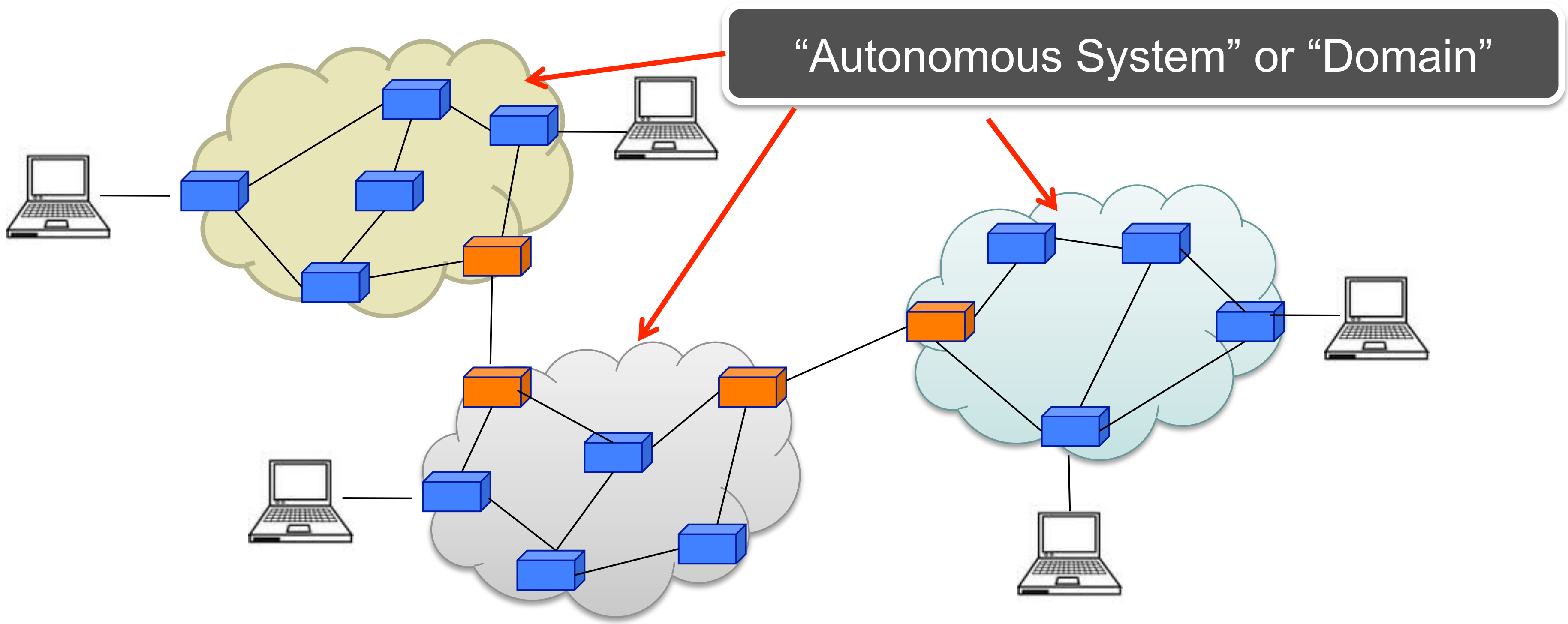
ECE 158A: Lecture 6

Fall 2015

Routing (2)

Review: Autonomous Systems

- The Internet is grouped into about 40,000 Autonomous Systems (AS) or “domains”
  - Groups of hosts/routers under a single administrative entity
- Each AS is assigned a unique identifier
  - 16 bit AS Number (ASN)



Review: Two-Level Routing

- Routing is performed at two levels:
  - Routing within a domain is called intra-domain routing
  - Routing across domains is called inter-domain routing

Review: LS vs DV

- Link State (LS):
  - Each node learns the complete network map (global information)
  - Each node computes shortest paths independently and in parallel

- Distance Vector (DV):
  - No node has the complete picture (local information)
  - Nodes cooperate to compute shortest paths in a distributed manner

- LS uses global information, while DV is asynchronous, and distributed.
- LS has higher messaging overhead and higher processing complexity, but is less vulnerable to looping

Distance Vector

Learning-by-Doing

- In-class networking experiment:
  - Source node: The instructor
  - Destination node: The youngest student in the room
  - Routers: Everybody else
- Goal: By only communicating to your neighbors
  - Identify the destination
  - Be ready to route a packet toward the destination
- Hint: maintain a vector for next hop to all ages and exchange it
- Ready-Set-Go!

Distance Vector Routing

- Each router knows only the next hop for each node, not the complete path
- Each router knows and updates provisional cost to every other router
  - E.g.: Router A: “I can get to router B with cost 11”
- Routers exchange this distance vector information with their neighboring routers
  - Vector because one entry per destination
- Routers look over the set of options offered by their neighbors and select the best one
- Iterative process converges to set of shortest paths

Bellman-Ford Algorithm

- INPUT:
  - Link costs to each neighbor
    (*Not full topology*)
- OUTPUT:
  - Next hop to each destination and the corresponding cost
    (*Not the complete path to the destination*)

- My neighbors tell me how far they are from destination
  - Compute: (cost to neighbors) + (neighborr's cost to destination)
  - Pick minimum as my choice
  - Advertise that cost to my neighbors

Bellman-Ford Overview

- Each router maintains a table
  - Best known distance from X to Y, via Z as next hop = $D_Z(X,Y)$
- Each local iteration caused by:
  - Local link cost change
  - Message from neighbor
- Notify neighbors *only* if least cost path to any destination changes
  - Neighbors then notify their neighbors if necessary

Each node:

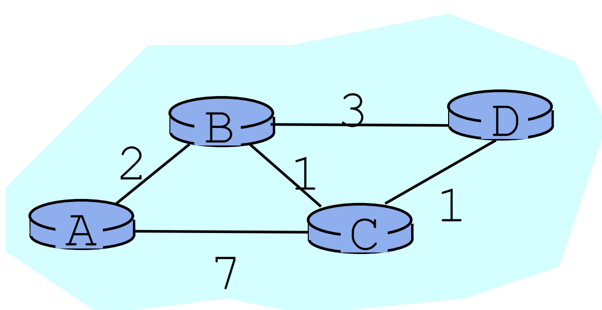
wait for (change in local link cost or msg from neighbor)

recompute distance table

if least cost path to any dest has changed, *notify* neighbors

Bellman-Ford Overview

- Each router maintains a table
  - Row for each possible destination
  - Column for each directly-attached neighbor to node
  - Entry in row Y and column Z of node X ⇒ best known distance from X to Y, via Z as next hop = $D_Z(X,Y)$



Node A

	B	C
B	2	8
C	3	7
D	4	8

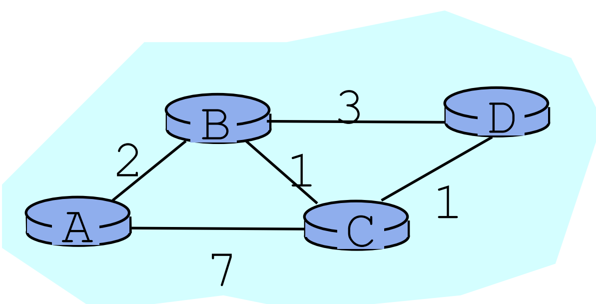
Neighbor (next-hop)

Destinations

$D_C(A, D)$

Bellman-Ford Overview

- Each router maintains a table
  - Row for each possible destination
  - Column for each directly-attached neighbor to node
  - Entry in row Y and column Z of node X ⇒ best known distance from X to Y, via Z as next hop = $D_Z(X,Y)$



Node A

	B	C
B	2	8
C	3	7
D	4	8

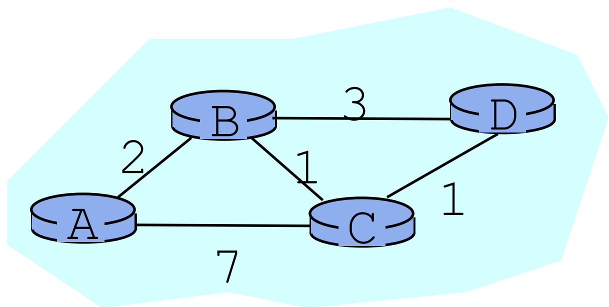
Smallest distance in row Y = shortest Distance of A to Y, **D(A, Y)**

Distance Vector Algorithm

- **c(i,j): link cost from node *i* to *j***
- **D_Z(A,V): cost from A to V via Z**
- **D(A,V): cost of A's best path to V**

```
Initialization:
   for all neighbors V  do
        if V adjacent to A
             D(A, V) = c(A,V);
        else
             D(A, V) = ∞;
     send D(A, Y) to all neighbors
 loop:
   wait (until A sees a link cost change to neighbor V  /* case 1 */
            or until A receives update from neighbor V)    /* case 2 */
   if (c(A,V) changes by ±d)  /* ⇐ case 1 */
           for all destinations Y that go through V do
                 D_V(A,Y) =  D_V(A,Y) ± d
   else if (update D(V, Y) received from V) /* ⇐ case 2 */
           /* shortest path from V to some Y has changed  */
           D_V(A,Y) = D_V(A,V) + D(V, Y);    /* may also change D(A,Y) */
   if (there is a new minimum for destination Y)
           send D(A, Y) to all neighbors
   forever
```

Example: Initialization



Node A

	B	C
B	2	∞
C	∞	7
D	∞	∞

Node B

	A	C	D
A	2	∞	∞
C	∞	1	∞
D	∞	∞	3

Node C

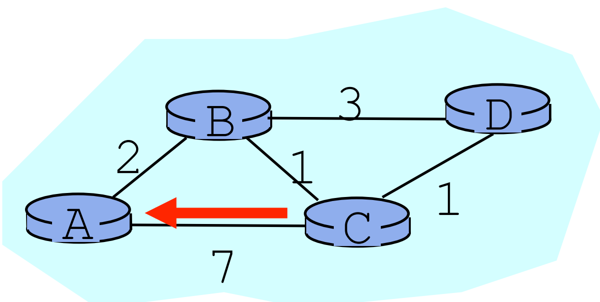
	A	B	D
A	7	∞	∞
B	∞	1	∞
D	∞	∞	1

Node D

	B	C
A	∞	∞
B	3	∞
C	∞	1

```
Initialization:
  for all neighbors V do
     if V adjacent to A
        D(A, V) = c(A,V);
     else
        D(A, V) = ∞;
   send D(A, Y) to all neighbors
```

Example: C sends update to A



Node A

	B	C
B	2	8
C	∞	7
D	∞	8

Node B

	A	C	D
A	2	∞	∞
C	∞	1	∞
D	∞	∞	3

$D_C(A, B) = D_C(A,C) + D(C, B) = 7 + 1 = 8$

$D_C(A, D) = D_C(A,C) + D(C, D) = 7 + 1 = 8$

Node C

	A	B	D
A	7	∞	∞
B	∞	1	∞
D	∞	∞	1

Node D

	B	C
A	∞	∞
B	3	∞
C	∞	1

loop:

…

else if (update D(*A, Y*) from *C*)

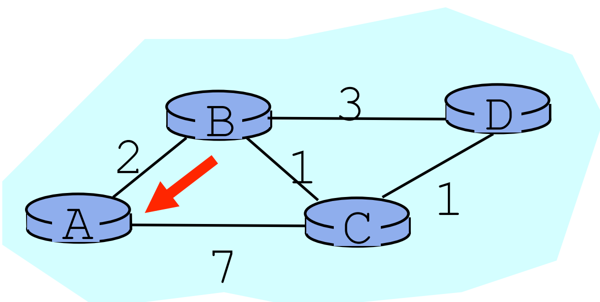
  $D_C(A,Y) = D_C(A,C) + D(C, Y)$;

if (new min. for destination Y)

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

forever

Example: Now B sends update to A



Node A

	B	C
B	2	8
C	3	7
D	5	8

Node B

	A	C	D
A	2	∞	∞
C	∞	1	∞
D	∞	∞	3

$D_B(A, C) = D_B(A,B) + D(B, C) = 2 + 1 = 3$

$D_B(A, D) = D_B(A,B) + D(B, D) = 2 + 3 = 5$

loop:

…

else if (update D(*A*, *Y*) from *B*)

  $D_B(A,Y) = D_B(A,B) + D(B, Y)$;

if (new min. for destination Y)

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

forever

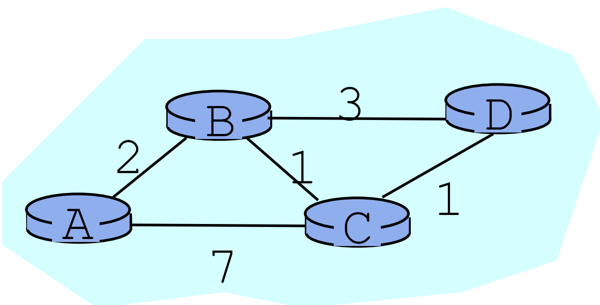
Node C

	A	B	D
A	7	∞	∞
B	∞	1	
D	∞	∞	1

Node D

	B	C
A	∞	∞
B	3	∞
C	∞	1

Example: After 1^{st} Full Exchange



Node A

	B	C
B	2	8
C	3	7
D	5	8

Node B

	A	C	D
A	2	8	∞
C	9	1	4
D	∞	2	3

Node C

	A	B	D
A	7	3	∞
B	9	1	4
D	∞	4	1

Node D

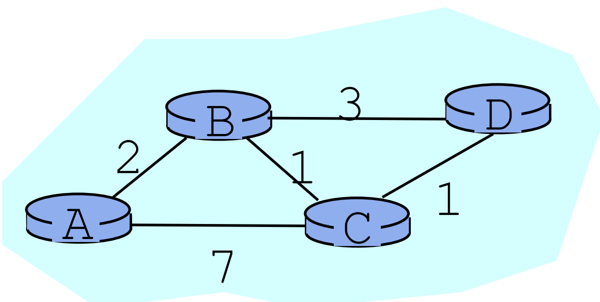
	B	C
A	5	8
B	3	2
C	4	1

Assume all send messages at same time

End of 1^{st} Iteration: All nodes know the best two-hop paths

Example: Now A ser

What harm does this cause?



Node A

	B	C
B	2	8
C	3	7
D	5	8

Node

		C	D
A		8	∞
C	5	1	4
D	7	2	3

$D_A(B, C) = D_A(B,A) + D(A, C) = 2 + 3 = 5$

$D_A(B, D) = D_A(B,A) + D(A, D) = 2 + 5 = 7$

loop:

…

else if (update D(*B, Y*) from *A*)

  $D_A(B,Y) = D_A(B,A) + D(A, Y)$;

if (new min. for destination Y)

  send D(*B, Y*) to all neighbors

forever

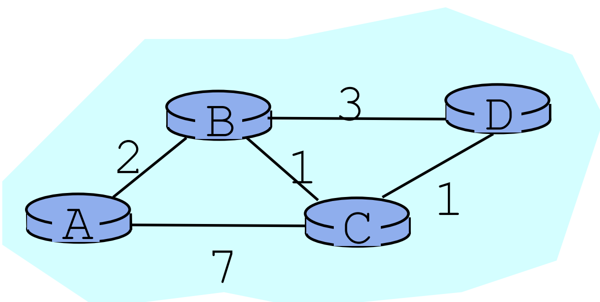
Node C

	A	B	D
A	7	3	∞
B	9	1	4
D	∞	4	1

Node D

	B	C
A	5	8
B	3	2
C	4	1

Example: End of 2nd Full Exchange



Node A

	B	C
B	2	8
C	3	7
D	4	8

Node B

	A	C	D
A	2	4	8
C	5	1	4
D	7	2	3

Node C

	A	B	D
A	7	3	6
B	9	1	3
D	12	3	1

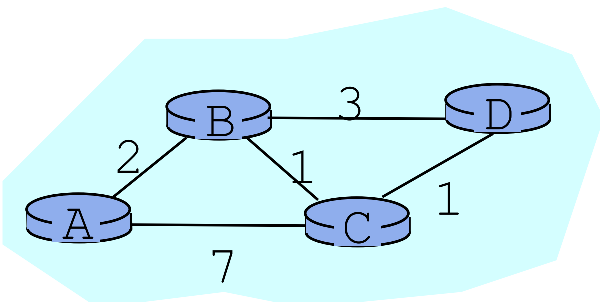
Node D

	B	C
A	5	4
B	3	2
C	4	1

Assume all send messages at same time

End of 2nd Iteration: All nodes knows the best three-hop paths

Example: End of 3rd Full Exchange



Node A

	B	C
B	2	8
C	3	7
D	4	8

Node B

	A	C	D
A	2	4	7
C	5	1	4
D	6	2	3

Assume all send messages at same time

Node C

	A	B	D
A	7	3	5
B	9	1	3
D	11	3	1

Node D

	B	C
A	5	4
B	3	2
C	4	1

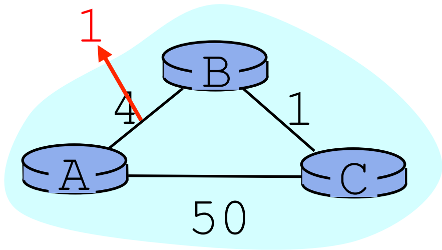
What route does this 11 represent?

Intuition

- Initial state: best one-hop paths
- One simultaneous round: best two-hop paths
- Two simultaneous rounds: best three-hop paths
- …
- Kth simultaneous round: best (k+1) hop paths

- Must eventually converge
  - as soon as it reaches longest best path
- …..but how does it respond to changes in cost?

DV: Link Cost Changes



“good news travels fast”

“bad news travels slowly”

Link State vs. Distance Vector

- Message Complexity:
  - LS: O(|N| |E|)
  - DV: O(|E| k), where k is the number of iterations
- Speed of convergence:
  - LS: $O(|N|^2)$
  - DV: slow in case of failures and can have routing loops while converging
- Robustness: what happens if router malfunctions?
  - LS: Each node computes only its own table. An advertised incorrect link cost might not impact all nodes
  - DV: Each node's table is used by other nodes; errors propagate through the entire network
- Privacy: LS offers no privacy---global sharing of information

Inter-domain Routing

- Issues of autonomy escalate in inter-domain routing

- AS want freedom to pick routes based on policy
  - *"My traffic can't be carried over my competitor's network"*
  - *"I don't want to carry A's traffic through my network"*
  - Not expressible as Internet-wide "shortest path"!

- AS want autonomy
  - Want to choose their own internal routing protocol
  - Want to choose their own policy

- AS want privacy
  - Other AS are business rivals
  - Choice of network topology, routing policies, etc.

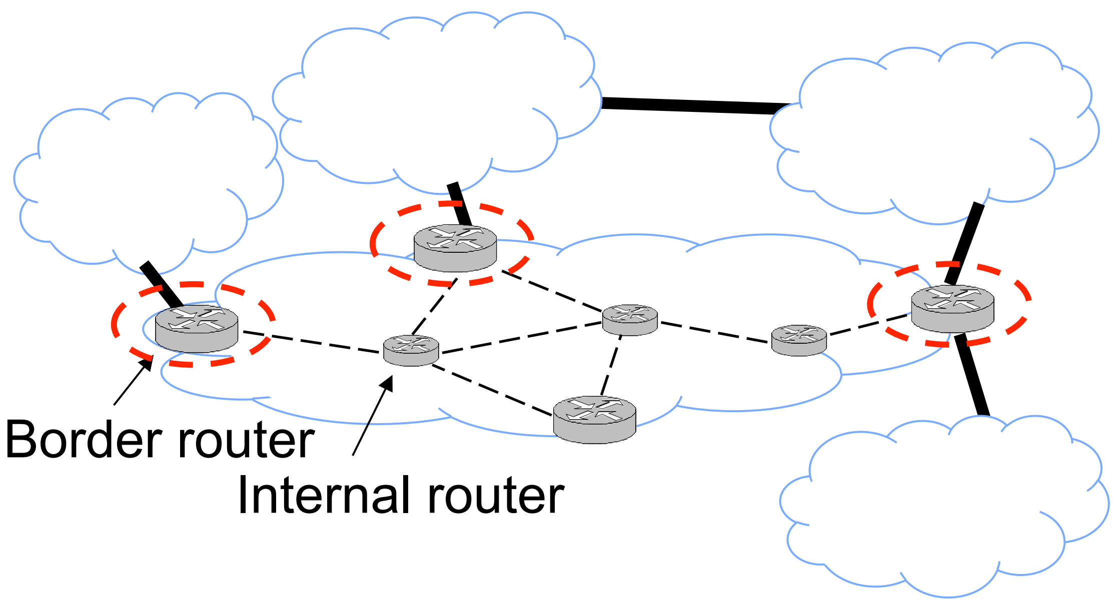
Business Relationships

- AS topology reflects business relationships between them

- Business relationships impact which routes are acceptable.
- Routing follows the money, not the shortest path

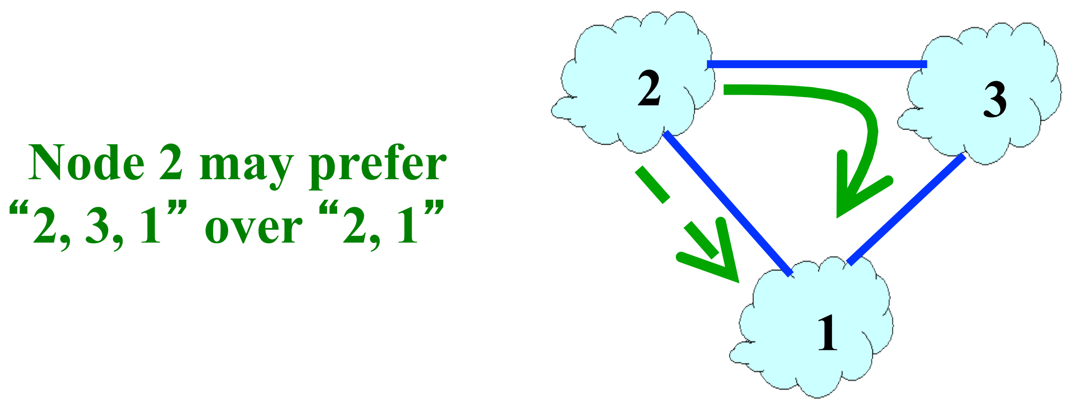
Border Gateway Protocol (BGP)

- Inspired by distance vector:
  - An AS advertises its best routes to one or more IP prefixes
  - Each AS selects the “best” route it hears advertised for a prefix
  - No global sharing of network topology information
  - Iterative and distributed convergence on paths
- Implemented at border routers of each domain

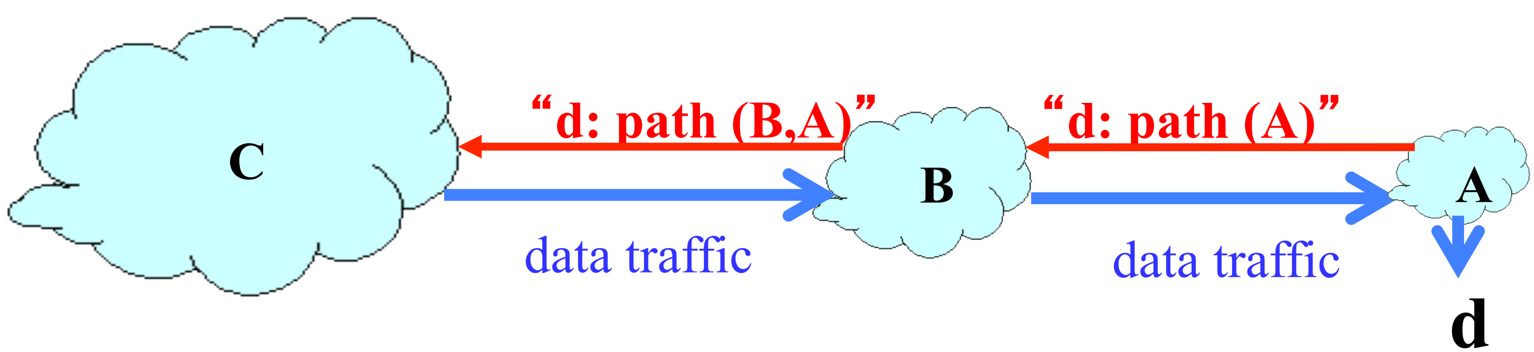


Differences between BGP and DV

- Best-policy path: BGP selects the best route based on policy, not shortest distance (least cost)

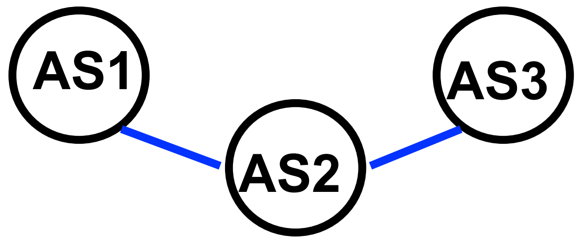


- Path-vector routing: To avoid loops, BGP advertises the entire path to destination rather than just the cost



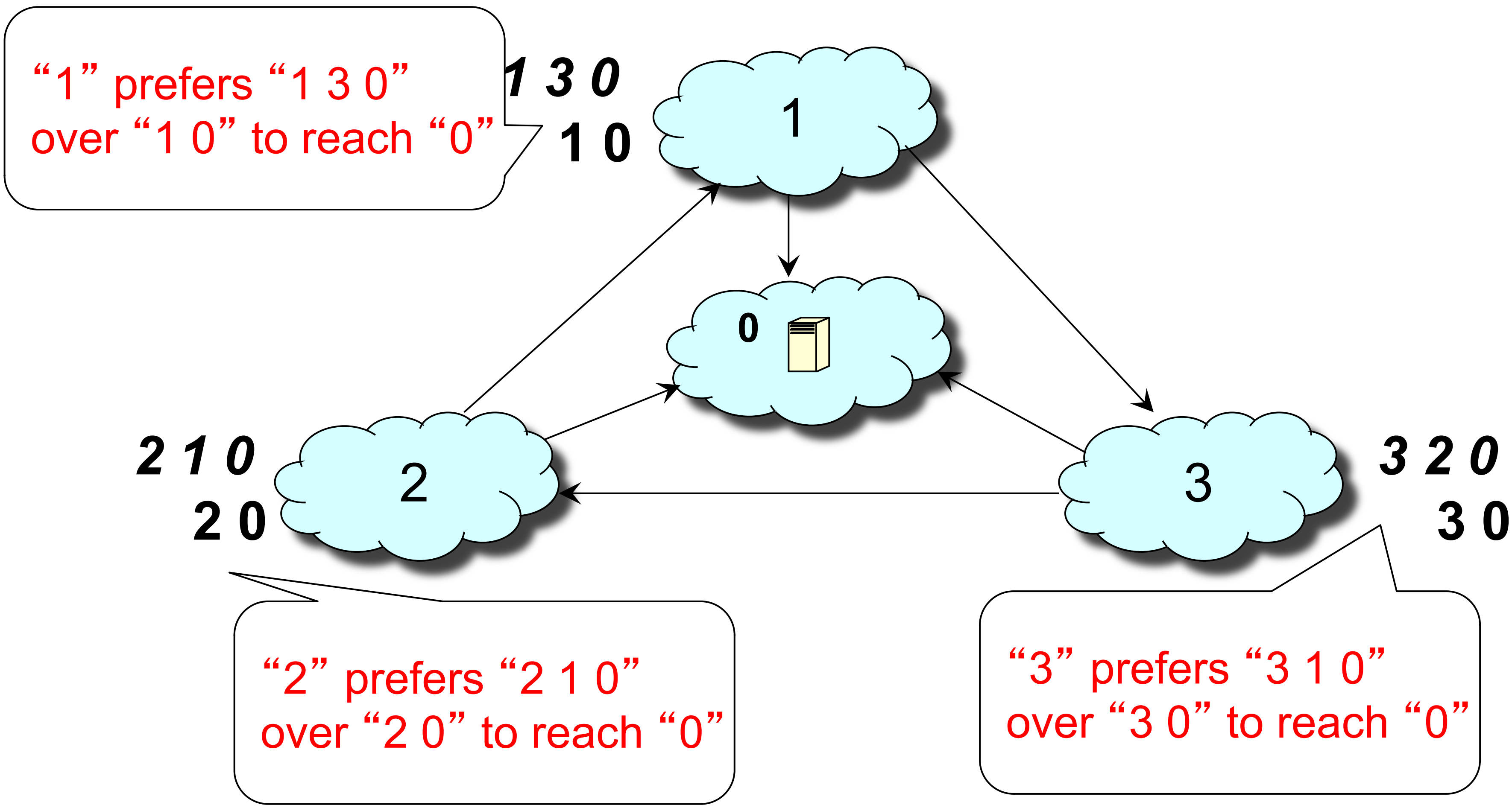
Differences between BGP and DV

- Selective routing: For policy reasons, an AS may choose not to advertise a route to a destination



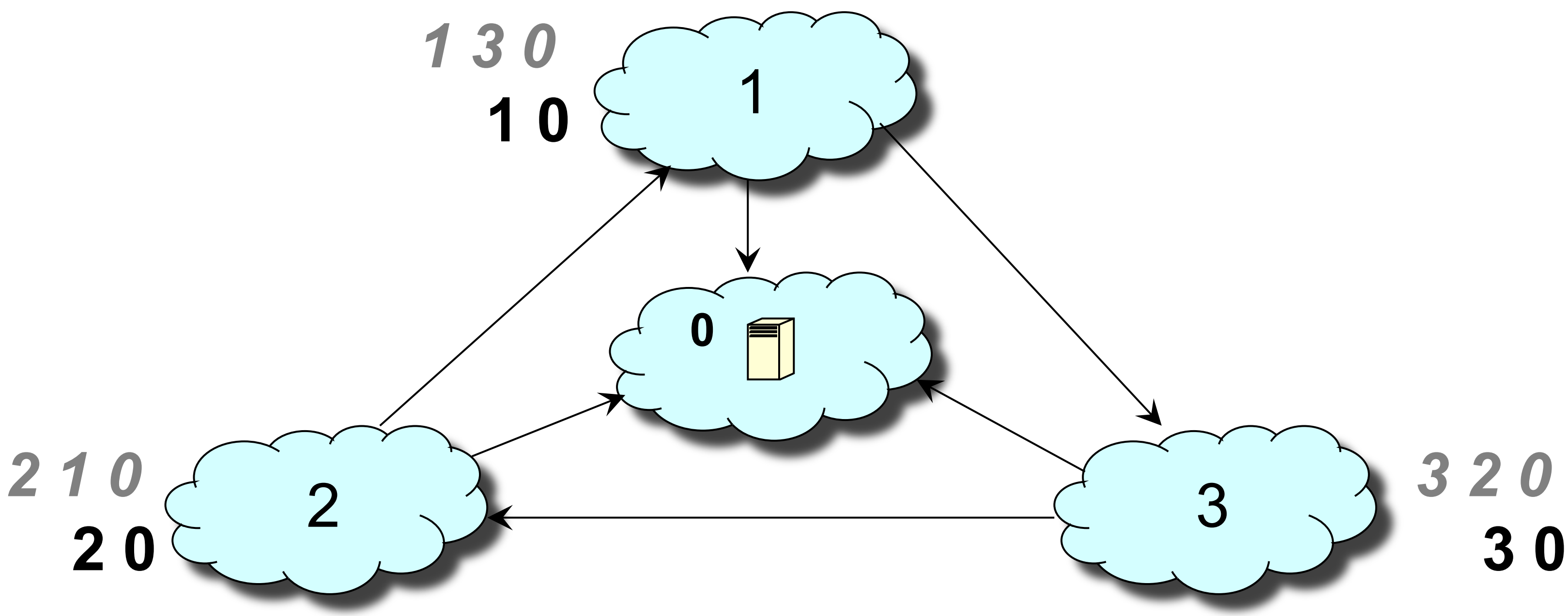
Example: AS2 does not want to carry traffic between AS1 and AS3

Example of Policy Oscillation



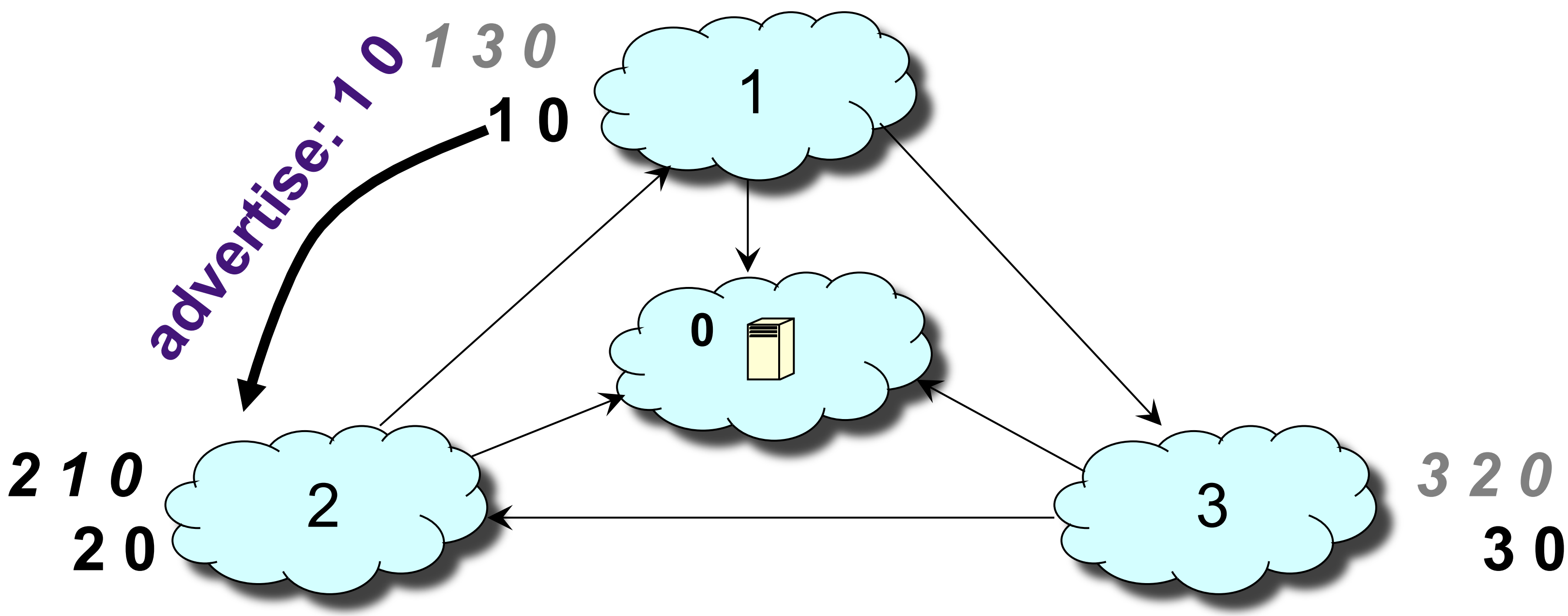
Step-by-Step of Policy Oscillation

Initially: Nodes 1, 2, 3 know only shortest path to 0

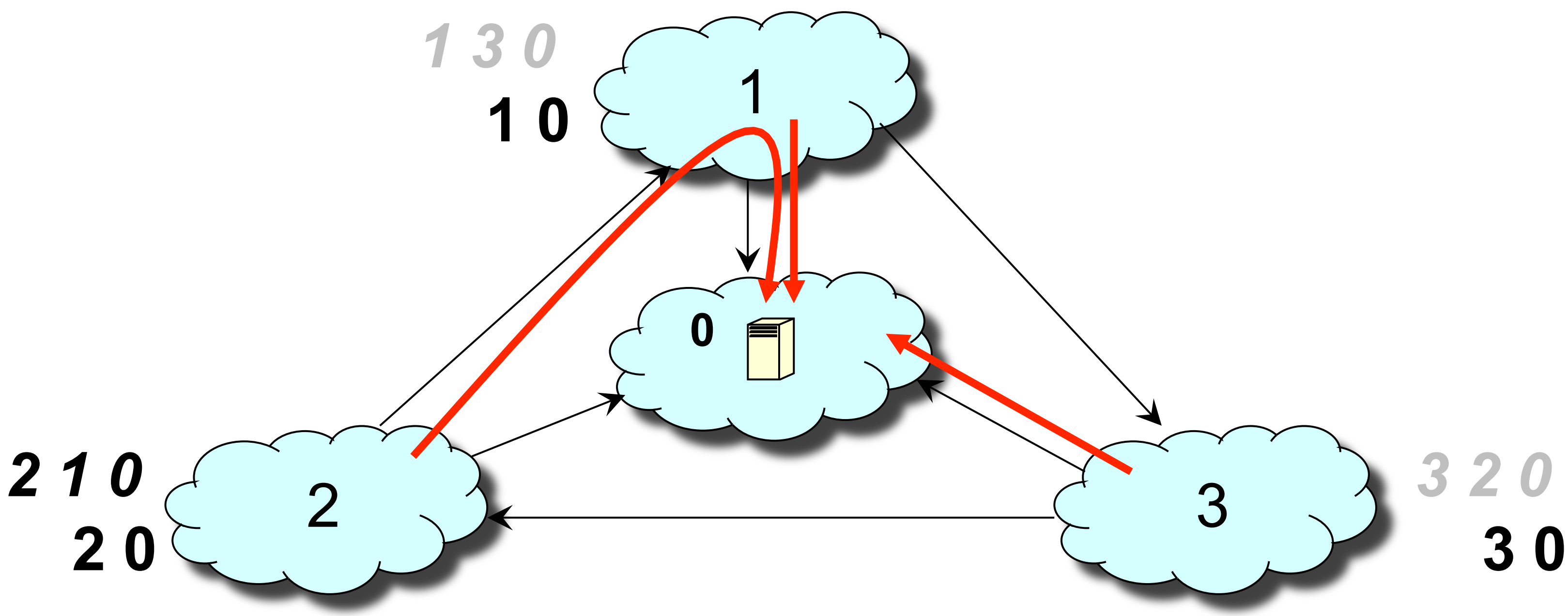


Step-by-Step of Policy Oscillation

1 **advertises** its path 1 0 to 2

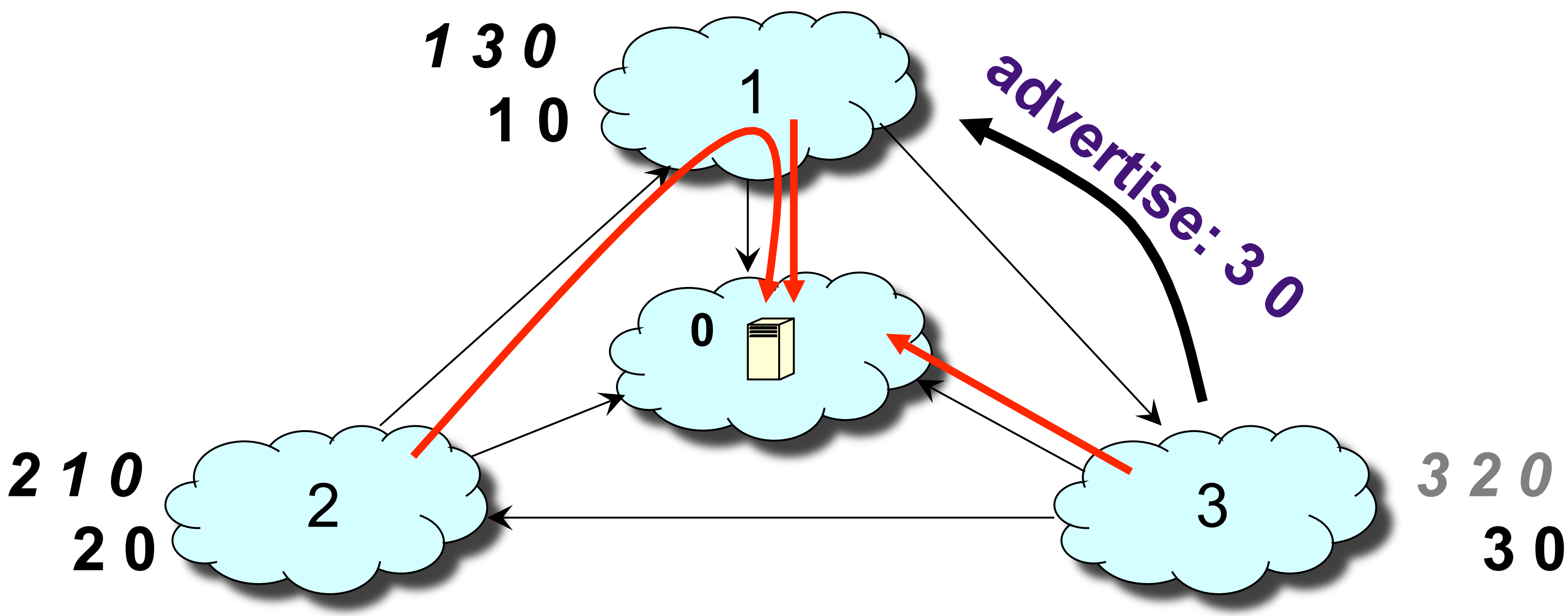


Step-by-Step of Policy Oscillation

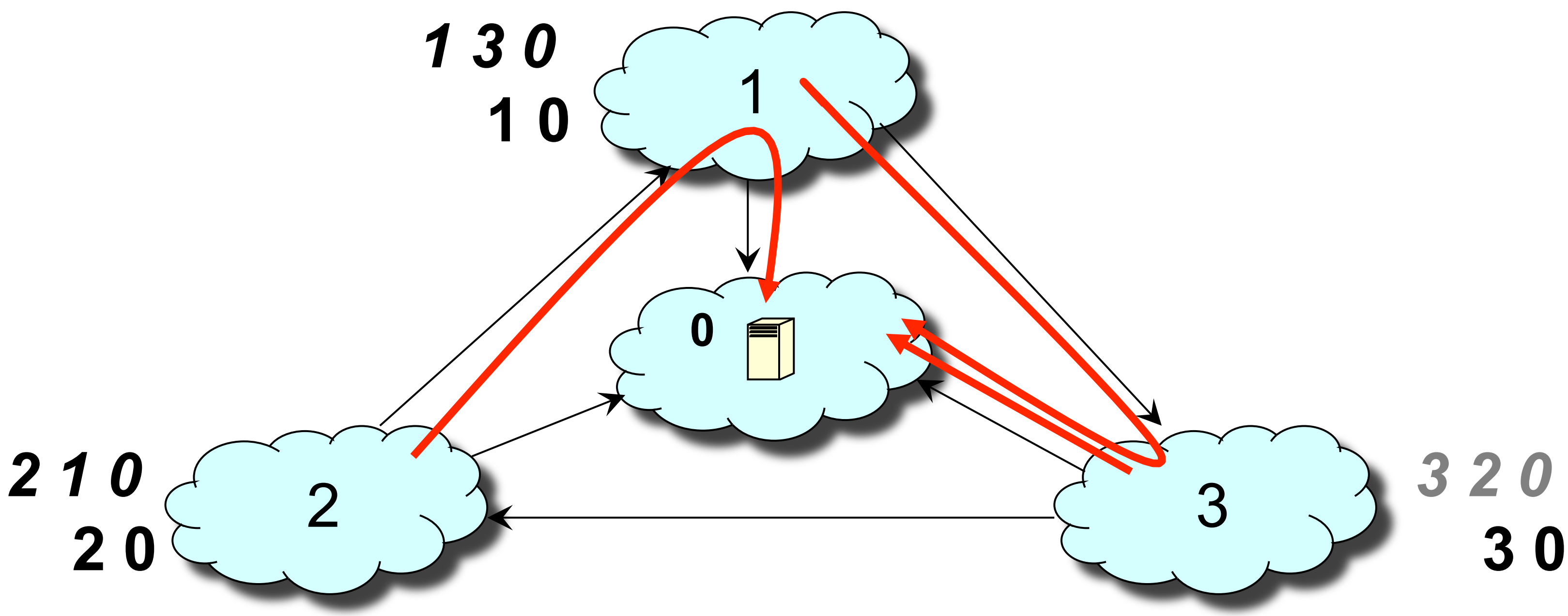


Step-by-Step of Policy Oscillation

3 **advertises** its path 3 0 to 1

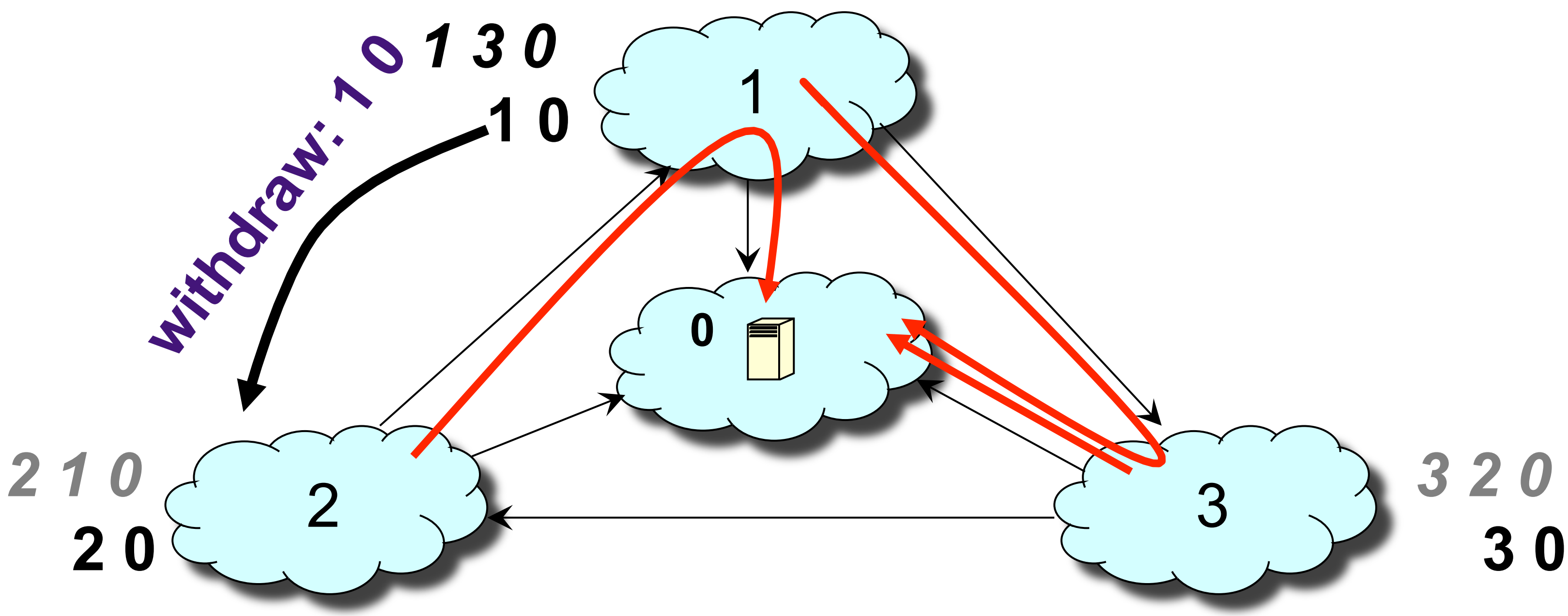


Step-by-Step of Policy Oscillation

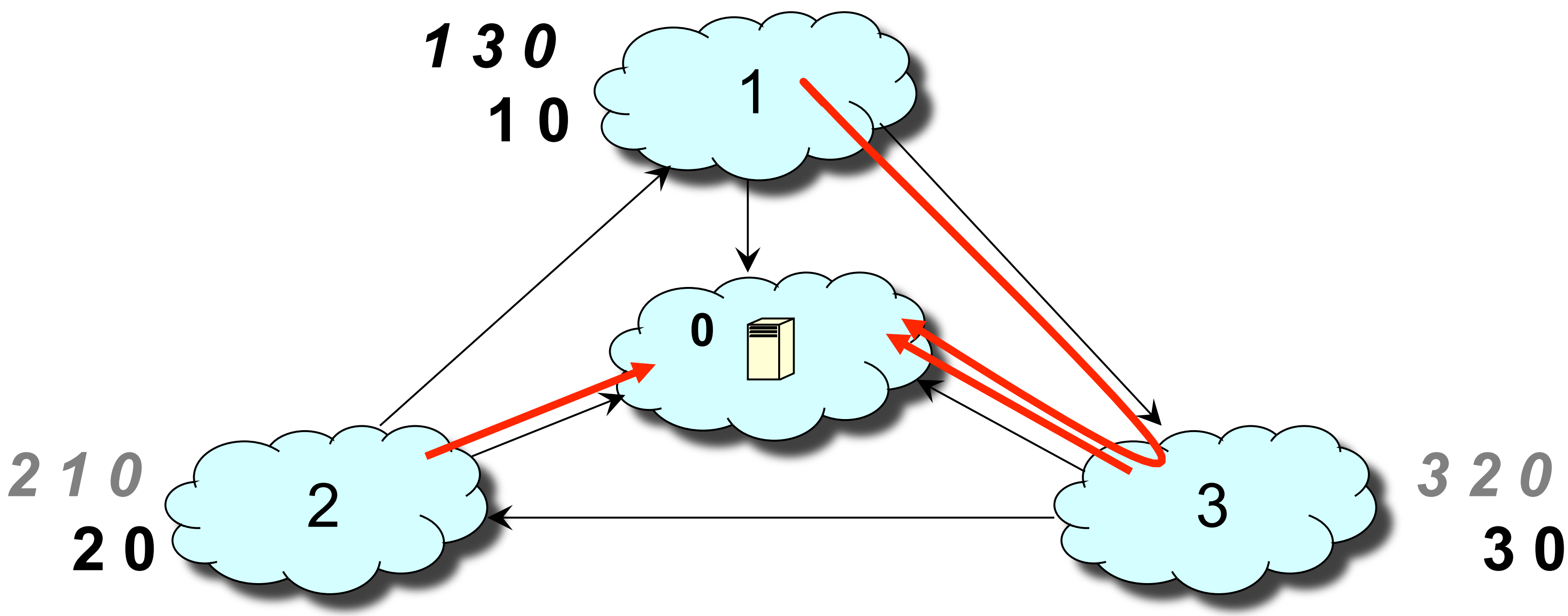


Step-by-Step of Policy Oscillation

1 **withdraws** its path 1 0 from 2

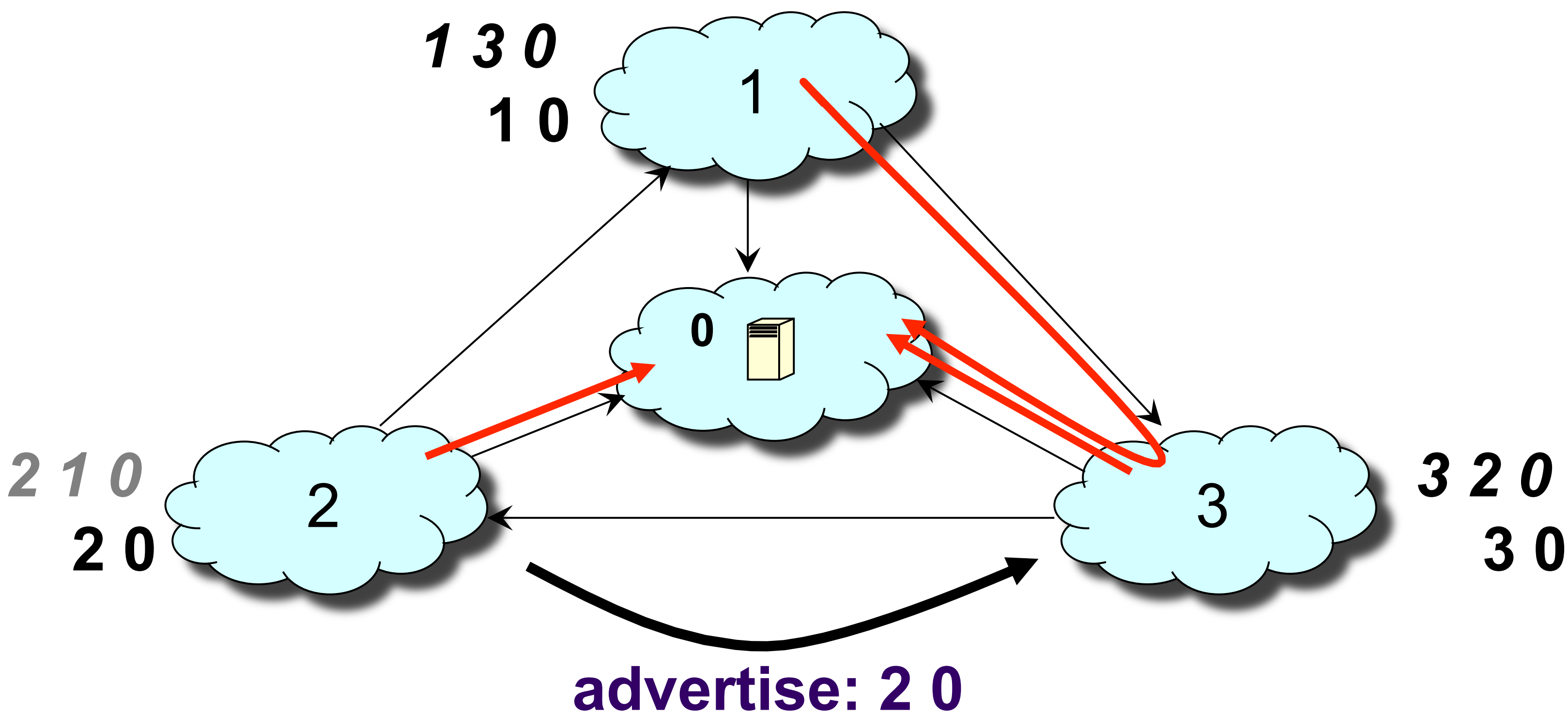


Step-by-Step of Policy Oscillation

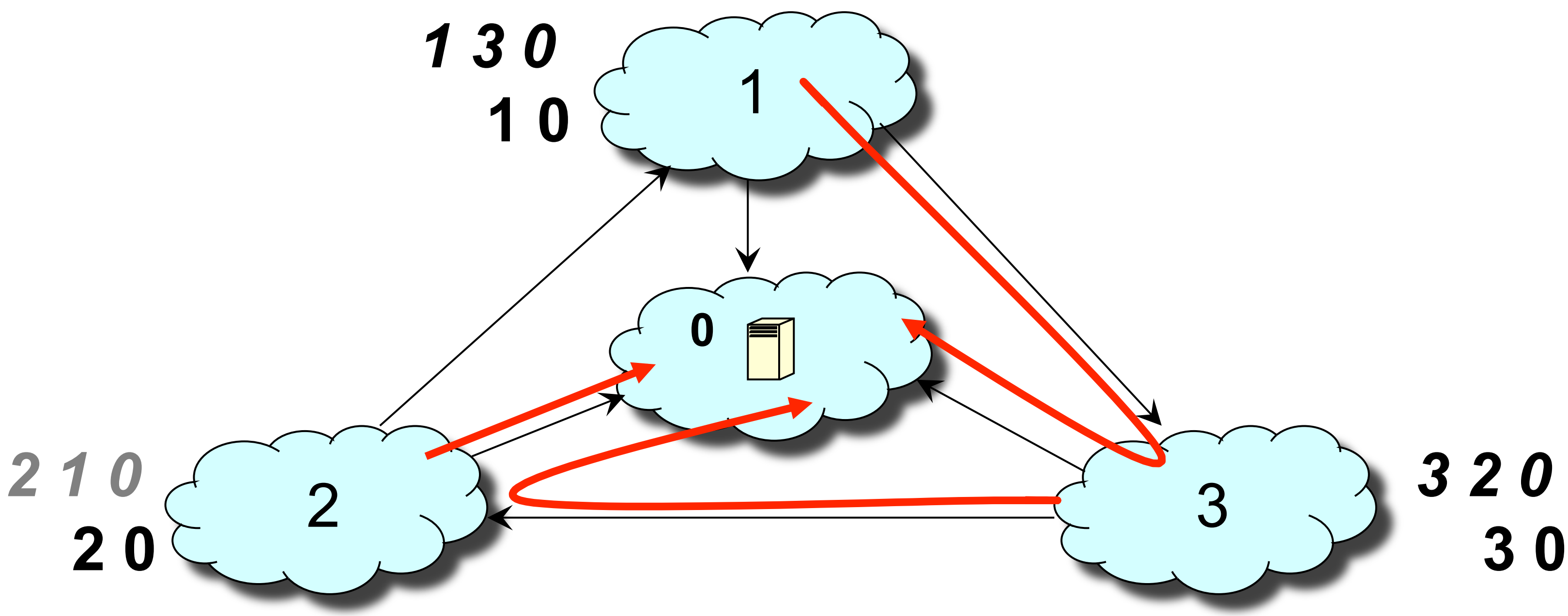


Step-by-Step of Policy Oscillation

2 **advertises** its path 2 0 to 3

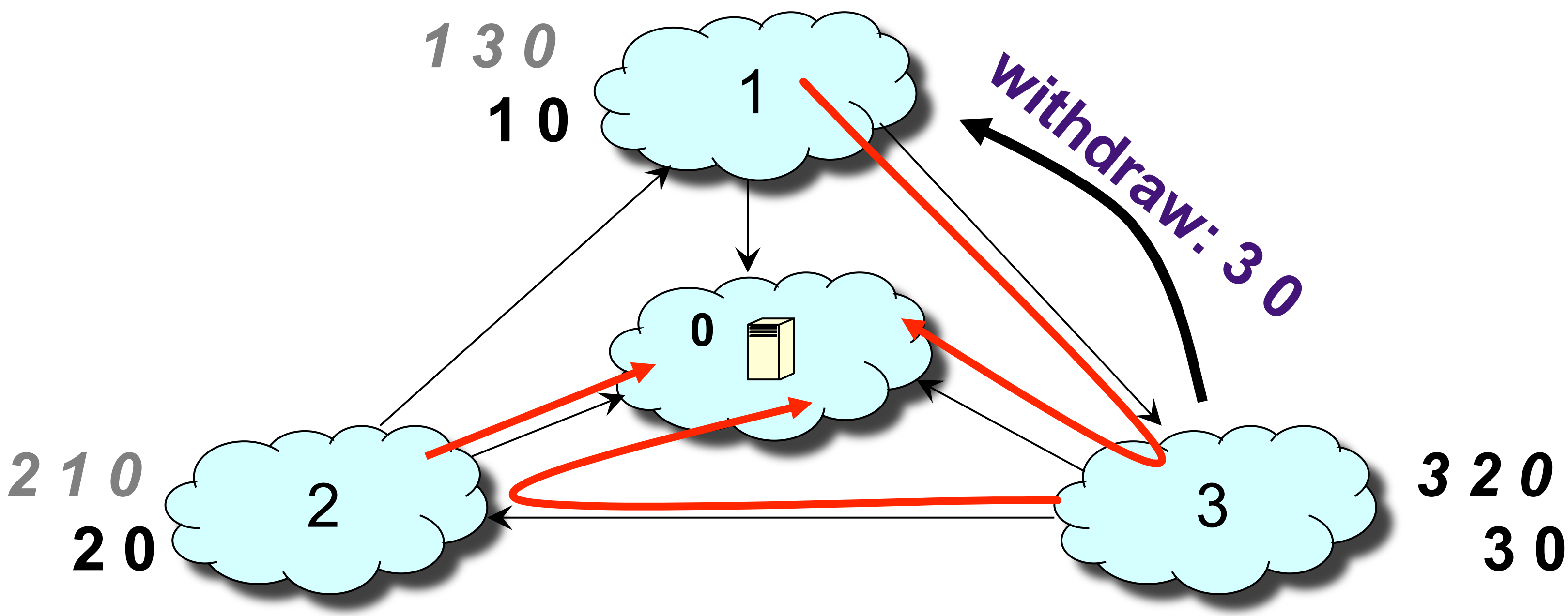


Step-by-Step of Policy Oscillation

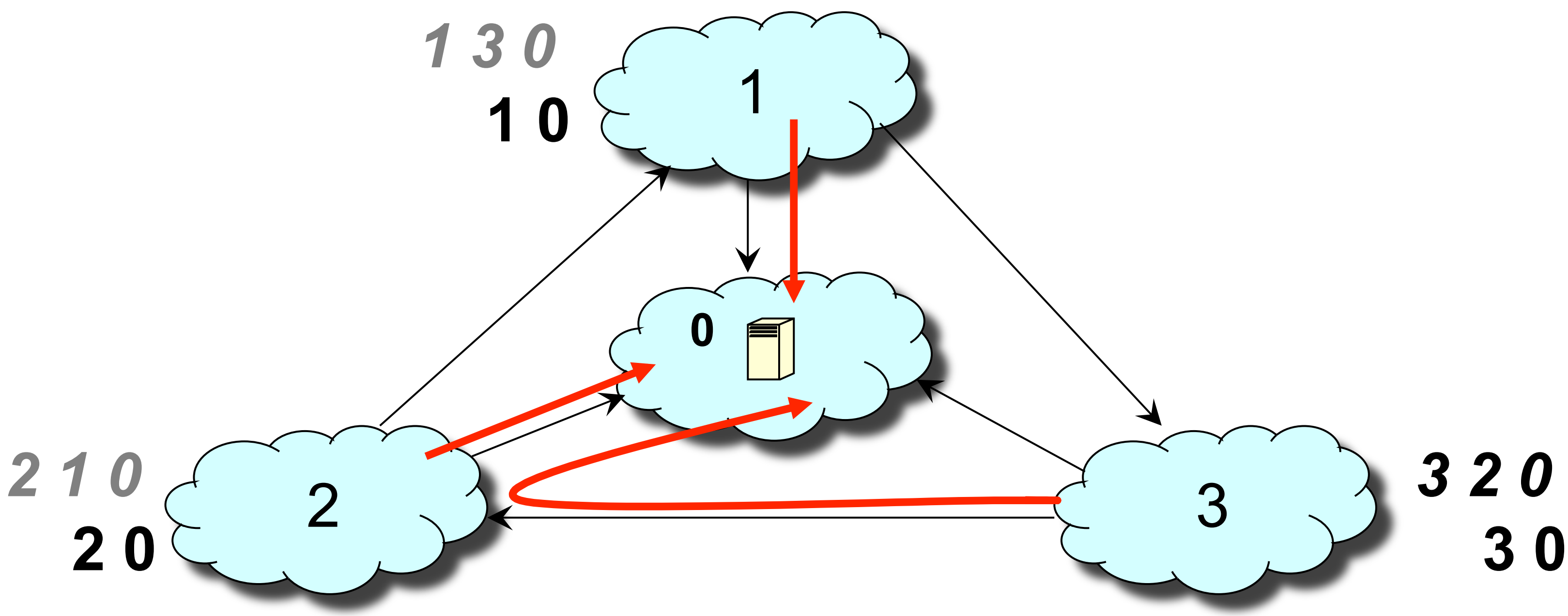


Step-by-Step of Policy Oscillation

3 **withdraws** its path 3 0 from 1

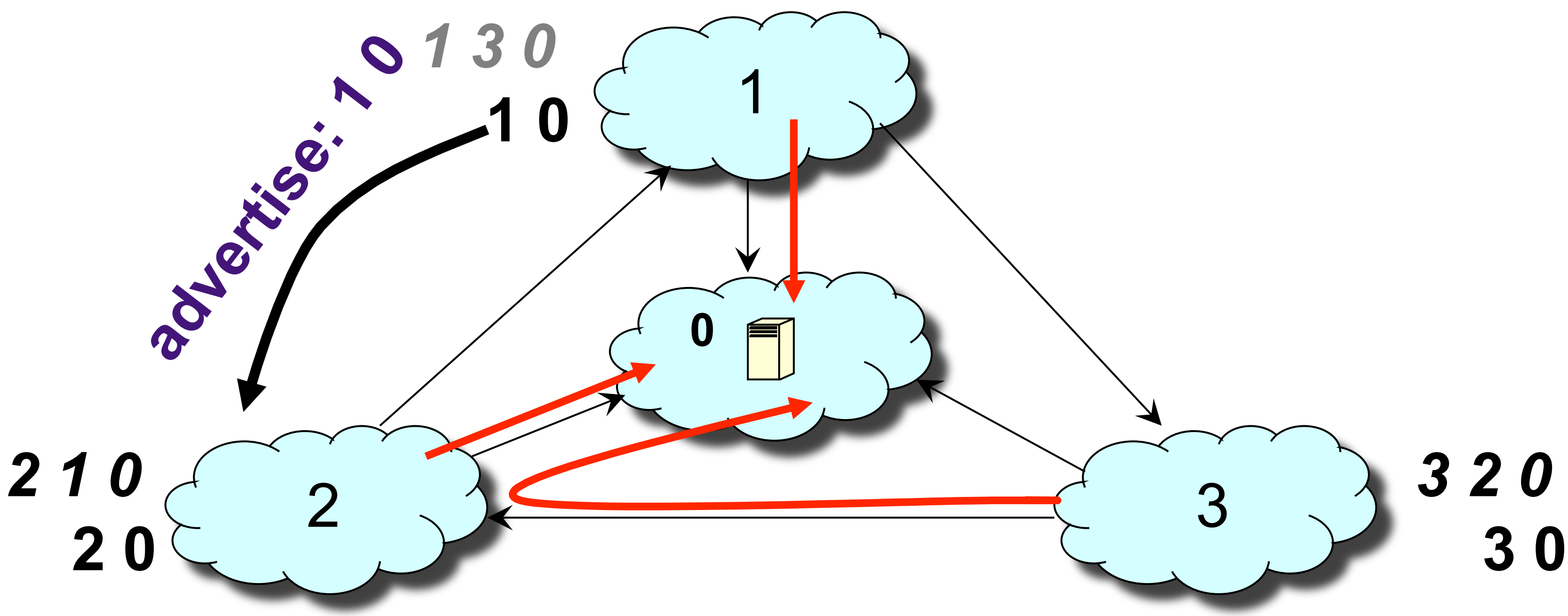


Step-by-Step of Policy Oscillation

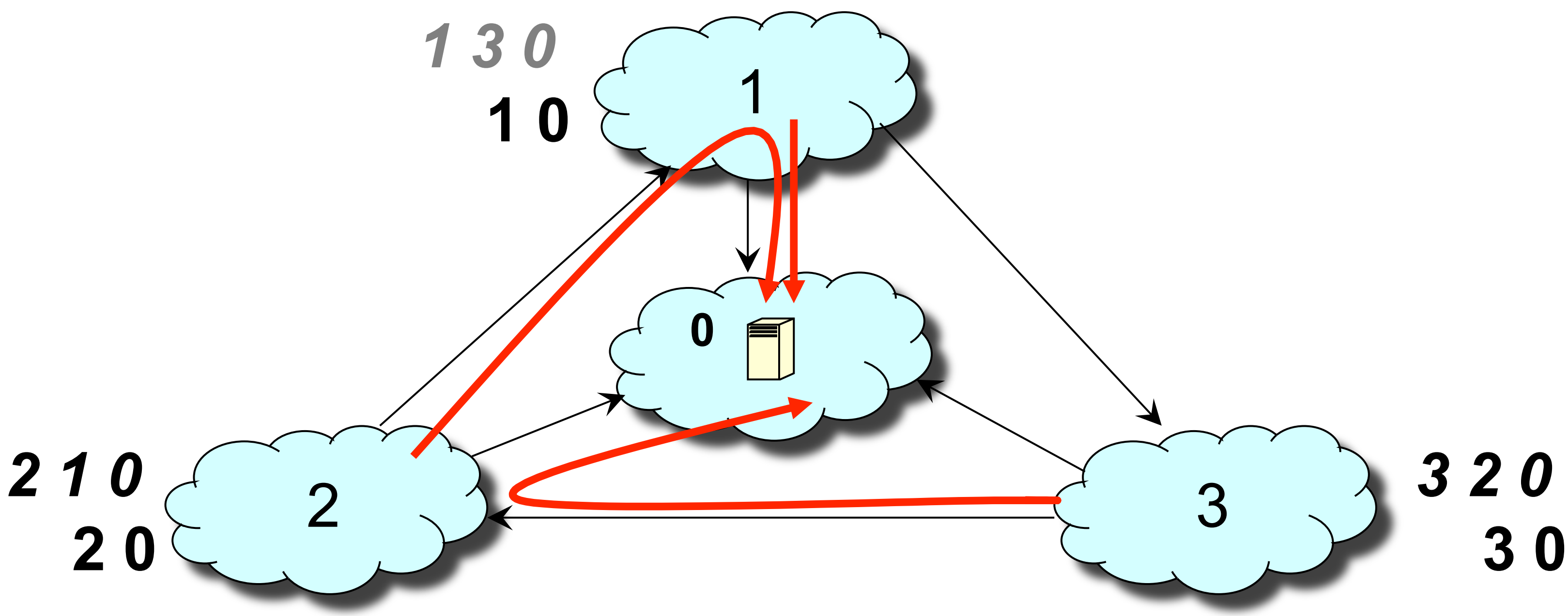


Step-by-Step of Policy Oscillation

1 **advertises** its path 1 0 to 2

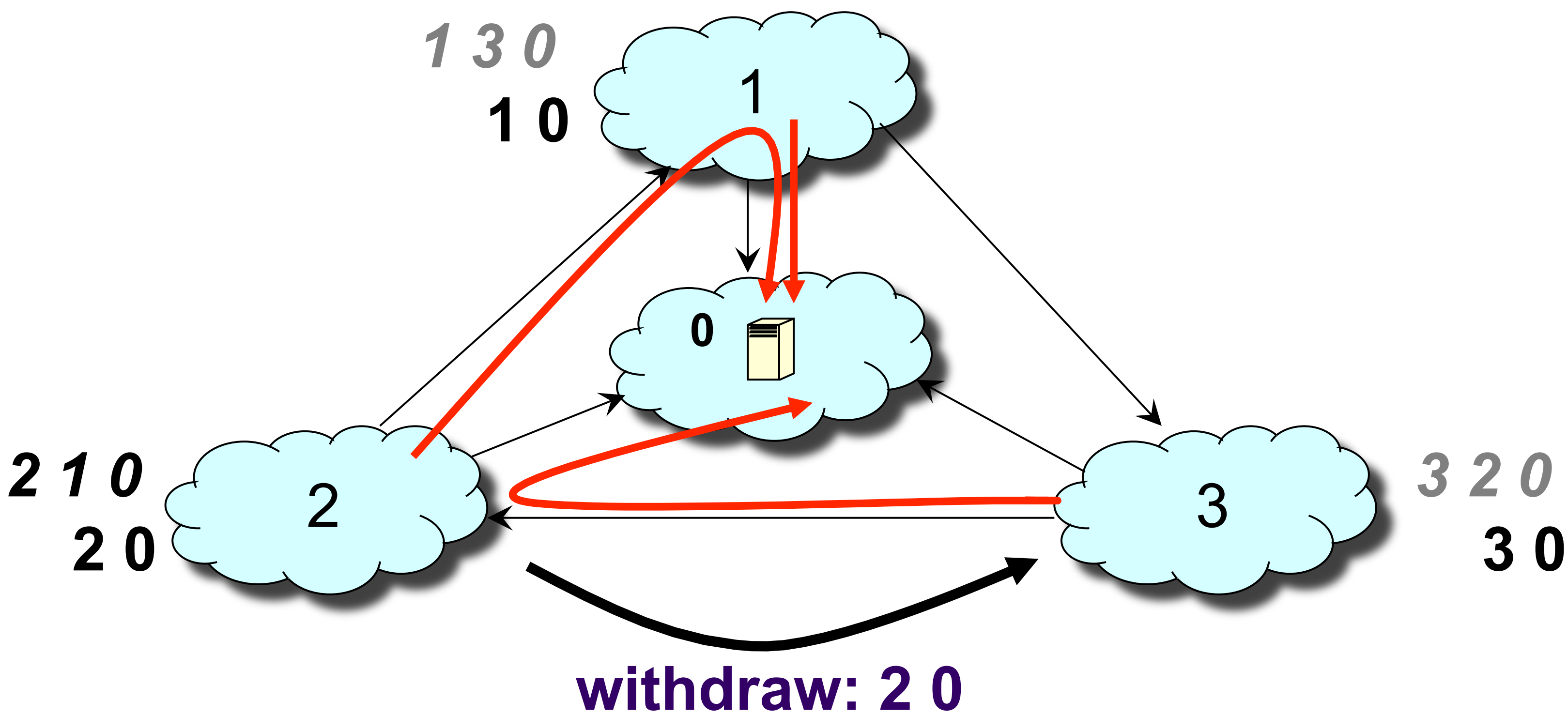


Step-by-Step of Policy Oscillation



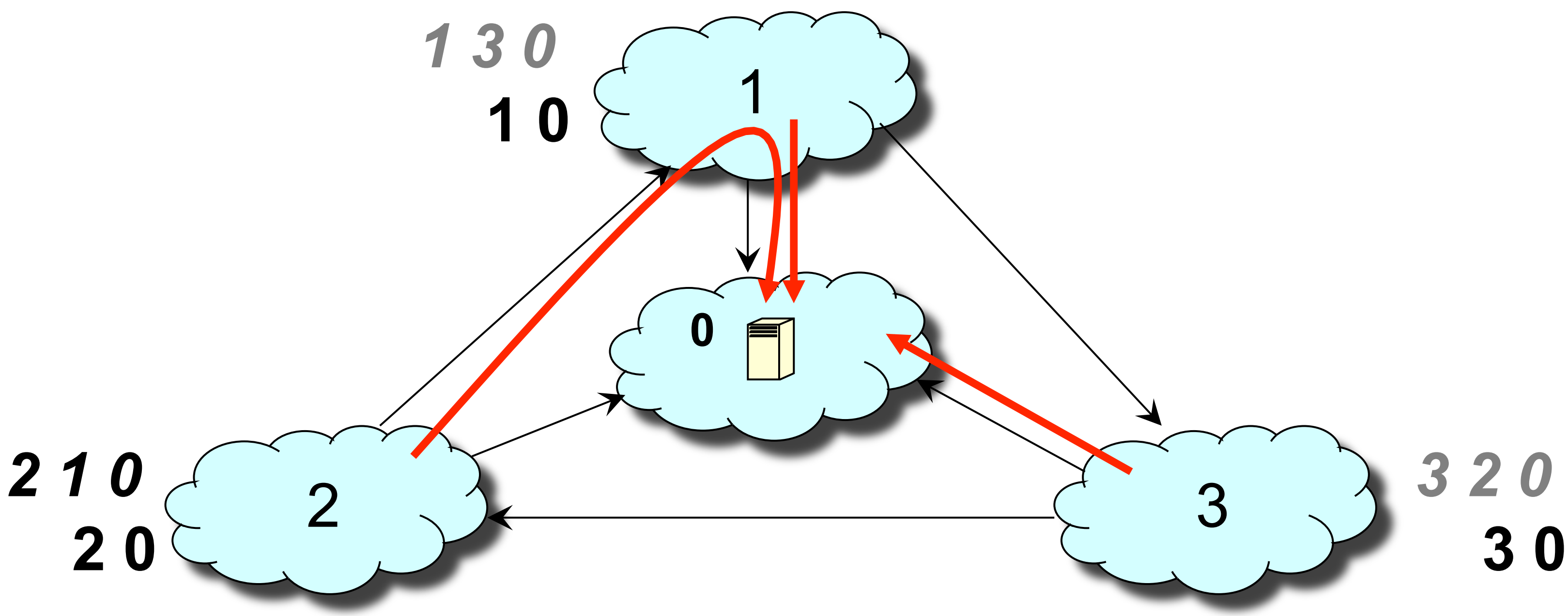
Step-by-Step of Policy Oscillation

2 **withdraws** its path 2 0 from 3



Step-by-Step of Policy Oscillation

- Back to the starting point



Performance Issue

- BGP outages are the biggest source of Internet problems
- BGP protocol is both bloated and underspecified
  - Lots of leeway in how to set and interpret attribute values, route selection rules, etc.
  - Necessary to allow autonomy, diverse policies
  - But also gives operators plenty of rope
- Much of this configuration is manual and *ad hoc*
- And the core abstraction is fundamentally flawed
  - per-router configuration to effect AS-wide policy
  - Strong industry interest in changing this!