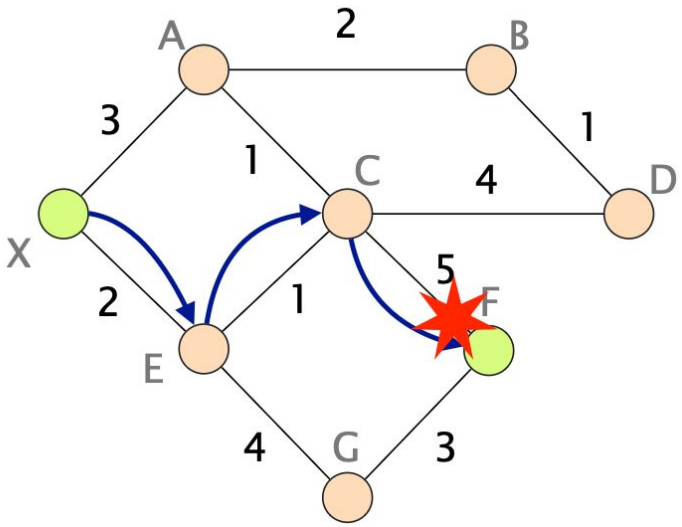
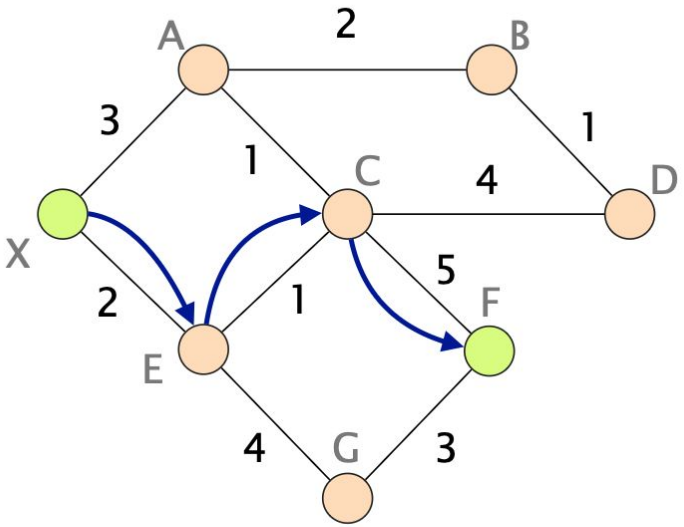
Black Holes - Due to Detection Delay as Routers Do Not Immediately Detect Failure



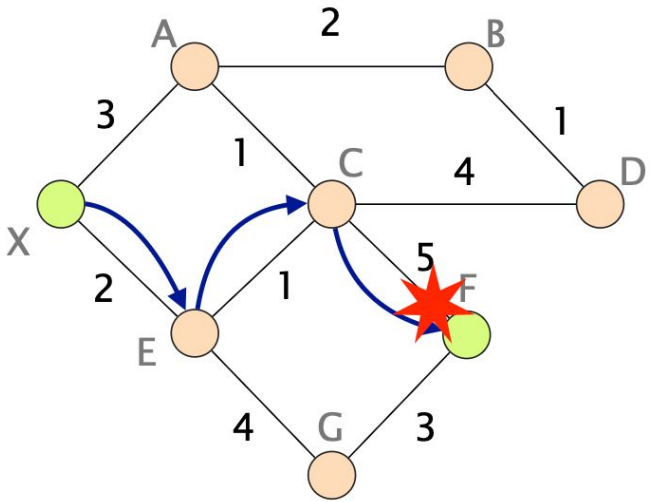
depends on the timeout for detecting lost hellos

Forwarding Loops - Due to Inconsistent Link-State DBs



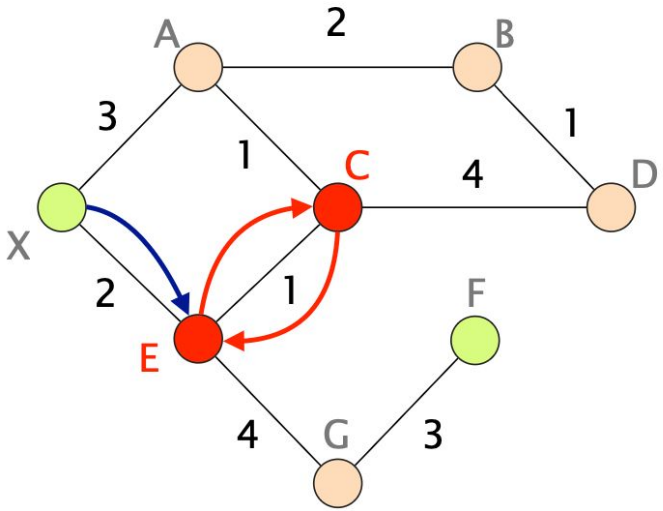
Initial forwarding state

Forwarding Loops - Due to Inconsistent Link-State DBs



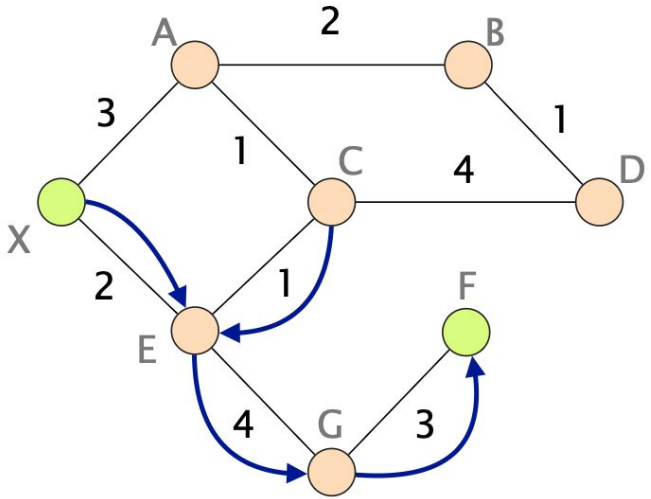
C learns about the failure
and immediately reroute to E

Forwarding Loops - Due to Inconsistent Link-State DBs



A loop appears as E
isn't yet aware of the failure

Forwarding Loops - Due to Inconsistent Link-State DBs

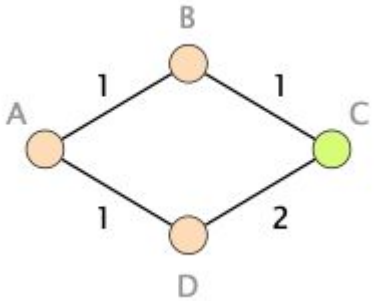


The loop disappears as soon as
E updates its forwarding table

Forwarding Loops - Due to Inconsistent Link-State DBs

Consider this simple network running OSPF as link-state routing protocol. Each link is associated with a weight that represents the cost of using it to forward packets. Link weights are bi-directional.

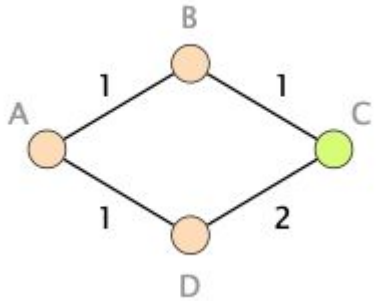
Assume that routers A, B and D transit traffic for an IP destination connected to C and that link (B,C) fails. Which nodes among A, B and D could potentially see their packets being stuck in a transient forwarding loop? Which ones would not?



Forwarding Loops - Due to Inconsistent Link-State DBs

Consider this simple network running OSPF as link-state routing protocol. Each link is associated with a weight that represents the cost of using it to forward packets. Link weights are bi-directional.

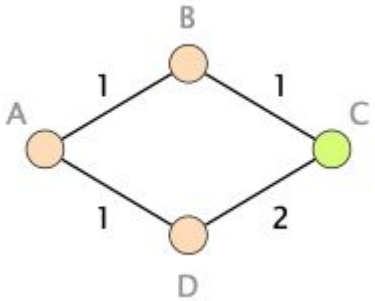
Assume that routers A, B and D transit traffic for an IP destination connected to C and that link (B,C) fails. Which nodes among A, B and D could potentially see their packets being stuck in a transient forwarding loop? Which ones would not?



Solution: Nodes A and B could see their packets stuck in a forwarding loop if B updates its forwarding table before A, which is likely to happen as B would be the first to learn about an adjacent link failure. On the other hand, D would not see any loop as it uses its direct link with C to reach any destination connected beyond it.

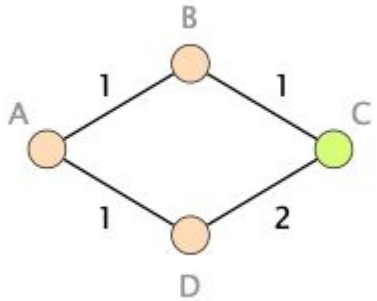
Forwarding Loops - Due to Inconsistent Link-State DBs

Assume now that the network administrator wants to take down the link (B,C), on purpose, for maintenance reasons. To avoid transient issues, the administrator would like to move away all traffic from the link before taking it down and this, without creating any transient loop (if possible). What is the minimum sequence of increased weights setting on link (B,C) that would ensure that no packet destined to C is dropped?



Forwarding Loops - Due to Inconsistent Link-State DBs

Assume now that the network administrator wants to take down the link (B,C), on purpose, for maintenance reasons. To avoid transient issues, the administrator would like to move away all traffic from the link before taking it down and this, without creating any transient loop (if possible). What is the minimum sequence of increased weights setting on link (B,C) that would ensure that no packet destined to C is dropped?



Solution: One example of a minimum sequence of (B,C) weights is [1, 3, 5].

Note: The problem highlighted above happens because B shifts traffic to A before A shifts traffic to D, hence creating a forwarding loop. By setting the (B,C) link weight to 3 first, (only) A shifts from using (A, B, C) to using (A, D, C). Once A has shifted, it is safe to shift B by setting the link weight to 5 (or higher). Once B has shifted as well, the link can be safely torn down.

Convergence - the Process During Which Routers Seek to Regain a Consistent View of the Network

Two Major Link-State Protocols in Wide Use

OSPF

IS-IS

Open Shortest Path First

Intermediate Systems[2]

used in many enterprise & ISPs

work on top of IP

only route IPv4 by default

Two Major Link-State Protocols in Wide Use

OSPF

Open Shortest Path First

IS-IS

Intermediate Systems[2]

used mostly in large ISPs

work on top of link-layer

network protocol agnostic

Distance Vector - Recall...

Use the Bellman Ford algorithm

$$d_x(y) = \min\{ c(x,v) + d_v(y) \} \qquad \text{over all neighbors } v$$

Routing by rumor

Good news travels fast

Bad news travels slowly - count to infinity

Similar to Link-State, Routers have three Situations to Send New DVs

Topology change	link or node failure/recovery
Configuration change	link cost change
Periodically	refresh the link-state information every (say) 30 minutes account for possible data corruption

How Do We Fix Count-to-Infinity?

How Do We Fix Count-to-Infinity?

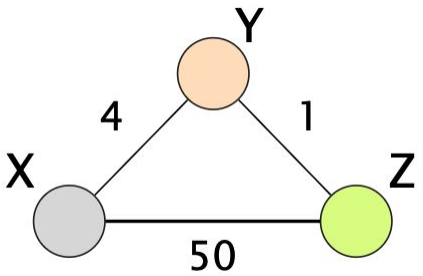
When a router uses another one, it will announce it as an infinite cost

- Technique known as poisoned reverse

How Do We Fix Count-to-Infinity?

When a router uses another one, it will announce it as an infinite cost

- Technique known as poisoned reverse

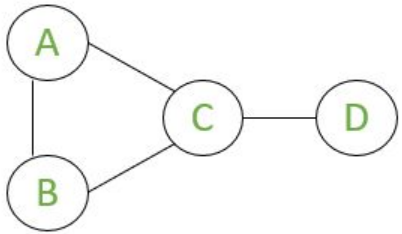


Poisoned Reverse Failure

Can you think of a case where poisoned reverse cannot prevent loops?

Poisoned Reverse Failure

Can you think of a case where poisoned reverse cannot prevent loops?

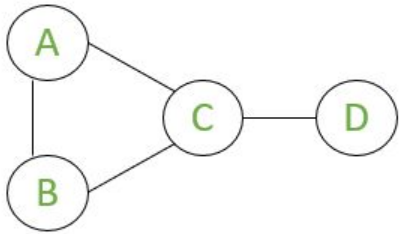


Poisoned Reverse Failure

Can you think of a case where poisoned reverse cannot prevent loops?

C tells A & B that D is unreachable

- A computes new route through B
  - Tells B that D is unreachable (poison reverse)
  - Tells C it has path of cost 3
- C computes new route through A
  - C tells B that D is now reachable
- Etc...



Poisoned Reverse

- In reality infinity ~= 16 for most protocols

Link-State vs Distance Vector

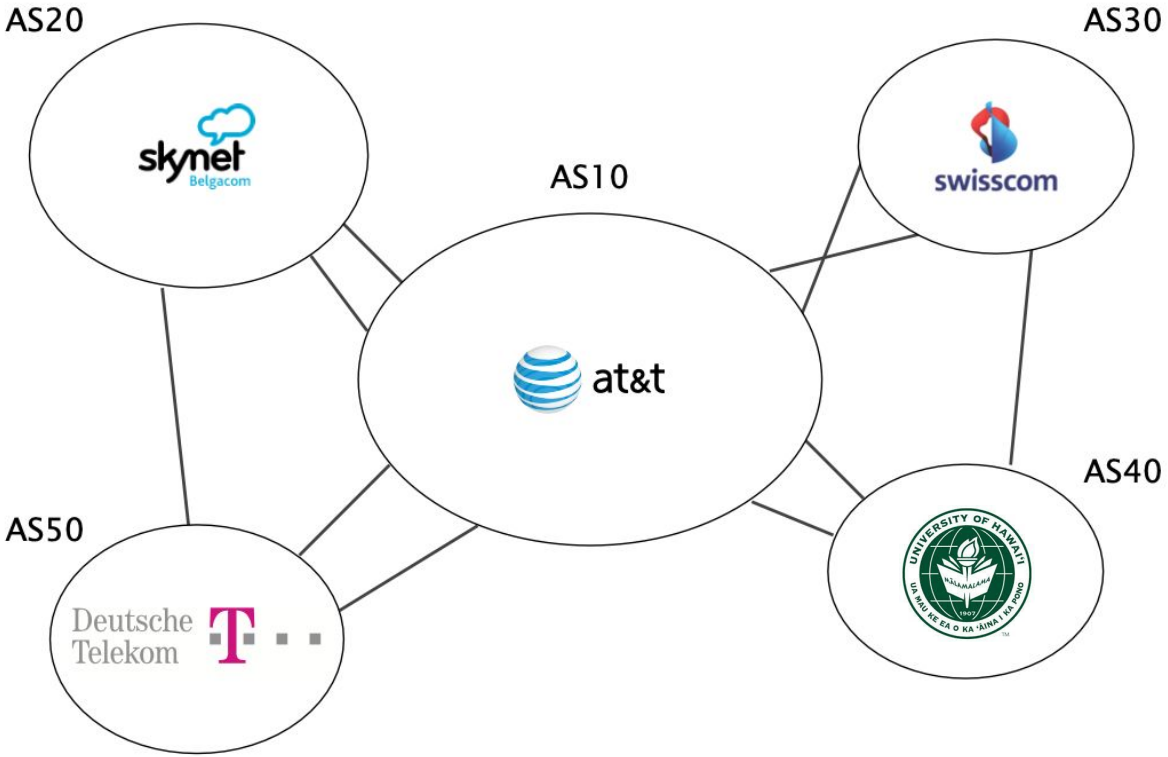
	Message complexity	Convergence speed	Robustness
Link-State	O(nE) message sent n: #nodes E: #links	relatively fast	node can advertise incorrect link cost nodes compute their own table
Distance-Vector	between neighbors only	slow	node can advertise incorrect path cost errors propagate

Internet Routing

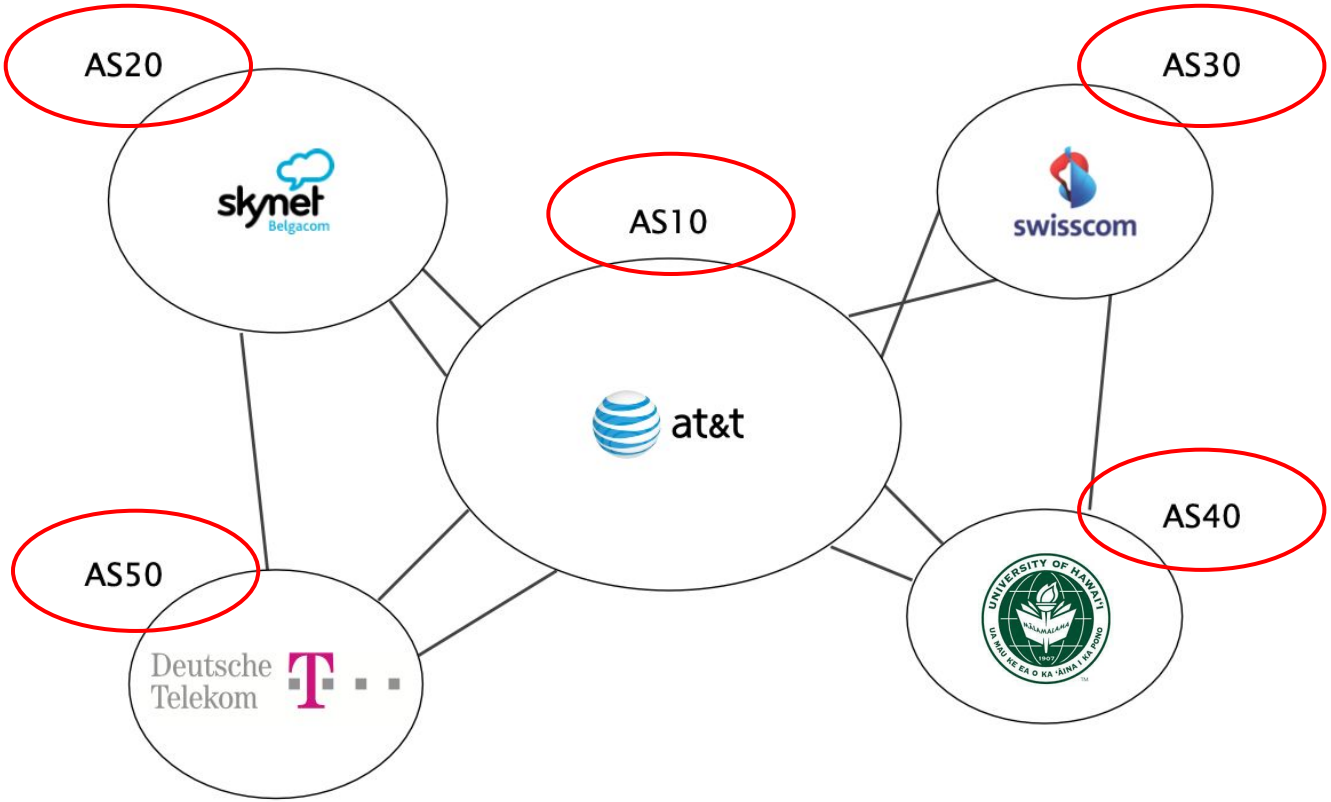
1. Intra-domain routing
    - Link-state protocols
    - Distance-vector protocols

2. Inter-domain routing
    - Path-vector protocols

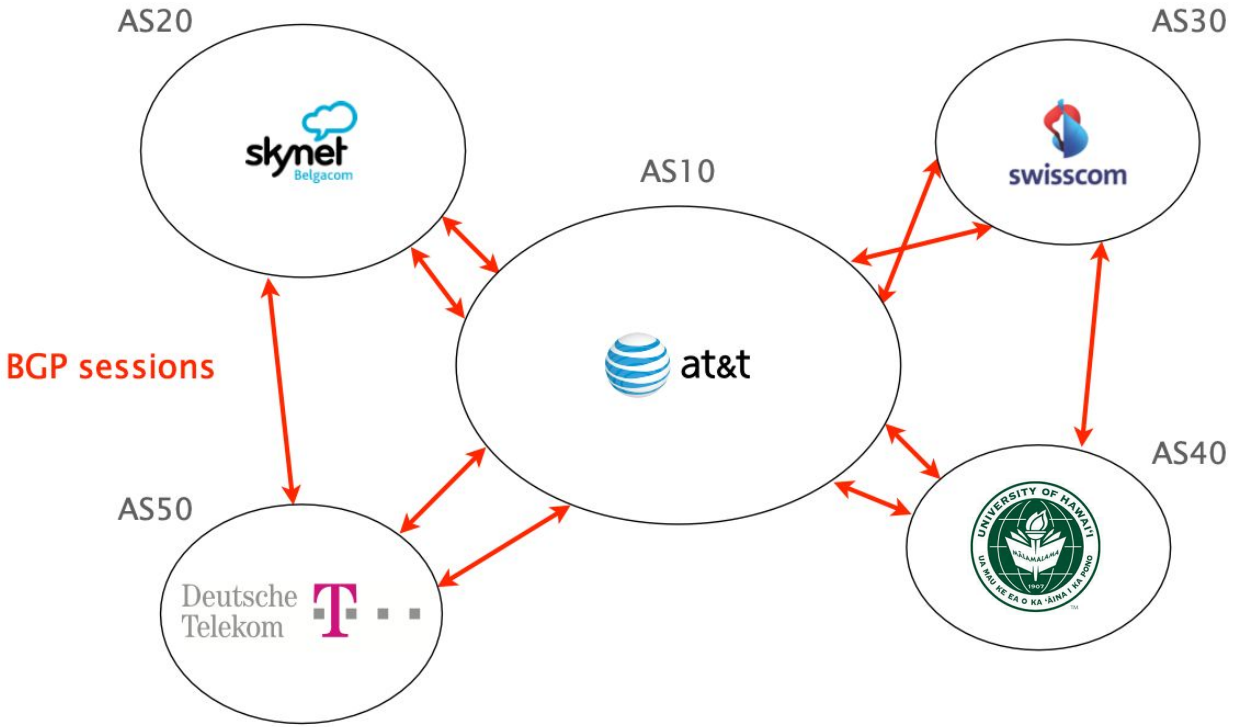
The Internet is a Network of Networks referred to as Autonomous Systems (AS)



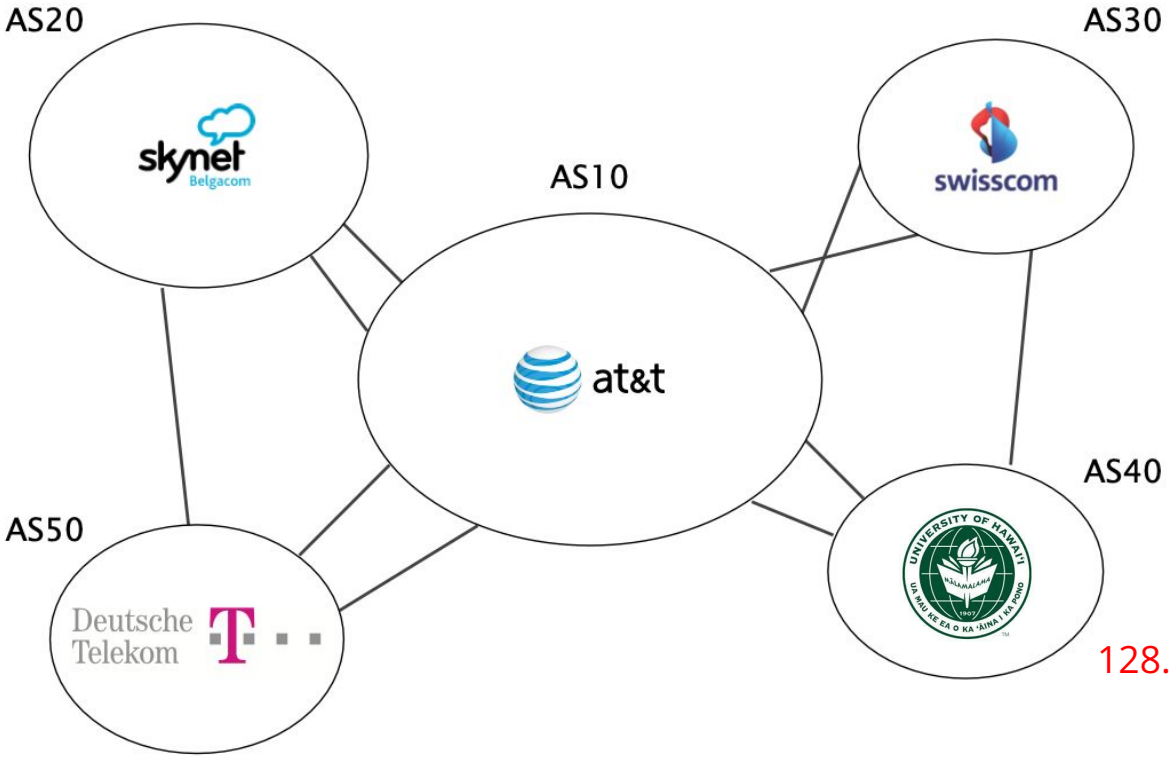
Each AS has a Number that Identifies it (16 bits)



Border Gateway Protocol is the Glue that Holds the Internet Together



ASes use BGP to Advertise IP Prefixes they can Reach, either Directly or Indirectly



BGP Needs to Solve Three Challenges: Scalability, Privacy and Policy Enforcement

- There is a huge # of networks and prefixes
  - 1M prefixes, >70,000 networks, millions of routers
- Networks don’t want to divulge internal topologies or their business relationships
- Networks need to control where to send and receive traffic without an Internet-wide notion of a link cost metric

Link-State DOES NOT Solve These Challenges

- Floods topology information
  - high processing overhead
- Requires each node to compute the entire path
  - high processing overhead
- Minimizes some notion of total distance
  - works only if the policy is shared and uniform

Distance Vector is *Better*

pros

Hide details of the network topology

nodes determine only “next-hop” for each destination

Distance Vector is *Better*, But Not Quite There

pros

Hide details of the network topology

nodes determine only “next-hop” for each destination

cons

It still minimizes some common distance

impossible to achieve in an inter domain setting

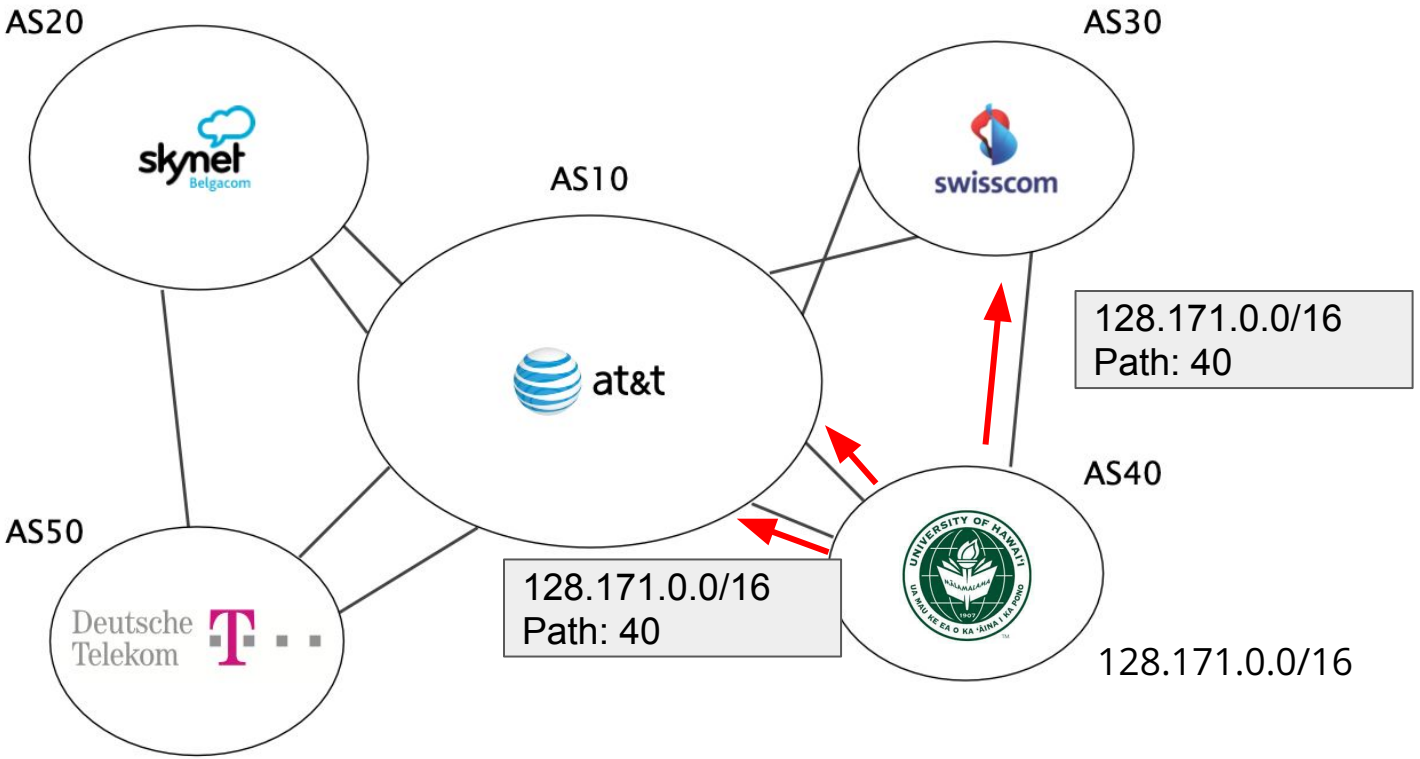
It converges slowly

counting-to-infinity problem

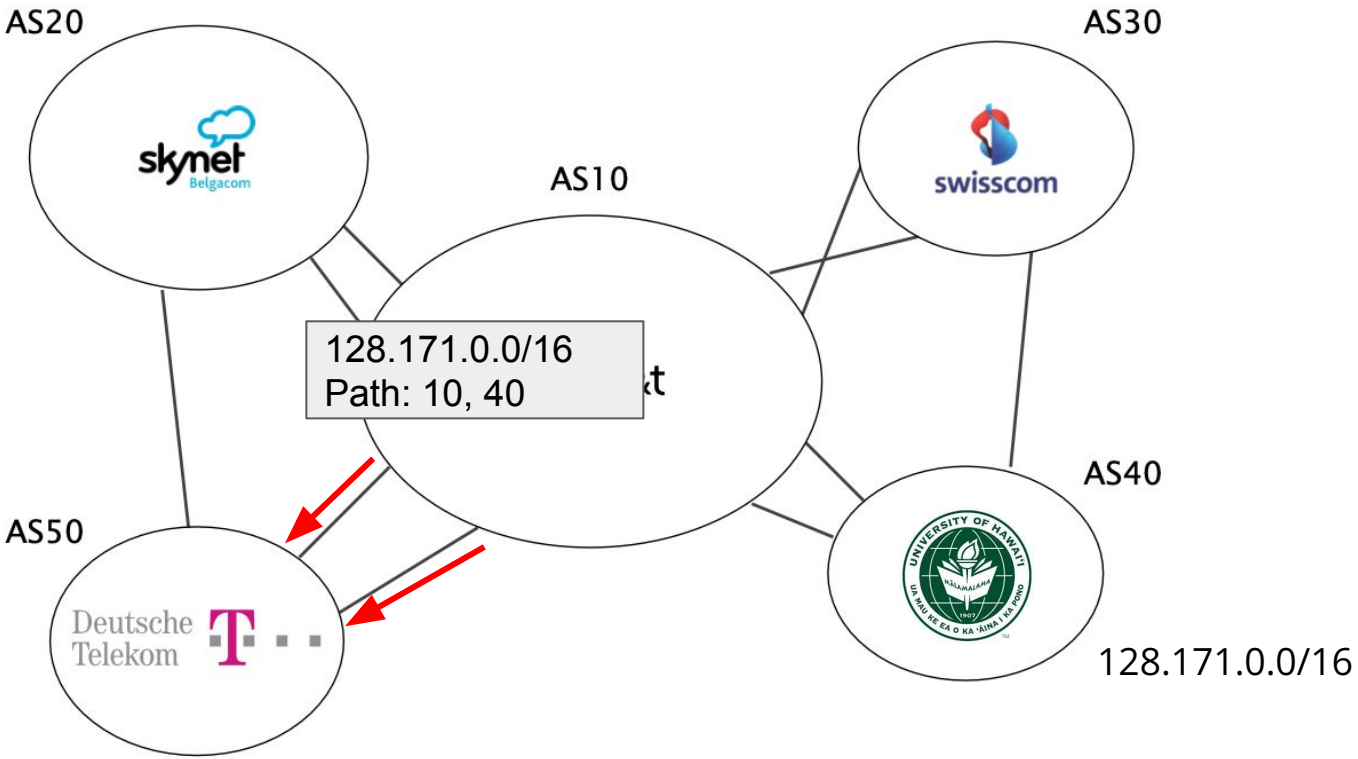
BGP Uses Path Vector Routing to Support Flexible Routing and Avoid Count to Infinity

key idea advertise the entire path instead of distances

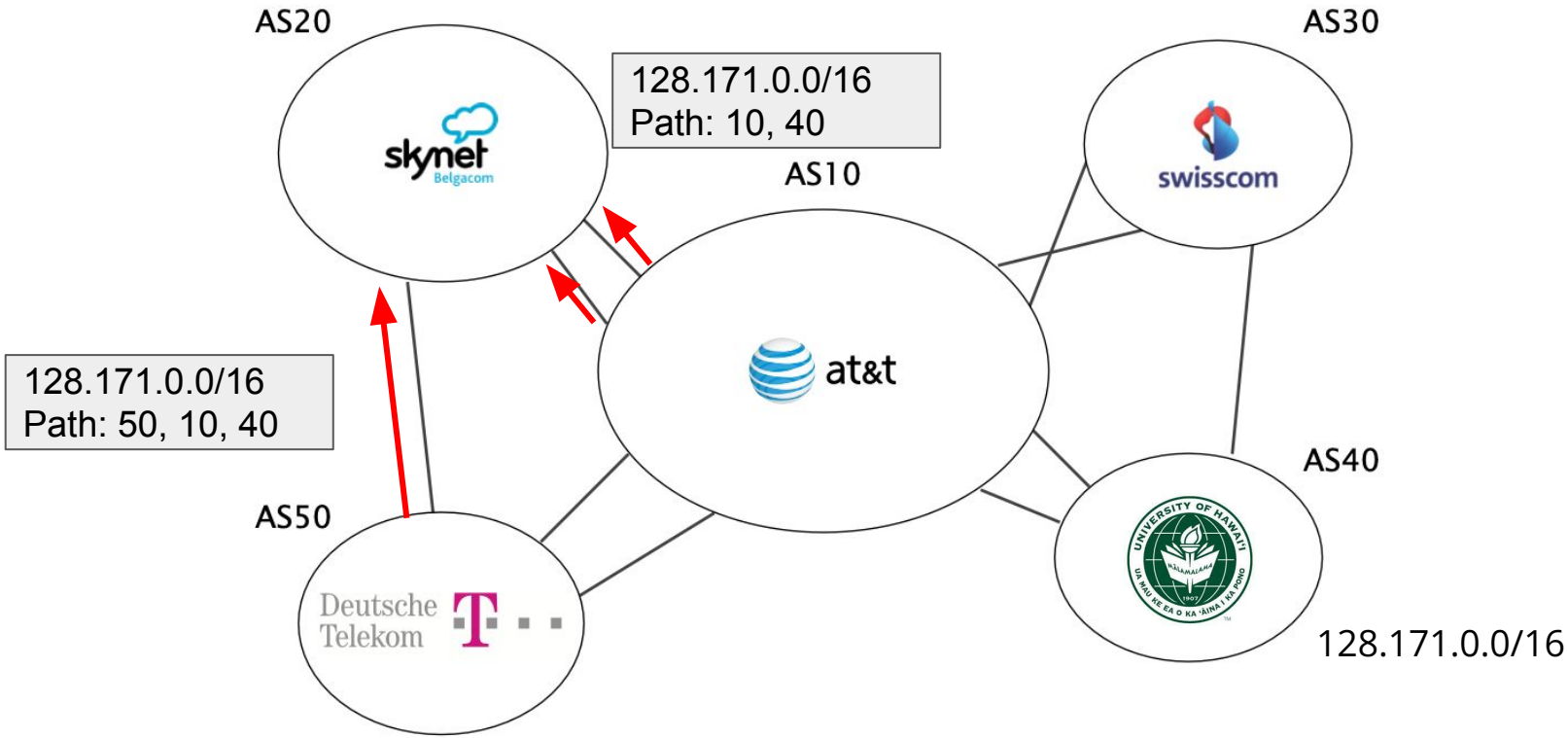
BGP Announcements Carry Complete Path Information Rather than DIstances



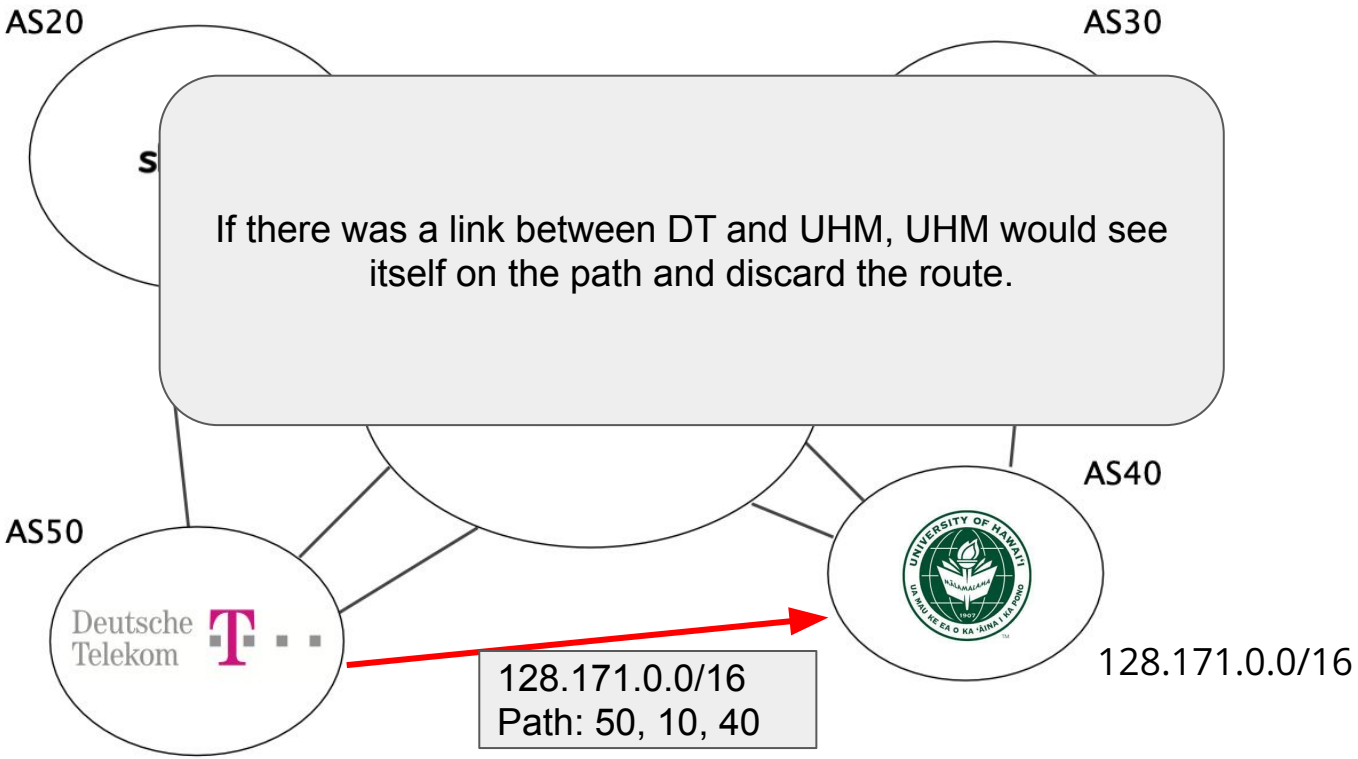
Each AS Appends Itself to the Path As it Propagates Announcements



Each AS Appends Itself to the Path As it Propagates Announcements



With a Complete Path, Loops are Easy to Detect



Life of a BGP Router: Three Steps

while true:

- receives routes from my neighbors
- select one best route for each prefix
- export the best route to my neighbors

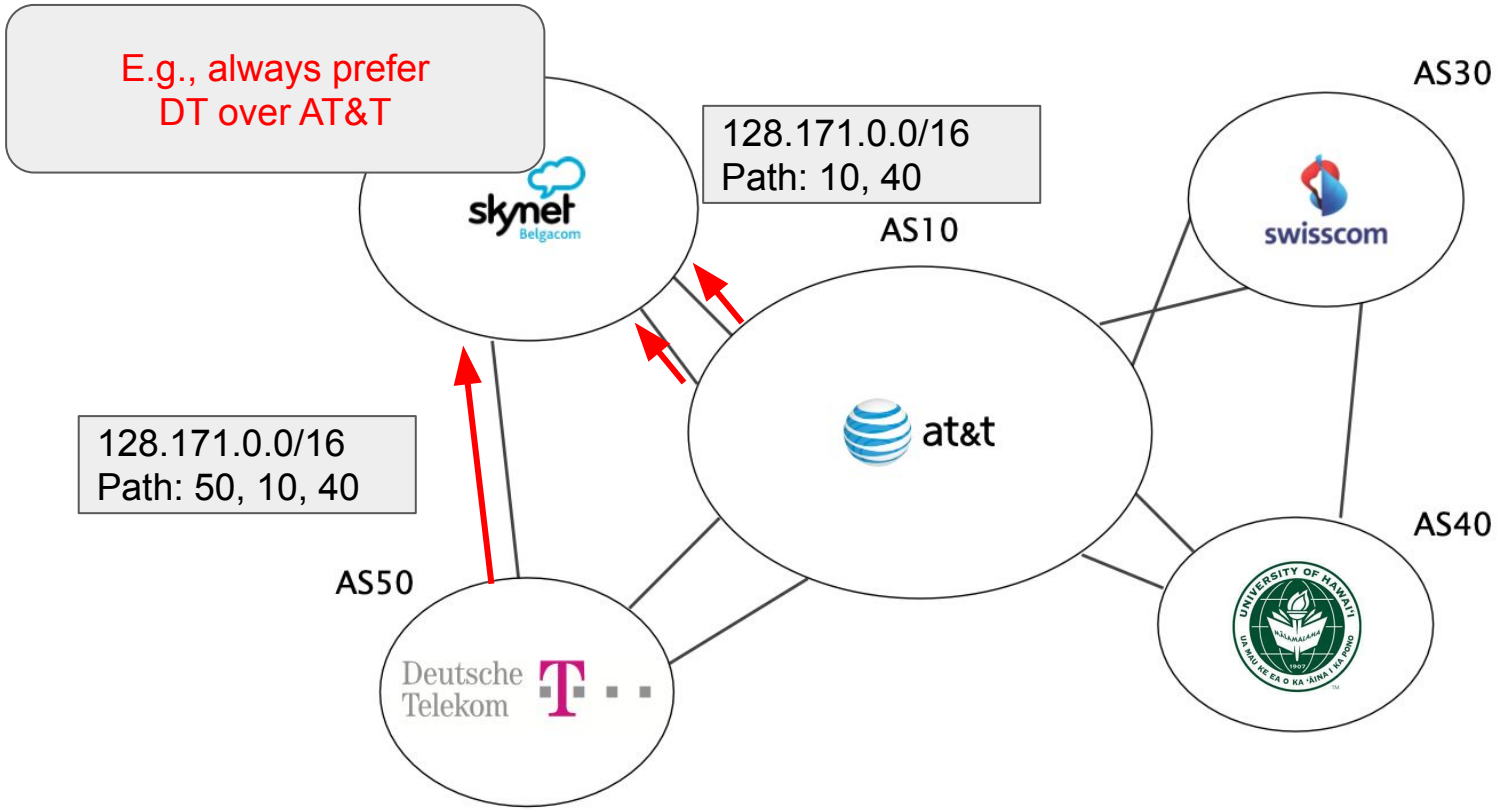
Each AS Can Apply Local Routing Policies

Each AS is **free to**

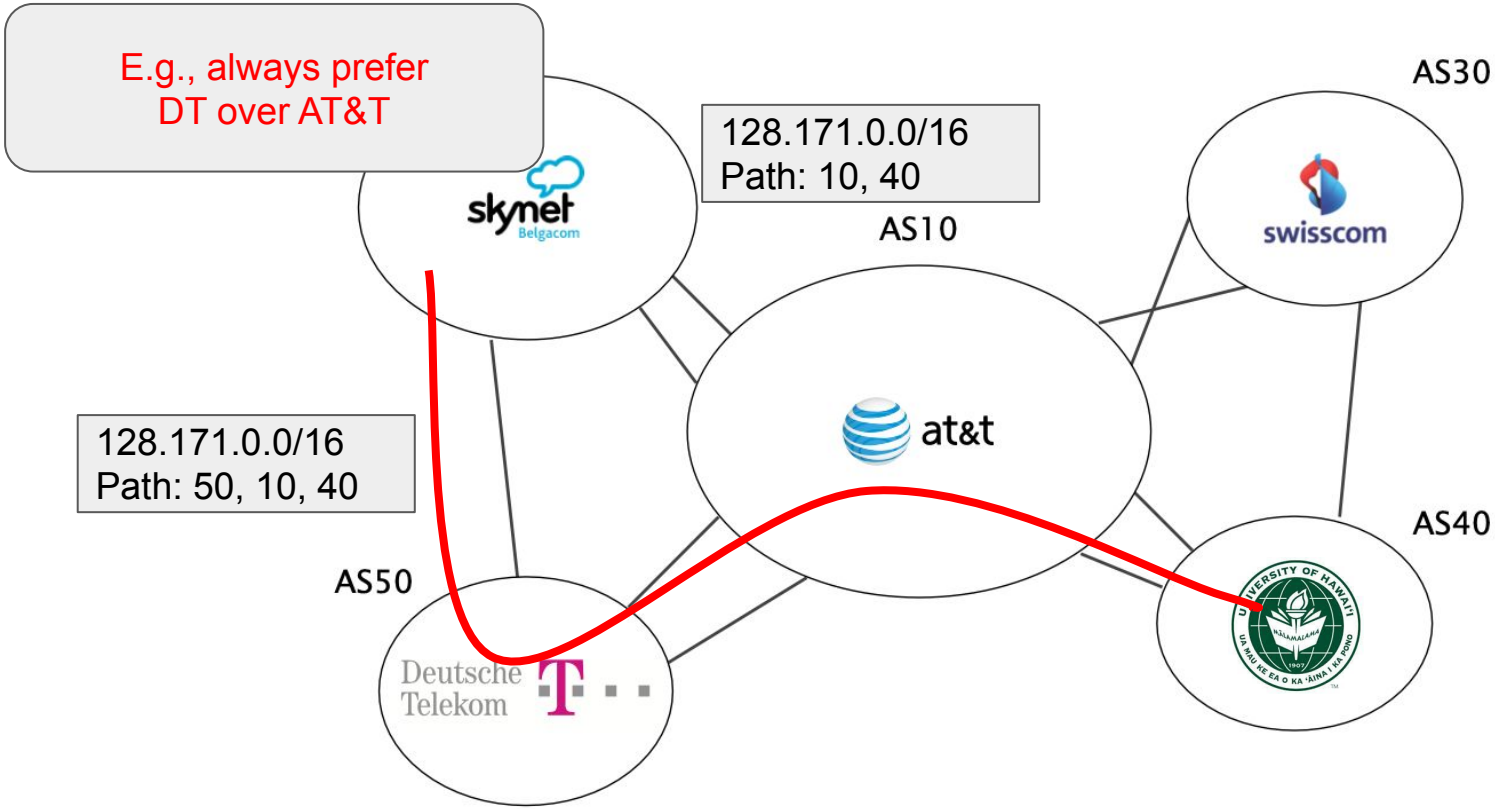
- **select and use any path**

  preferably, the cheapest one

Each AS Can Apply Local Routing Policies



Each AS Can Apply Local Routing Policies



Each AS Can Apply Local Routing Policies

Each AS is free to

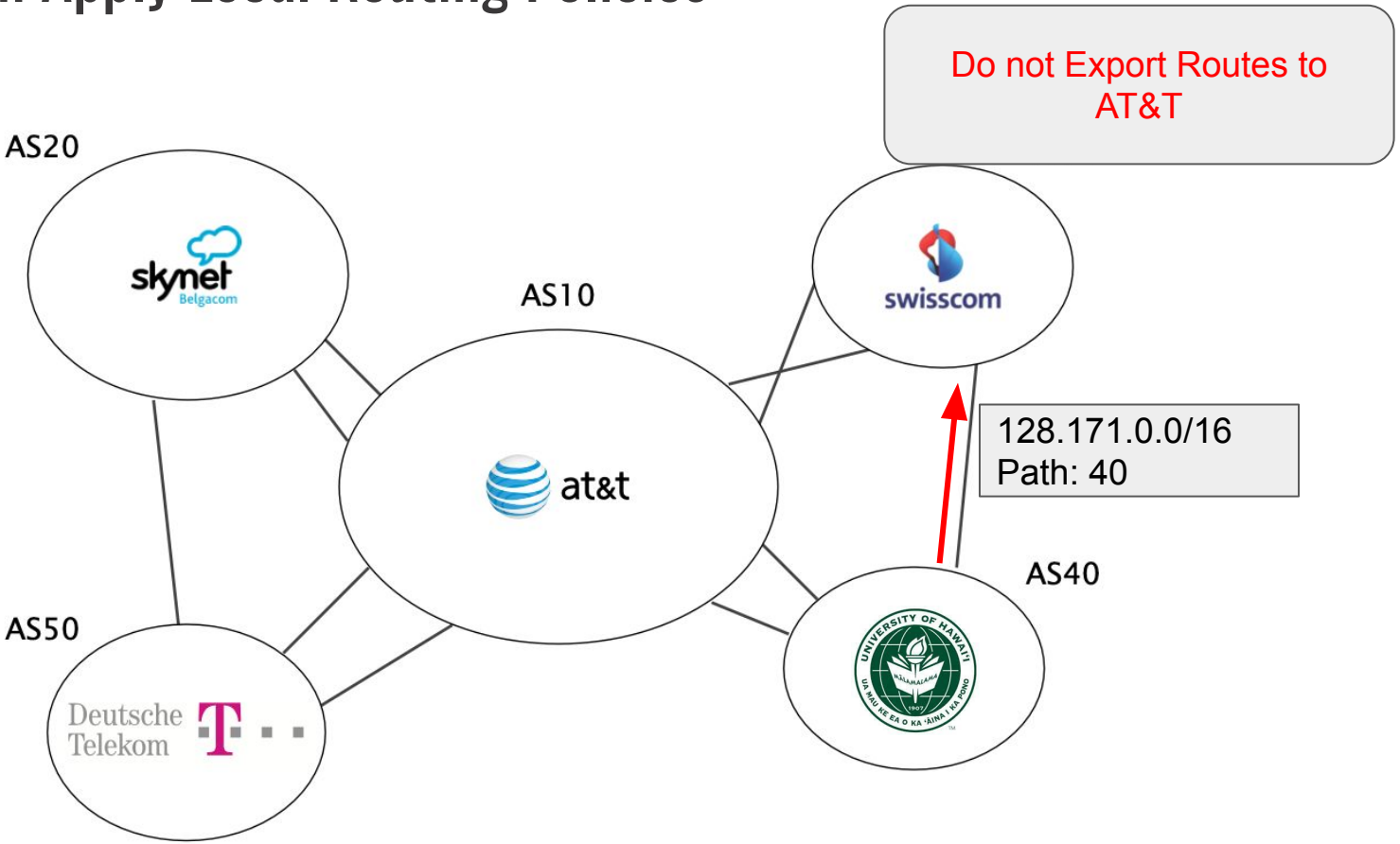
- select and use any path

  preferably, the cheapest one

- decide which path to export (if any) to which neighbor

  preferably, none to minimize carried traffic

Each AS Can Apply Local Routing Policies



Each AS Can Apply Local Routing Policies