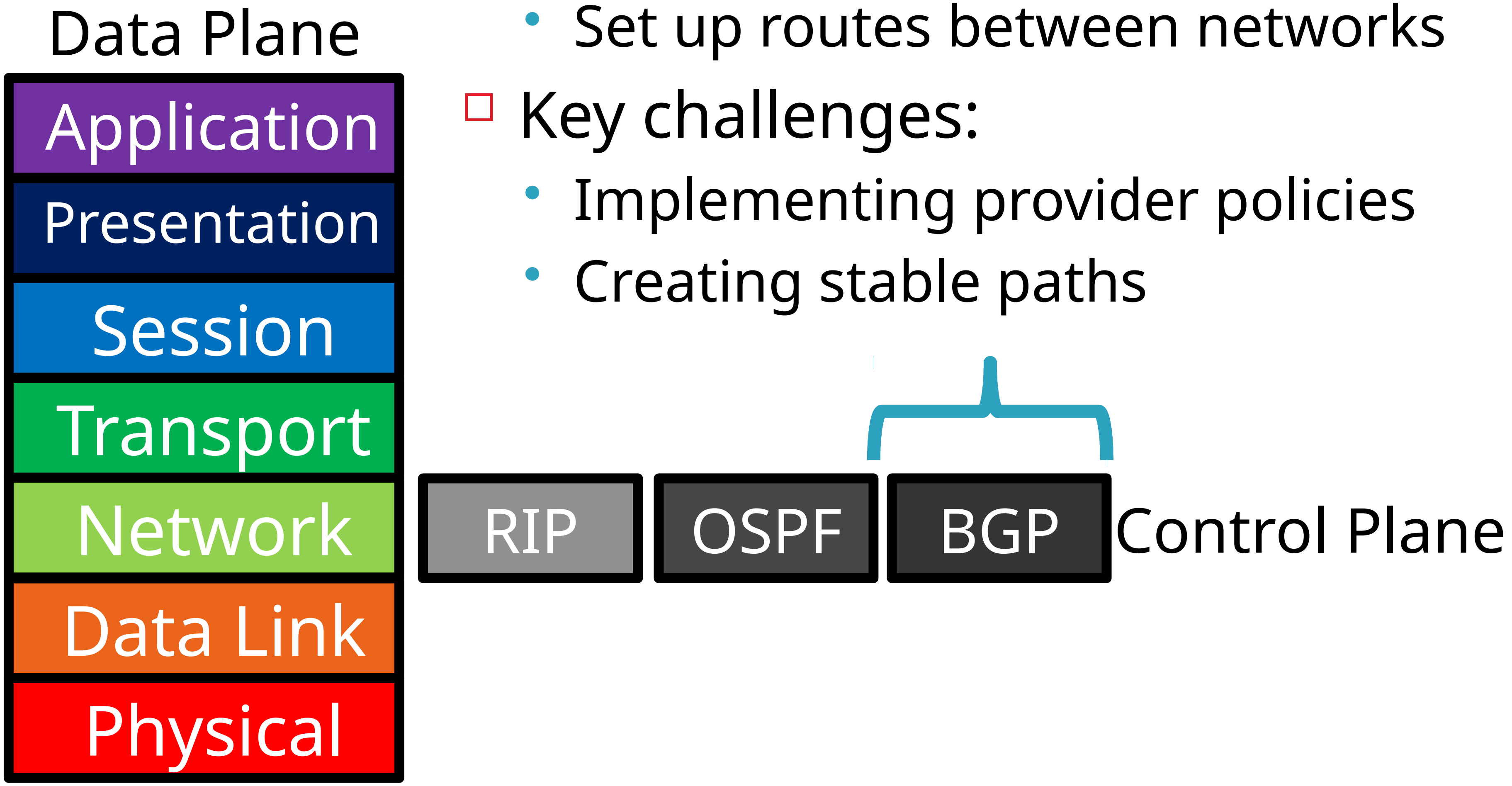
Link State vs. Distance Vector

	Link State	Distance Vector
Message Complexity	$O(n^2*e)$	O(d*n*k)
Time Complexity	O(n*log n)	O(n)
Convergence Time	O(1)	O(k)
Robustness	• Nodes may advertise incorrect link costs • Each node computes	• Nodes may advertise incorrect path cost • Errors propagate due to

- Which is best?
- In practice, it depends.
- In general, link state is more popular.

Network Layer, Control Plane

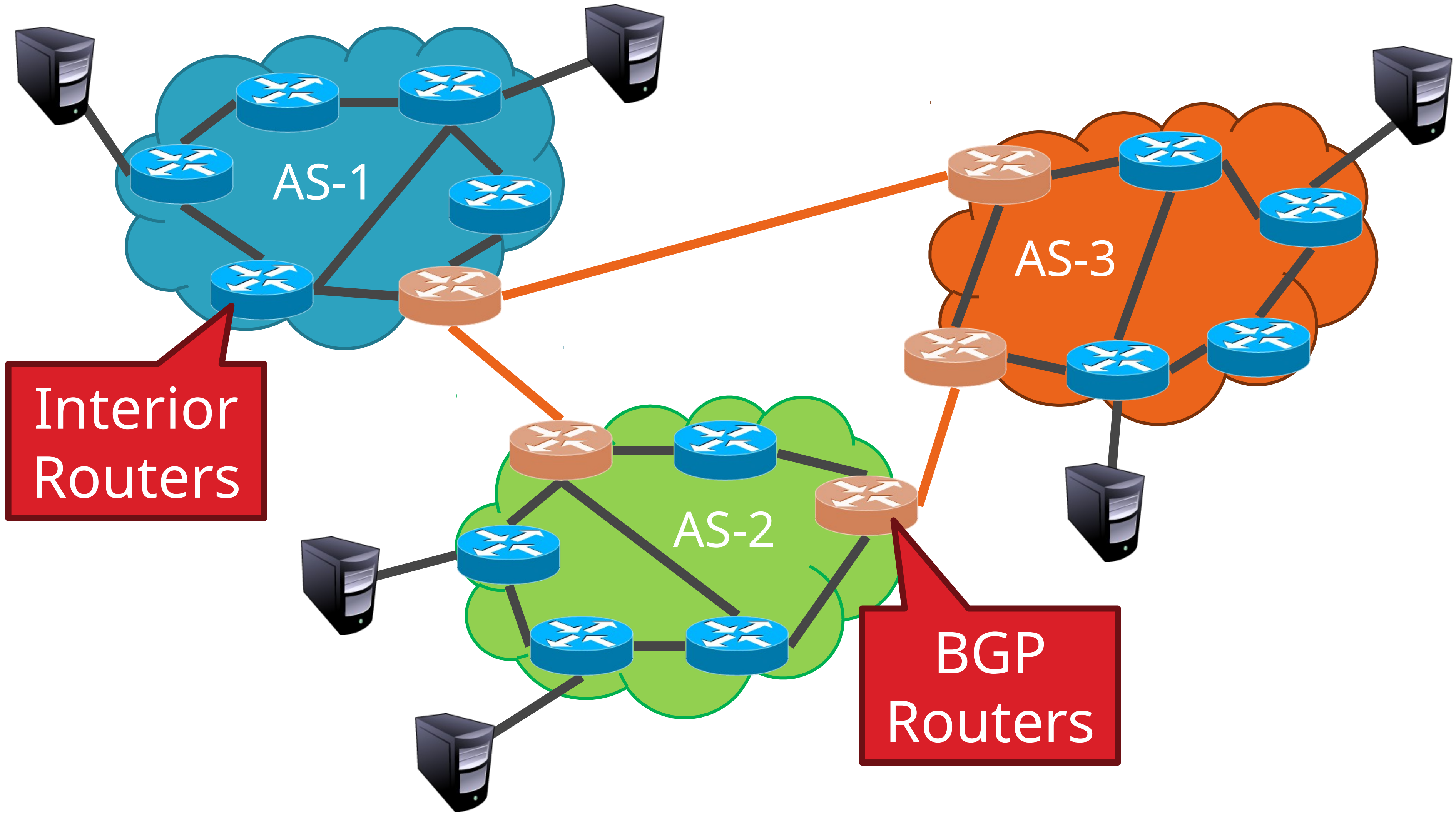
- Function:
  - Set up routes between networks
- Key challenges:
  - Implementing provider policies
  - Creating stable paths



Outline

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

ASs, Revisited



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

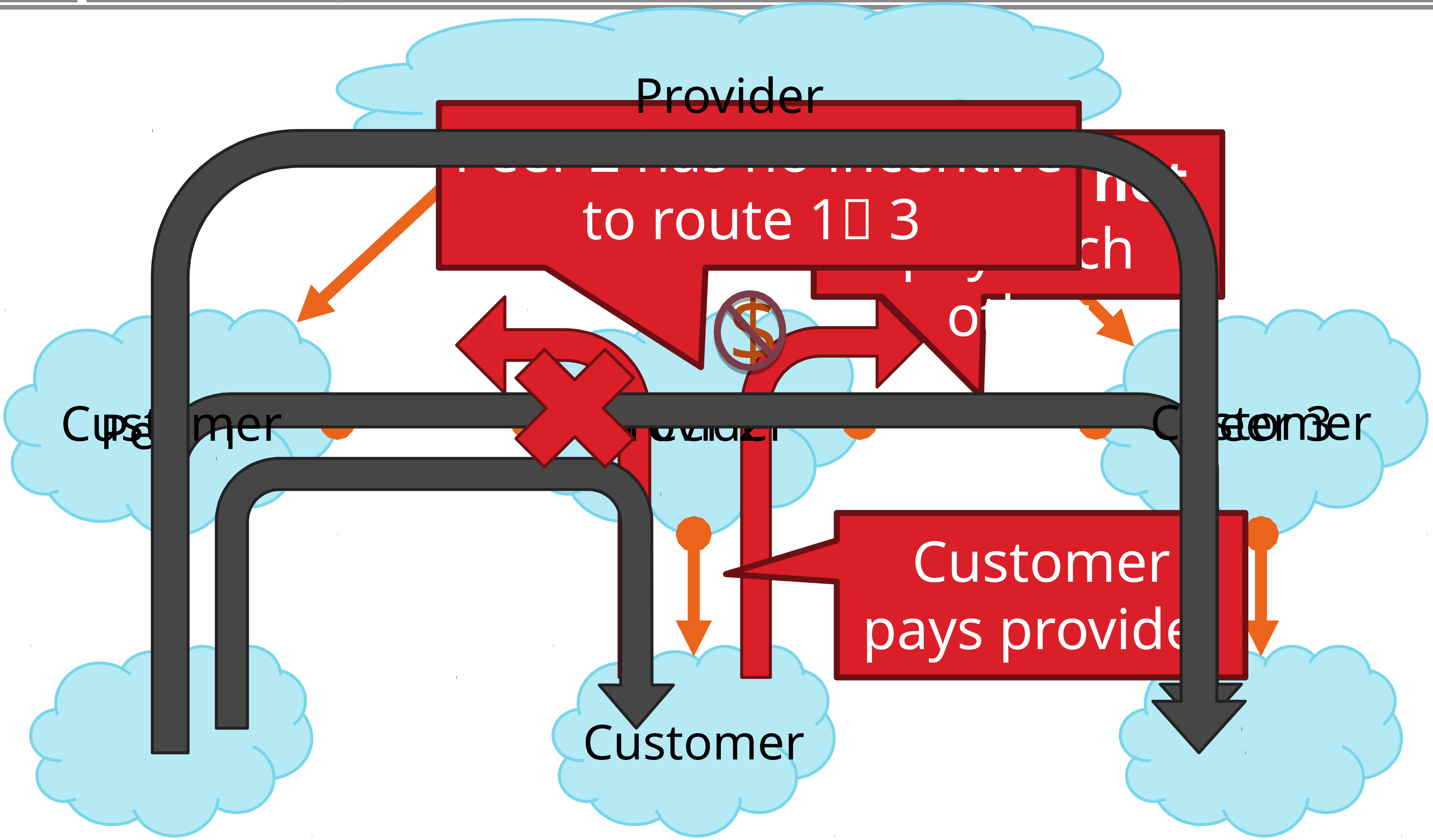
Inter-Domain Routing

- 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

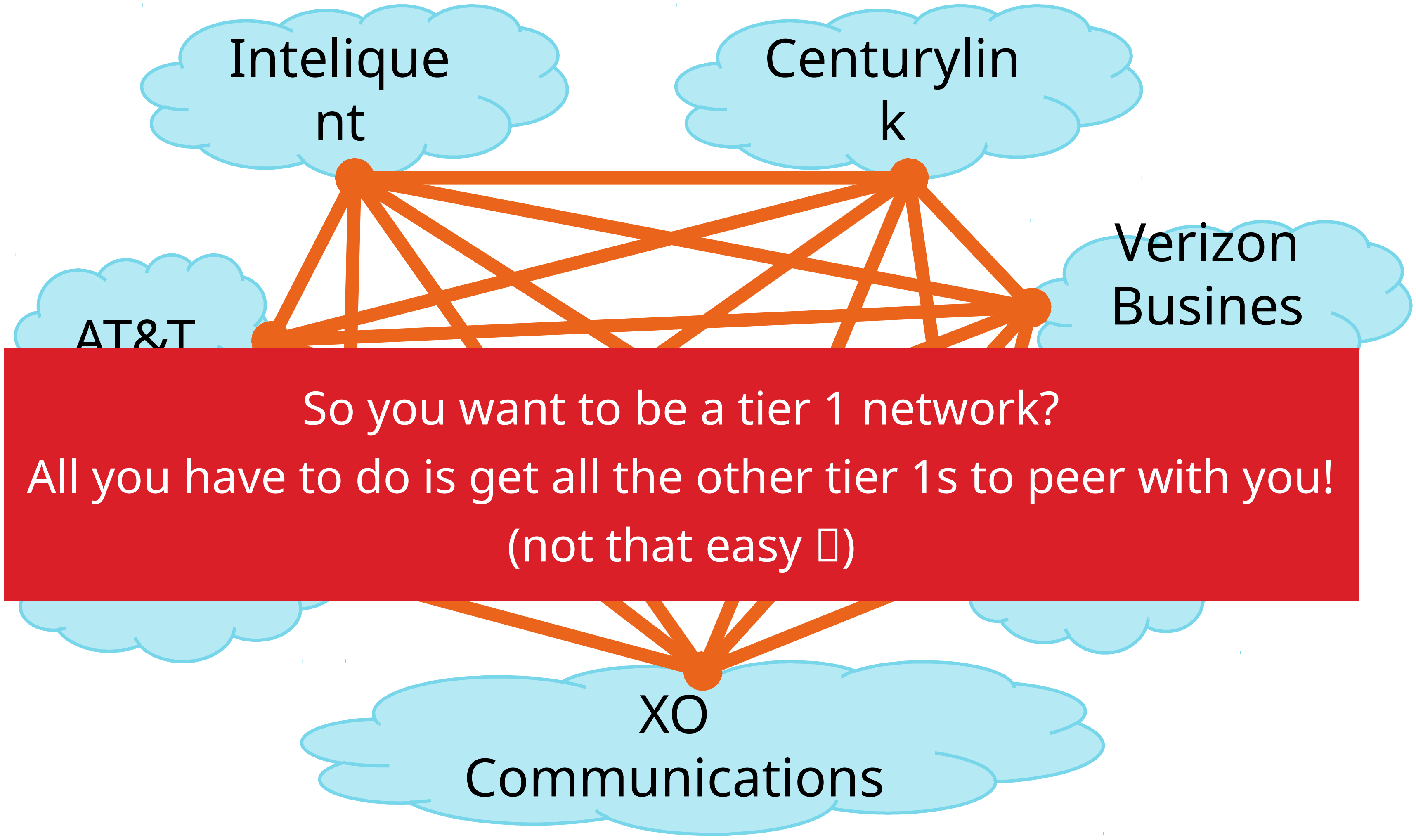
BGP

- 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



Tier-1 ISP Peering

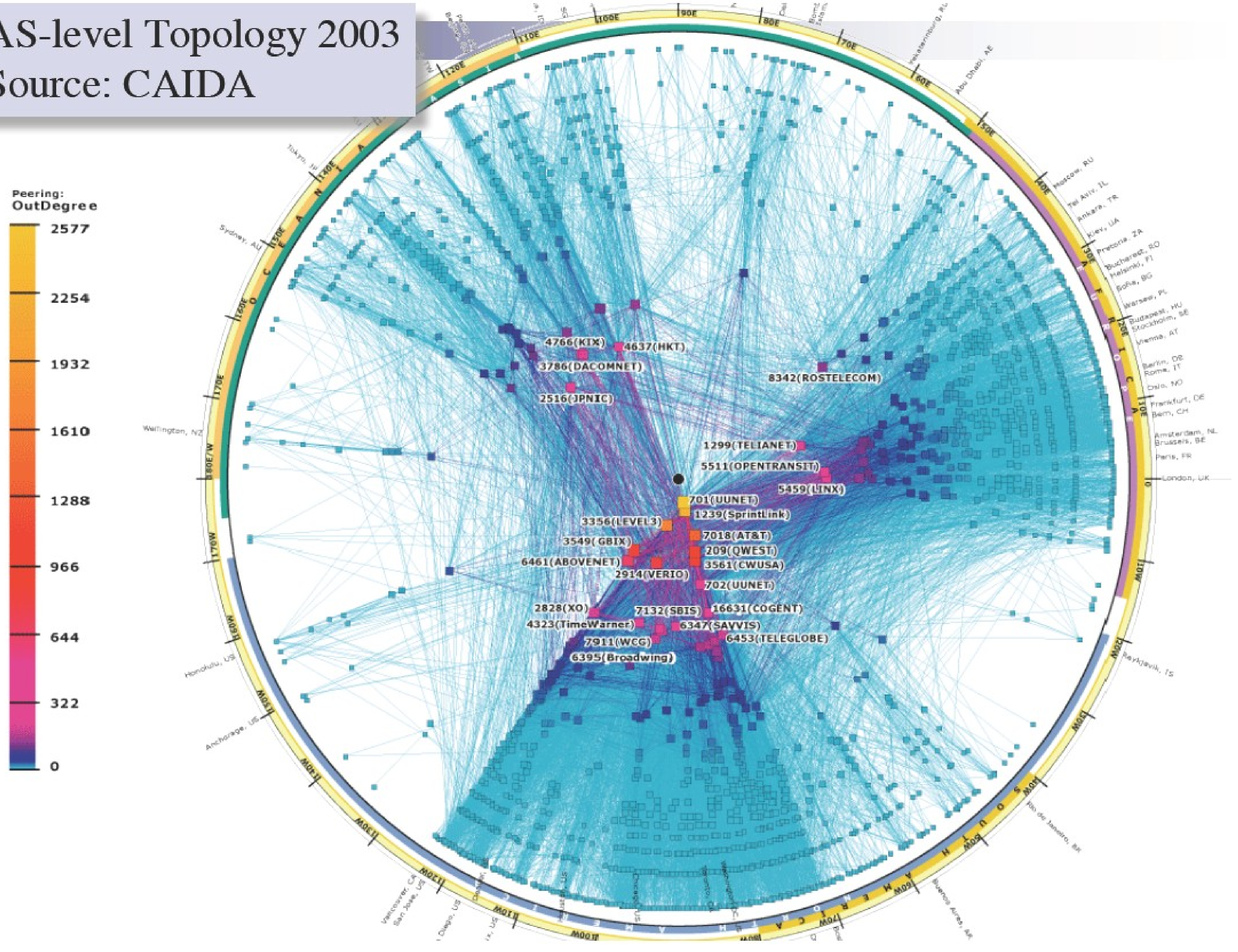


So you want to be a tier 1 network?

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 c
- I p
- M t t

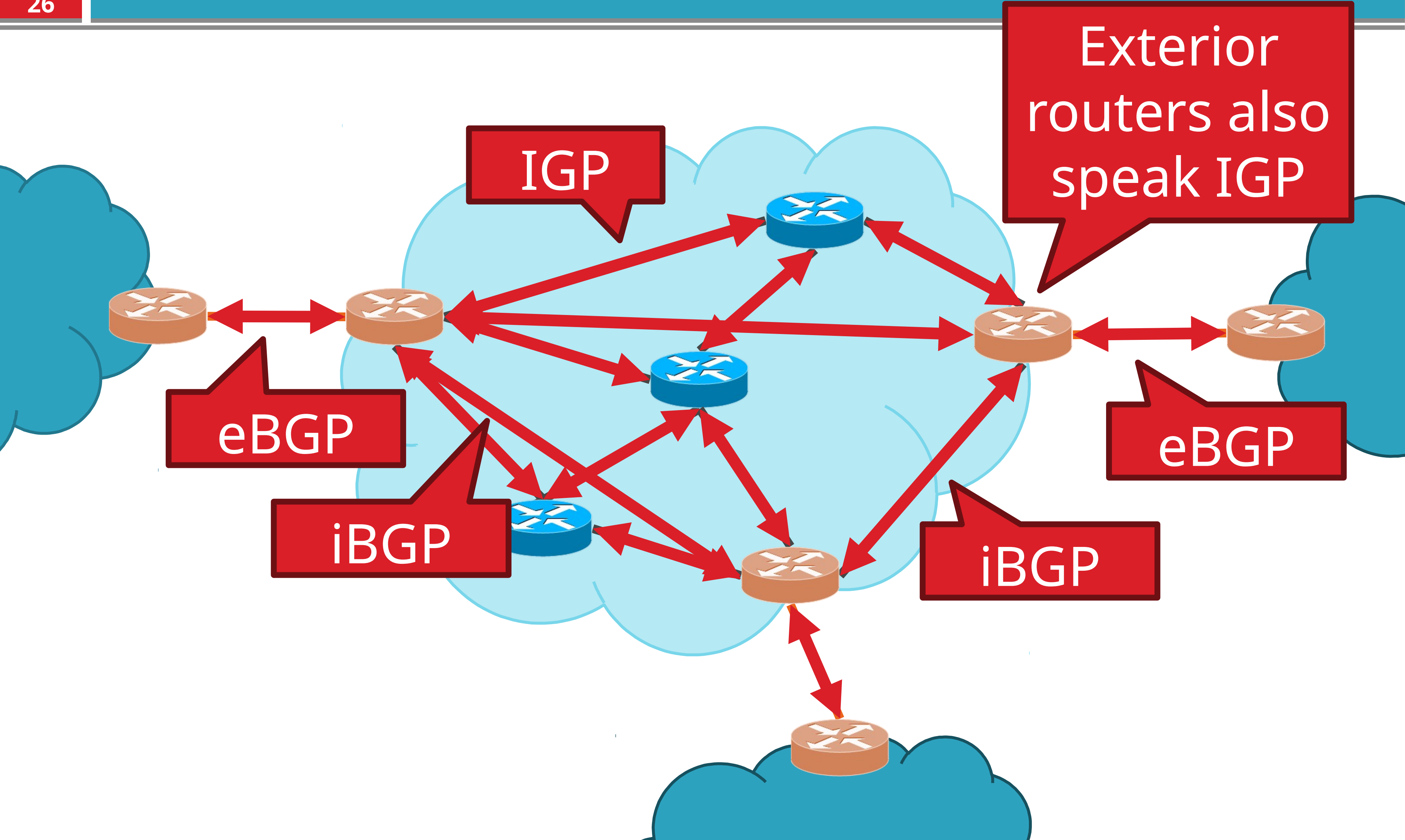
- You would rather
- ts

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

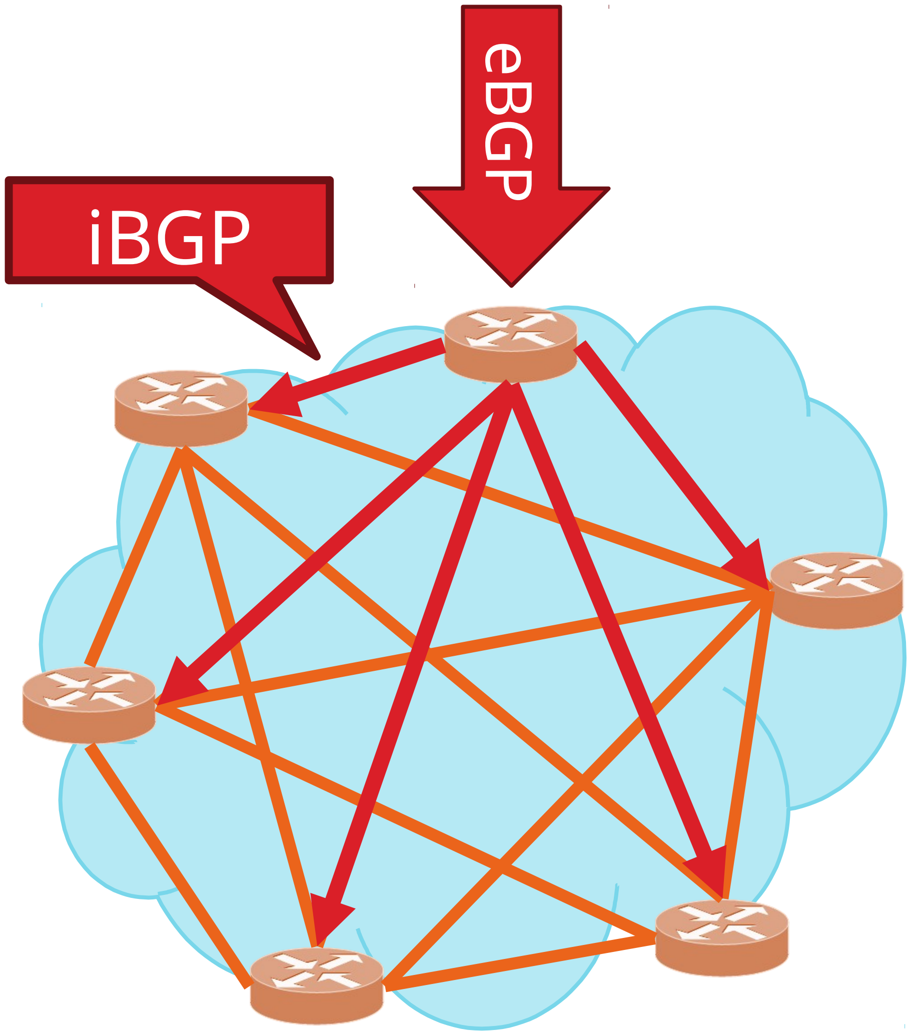
Example: If you are a customer of my peer why should I peer with you? You should pay me too!

Incentive to keep relationships private!

Two Types of BGP Neighbors



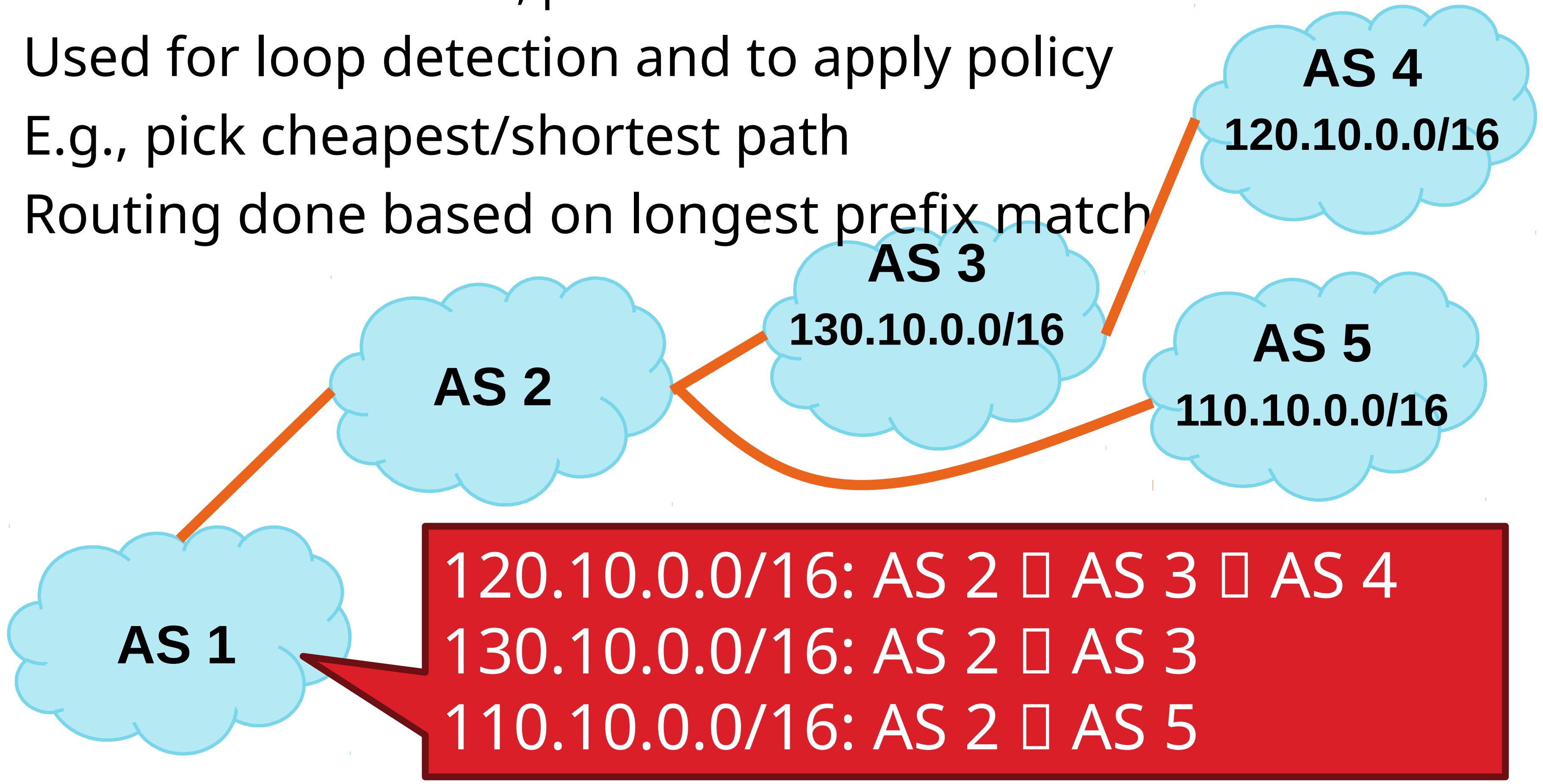
Full iBGP Meshes



- 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

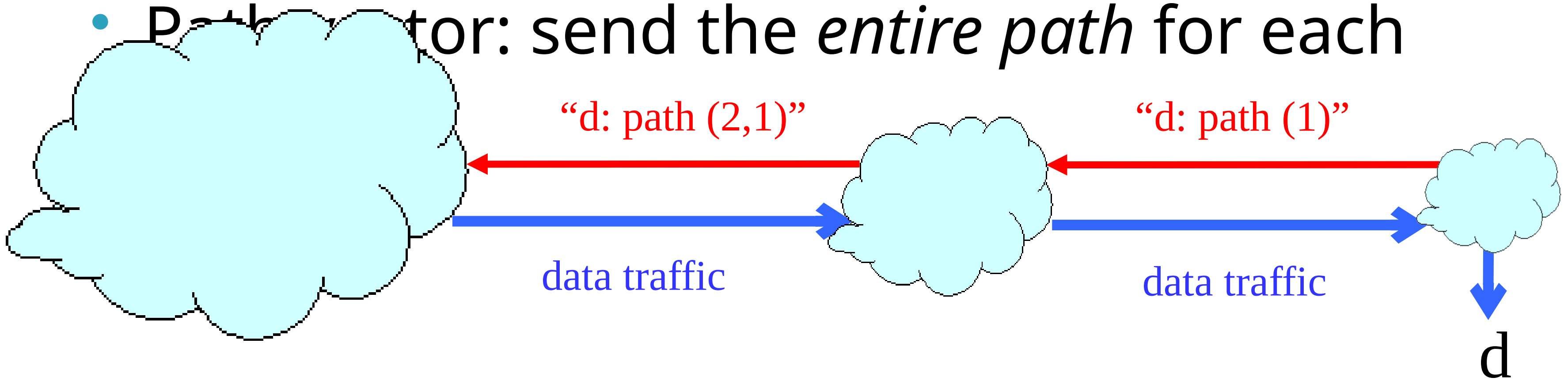
- AS-path: sequence of ASs a route traverses
  - Like distance vector, plus additional information
- Used for loop detection and to apply policy
- E.g., pick cheapest/shortest path
- Routing done based on longest prefix match



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

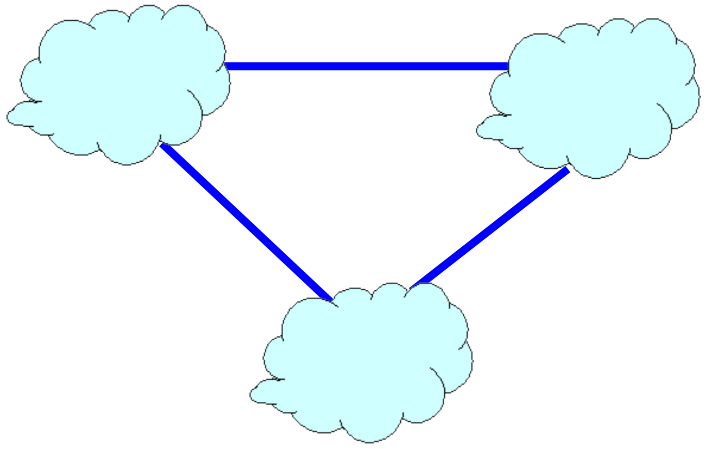
Path-Vector Routing

- 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

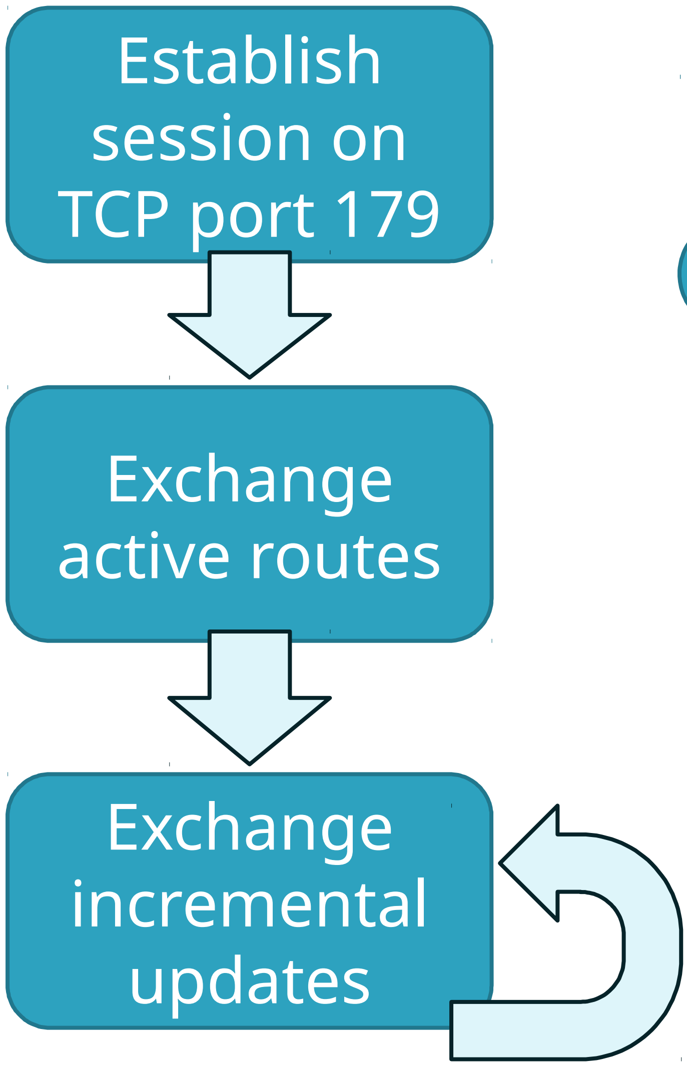


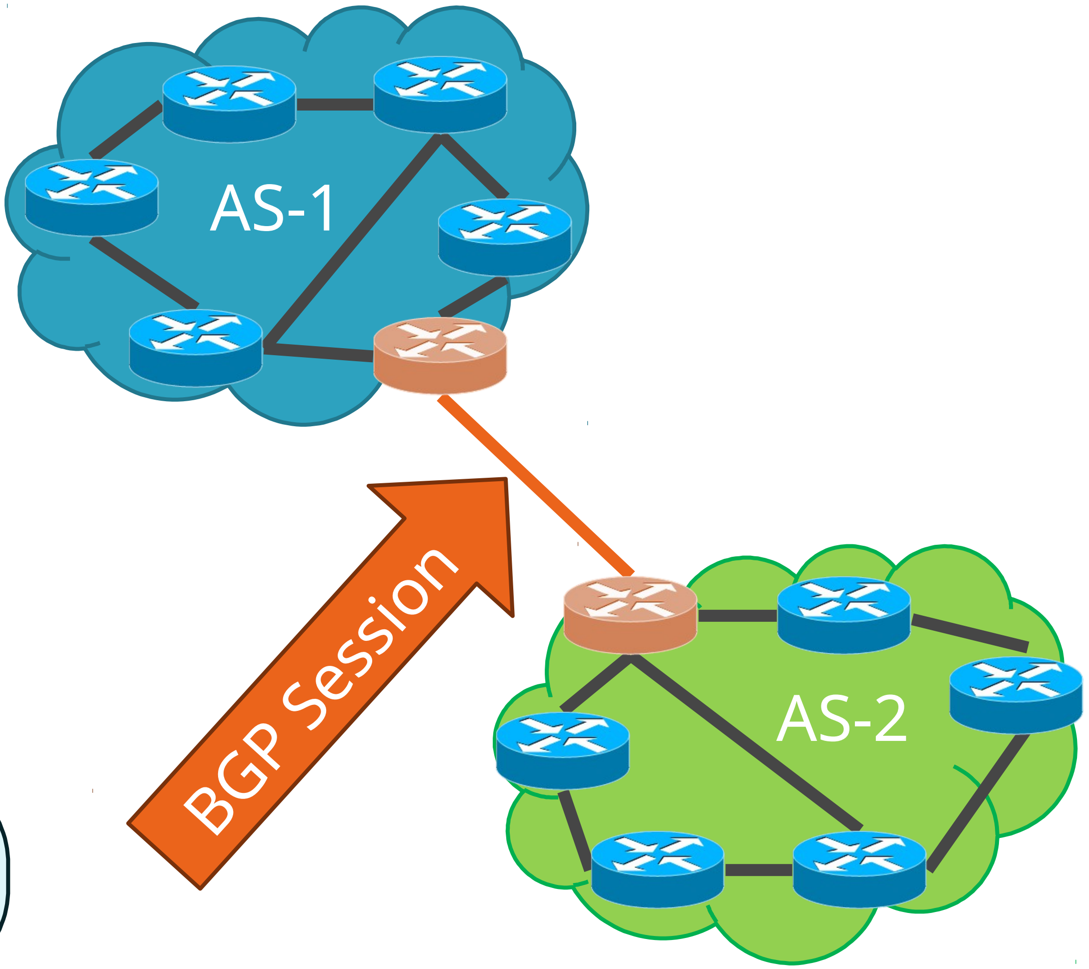
Flexible Policies

- 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

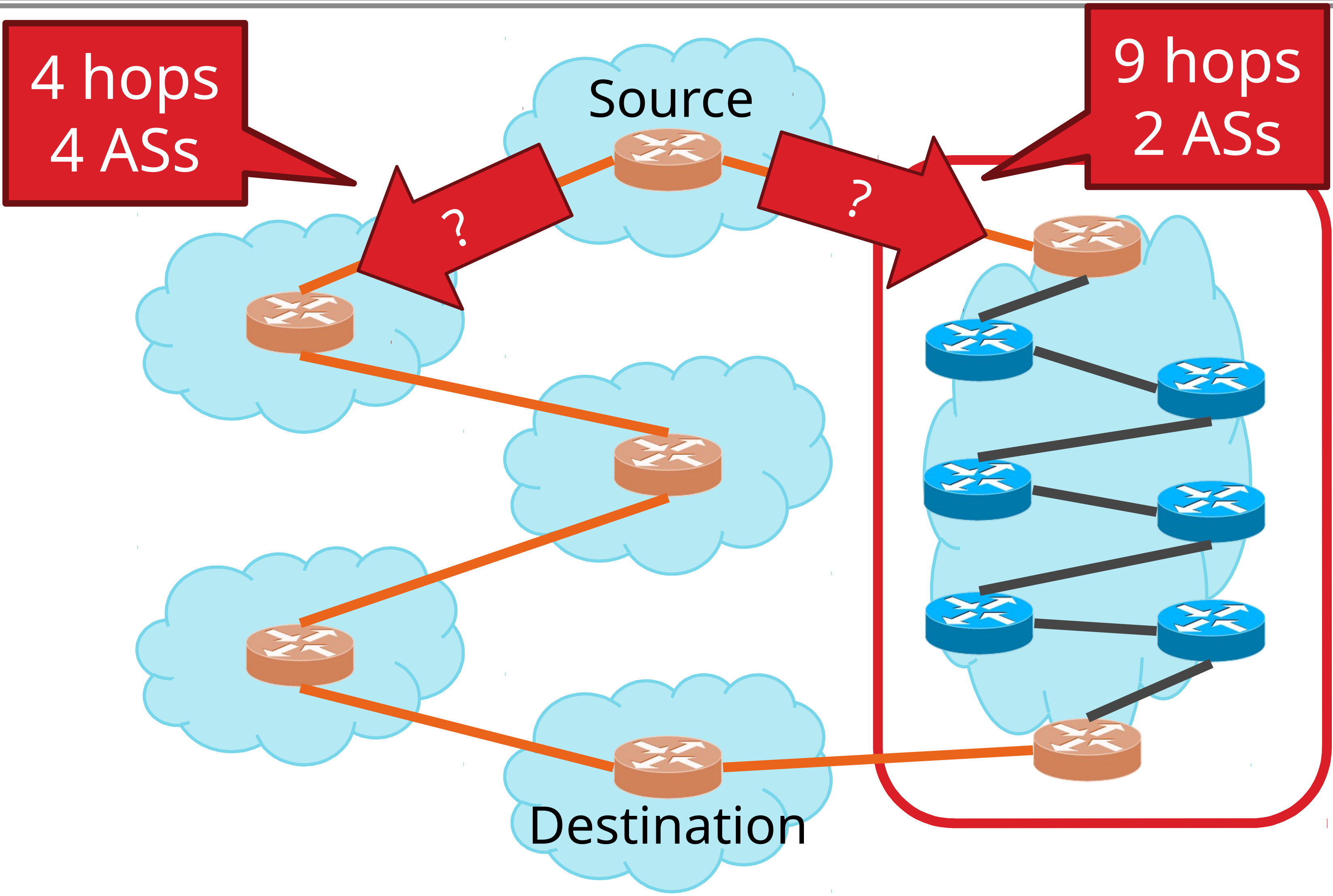
- Open: Establish a peering session.
- Keep Alive: Handshake at regular intervals.
- Notification: Shuts down a peering session.
- Update: Announce new routes or withdraw previously announced routes.

announcement = IP prefix + attributes values

BGP Attributes

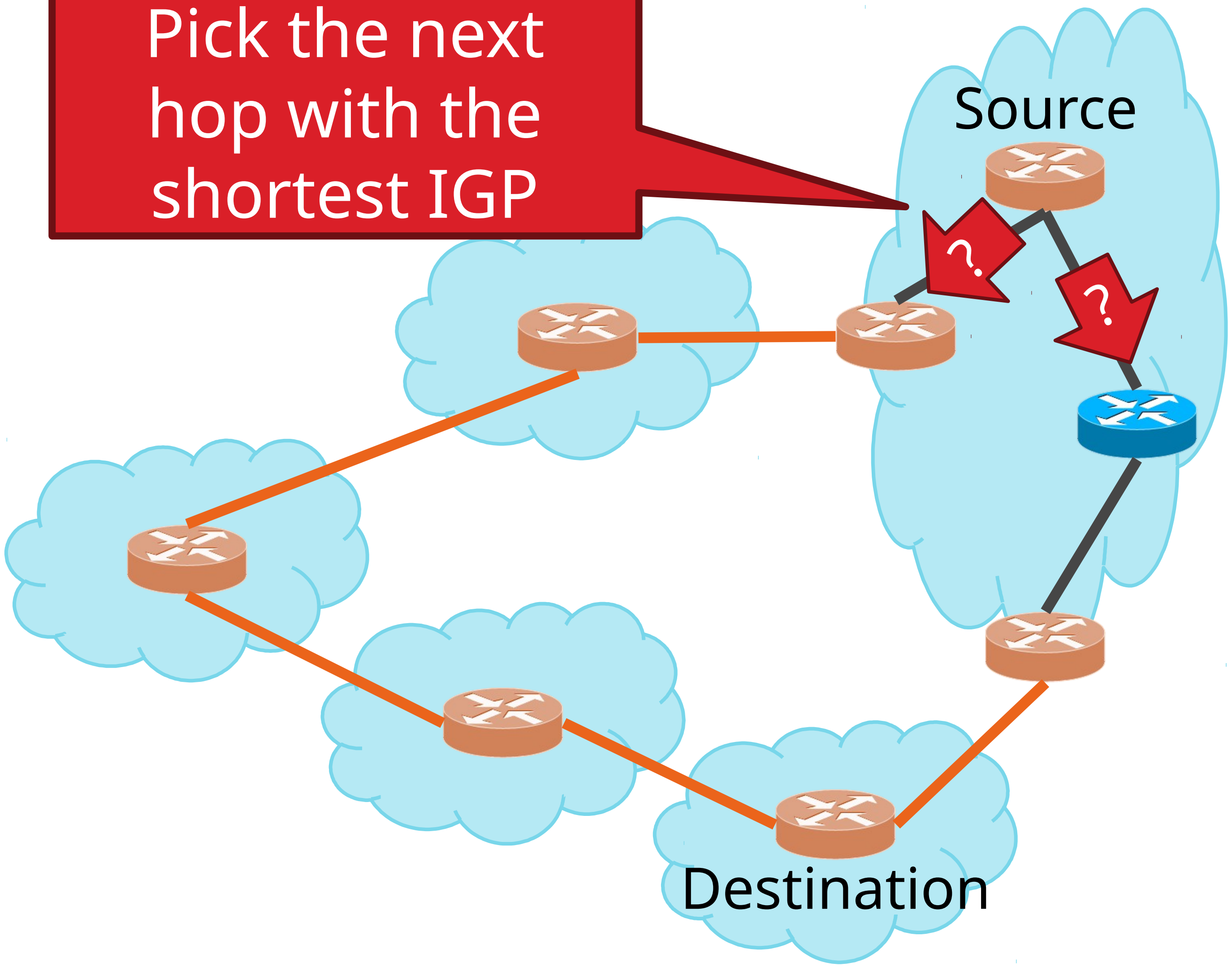
- Attributes used to select “best” path
  - LocalPref
    - Local preference policy to choose most preferred route
    - Overrides default fewest AS behavior
  - Multi-exit Discriminator (MED)
    - Specifies path for external traffic destined for an internal network
    - Chooses peering point for your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

Shortest AS Path != Shortest Path

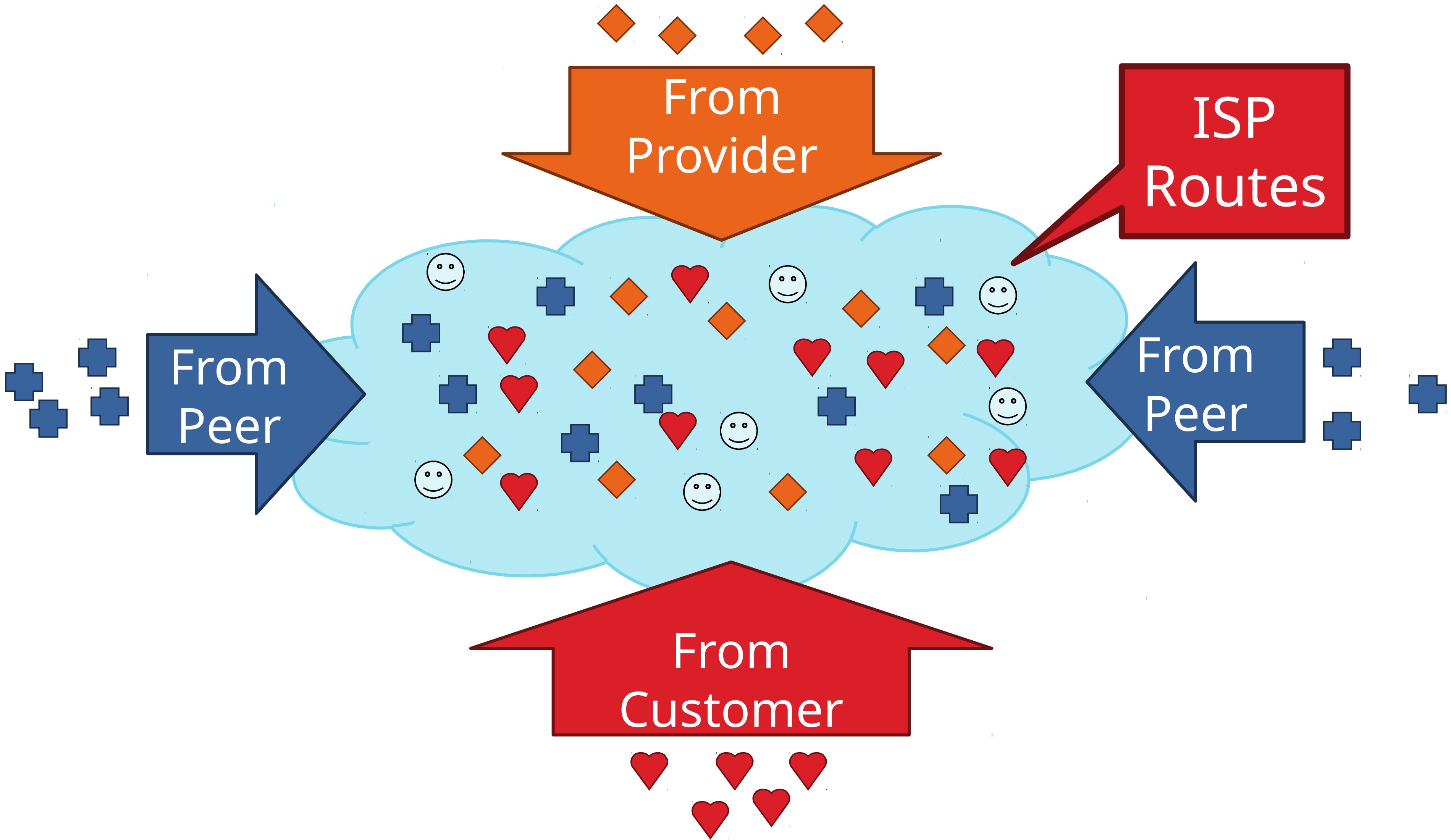


Hot Potato Routing

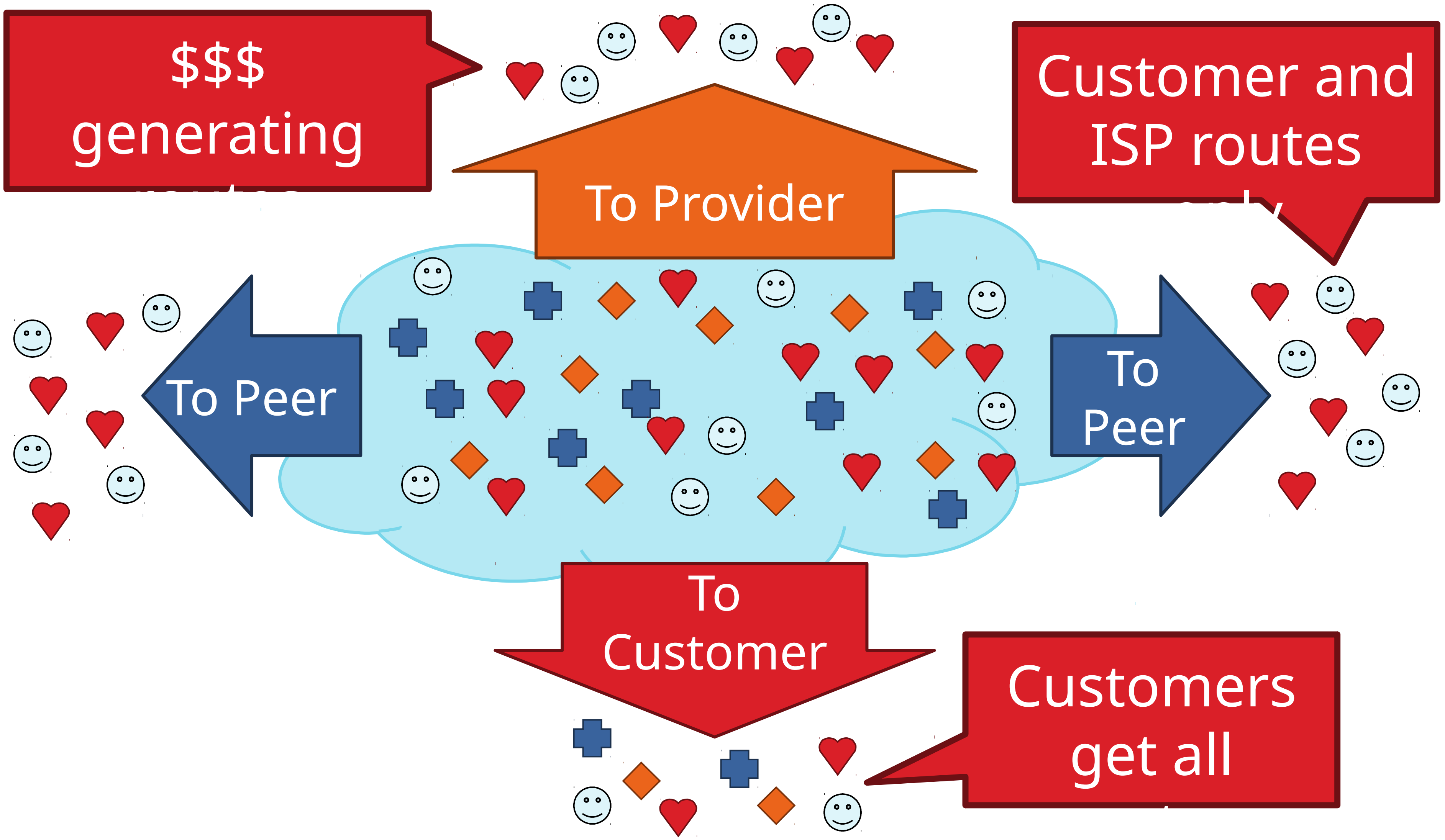
Pick the next hop with the shortest IGP



Importing Routes

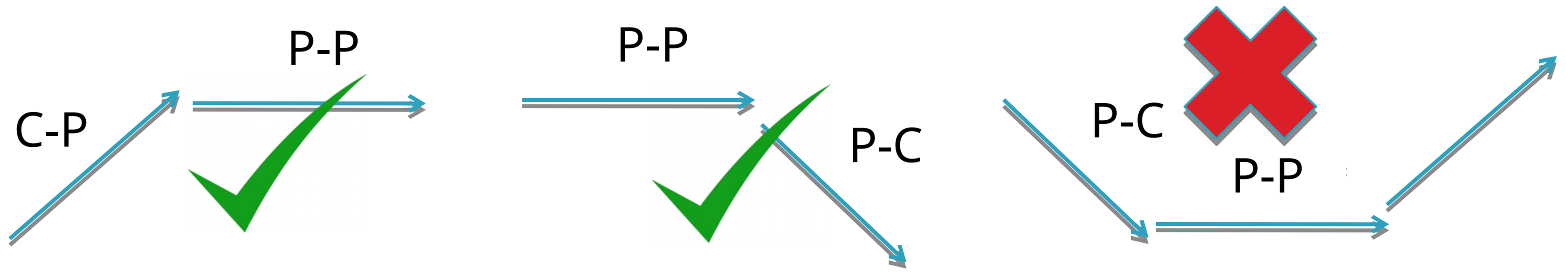


Exporting Routes



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

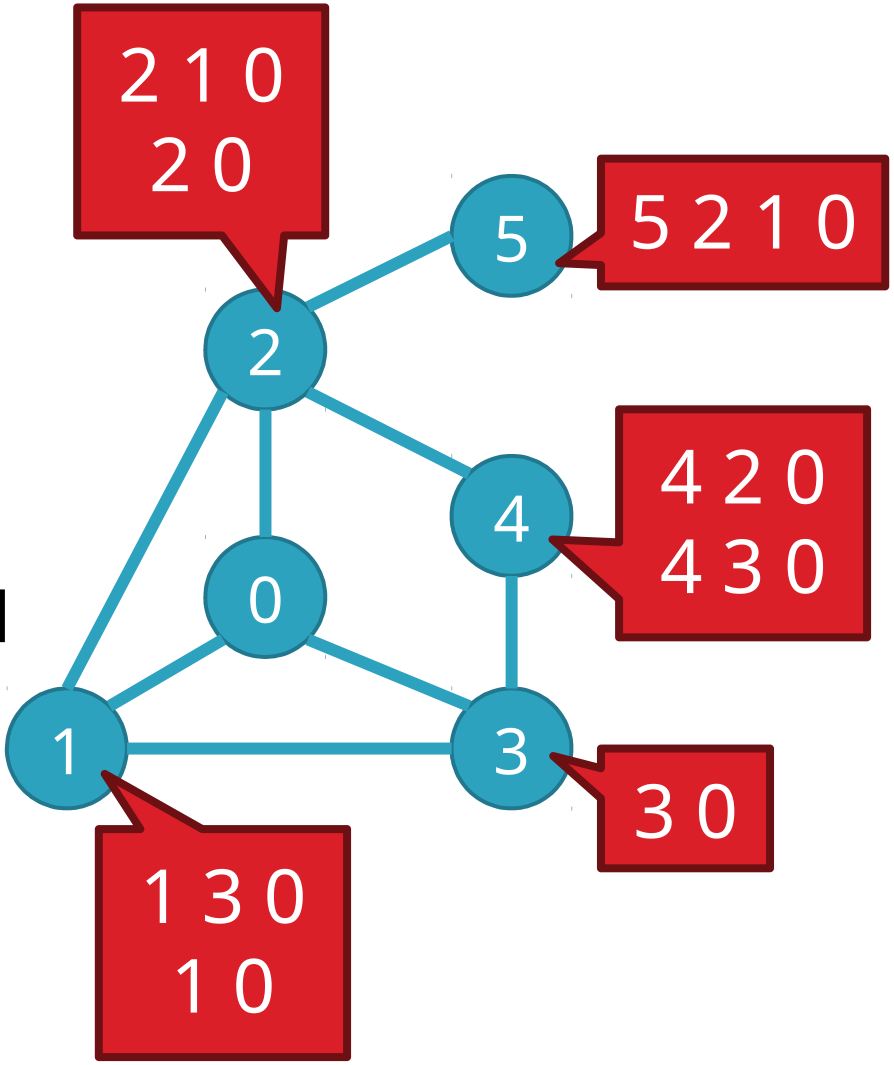
- AS_SET
  - Instead of a single AS appearing at a slot, it’s a set of Ases
- Communities
  - Arbitrary number that is used by neighbors for routing decisions
    - Export this route only in Europe
    - Do not export to your peers
  - Usually stripped after first interdomain hop
  - Why?
- Prepending
  - Lengthening the route by adding multiple instances of ASN
  - Why?

Outline

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

The Stable Paths Problem

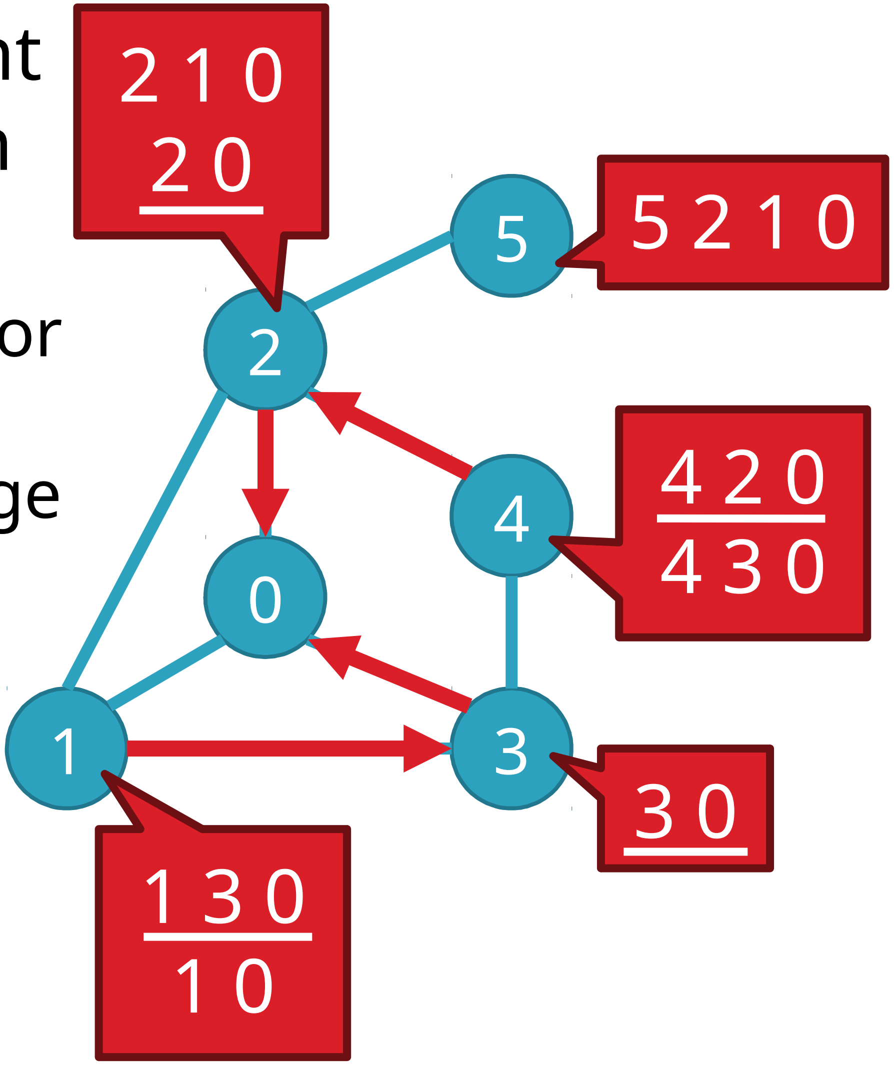
- An instance of the SPP:
  - Graph of nodes and edges
  - Node 0, called the origin
  - A set of permitted paths from each node to the origin
  - Each set of paths is ranked



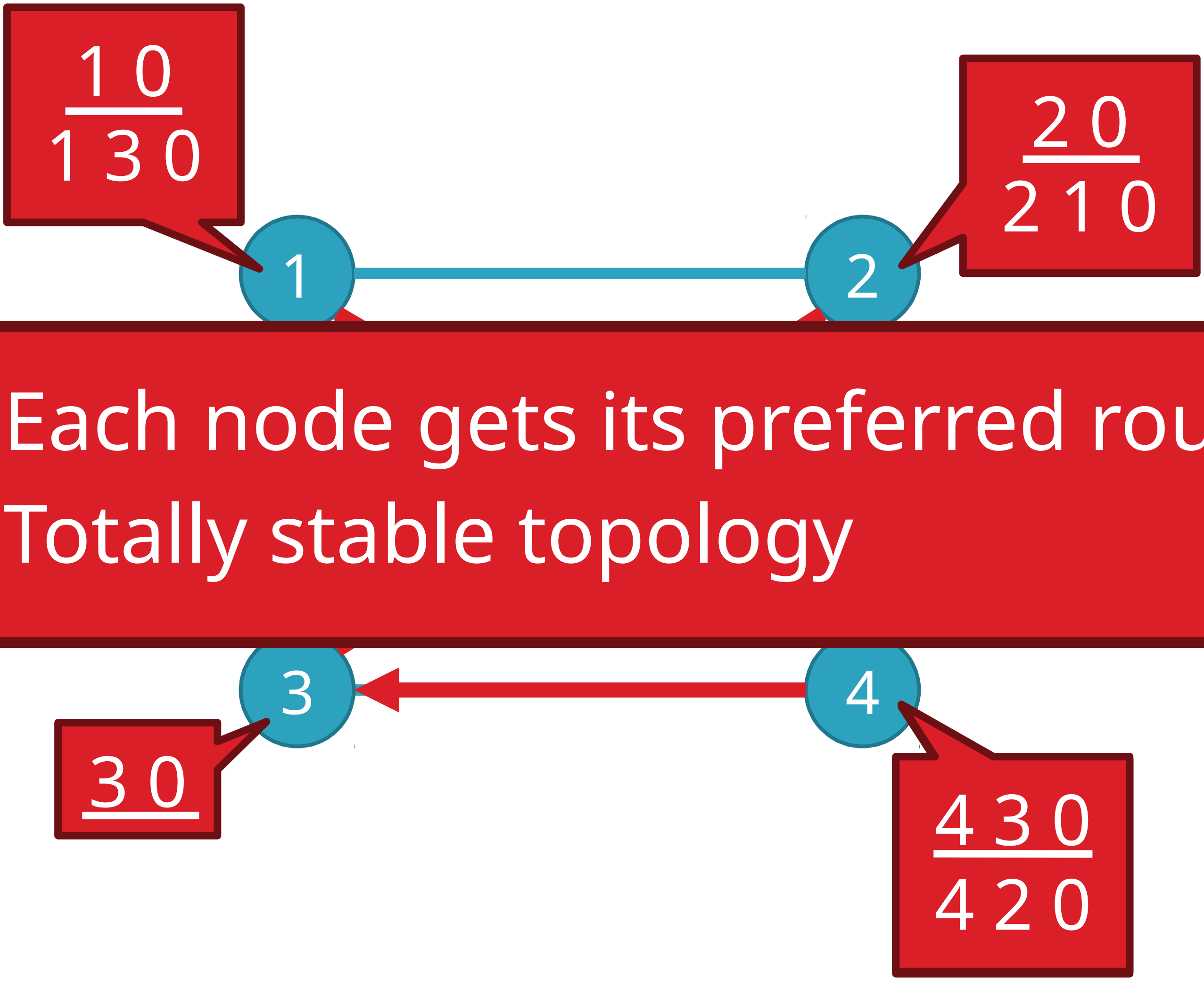
A Solution to the SPP

- A solution is an assignment of permitted paths to each node such that:
  - or
  - ge
  - consistent with their neighbors

Solutions need not use the shortest paths, or form a spanning tree

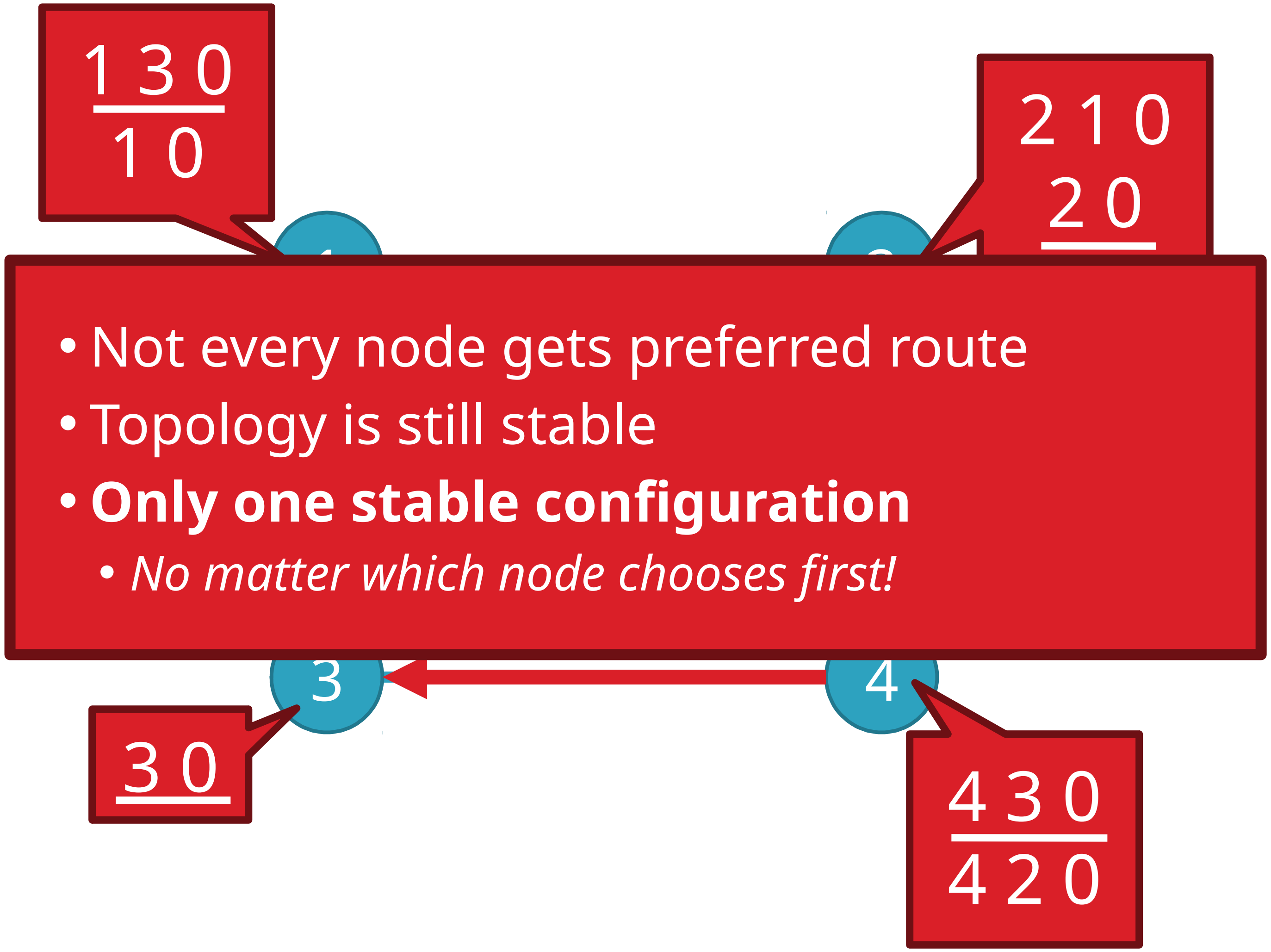


Simple SPP Example



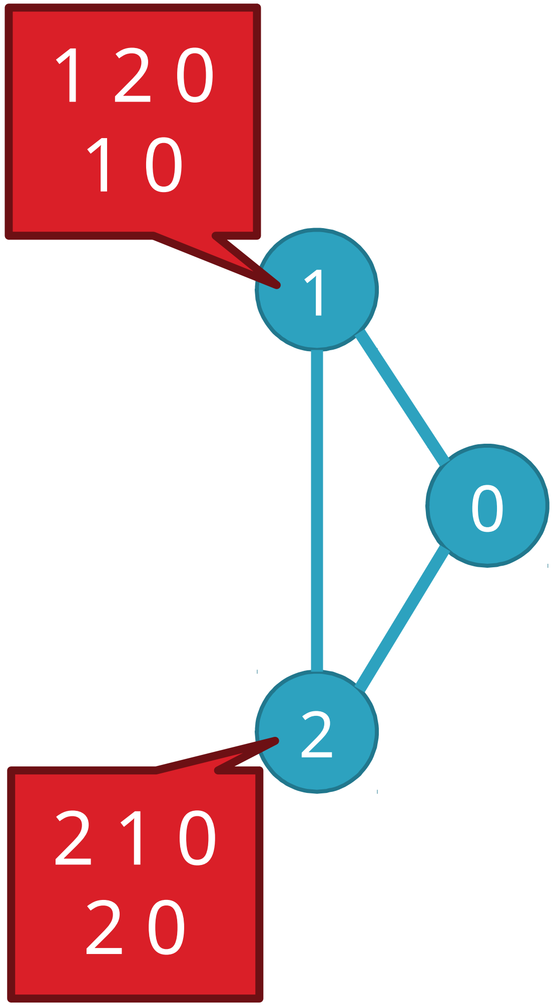
- Each node gets its preferred route
- Totally stable topology

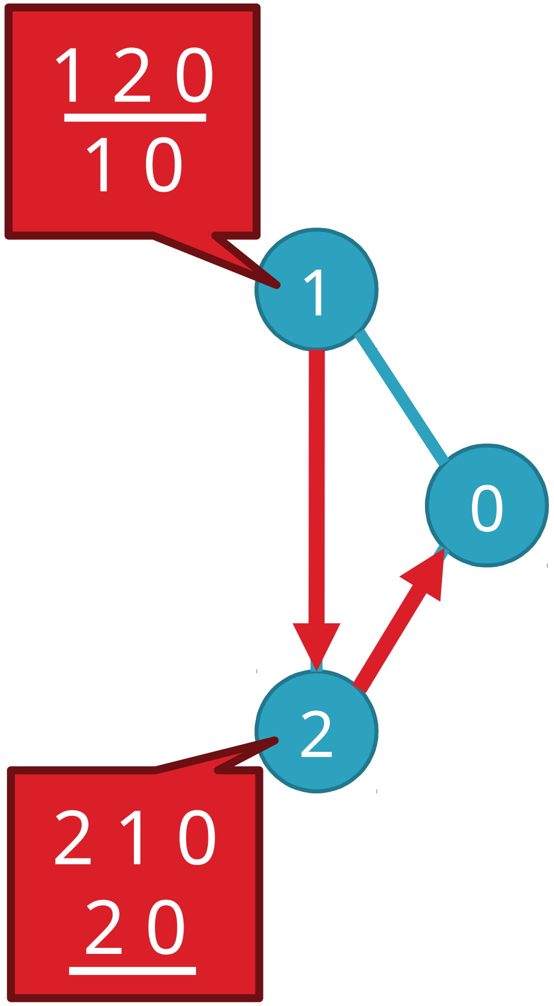
Good Gadget

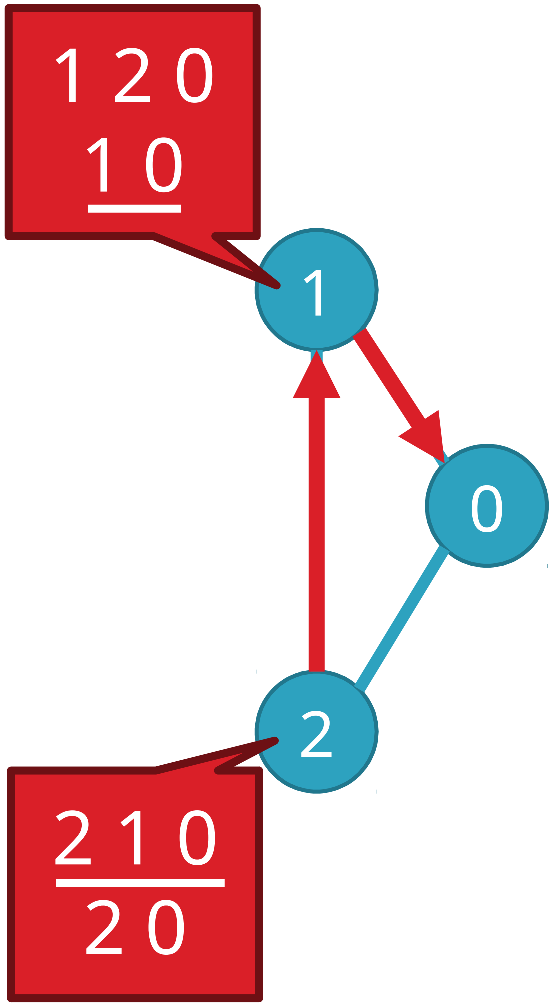


- Not every node gets preferred route
- Topology is still stable
- **Only one stable configuration**
  - *No matter which node chooses first!*

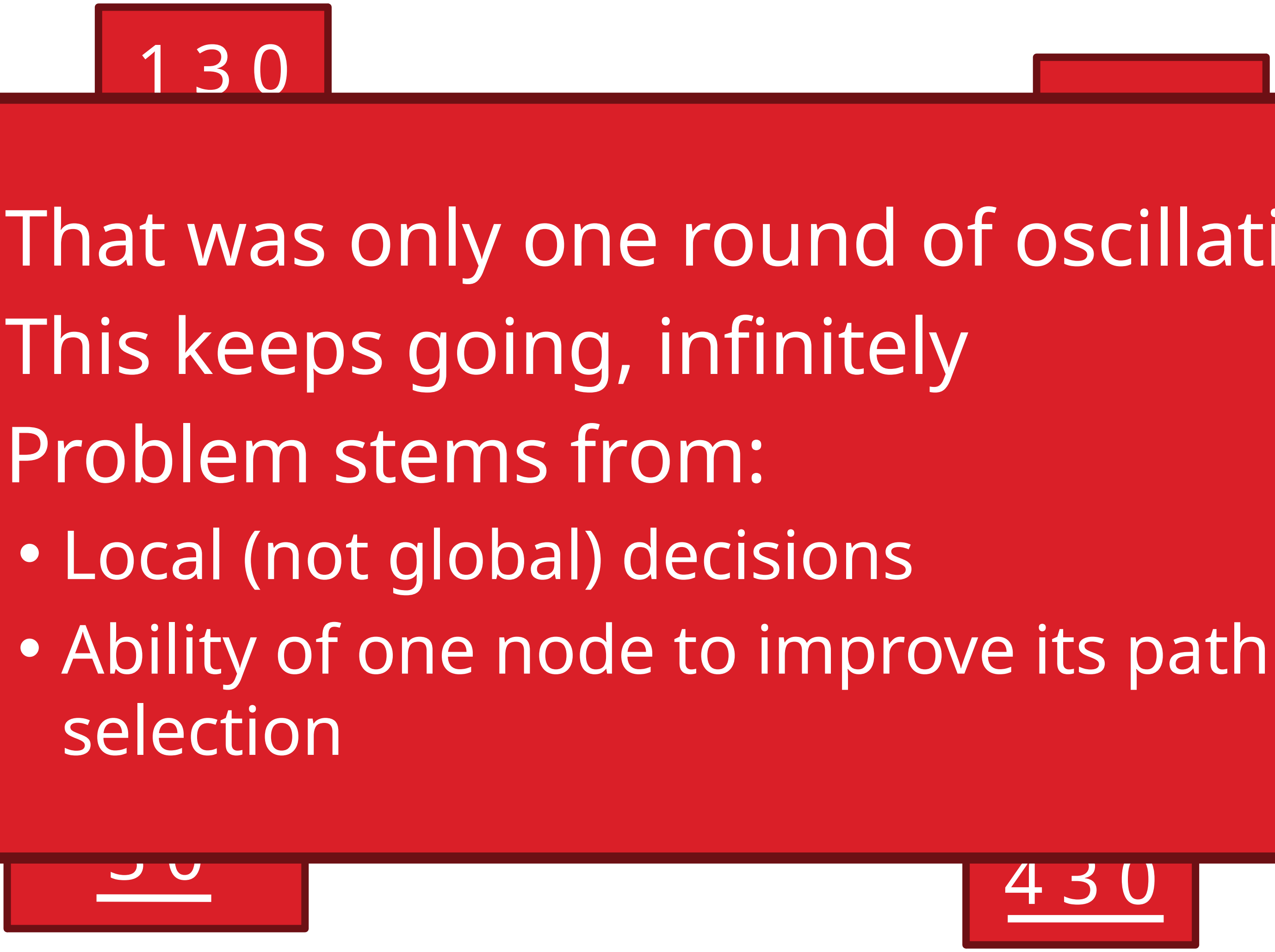
SPP May Have Multiple Solutions







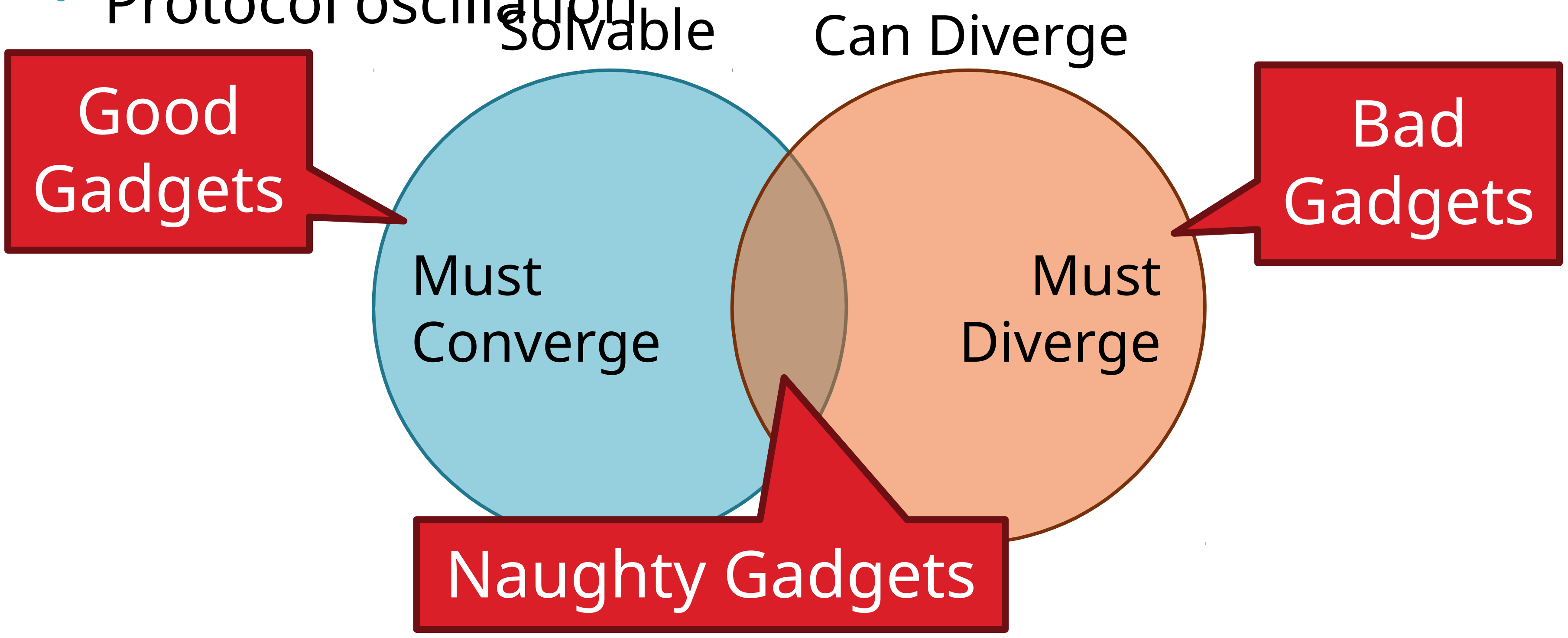
Bad Gadget



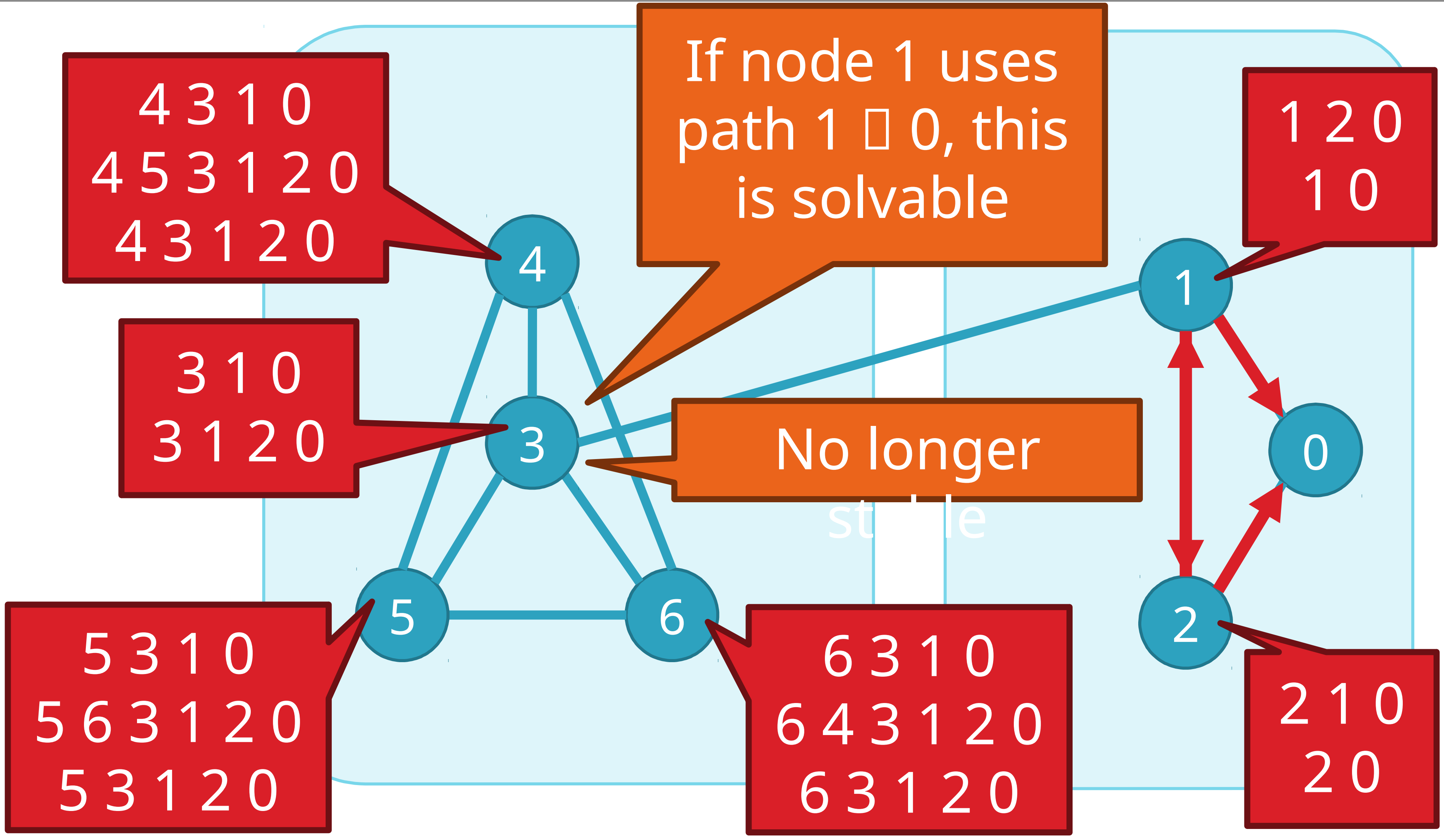
- That was only one round of oscillation!
- This keeps going, infinitely
- Problem stems from:
  - Local (not global) decisions
  - Ability of one node to improve its path selection

SPP Explains BGP Divergence

- BGP is not guaranteed to converge to stable routing
  - Policy inconsistencies may lead to “livelock”
  - Protocol oscillation

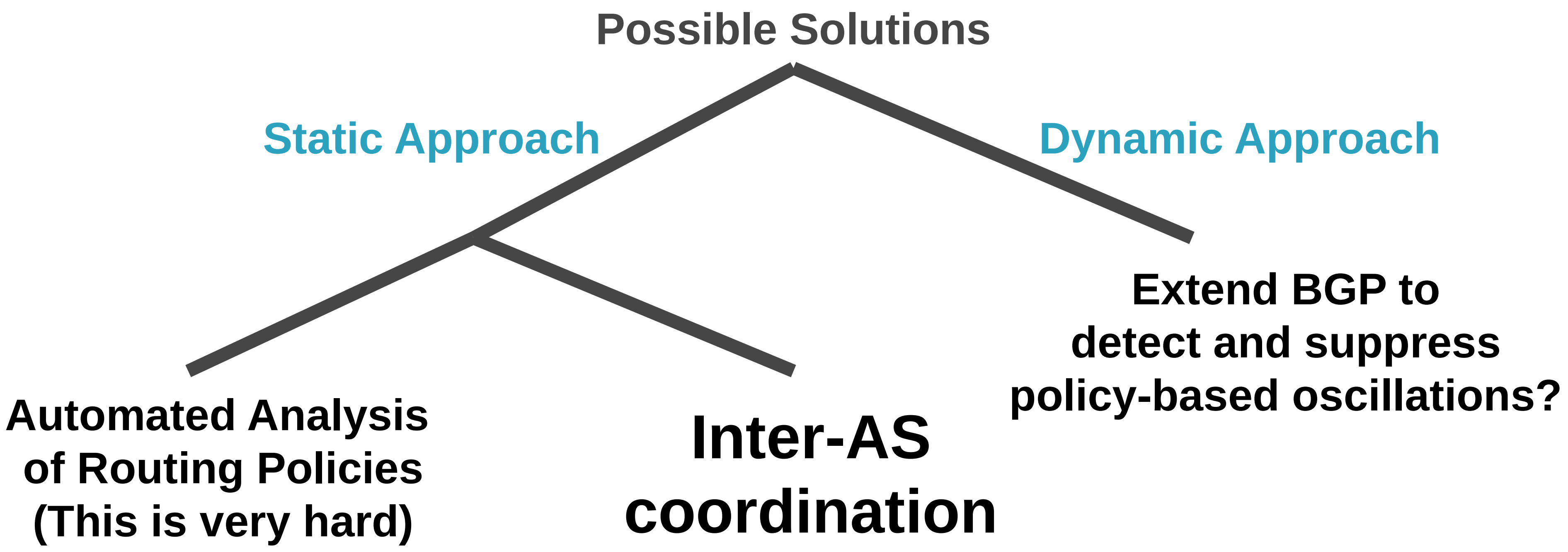


BGP is Precarious



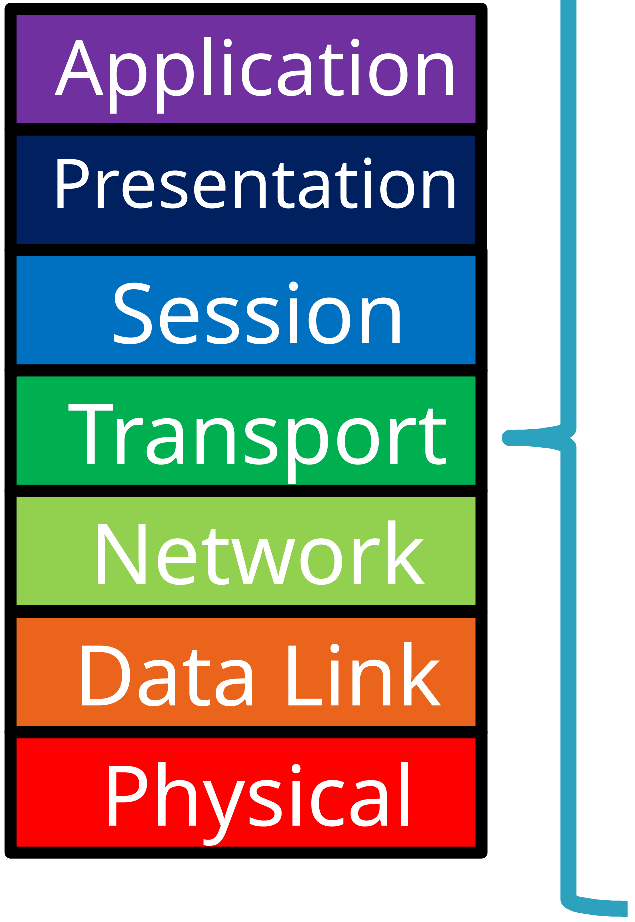
Can BGP Be Fixed?

- Unfortunately, SPP is NP-complete


Possible Solutions
Static Approach
Dynamic Approach
Automated Analysis of Routing Policies (This is very hard)
Inter-AS coordination
Extend BGP to detect and suppress policy-based oscillations?

These approaches are complementary

Transport Layer



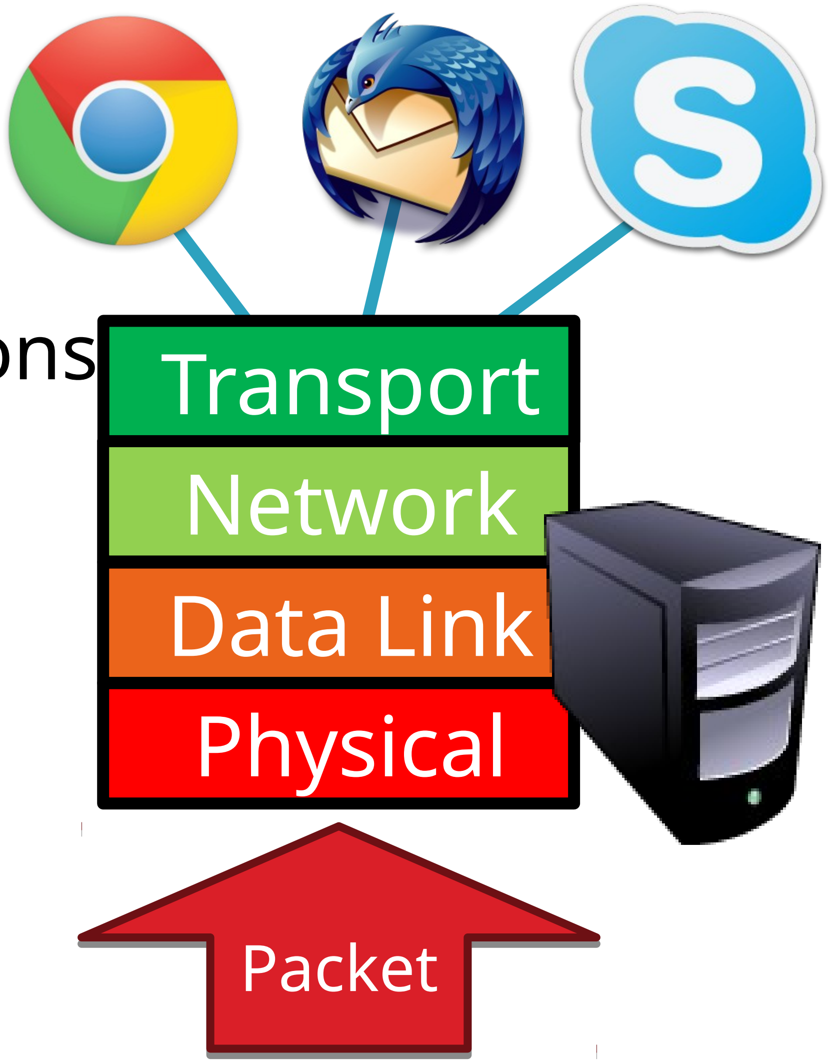
- Function:
  - Demultiplexing of data streams
- Optional functions:
  - Creating long lived connections
  - Reliable, in-order packet delivery
  - Error detection
  - Flow and congestion control
- Key challenges:
  - Detecting and responding to congestion
  - Balancing fairness against high utilization

Outline

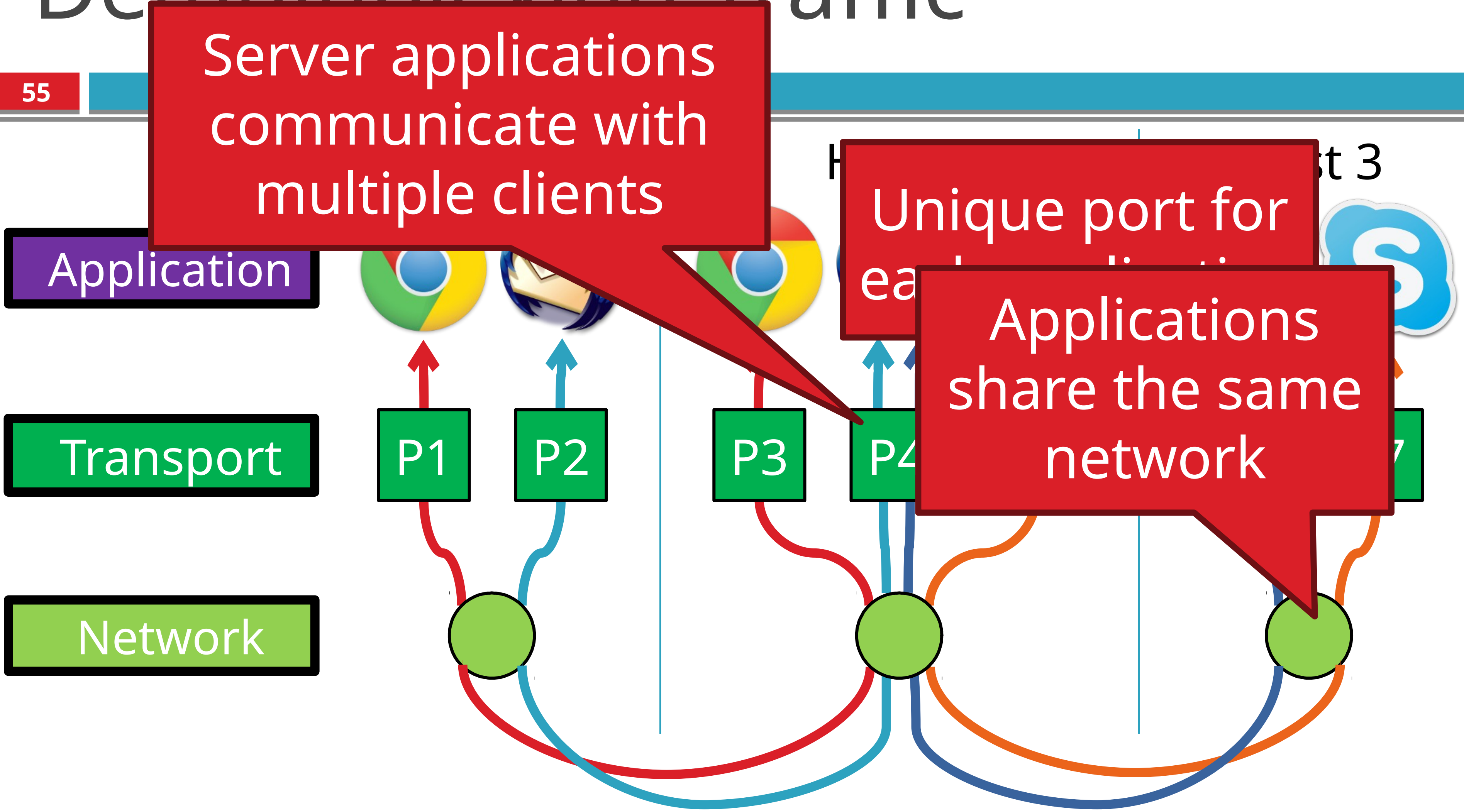
- UDP
- TCP
- Congestion Control
- Evolution of TCP
- Problems with TCP

The Case for Multiplexing

- Datagram network
  - No circuits
  - No connections
- Clients run many applications at the same time
  - Who to deliver packets to?
- IP header "protocol" field
  - 8 bits = 256 concurrent streams
- Insert Transport Layer to handle demultiplexing



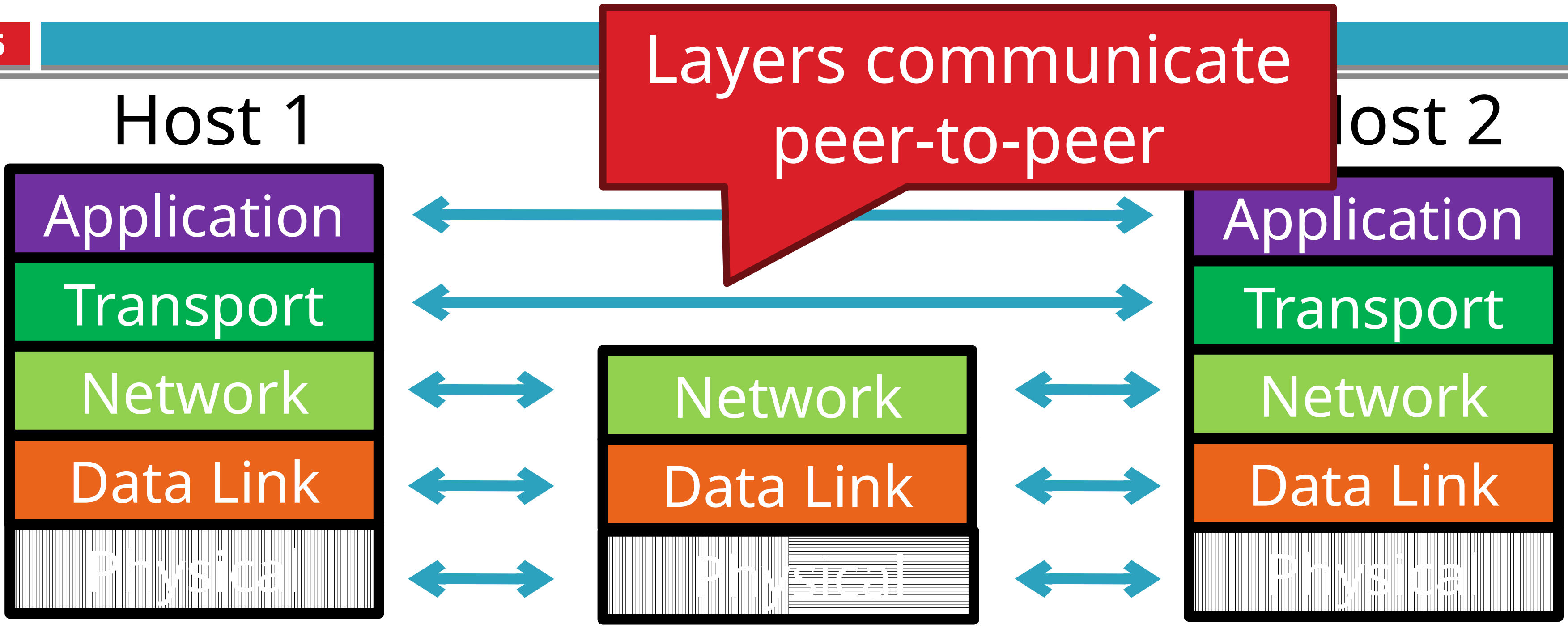
Demultiplexing Traffic



Server applications communicate with multiple clients
Unique port for each application
Applications share the same network
Application
Transport
Network
P1
P2
P3
P4
st 3

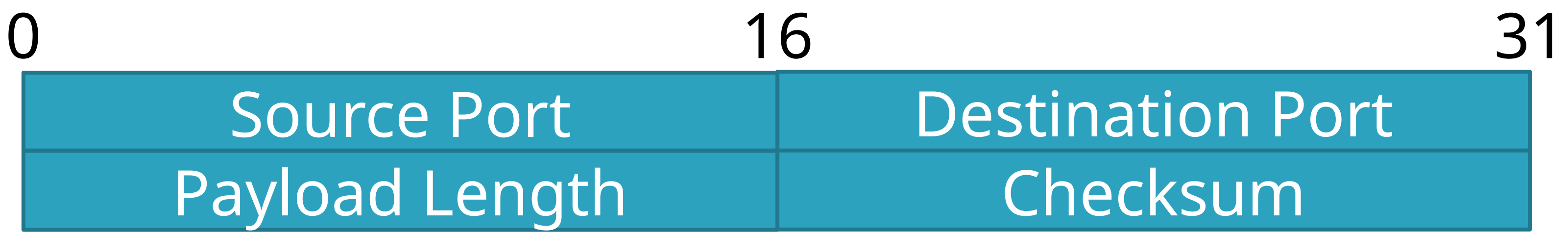
Endpoints identified by <*src_ip, src_port, dest_ip, dest_port*>

Layering, Revisited



- Lowest level end-to-end protocol
  - Transport header only read by source and destination
  - Routers view transport header as payload

User Datagram Protocol (UDP)



- Simple, connectionless datagram
  - C sockets: SOCK_DGRAM
- Port numbers enable demultiplexing
  - 16 bits = 65535 possible ports
  - Port 0 is invalid
- Checksum for error detection
  - Detects (some) corrupt packets
  - Does not detect dropped, duplicated, or reordered packets

Uses for UDP

- Invented after TCP
  - Why?
- Not all applications can tolerate TCP
- Custom protocols can be built on top of UDP
  - Reliability? Strict ordering?
  - Flow control? Congestion control?
- Examples
  - RTMP, real-time media streaming (e.g. voice, video)
  - Facebook datacenter protocol