CS-340 Introduction to Computer Networking

Lecture 11: BGP Routing

Steve Tarzia

Last Lecture: Router internals & Routing algorithms

- *Weighted Fair Queueing* can prioritize classes of packets in router queue.
- *Routing algorithms* determine each router's forwarding table. It's a a *shortest path* problem on the *weighted graph* graph representing the network.
  - May be *centralized/global* or *distributed.*
- *Dijkstra's Algorithm* is a fast *centralized* (LS) algorithm for shortest path.
  - Used by *Open Shortest Path First (OSPF)* protocol within an AS.
  - Routers initially *flood/broadcast* local link information to entire network.
  - Each router then solves shortest path from itself to all other routers.
- *Distance Vector (DV)* algorithm is a *distributed* shortest path algorithm
  - Used by the *Border Gateway Protocol (BGP)* to route between AS's.
  - Initially, routers only knows distance to neighbors – broadcast to neighbors.
  - When receive a neighbor's DV, update own DV, & broadcast if DV changed.

What's in a Distance Vector?

- Each network node has its own DV.
- For that node, DV lists the ⟨**cost**, **next_hop**⟩ for every **destination**.
    - Ie., it's a dictionary mapping from *destination* to ⟨*cost*, *next_hop*⟩.

How is the DV used?

- The *next_hop* tells which direction to travel to reach the destination most quickly. At each hop, check the node's DV to find the next hop.
- When we share our DV with a neighbor, the *cost* lets the neighbor decide its best next_hop for that destination.

Distance Vector (DV) calculation

- Each node operates independently (a distributed algorithm).

Each node must store:

1. Its **outbound links**: who do they connect to, and at what cost.
2. **Each neighbor's latest DV**: who can I reach through neighbors?

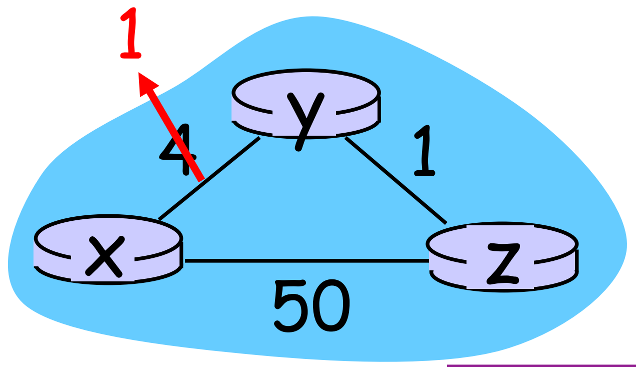
Each node must compute:

- Its own DV using 1 & 2. This will be optimal, given the info I have.
- Whenever either 1 or 2 changes, I must recompute my DV.
- Whenever my DV changes, send my updated DV to all neighbors.

When DV messages are no longer sent, we have converged to an optimal solution. Each node's DV is an optimal **forwarding table** for routing.

What happens when link costs change?

- Node detects local link cost change
- Recalculates distance vector
- If DV changed, notify neighbors

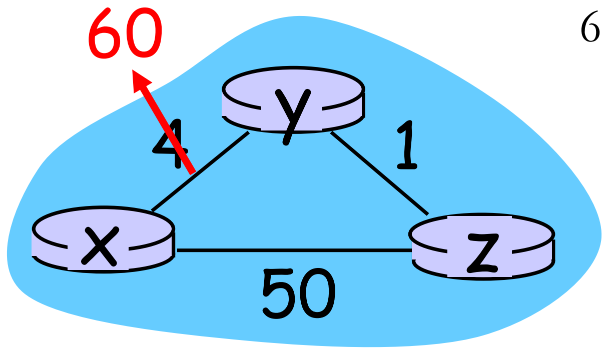


What if my DV does not change?

Link cost was decreased – *Good news travels quickly*

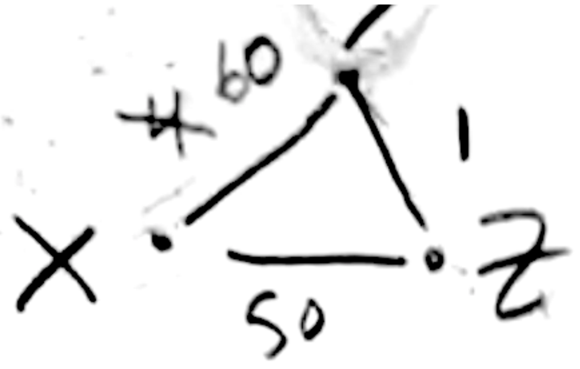
1. *y* detects new link-cost, updates its DV, sends new DV to neighbors.
2. *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.
3. *y* receives *z*'s update, updates its distance table. *y*'s least costs do *not* change, so *y* does *not* send a message to *z*.

Bad news travels slowly



- When link costs are **increased**, updates can be slow.
- In this example, it will take 44 iterations for the algorithm to re-converge to the correct solutions.
- **Count to infinity** problem:
  - Neighbors think that the other still has a good path and keep trying to route through each other, meanwhile each node's estimated distance slows increases.

Count to Infinity Demo



fore :

$$DV_x = \begin{cases} X : 0, \emptyset \\ Y : 4, Y \\ Z : 5, Y \end{cases}$$

$$DV_y = \begin{cases} X : 4, X \\ Y : 0, \emptyset \\ Z : 1, Z \end{cases}$$

$$DV_z = \begin{cases} X : 5, Y * \\ Y : 1, Y \\ Z : 0, \emptyset \end{cases}$$

*problem!

link changes!

Y recalculates its DV

direct costs $\begin{cases} Z : 1 \\ X : 60 \end{cases}$

$$DV'_y = \begin{cases} X : 6, Z * \\ Y : 0, \emptyset \\ Z : 1, Z \end{cases}$$

Z recalculates DV_z

direct costs : $\begin{cases} X : 50 \\ Y : 1 \end{cases}$

$$DV'_z = \begin{cases} X : 7, Y * \\ Y : 1, Y \\ Z : 0, \emptyset \end{cases}$$

Count to Infinity solutions

- Can you think of a way to avoid *count to infinity*?
- **Poisoned reverse** solves this problem for pairs:
  - If Z routes through Y to get to X:
    Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z).
- But **routing loops** can still temporarily arise (involving ≥3 nodes).
- How can you avoid routing loops in general?
  - List entire path in DV and avoid routes that travel through me.
  - This is done in BGP with the AS_PATH.

Centralized (LS) *vs* Distributed (DV)

Message complexity:

Centralized (LS)	Distributed (DV)
• $\Theta(\|V\|\|E\|)$ messages sent to propagate complete graph info.	• Depends on convergence behavior

Convergence speed:

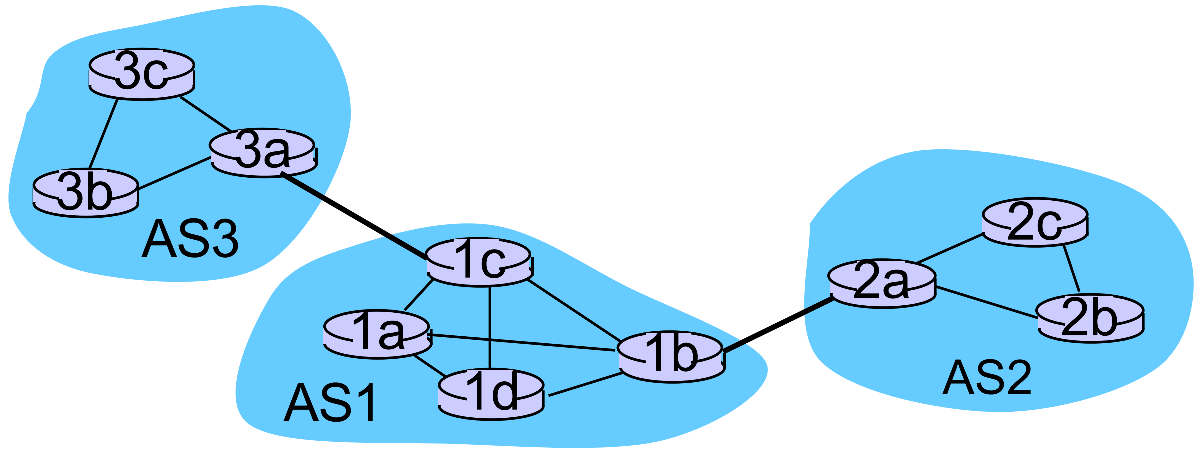
Centralized (LS)	Distributed (DV)
• $\Theta(\|E\|+\|V\|\log\|V\|)$	• Depends on convergence behavior
• "oscillations" are possible when costs change (like to count-to-inf.)	• Count-to infinity problem

Robustness to router malfunction:

Centralized (LS)	Distributed (DV)
• Incorrect *link cost* may be advertised.	• Incorrect *path cost* may be advertised.
• But this can be checked with cost advertised by other node on link.	• Bad router can attract traffic if it claims to be close to everywhere.

Hierarchical routing

- In reality, routers are not all equal. Network graph is ***hierarchical***.

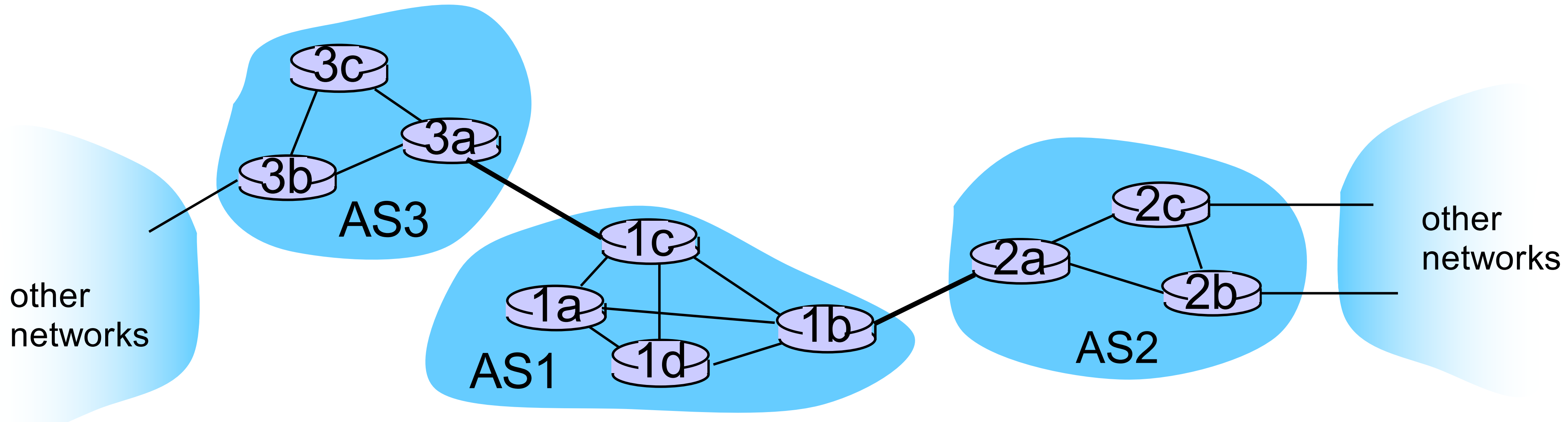


Why?

- ***Scaling***:
  - Too many routers (>100M) to solve one big shortest path problem.
- ***Administrative autonomy***:
  - Internet is a *network of networks*
  - Each network operator prefers to control its own *internal* routing policy.

Autonomous System (AS)

- The Internet is divided into about 100k **autonomous systems**.
  - Each has an **AS number**, distributed by the ICANN's regional authorities.
- **Gateway routers** *(border gateways)* at *edge* of the AS connect to other AS's.
- Routers in an AS run one *interior-gateway* (intra-AS) routing protocol.
- **Border Gateway Protocol (BGP)** is used by all border gateways to route traffic between AS's.



List of BGP autonomous systems is here:

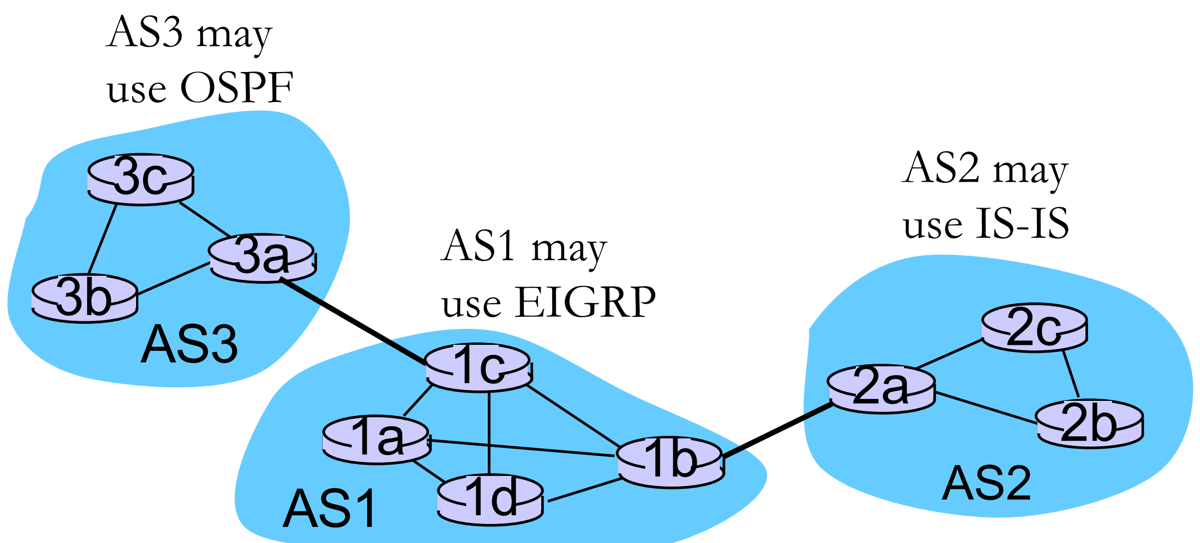
http://www.bgplookingglass.com/list-of-autonomous-system-numbers

Northwestern is AS #103

H-MART - Grandsuper Center Inc is AS #32401

Interior Gateway Protocols (Intra-AS routing)

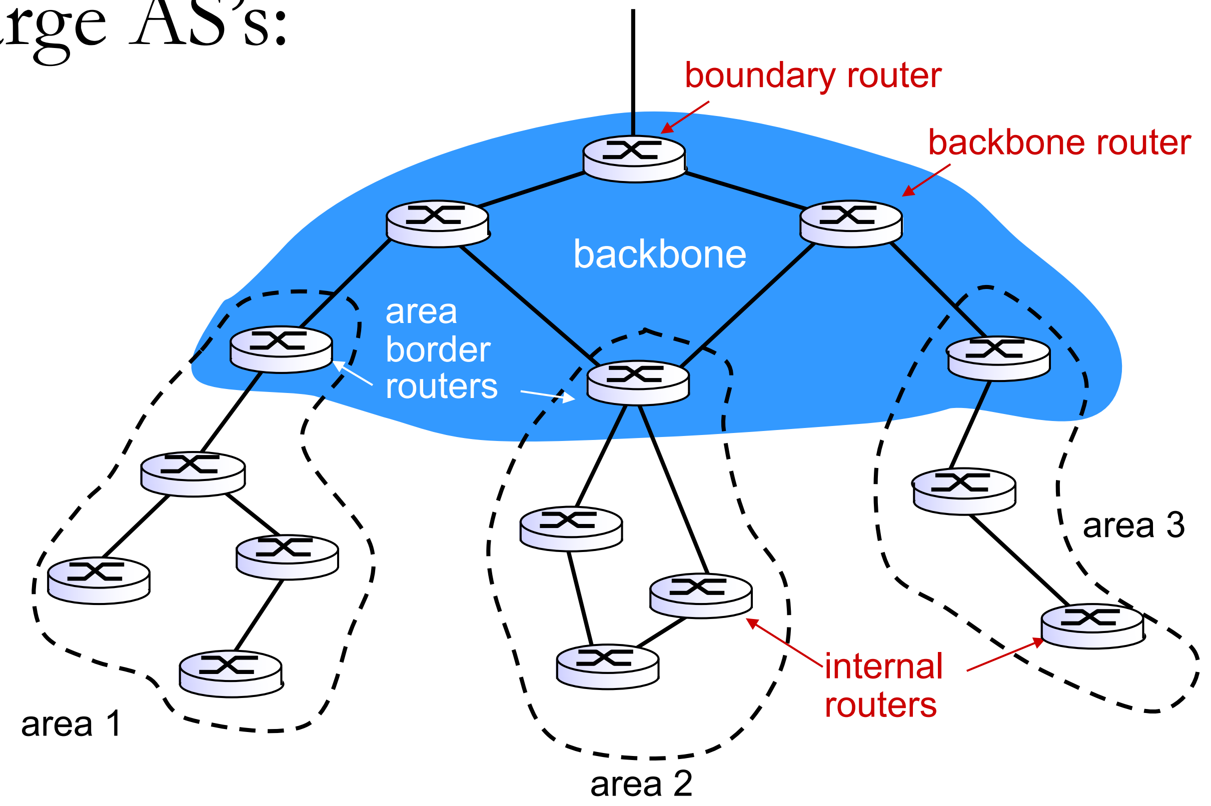
- Within an AS, network operator has full control.
  - A centralized or distributed routing algorithm can be used.
- Centralized IGPs (Link-State algorithms), using Dijkstra's algorithm:
  - *Open Shortest Path First (OSPF)* – "open" as in "open source"
  - *Intermediate System to Intermediate System (IS-IS)*
- Distributed IGPs (Distance-Vector algorithms)
  - *Routing Information Protocol (RIP)*
  - *Enhanced Interior Gateway Routing Protocol (EIGRP)*



OSPF highlights

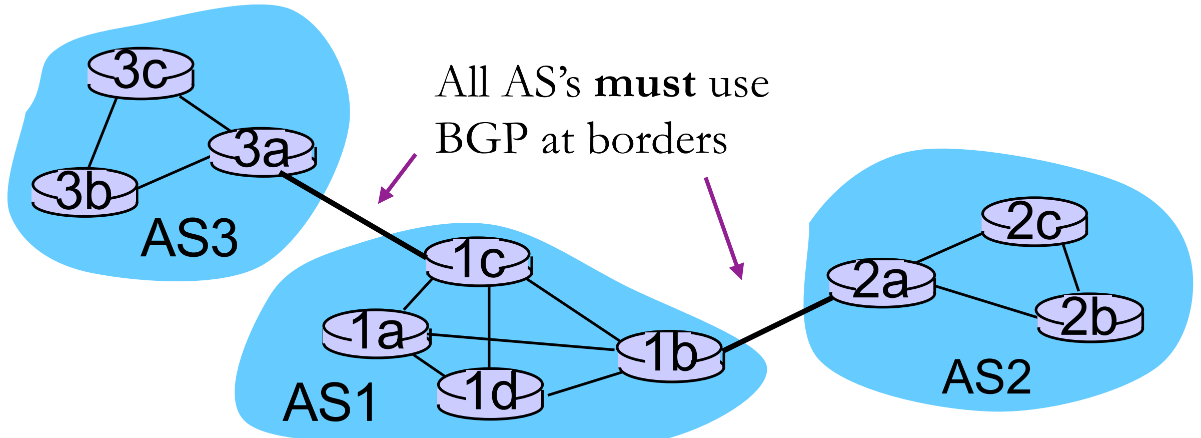
- Messages are authenticated to prevent malicious tampering.
- Multiple same-cost paths are allowed.
- Links can have different costs for different types of traffic
- *Hierarchical* OSPF can be used in large AS's:

- Generally, an AS can be divided into multiple private AS's internally, but look like one AS to outside Internet.



Border Gateway Protocol (Inter-AS routing)

- BGP is the Internet-standard routing protocol for connecting AS's
  - The ***glue that holds the Internet together***.
- eBGP: allows neighboring AS's to share their subnet reachability information: "I can reach 3.2.4.0/24 in 4 hops"
  - Prefix reachability table is the distance-vector that is shared and updated.
  - Eventually finds the fewest-AS-hop path to every subnet on Internet.
- iBGP: propagates reachability information to all AS-internal routers.



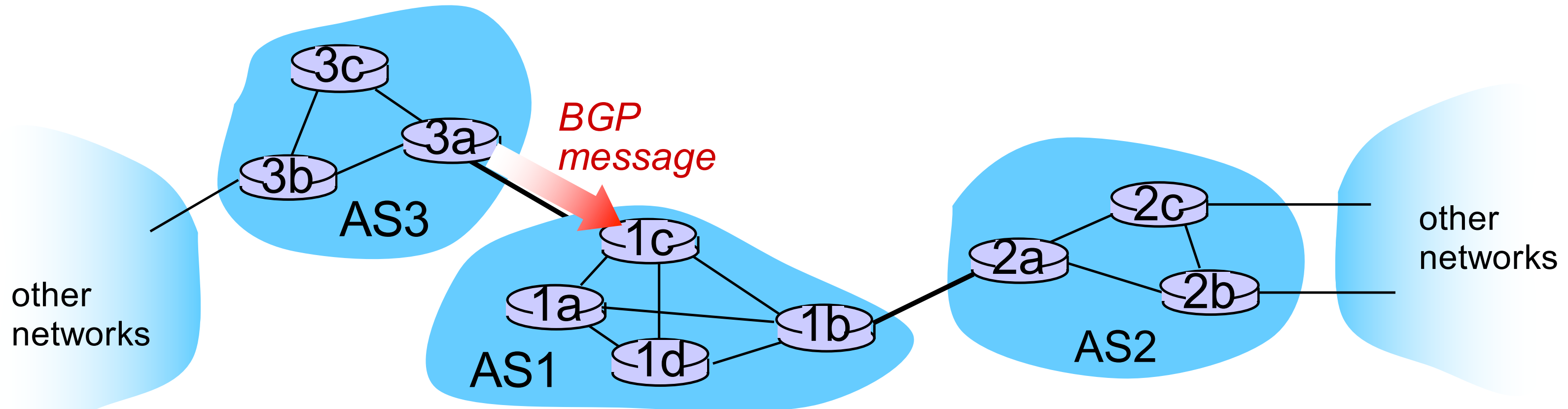
DV vs. BGP advertisement

$$\begin{array}{ccc} \left[\begin{array}{ccc} 0 & 4 & 1 \\ 1 & 7 & 2 \\ 2 & 8 & 1 \\ 3 & 10 & 1 \\ 4 & 3 & 2 \end{array}\right] \\ \begin{array}{ccc} \uparrow & \uparrow & \uparrow \\ \text{Destination} & \text{Path cost} & \text{Next Hop} \end{array} \end{array} \Rightarrow \begin{array}{c} \left[\begin{array}{ccc} 1.1.1.2/16 & [2,34,3] & 2.2.4.3 \\ 3.4.7.0/24 & [4,7] & 4.3.7.1 \\ \ldots & & \\ & & \end{array}\right] \\ \begin{array}{ccc} \uparrow & \uparrow & \uparrow \\ \text{Destination Subnet} & \text{AS\_PATH} & \text{Next Hop} \end{array} \end{array}$$

BGP Basics

- **BGP session:** two BGP routers ("peers") exchange BGP messages:
  - advertise paths to different destination network prefixes (network prefixes are the nodes/endpoints in this shortest-path problem)
  - exchanged over semi-permanent TCP connections
- When AS3 advertises a prefix to AS1:
  - AS3 promises it will forward packets towards that prefix
  - AS3 can aggregate prefixes in its advertisement:

20.1.**0**.0/24 + 20.1.**1**.0/24 + 20.1.**2**.0/24 + 20.1.**3**.0/24 = 20.1.0.0/**22**



BGP Advertisements (*DVs in practice*)

- Border gateways share lists of *BGP routes*, each has a prefix & attributes
- Two important route attributes:
  - **AS-PATH**: AS numbers through which the advertisement has passed. Indicates the AS-path that will be followed to reach the prefix.
  - **NEXT-HOP**: IP address of the router beginning the AS-PATH

{PREFIX: 43.5.0.0/16, AS-PATH: [AS4, AS65, AS1], NEXT-HOP: 5.6.7.200)}

- Above, a router in AS4 is advertising:
  - You can send traffic to 43.5.0.0/16 through my router 5.6.7.200, and it will travel through three AS's before arriving.

AS_PATH

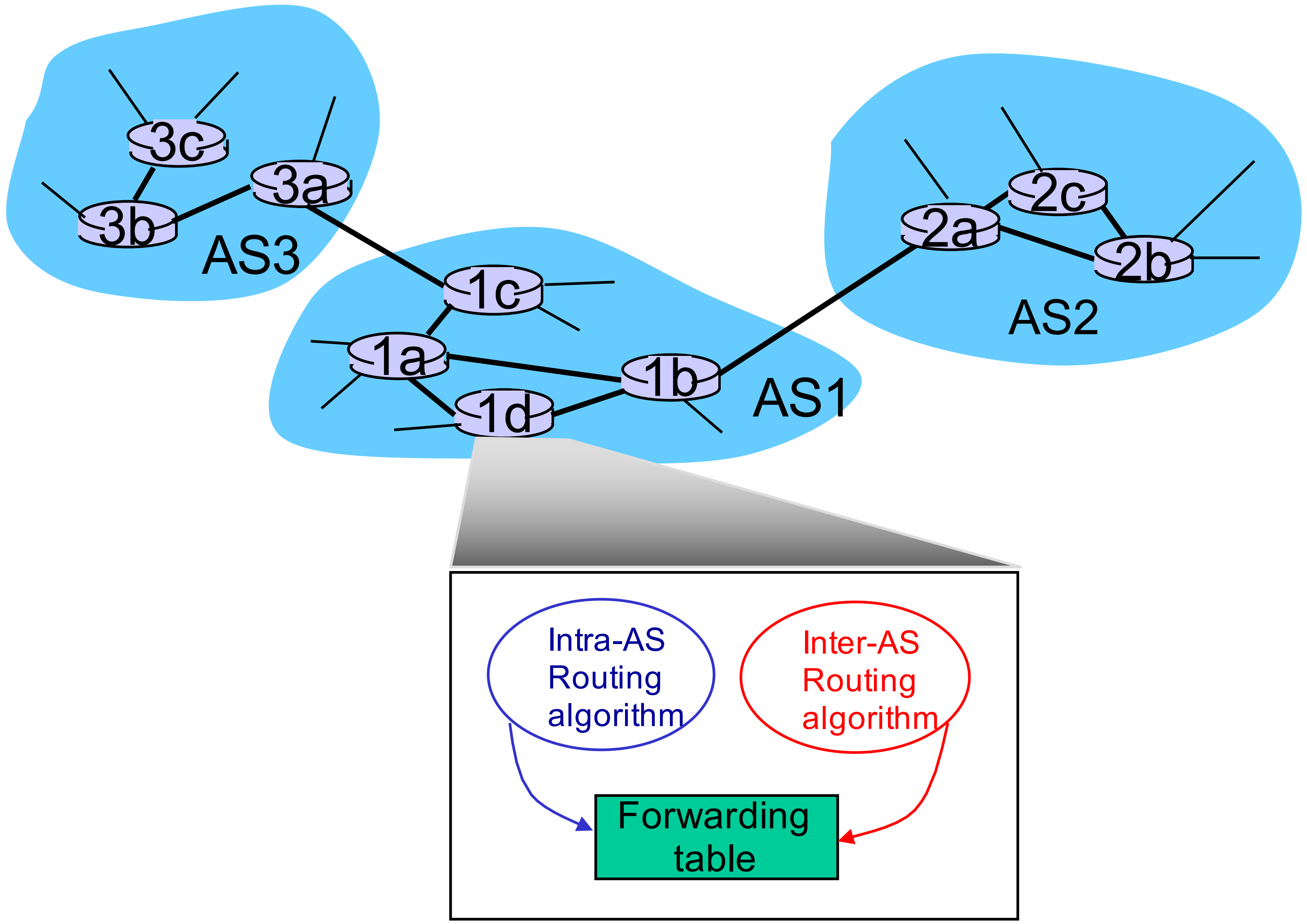
STOP and THINK

- Why does BGP advertise the full AS_PATH instead of just the path length?
- Allows routers to prevent loops:
    - RFC 4271: “AS loop detection is done by scanning the full AS path (as specified in the AS_PATH attribute), and checking that the autonomous system number of the local system does not appear in the AS path.”
    - If I am AS0 and I receive a route with path [2, 4, 0, 5], I should not use it.

BGP Route Selection in practice

1. AS policy determines ***local preference*** for various routes.
   (A hard-coded preference based on financial cost, agreements, etc.)
2. Among routes with the highest local preference,
   choose route with ***shortest AS-PATH***. (DV algorithm.)
3. If multiple options remain, use ***hot-potato routing***, that is, choose the route whose NEXT-HOP is closest.
   - This considers the within-AS distance using an IGP such as OSPF.
4. If multiple options still exist, use a random tie-breaker
   (eg., BGP router id)

Combining Intra- and Inter-AS decisions



- Forwarding table is controlled by both inter- and intra-AS routing algorithms.
  - Local destinations configured by intra-AS algorithm
  - External destinations configured by intra+inter algorithms.
- Border gateways share BGP routing results with all internal routers (using iBGP).

Summary: Why different Intra-, Inter-AS routing?

Policy:

- **Inter-AS:** admin wants control over how its traffic routed, who routes through its net. BGP ads can differ for different neighbors.
- **Intra-AS:** single admin, so can simply optimize for shortest path.

Scale:

- Hierarchical routing reduces table size, reduces update traffic.
- BGP optimizes global Internet routing only at the AS-level.

Performance:

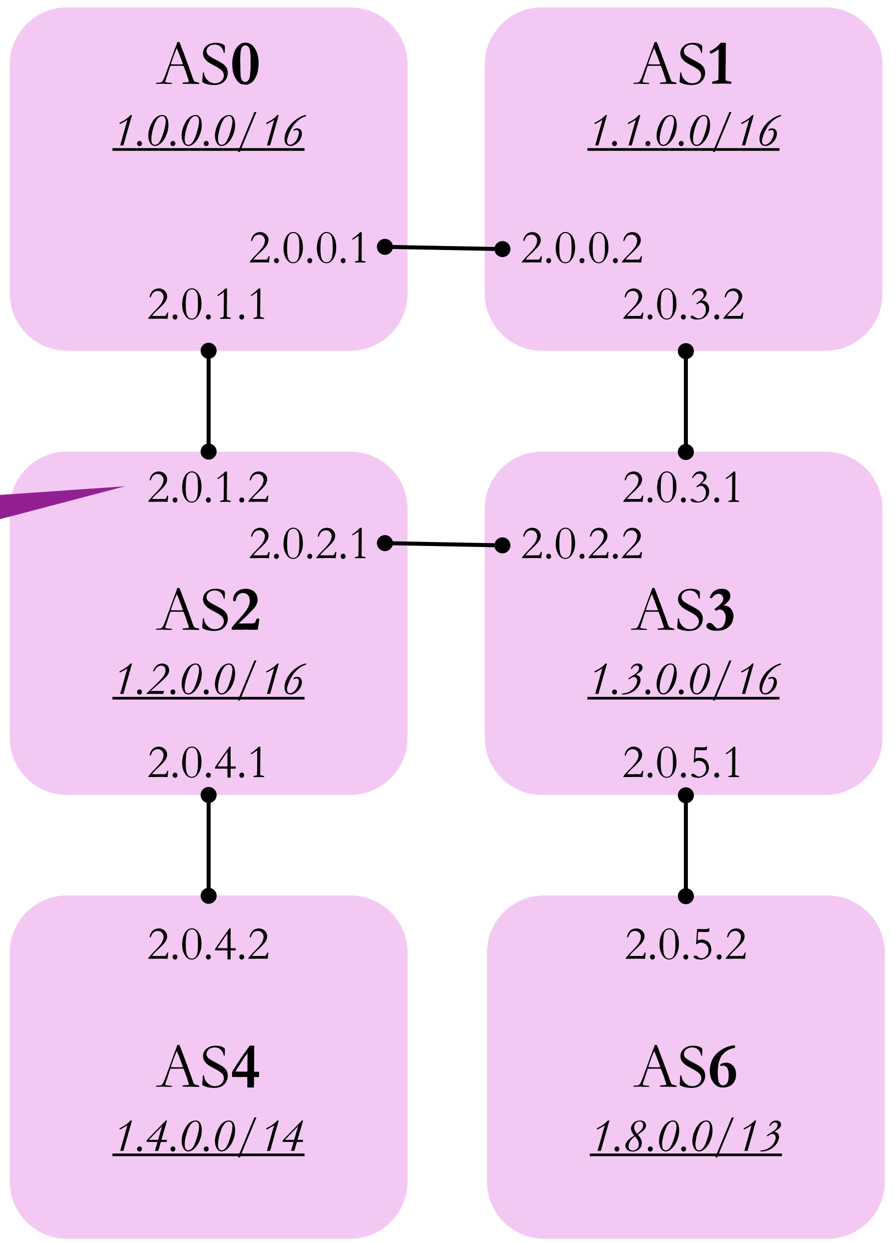
- **Intra-AS:** can focus on performance
- **Inter-AS:** policy and $ may dominate over performance

Intermission

An extended BGP example

- ICANN has distributed a total of ~1 million IP addresses to these organizations (overall: 1.0.0.0/12)
  - /16 = 64k
  - /14 = 256k
  - /13 = 512k
- Each has also been given an AS number 0-5.
- Each AS is reachable from neighbor AS's, through border routers.

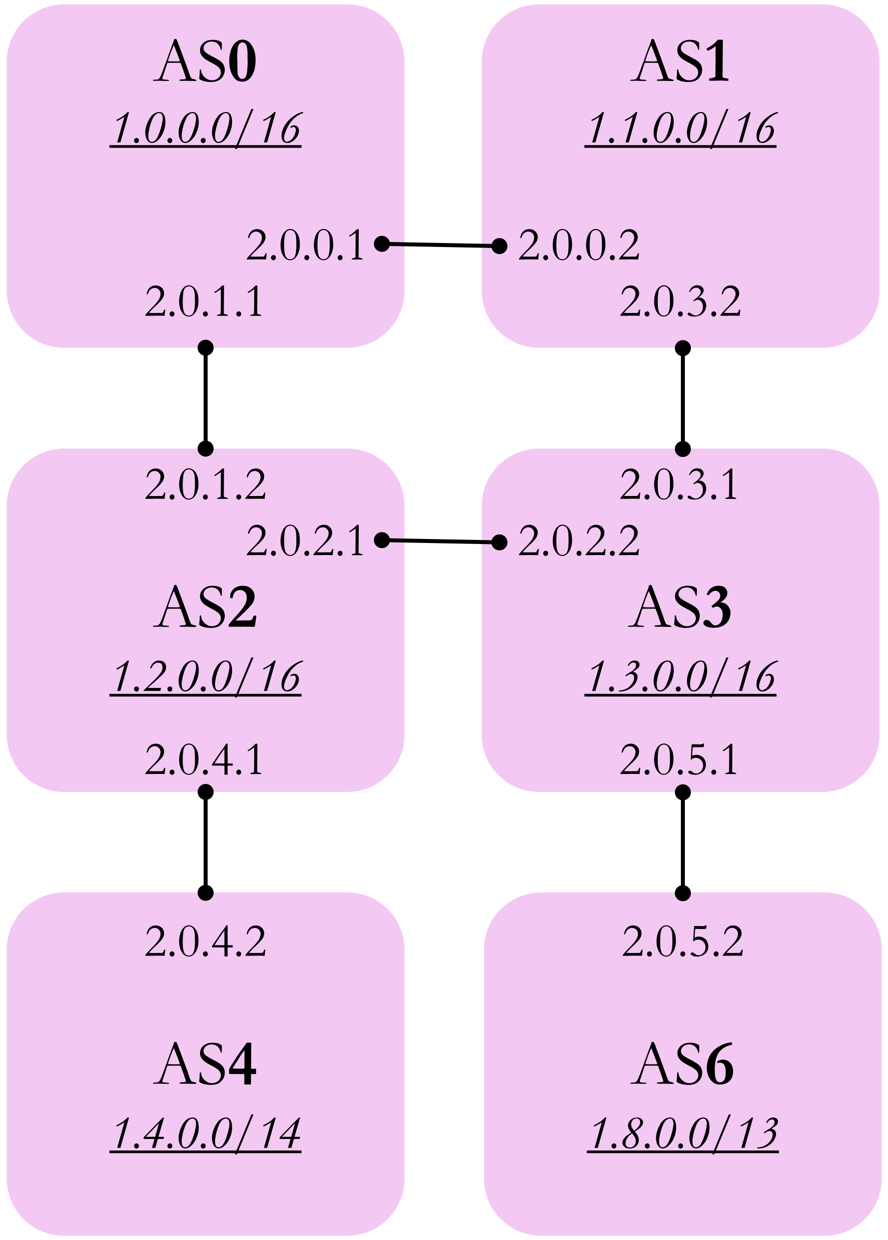
2.0.*.* addresses are used by border routers for convenience



Basic shortest-hop

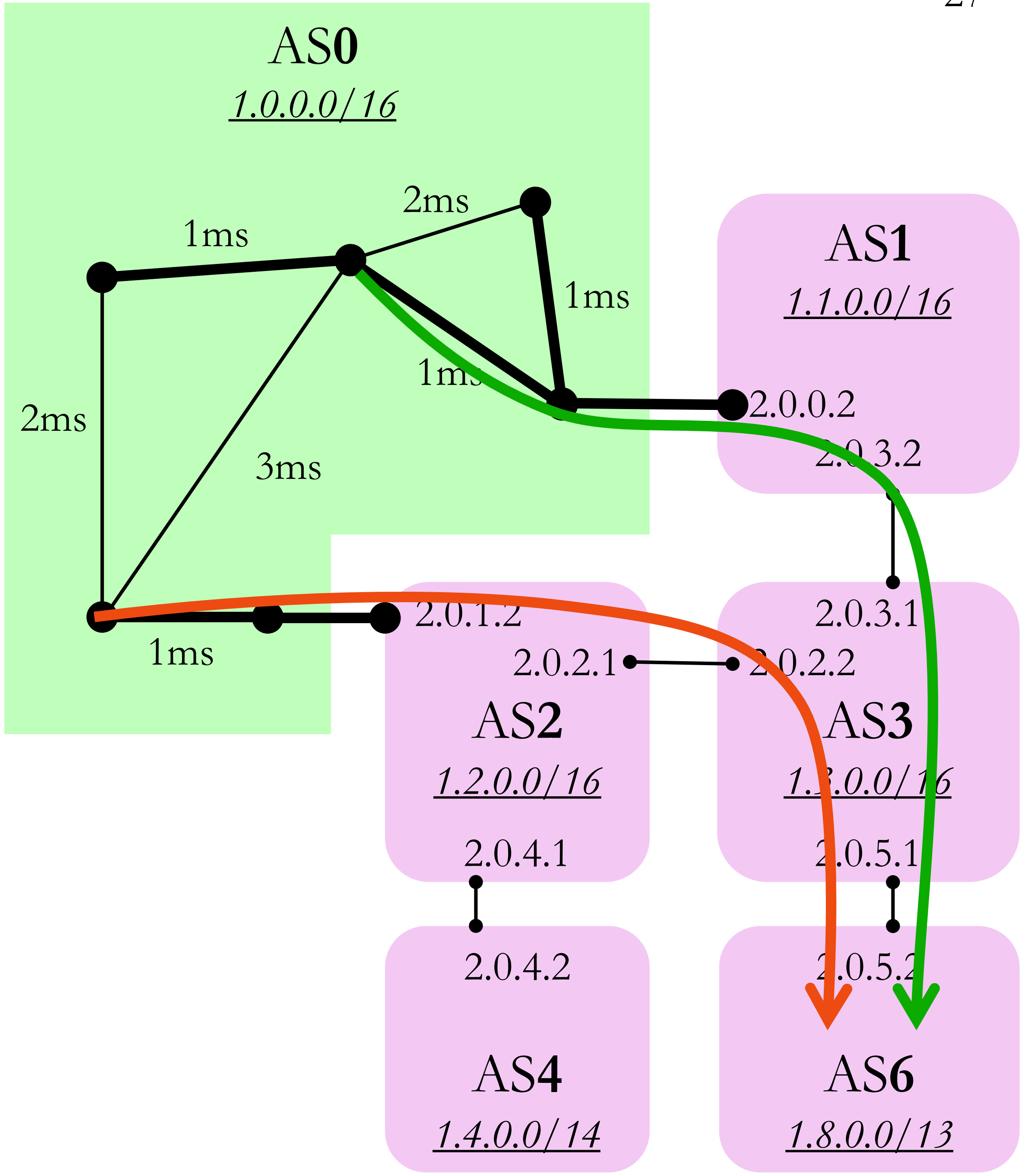
- First, let's assume *no local preference*:
  - All links can be used equally
- BGP should converge to a solution that minimizes AS-hop count.
- A BGP route contains:
  {Prefix, AS_PATH, NEXT_HOP}
- We expect AS0's routing table to be:
  - {1.0.0.0/16, [], … *varies internally by IGP*}
  - {1.1.0.0/16, [1], 2.0.0.2}
  - {1.2.0.0/16, [2], 2.0.1.2}
  - {1.3.0.0/16, [2, 3], 2.0.1.2}
  - … or {1.3.0.0/16, [1, 2], 2.0.0.2}
  - {1.4.0.0/14, [2, 4], 2.0.1.2}
  - {1.8.0.0/13, [2, 3, 6], 2.0.1.2}
  - … or {1.8.0.0.3/13, [1, 3, 6], 2.0.0.2}

see next slides



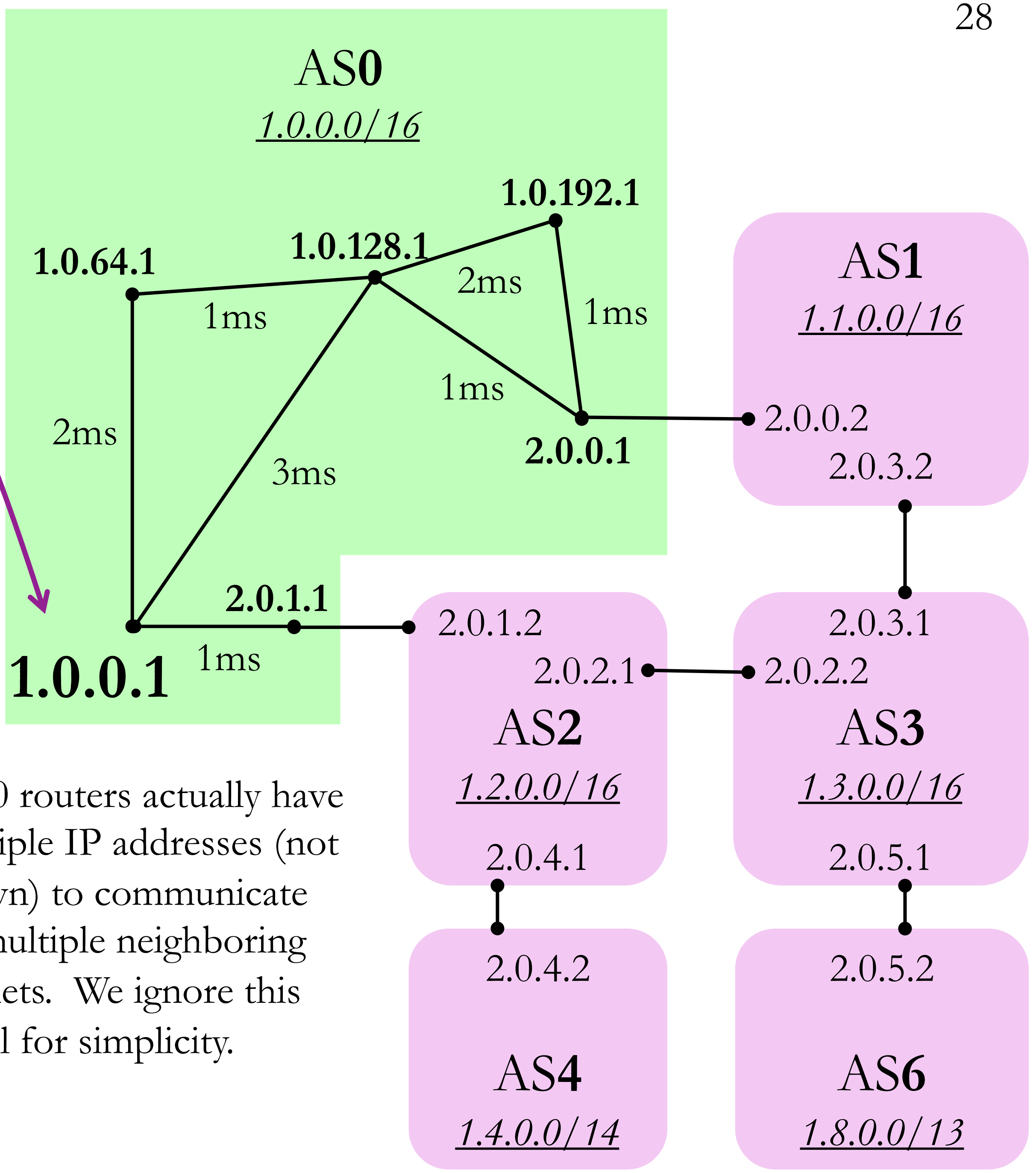
Intra-AS routing

- Within AS0, an IGP like OSPF will find the shortest-path route between each interior prefix and each border gateway.
- For BGP routes with equal AS-hop count, ***hot potato routing*** will choose a different route for different routers within AS0.
  - To break ties in external AS-hop-count, choose the route with fewest internal hops.
  - Eg., when routing to AS3 and AS6, some routers in AS0 will go through AS1 and others will go through AS2.



Forwarding tables

- At the router 1.0.0.1:
  - local subnet is 1.0.0.1/18 (those IPs can be reached directly)
  - 1.0.64.0/18 → 1.0.64.1
  - 1.0.128.0/18 → 1.0.128.1
  - 1.0.192.0/18 → 1.0.128.1
  - 1.1.0.0/16 → 1.0.128.1
  - 1.2.0.0/16 → 2.0.1.1
  - 1.3.0.0/16 → 2.0.1.1
  - 1.4.0.0/14 → 2.0.1.1
  - 1.8.0.0/13 → 2.0.1.1
- These routes can be aggregated…



*AS0 routers actually have multiple IP addresses (not shown) to communicate on multiple neighboring subnets. We ignore this detail for simplicity.

Aggregating routes

Original routes:

- 1.0.0.0/18 → local
- 1.0.64.0/18 → 1.0.64.1
- 1.0.128.0/18 → 1.0.128.1
- 1.0.192.0/18 → 1.0.128.1
- 1.1.0.0/16 → 1.0.128.1
- 1.2.0.0/16 → 2.0.1.1
- 1.3.0.0/16 → 2.0.1.1
- 1.4.0.0/14 → 2.0.1.1
- 1.8.0.0/13 → 2.0.1.1

After aggregation:

- 1.0.0.0/18 → local
- 1.0.64.0/18 → 1.0.64.1
- 1.0.128.0/17 → 1.0.128.1
- 1.1.0.0/16 → 1.0.128.1
- 1.0.0.0/12 → 2.0.1.1

Last route contains all the others, but *longest-prefix matching* is used.

Prefix aggregation detail

$$1.0.128.0/18 = \texttt{00000001.00000000.}\mathbf{\texttt{10xxxxxx}}\texttt{.xxxxxxxx}$$

$$1.0.192.0/18 = \underbrace{\texttt{00000001.00000000.}\mathbf{\texttt{11}}}_{/18}\mathbf{\texttt{xxxxxx}}\texttt{.xxxxxxxx}$$

Two prefixes above are equivalent to:

$$1.0.128.0/17 = \underbrace{\texttt{00000001.00000000.}\mathbf{\texttt{1}}}_{/17}\mathbf{\texttt{xxxxxxx}}\texttt{.xxxxxxxx}$$

Prefix aggregation detail (2)

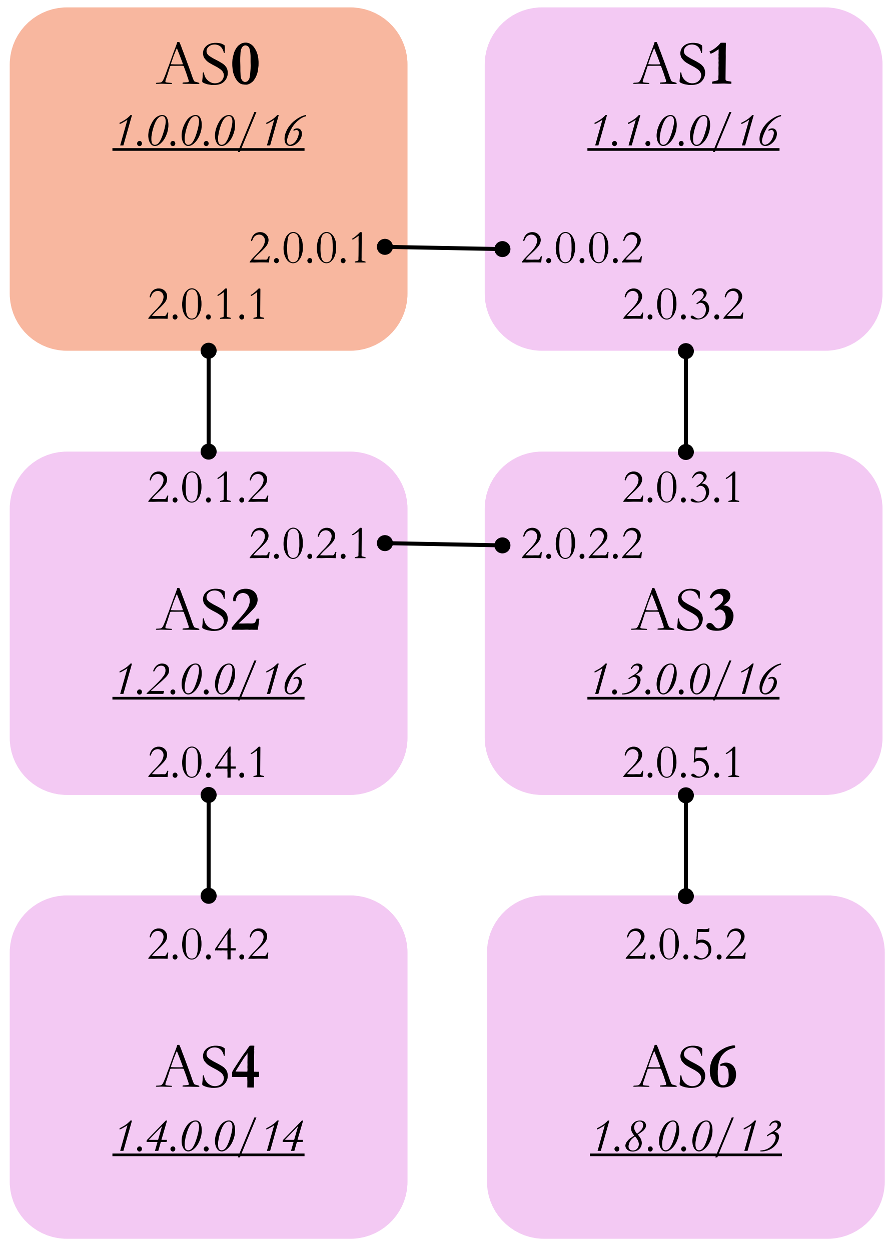
1.0.0.0/18	=	**00000001.00000000.00**XXXXXX.XXXXXXXX
1.0.64.0/18	=	**00000001.00000000.01**XXXXXX.XXXXXXXX
1.0.128.0/17	=	**00000001.00000000.1**XXXXXXX.XXXXXXXX
1.1.0.0/16	=	**00000001.00000001.**XXXXXXXX.XXXXXXXX
1.2.0.0/16	=	**00000001.00000010.**XXXXXXXX.XXXXXXXX
1.3.0.0/16	=	**00000001.00000011.**XXXXXXXX.XXXXXXXX
1.4.0.0/14	=	**00000001.000001**XX.XXXXXXXX.XXXXXXXX
1.8.0.0/13	=	**00000001.00001**XXX.XXXXXXXX.XXXXXXXX

All prefixes above are equivalent to:

1.0.0.0/12 = **00000001.0000**XXXX.XXXXXXXX.XXXXXXXX

BGP step-by-step

- Initially, AS0 knows only about itself:
  - {1.0.0.0/16, [], local}
- AS0 advertises its routes to neighbors:
  - AS1 gets: {1.0.0.0/16, [0], **2.0.0.1**}
  - AS2 gets: {1.0.0.0/16, [0], **2.0.1.1**}
- Neighbors advertise their routes, too:
  - AS1 sends: {1.1.0.0/16, [1], 2.0.0.2}
  - AS2 sends: {1.2.0.0/16, [2], 2.0.1.2}
- iBGP coordinates routers within AS0 to share these new advertisements internally, but we ignore iBGP for now.



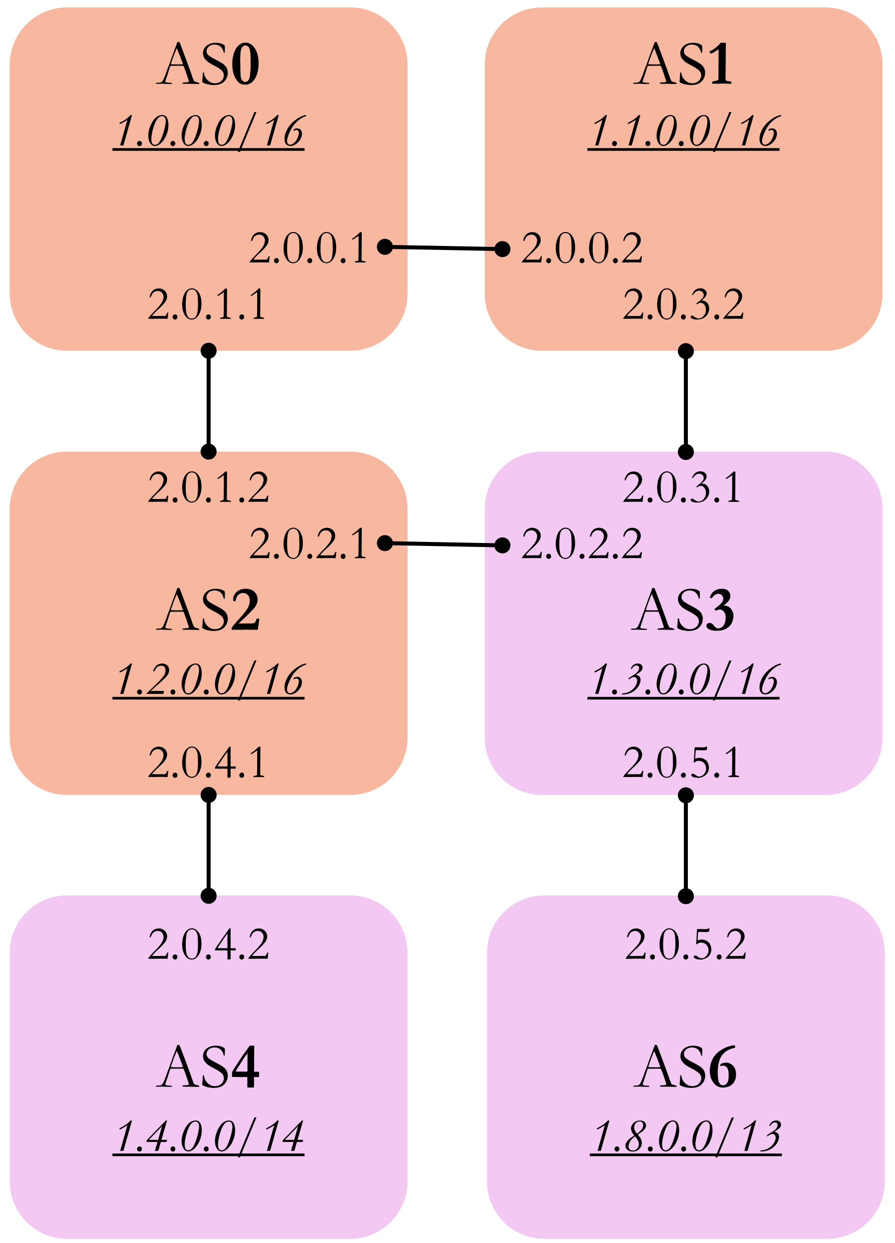
After hearing from neighbors

AS0 now has the following routes

- {1.0.0.0/16, [], local}
- {1.1.0.0/16, [1], 2.0.0.2}
- {1.2.0.0/16, [2], 2.0.1.2}

Eventually neighbor AS1's routes are:

- {1.1.0.0/16, [], local}
- {1.0.0.0/16, [0], 2.0.0.1}
- {1.3.0.0/16, [3], 2.0.3.1}
- {1.8.0.0/13, [3, 6], 2.0.3.1}
- {1.2.0.0/16, [3, 2], 2.0.3.1}
- {1.4.0.0/14}, [3, 2, 4], 2.0.3.1}



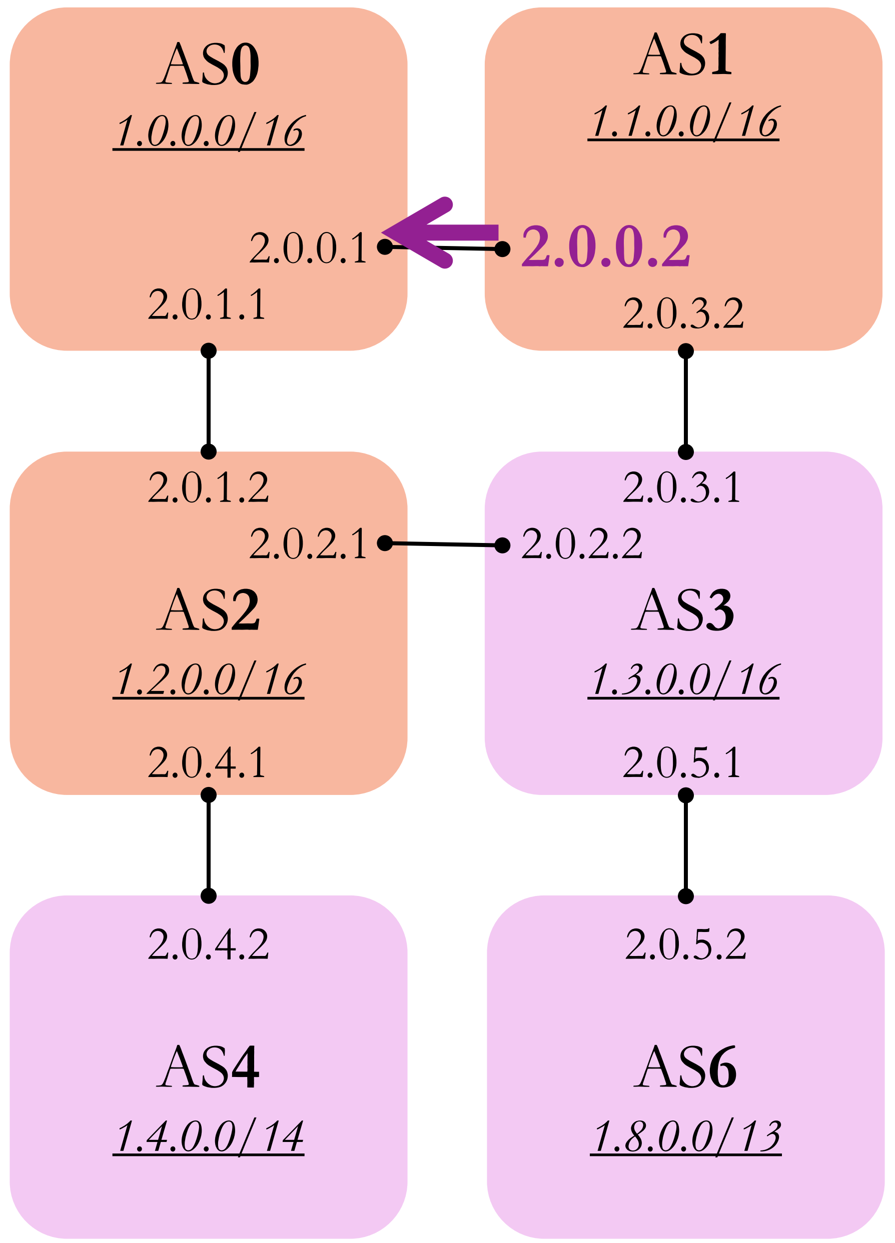
BGP advertisement from AS1

AS1's routes are:

- {1.1.0.0/16, [], local}
- {1.0.0.0/16, [0], 2.0.0.1}
- {1.3.0.0/16, [3], 2.0.3.1}
- {1.8.0.0/13, [3, 6], 2.0.3.1}
- {1.2.0.0/16, [3, 2], 2.0.3.1}
- {1.4.0.0/14}, [3, 2, 4], 2.0.3.1}

AS1 advertises to AS0:

- {1.1.0.0/16, [**1**], **2.0.0.2**}
- {1.0.0.0/16, [**1**, 0], **2.0.0.2**}
- {1.3.0.0/16, [**1**, 3], **2.0.0.2**}
- {1.8.0.0/13, [**1**, 3, 6], **2.0.0.2**}
- {1.2.0.0/16, [**1**, 3, 2], **2.0.0.2**}
- {1.4.0.0/14}, [**1**, 3, 2, 4], **2.0.0.2**}



AS0 recalculates its routes

Formerly:

- {1.0.0.0/16, [], local}
- {1.1.0.0/16, [1], 2.0.0.2}
- {1.2.0.0/16, [2], 2.0.1.2}

Advertisement from AS1 said:

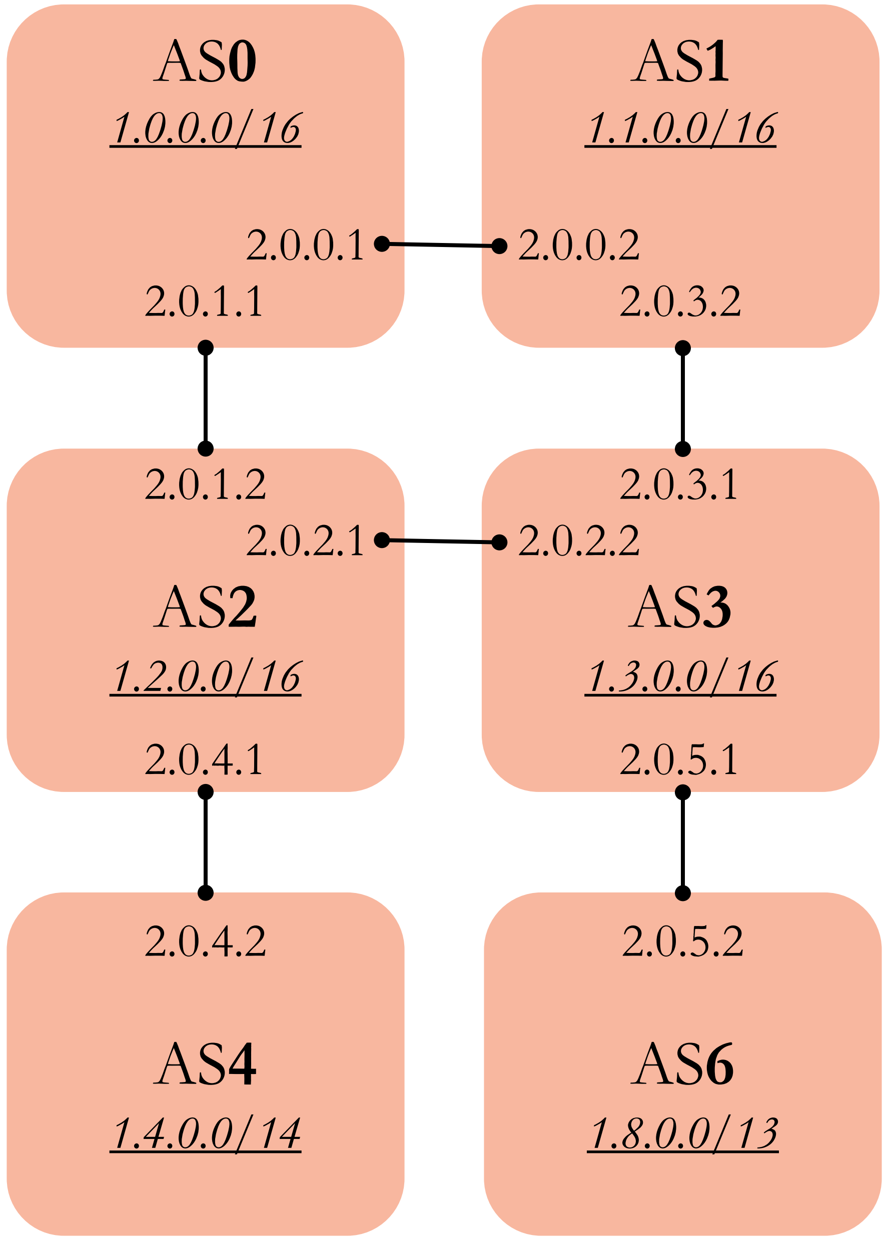
- {1.1.0.0/16, [1], 2.0.0.2}
- {1.0.0.0/16, [1, 0], 2.0.0.2} ← *an inferior & loopy route*
- {1.3.0.0/16, [1, 3], 2.0.0.2}
- {1.8.0.0/13, [1, 3, 6], 2.0.0.2}
- {1.2.0.0/16, [1, 3, 2], 2.0.0.2} ← *an inferior route*
- {1.4.0.0/14}, [1, 3, 2, 4], 2.0.0.2}

Updated AS0 routes:

- {1.0.0.0/16, [], local}
- {1.1.0.0/16, [1], 2.0.0.2}
- {1.2.0.0/16, [2], 2.0.1.2}
- {1.3.0.0/16, [1, 3], 2.0.0.2}
- {1.8.0.0/13, [1, 3, 6], 2.0.0.2}
- {1.4.0.0/14}, [1, 3, 2, 4], 2.0.0.2}

} *added*

This is like a DV update



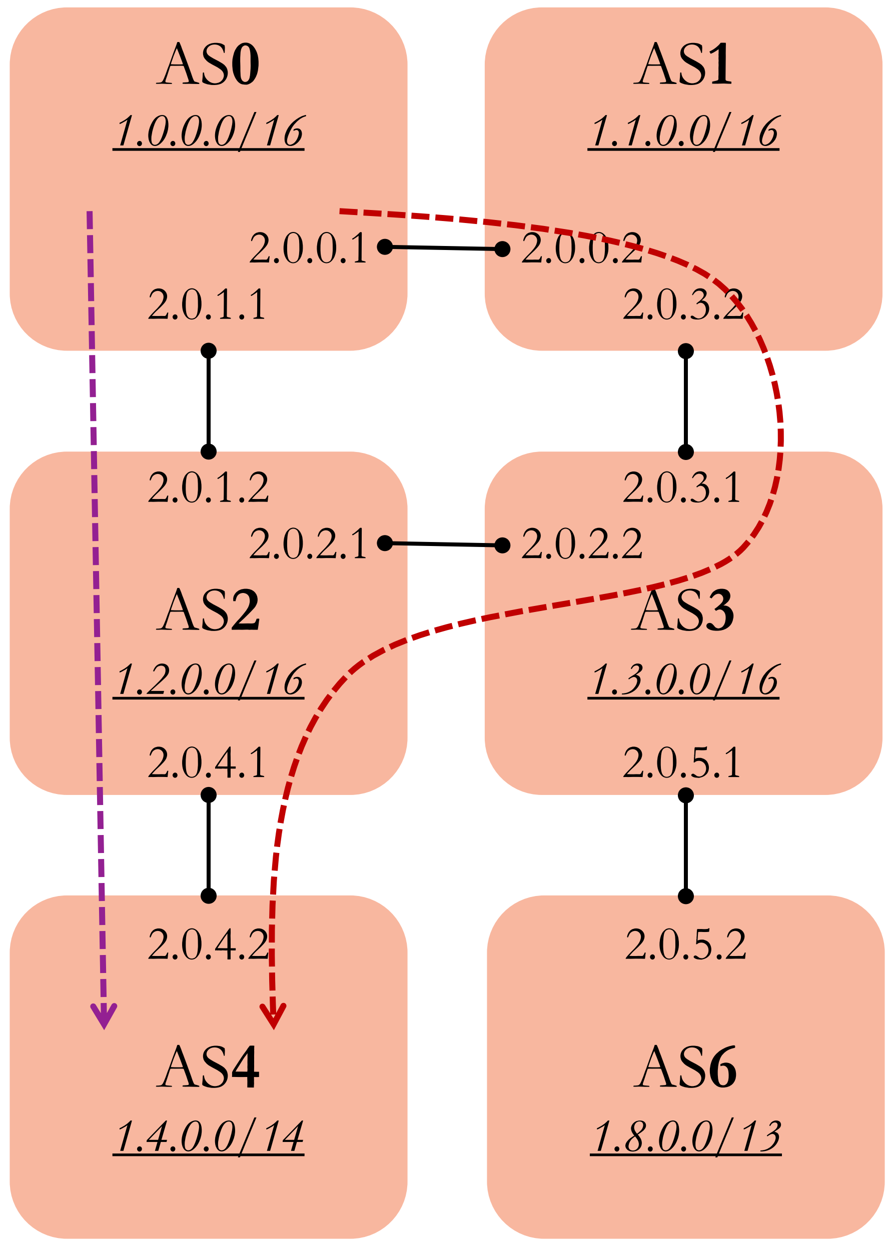
Next, AS0 hears from AS2 & updates

~~Formerly:~~ Updated:

- {1.0.0.0/16, [], local}
- {1.1.0.0/16, [1], 2.0.0.2}
- {1.2.0.0/16, [2], 2.0.1.2}
- {1.3.0.0/16, [1, 3], 2.0.0.2}
- ~~{1.4.0.0/14}, [1, 3, 2, 4], 2.0.0.2}~~
- {1.8.0.0/13, [1, 3, 6], 2.0.0.2}
- {1.4.0.0/14, [2, 4], 2.0.1.2} ← *shorter route*
- {1.3.0.0/16, [2, 3], 2.0.1.2} ← *equal alternative*
- {1.8.0.0/13, [2, 3, 6], 2.0.1.2} ← *equal alternative*

Advertisement from AS2 said:

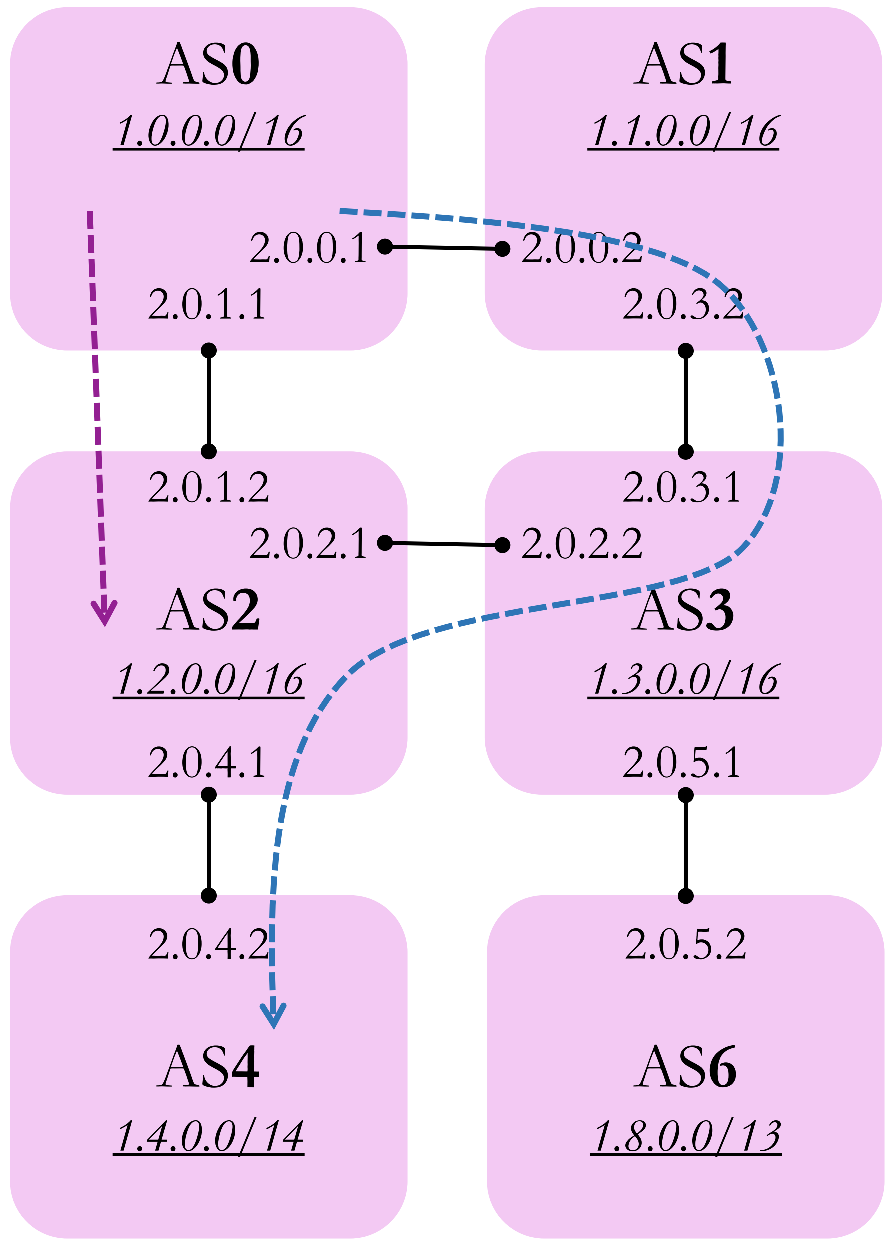
- {1.0.0.0/16, [2, 0], 2.0.1.2}
- {1.2.0.0/16, [2], 2.0.1.2}
- {1.3.0.0/16, [2, 3], 2.0.1.2}
- {1.1.0.0/16, [2, 3, 1], 2.0.1.2} ← *an inferior route*
- {1.4.0.0/14, [2, 4], 2.0.1.2} ← *a better route!*
- {1.8.0.0/13, [2, 3, 6], 2.0.1.2}



Why not use transitive routes?

- AS0 has two routes with AS_PATHs [2] and [1, 3, 2, 4].
- Does this mean that [2, 4] is a valid route?
- BGP does ***not*** make this inference because:
    - Until AS2 advertises the [2, 4] route to me, I cannot be sure that AS2 will accept traffic from me to AS4. AS2 may offer the route to some neighbors (AS3), but not others. (A **policy** distinction.)

STOP and THINK



Choosing among alternative routes

After connecting to AS7, AS0's border router must *choose one* of each duplicate route to advertise to AS7:

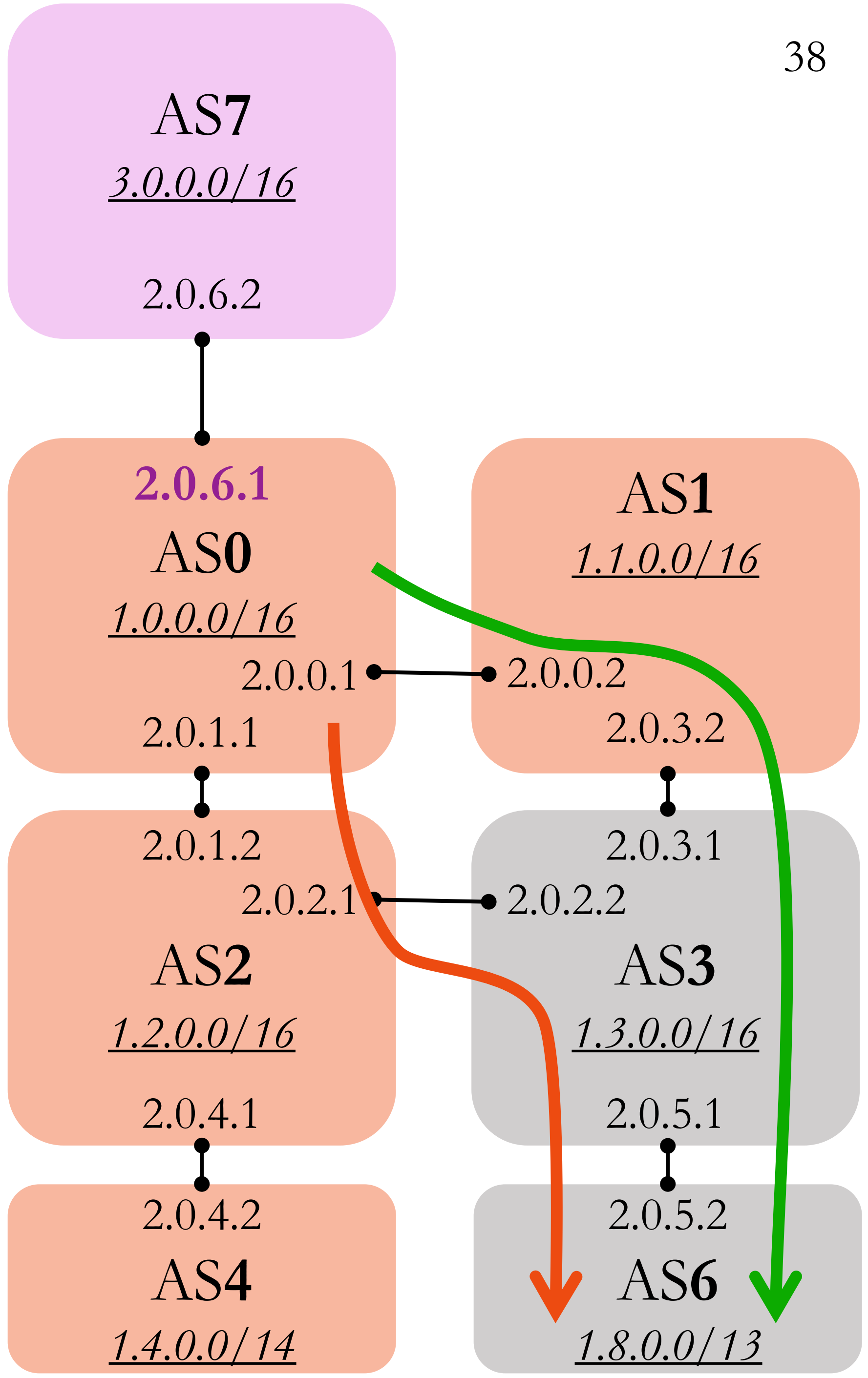
- {1.3.0.0/16, [1, 3], 2.0.0.2}
- {1.3.0.0/16, [2, 3], 2.0.1.2}

} Choose one?

- {1.8.0.0/13, [1, 3, 6], 2.0.0.2}
- {1.8.0.0/13, [2, 3, 6], 2.0.1.2}

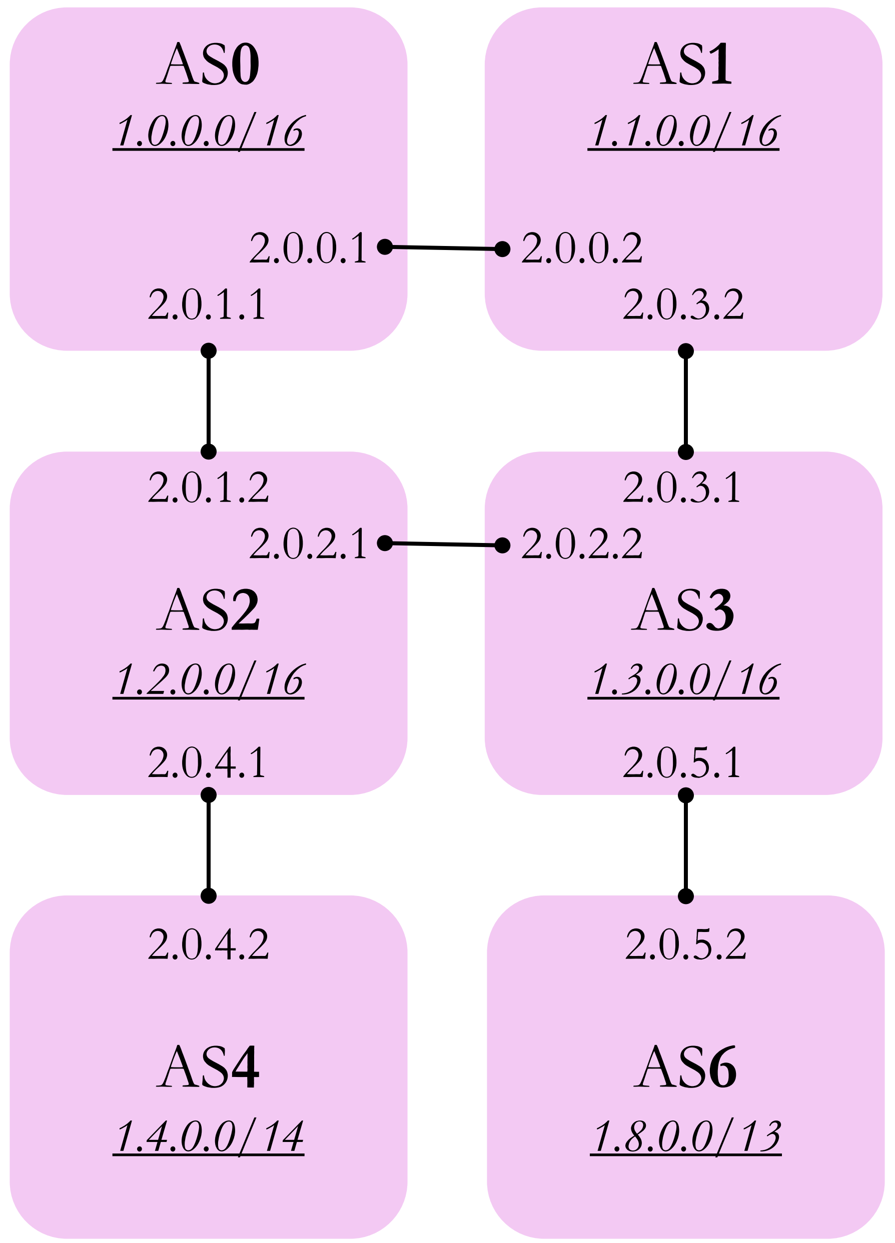
} Choose one?

- Choice depends on ***hot potato routing***:
  - Which of the next-hops is closest to the border router 2.0.6.1? (Fewest hops within AS0)
- If 2.0.0.1 is closer, then it will advertise:
  - {1.3.0.0/16, [0,1,3], 2.0.6.1}
  - {1.8.0.0/16, [0,1,3,6], 2.0.6.1}
- Routing table in AS7 can use one aggregated forwarding rule!
  - {1.0.0.0/12, 2.0.6.1}



Local preference

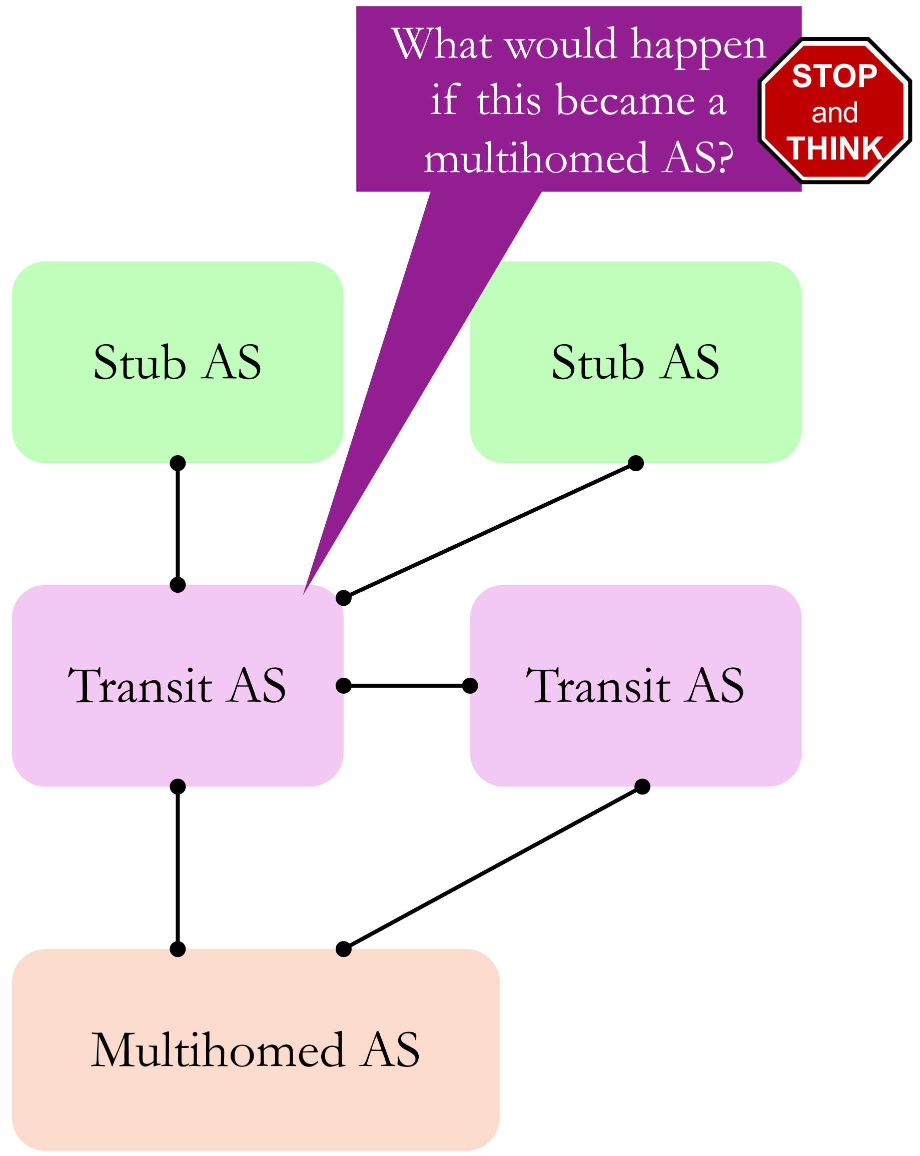
- Local preferences are optional, hard-coded forwarding rules.
- Local preference is given priority over the shortest path.
  - Often due to business considerations or due to links having different bandwidth/latency.
- AS0 has two next-hop choices to route to *all* of the AS's.
- For example, a worst-case choice of local preference at AS0 could yield:
  - {1.0.0.0/16, [], local}
  - {1.1.0.0/16, [2, 3, 1], 2.0.1.2}
  - {1.2.0.0/16, [1, 3, 2], 2.0.0.2}
  - {1.3.0.0/16, [2, 3], 2.0.1.2}
  - {1.4.0.0/14, [1, 3, 2, 4], 2.0.0.2}
  - {1.8.0.0/13, [2, 3, 6], 2.0.1.2}



Types of Autonomous Systems

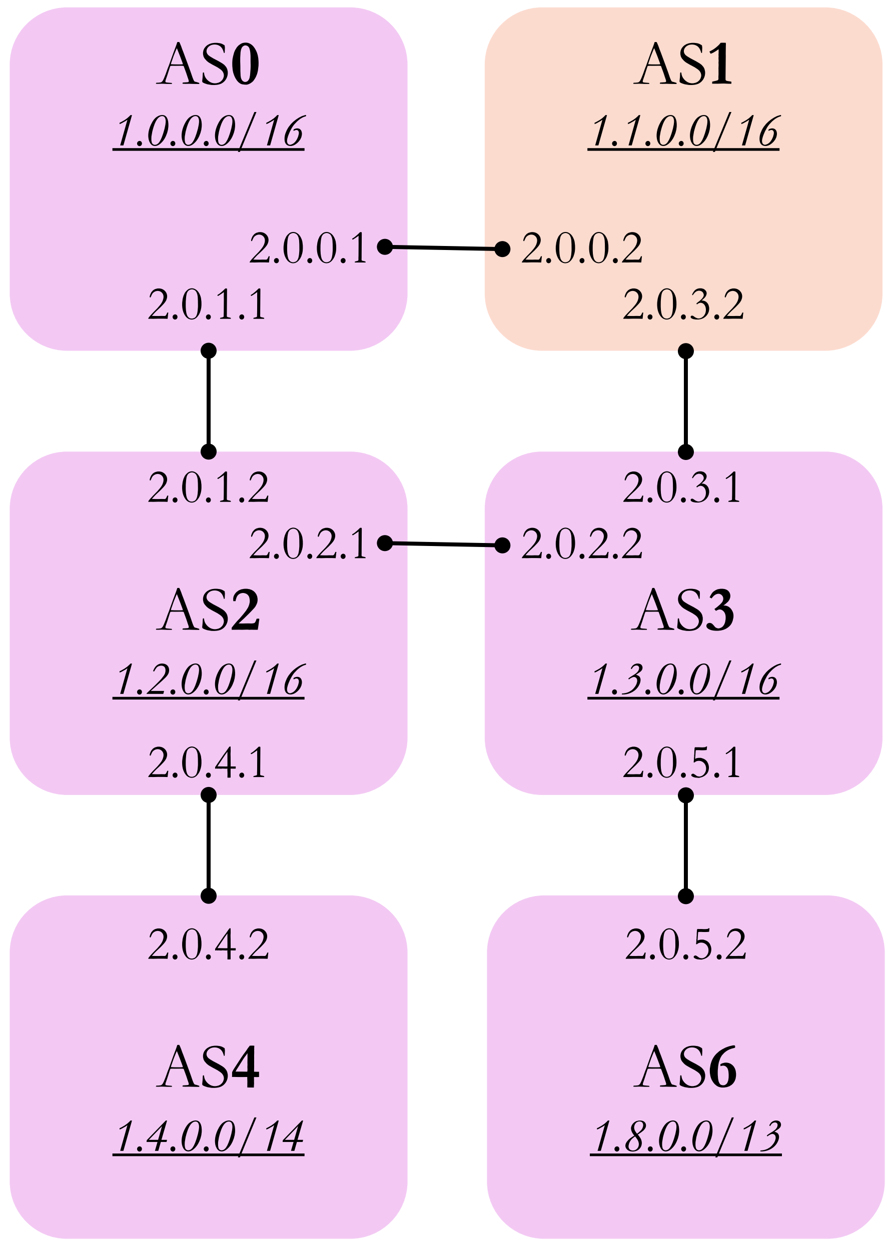
- ***Stub***: connects to just *one* other AS.
- ***Transit***: connects to multiple AS's and routes between them.
  - Eg., Tier 1 ISPs
- ***Multihomed***: connects to multiple other AS's, but do not route between.
  - Links for performance and fault tolerance.
  - Only advertises routes to its own subnets.

- Multihomed and transit AS's both have multiple connections but have different BGP advertisement policies.



Multihoming

- Let's say that AS1 decides to become a *multihomed* AS instead of a *transit* AS.
  - It will stop advertising routes to outside subnets.
- At AS0, we will only route to AS1 for its 1.1.0.0/16 subnet:
  - {1.0.0.0/16, [], local}
  - {1.1.0.0/16, [1], 2.0.0.2}
  - {1.2.0.0/16, [2], 2.0.1.2}
  - {1.3.0.0/16, [2, 3], 2.0.1.2}
  - ~~{1.3.0.0/16, [1, 2], 2.0.0.2}~~ ← *not advertised*
  - {1.4.0.0/14, [2, 4], 2.0.1.2}
  - {1.8.0.0/13, [2, 3, 6], 2.0.1.2}
  - ~~{1.8.0.0.3/13, [1, 3, 6], 2.0.0.2}~~ ← *not advertised*



BGP routing review

- DV **Count to infinity** leads to slower convergence when links get worse.
  - Good news travels quickly, bad news travels slowly.
- Internet routing is hierarchical.
- **Autonomous systems** (ASes) are grouped routers with one routing policy.
- An **Interior Gateway Protocol** (IGP) (eg., OSPF) determines optimal routes within an AS.
  - Can use a centralized (Link State) shortest path algorithm, like Dijkstra's.
- The **Border Gateway Protocol** (BGP) determines routes between ASes.
  - Uses a distributed shortest-AS-hop path algorithm (Distance Vector).
- BGP **advertisement** includes a list of routes, each looking like:
  - {PREFIX: 43.5.0.0/16, AS-PATH: [AS4, AS65, AS1], NEXT-HOP: 5.6.7.200)}
  - This tells a neighboring AS that it can forward packets to the prefix.