# Advanced Networking Concepts
# Network layer part 2

Lennart Franked

November 13, 2025

Department of Computer and Electrical Engineering
Mid Sweden University

# 1. Introduction

# 2. Routing

2.1 IGP

Link state

Distance vector

2.2 EGP

Path Vector

# 3. References

# Reading instructions

This lecture and its figures are based on and adapted from:

- [2, Chapter 5]

It should be regarded as a complement to the assigned reading in the chapter above.

# Table of Contents

# Network layer functions

**Forwarding — Data plane**

Move packets from an incoming port to the appropriate outgoing port

**Routing — Control plane**

- Determines the path a packet must take from source to destination.
- Two approaches
  - per-router
  - centralized controller

# Traditional routing — Per-router

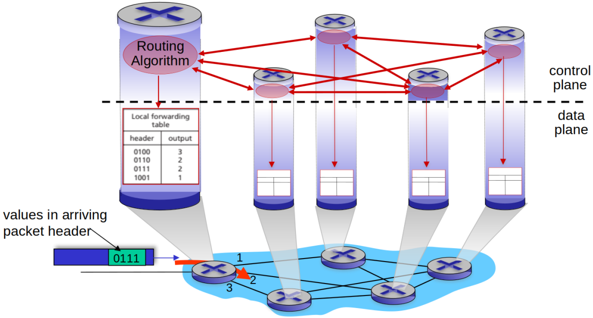



Figure: [2, Figure 4.2]

# SDN — Centralized controller

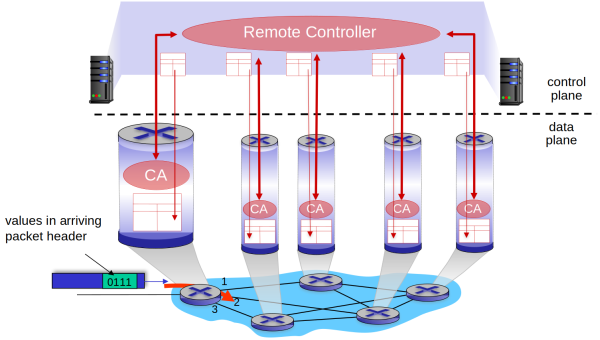
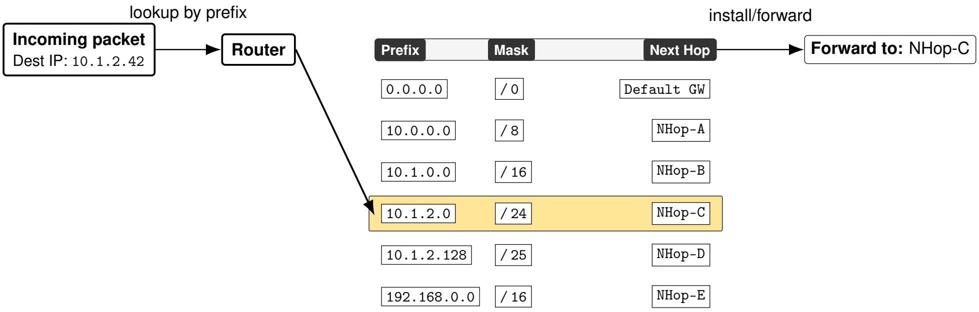



Figure: [2, Figure 4.3]

# Table of Contents

# Routing table



# Manually added routes

## Advantages

- Low overhead
- Simple
- Suitable in simple stable topologies

## Disadvantages

- Necessary to have a complete picture of the network.
- Slow to adapt the network changes.
- Not manageable in large complicated networks.
- Easy to misconfigure

# Dynamically added routes

## Advantages

- Quick to adapt to topology changes
- Handles complex networks

## Disadvantages

- Resource-intensive
- Overhead
- Can be complex to set up
- Easy to misconfigure

# Interior Gateway Protocols

- Dynamic routing protocol run within a routing domain.
- There are two main types of IGP
  - Link state
  - Distance Vector
- Which protocol to use depends on:
  1. Currently running protocols
  2. Needs
  3. Hardware

# How do we find the best path?

## Driving by road signs


Figure: Navigating by road signs, similar to distance vector routing

# How do we find the best path?

## Driving by GPS

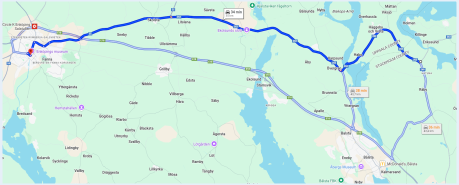


Figure: Navigating by GPS, similar to link state routing

# Link state algorithms

- All routers have a complete picture of the network in its routing domain.
- Common mechanisms to achieve this is link-flooding.
- Allows each router to run its own SPF-algorithm to find the best path to a network prefix.
- Most commonly used link state protocol is OSPF and IS-IS.

## Dijkstra's Algorithm

- The cost between all routers is known, and all link costs are non-negative.
- Calculates the lowest cost to reach each destination starting from a given source router.
- Iterative algorithm: after $k$ iterations, the lowest cost to $k$ destinations is known.

# Dijkstra's Algorithm

- $c(x, y)$ — the cost from $x$ to $y$ (default $= \infty$ if no direct link exists)
- $D(v)$ — the current known minimum cost to reach router $v$
- $p(v)$ — the predecessor router along the shortest path to $v$
- $N'$ — the set of routers for which the shortest path from the source is known

# Dijkstra's Algorithm

```
Initialization (source router x):
 N' = {x}
 For all routers v:
   if v is adjacent to x:
     D(v) = c(x,v)
   else:
     D(v) = ∞

Loop until all routers are added to N':
  Find w not in N' such that D(w) is a minimum
  Add w to N'
  Update D(v) for all v adjacent to w and not in N':
    D(v) = min(D(v), D(w) + c(w,v))
```

Listing 1: Dijkstra's algorithm (source router x) [2]

# Example: Dijkstra's Algorithm

| Step | $N'$ | D(v) p(v) | D(w) p(w) | D(x) p(x) | D(y) p(y) | D(z) p(z) |
|---|---|---|---|---|---|---|
| 0 | u (source) | 7,u | 3,u | 5,u | ∞ | ∞ |
| 1 | uw | 6,w | | 5,u | 11,w | ∞ |
| 2 | uwx | 6,w | | | 11,w | 14,x |
| 3 | uwxv | | | | 10,v | 14,x |
| 4 | uwxvy | | | | | 12,y |
| 5 | uwxvyz | | | | | |

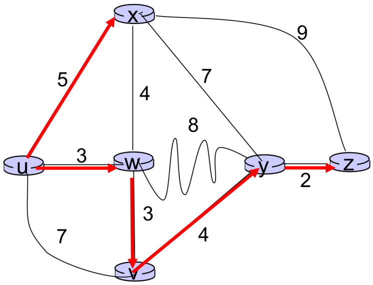



Figure: Dijkstra topology example (source router *u*) [2]

# Example: Dijkstra’s Algorithm

| Step | $N'$ | D(v) p(v) | D(w) p(w) | D(x) p(x) | D(y) p(y) | D(z) p(z) |
|---|---|---|---|---|---|---|
| 0 | u (source) | 7,u | 3,u | 5,u | $\infty$ | $\infty$ |
| 1 | uw | 6,w | | 5,u | 11,w | $\infty$ |
| 2 | uwx | 6,w | | | 11,w | 14,x |
| 3 | uwxv | | | | 10,v | 14,x |
| 4 | uwxvy | | | | | 12,y |
| 5 | uwxvyz | | | | | |

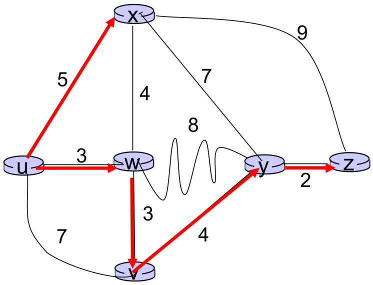



Figure: Dijkstra topology example (source router *u*) [2]

# Example: Dijkstra's Algorithm

| Step | $N'$ | D(v) p(v) | D(w) p(w) | D(x) p(x) | D(y) p(y) | D(z) p(z) |
|---|---|---|---|---|---|---|
| 0 | u (source) | 7,u | 3,u | 5,u | $\infty$ | $\infty$ |
| 1 | uw | 6,w | | 5,u | 11,w | $\infty$ |
| 2 | uwx | 6,w | | | 11,w | 14,x |
| 3 | uwxv | | | | 10,v | 14,x |
| 4 | uwxvy | | | | | 12,y |
| 5 | uwxvyz | | | | | |

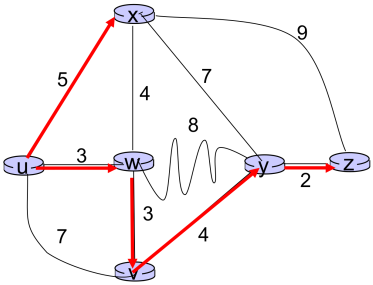



Figure: Dijkstra topology example (source router *u*) [2]

# Example: Dijkstra's Algorithm

| Step | $N'$ | D(v) p(v) | D(w) p(w) | D(x) p(x) | D(y) p(y) | D(z) p(z) |
|---|---|---|---|---|---|---|
| 0 | u (source) | 7,u | 3,u | 5,u | $\infty$ | $\infty$ |
| 1 | uw | 6,w | | 5,u | 11,w | $\infty$ |
| 2 | uwx | 6,w | | | 11,w | 14,x |
| 3 | uwxv | | | | 10,v | 14,x |
| 4 | uwxvy | | | | | 12,y |
| 5 | uwxvyz | | | | | |

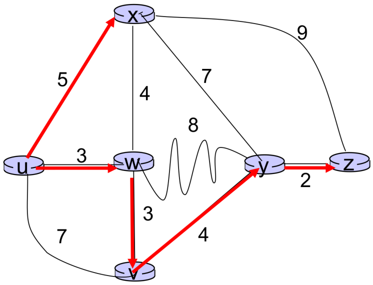



Figure: Dijkstra topology example (source router *u*) [2]

# Example: Dijkstra's Algorithm

| Step | $N'$ | D(v) p(v) | D(w) p(w) | D(x) p(x) | D(y) p(y) | D(z) p(z) |
|---|---|---|---|---|---|---|
| 0 | u (source) | 7,u | 3,u | 5,u | $\infty$ | $\infty$ |
| 1 | uw | 6,w | | 5,u | 11,w | $\infty$ |
| 2 | uwx | 6,w | | | 11,w | 14,x |
| 3 | uwxv | | | | 10,v | 14,x |
| 4 | uwxvy | | | | | 12,y |
| 5 | uwxvyz | | | | | |

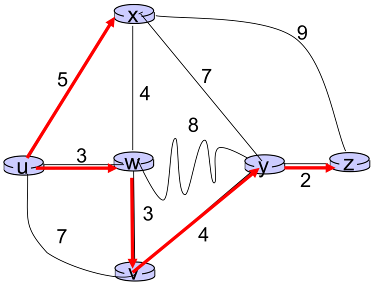



Figure: Dijkstra topology example (source router *u*) [2]

# Example: Dijkstra's Algorithm

| Step | $N'$ | D(v) p(v) | D(w) p(w) | D(x) p(x) | D(y) p(y) | D(z) p(z) |
|---|---|---|---|---|---|---|
| 0 | u (source) | 7,u | 3,u | 5,u | $\infty$ | $\infty$ |
| 1 | uw | 6,w | | 5,u | 11,w | $\infty$ |
| 2 | uwx | 6,w | | | 11,w | 14,x |
| 3 | uwxv | | | | 10,v | 14,x |
| 4 | uwxvy | | | | | 12,y |
| 5 | uwxvyz | | | | | |

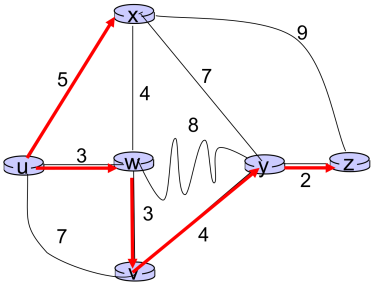



Figure: Dijkstra topology example (source router *u*) [2]

## Drawbacks

- Each router must maintain an up-to-date map of the entire network topology.
- Requires significant memory and processing resources.
- Higher computational complexity due to shortest-path calculations (e.g., Dijkstra's algorithm).
- Periodic flooding of link-state information increases control traffic.
- Susceptible to temporary inconsistencies during topology changes.
- Synchronization or flooding errors can lead to incorrect routing information.

# Distance Vector Algorithms

- Every router informs their neighbours on which network prefixes they can reach, and to what cost
- Hence distance vector
  - Direction: Which port to send packet to.
  - Magnitude: Cost to reach destination.
- Common DV protocols are RIP and EIGRP.

# Bellman-Ford

$$d_x(y) := \text{minimum cost of the path between } x \text{ and } y$$
$$d_x(y) = \min_v\{c(x,v) + d_v(y)\}$$

- $\min_v$ checks all neighbors $v$ of $x$
- $c(x,v)$ is the cost between $x$ and $v$
- $d_v(y)$ is the cost between $v$ and $y$

# Example: Bellman-Ford

$$
\begin{aligned}
d_u(z) = \min\{ & c(u,v) + d_v(z), \\
& c(u,x) + d_x(z), \\
& c(u,w) + d_w(z)\} \\
= \min\{ & 2+5,\ 1+3,\ 5+3\} = 4
\end{aligned}
$$

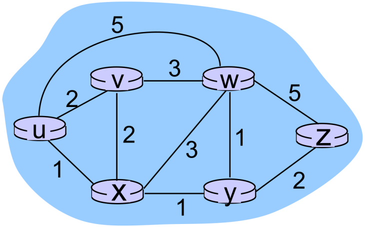


Figure: Bellman-Ford example [2]

# Bellman-Ford

$$D_x(y) := \text{estimated minimum cost from } x \text{ to } y$$

- Node $x$ maintains a list $D_x = [D_x(y) : y \in N]$
- This means that:
  - $x$ must know the cost to all its neighbors $v$: $c(x, v)$
  - For each neighbor $v$, $x$ must know its distance vector $D_v = [D_v(y) : y \in N]$

# Bellman-Ford

- All routers periodically exchange their distance vectors
- Each time a router receives a new distance vector, it updates its routing table
- For each router $y \in N$:
  $D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\}$

# Drawbacks

- Sensitive to routing loops.
- Does not know the overall network topology.
- Slow convergence after topology changes.
- Can suffer from the *count-to-infinity* problem.

# Convergence in DV

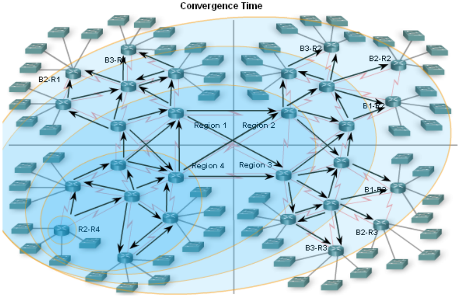



Figure: Convergence in DV [1]

# Routing Loop Example in Distance Vector Routing

- Link between A and X fails.
- A updates its table: $D_A(X) = \infty$.
- B and C have not yet learned about the failure.
- B tells A it can reach X via C (and vice versa).
- A updates its table based on outdated info, creating a routing loop.
- A and B forward packets to each other for X forming a routing loop.

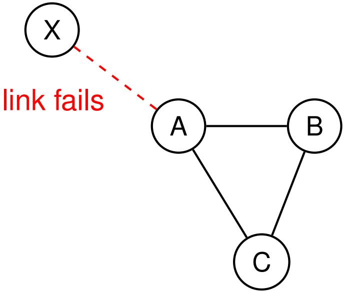



Figure: Routing loop after link failure.

# Routing at Scale

- We don’t have a flat network with identical routers
- There are billions of addresses, it is not reasonable for a router to know them all.
- Different organizations with different needs, policies, and budget constraints.

# Autonomous Systems

Let's scale this up!

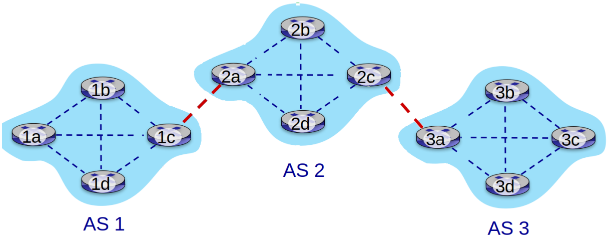



Figure: [2, Figure 5.8]

# IGP vs EGP

## IGP

- Routing within an Autonomous system (intra-AS routing)
- Focus on network efficiency
- All routers are trusted
- All packets originate from within.
- Different routing protocols can be run within an AS (Not to confuse with that different RP can communicate with each other)

## EGP

- Routing between autonomous systems (inter-AS routing)
- Externally sourced or destined packets.
- trust only peering ASes.
- Focus on economy and politics
- Same routing protocol must be run by all AS.

# Border Gateway Protocol

- “The de facto inter-domain routing protocol”
- Obtain destination network reachability info from neighboring ASes
- Determine routes to other networks based on reachability information and policy
- Propagate reachability information to all AS-internal routers (iBGP)
- Advertise (to neighboring networks) destination reachability info

# eBGP iBGP

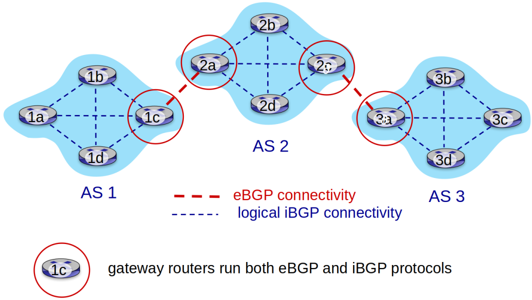
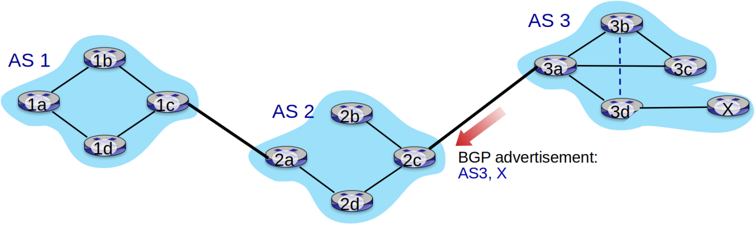



Figure: eBGP, iBGP [2]

# BGP

- two BGP routers ("peers") exchange BGP messages over semi-permanent TCP connection
- advertising paths to different destination network prefixes (BGP is a "path vector" protocol)
- when AS3 gateway 3a advertises path AS3,X to AS2 gateway 2c it promises it will forward datagrams towards X.

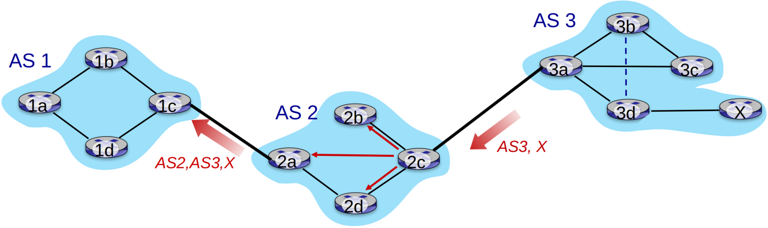


## Path Vector

### BGP Advertisements

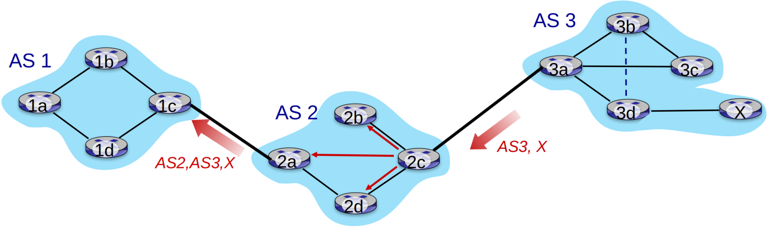
- Network prefix
- Attributes (13 attributes)
  - AS_PATH
  - NEXT-HOP
- Policy-based routing
  - gateway receiving route advertisement uses import policy to accept/decline path (e.g., never route through AS Y).
  - AS policy also determines whether to advertise path to other neighboring ASes

# BGP Path



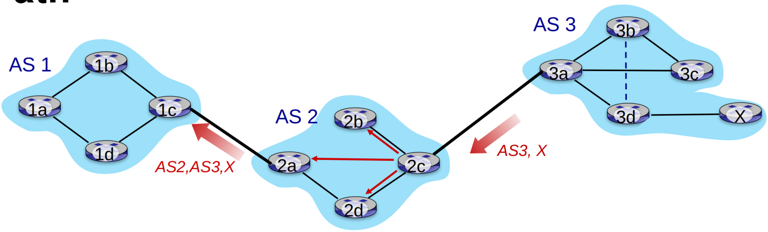

- AS2 router 2c receives path advertisement AS3,X (via eBGP) from AS3 router 3a
- Based on AS2 policy, AS2 router 2c accepts path AS3,X, propagates (via iBGP) to all AS2 routers
- Based on AS2 policy, AS2 router 2a advertises (via eBGP) path *AS2, AS3, X* to AS1 router 1c

# BGP Path



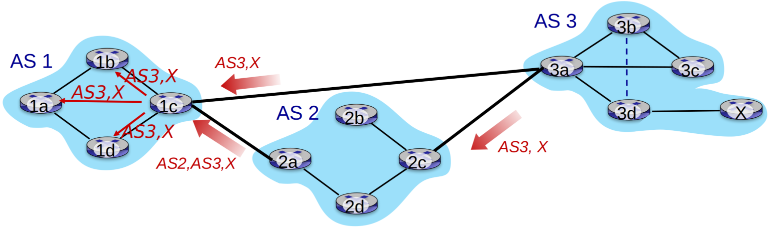

- AS2 router 2c receives path advertisement AS3,X (via eBGP) from AS3 router 3a
- Based on AS2 policy, AS2 router 2c accepts path AS3,X, propagates (via iBGP) to all AS2 routers
- Based on AS2 policy, AS2 router 2a advertises (via eBGP) path *AS2, AS3, X* to AS1 router 1c

# BGP Path



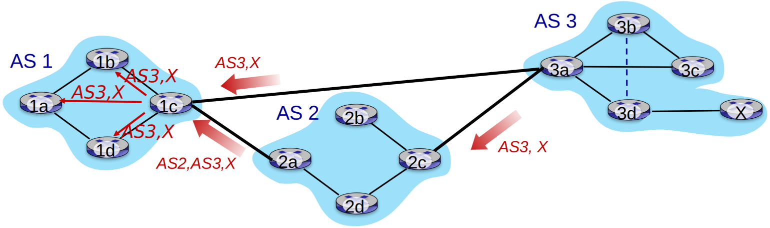

- AS2 router 2c receives path advertisement AS3,X (via eBGP) from AS3 router 3a
- Based on AS2 policy, AS2 router 2c accepts path AS3,X, propagates (via iBGP) to all AS2 routers
- Based on AS2 policy, AS2 router 2a advertises (via eBGP) path *AS2, AS3, X* to AS1 router 1c

# BGP Multi Path



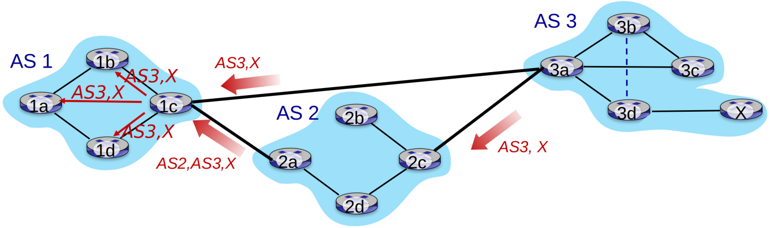

- AS1 gateway router 1c learns path AS2,AS3,X from 2a
- AS1 gateway router 1c learns path AS3,X from 3a
- Based on policy, AS1 gateway router 1c chooses path AS3,X and advertises path within AS1 via iBGP

# BGP Multi Path



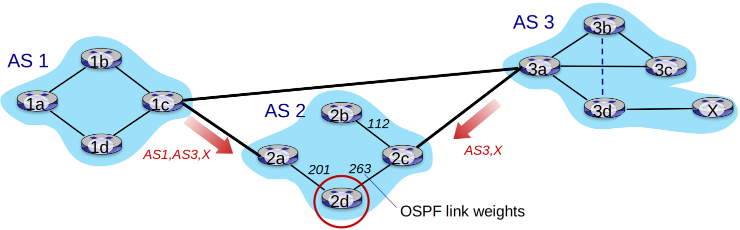

- AS1 gateway router 1c learns path AS2,AS3,X from 2a
- AS1 gateway router 1c learns path AS3,X from 3a
- Based on policy, AS1 gateway router 1c chooses path AS3,X and advertises path within AS1 via iBGP

# BGP Multi Path



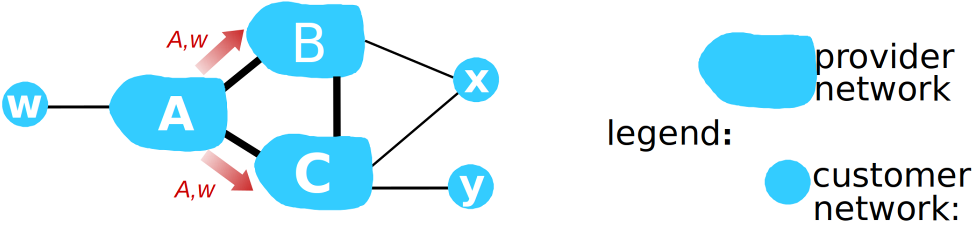

- AS1 gateway router 1c learns path AS2,AS3,X from 2a
- AS1 gateway router 1c learns path AS3,X from 3a
- Based on policy, AS1 gateway router 1c chooses path AS3,X and advertises path within AS1 via iBGP

# Hot potato routing




- 2d learns (via iBGP) it can route to X via 2a or 2c
- Hot Potato Routing: choose local gateway that has least intra-domain cost.
- IGP metrics override EGP considerations

# Inter-AS Routing




- An AS owner only wants to route traffic to/from its network.
- A advertises path Aw to B and to C
- B and C chooses not to advertise Aw to each other
- B will inform x about BAw
- C will inform x,y about CAw

# Summary Intra- Inter-AS routing

- Policy
  - Inter-AS: AS owner wants control over how and what traffic is routed
  - Intra-AS: No issue since all data is its own.
- Scale
  - Hierarchical routing saves table sizes
- Performance
  - Intra-AS: Focuses on performance
  - Inter-AS: Policy is more important than performance

# Table of Contents

# References

[1] Cisco Networking Cisco Networking Academy. *Scaling Networks V6 Companion Guide*. Companion Guide. Cisco Press, 2017. ISBN: 9781587134340.

[2] James F. Kurose and Keith W. Ross. *Computer networking : a top-down approach*. 7th ed. Boston: Pearson/Addison Wesley, 2017. ISBN: 9781292153599.