Master Course Computer Networks IN2097

Prof. Dr.-Ing. Georg Carle
Christian Grothoff, Ph.D.
Stephan Günther

Chair for Network Architectures and Services
Department of Computer Science
Technische Universität München
http://www.net.in.tum.de

Routing

Short note on pronunciation of the word “routing”

- [‘ru:tıŋ] /r-oo-ting/ = British English
- [‘raʊdıŋ] /r-ow-ding/ = American English
- Both are correct!

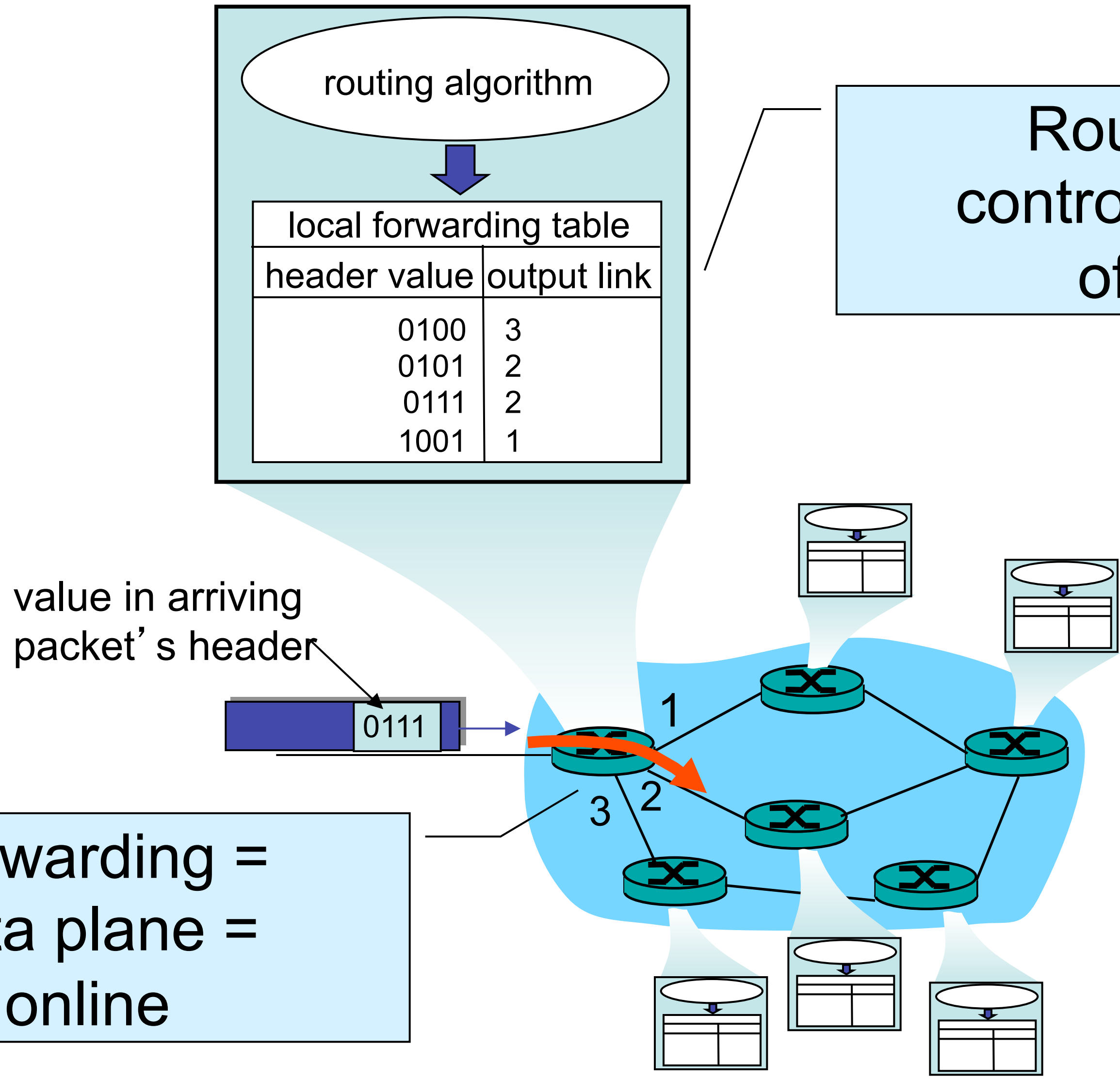
Topics

- Routing and forwarding
- Routing algorithms recapitulated
  - Link state
  - Distance Vector
  - Path Vector
- Intradomain routing protocols
  - RIP
  - OSPF
- Interdomain routing
  - Hierarchical routing
  - BGP
- Business considerations
  - Policy routing
  - Traffic engineering
- Routing security
- Multicast routing

Routing ≠ Forwarding

- Routing:
  - The process of determining the best path for a specific type of packets (usually: all packets with same destination) through the network
  - Analogy: Read street map, plan journey
  - Decentralized routing: performed jointly by all routers *of a network* by exchanging many messages
  - Alternative: centralization, c.f. Software Defined Networking - SDN
- Forwarding:
  - The process of a router relaying a packet to a neighbouring router. (Choice of the neighbouring router depends on the previous routing protocol calculations)
  - Performed by one router on one packet
  - Analogy: Read a street sign and determine if we should take the next exit
- In practice, this distinction is often ignored
  - "If router A routes packet X, then …"
  - Actually, it doesn't – it *forwards* X.

Signalling Plane and Data Plane



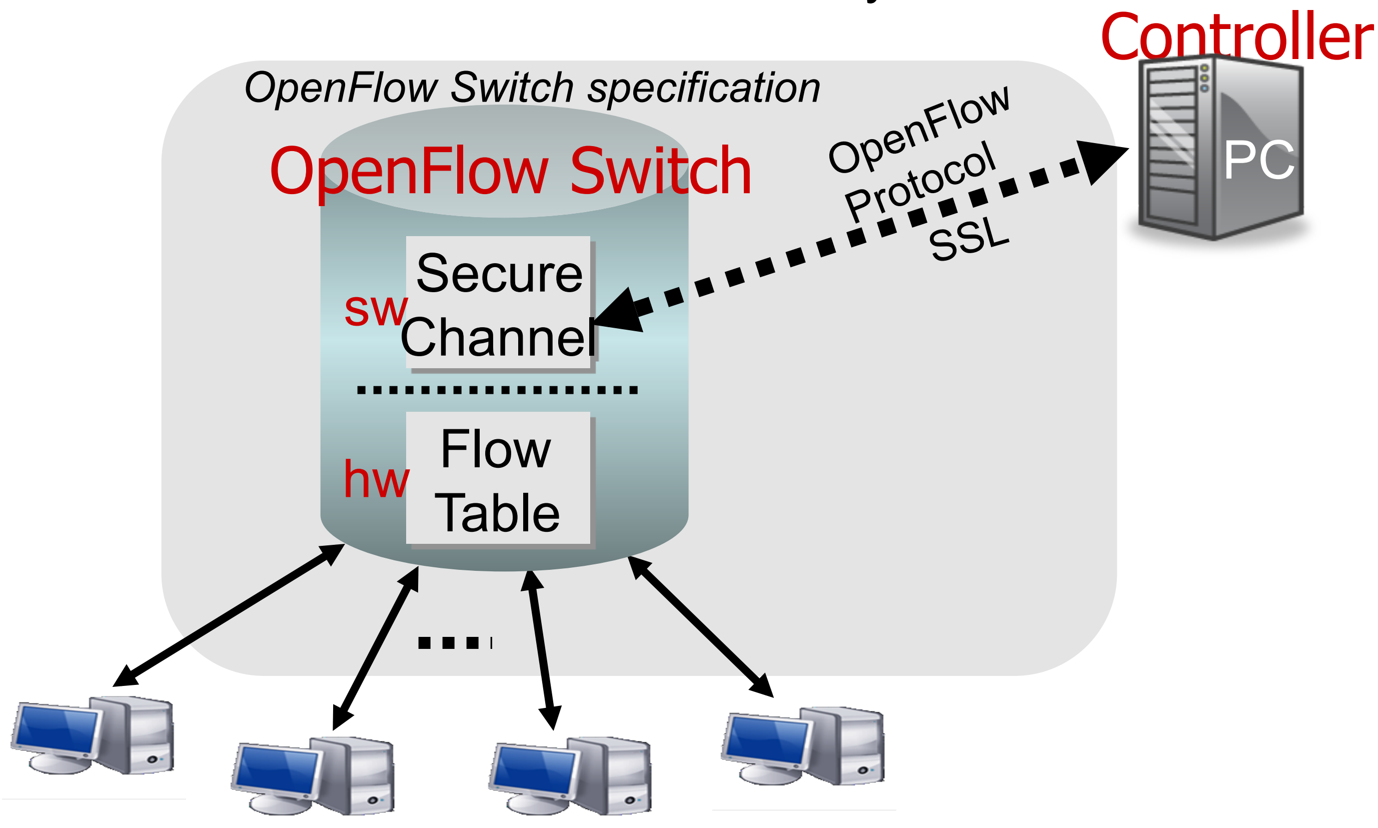
Routing =
control plane =
offline

Forwarding =
data plane =
online

Software Defined Networking

- Example: OpenFlow Switch architecture, Stanford University
- Concept: separation of switch fabric, and switch control
- Allows for cheap switches, centrally controlled by switch manager

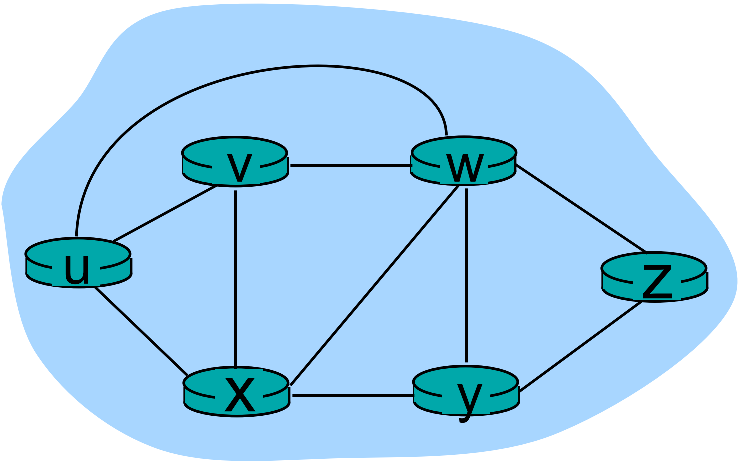
⇨ Assessment: suitable for low-latency data center communication



The Stanford Clean Slate Program

http://cleanslate.stanford.edu

Graph Abstraction



Graph: G = (N,E)

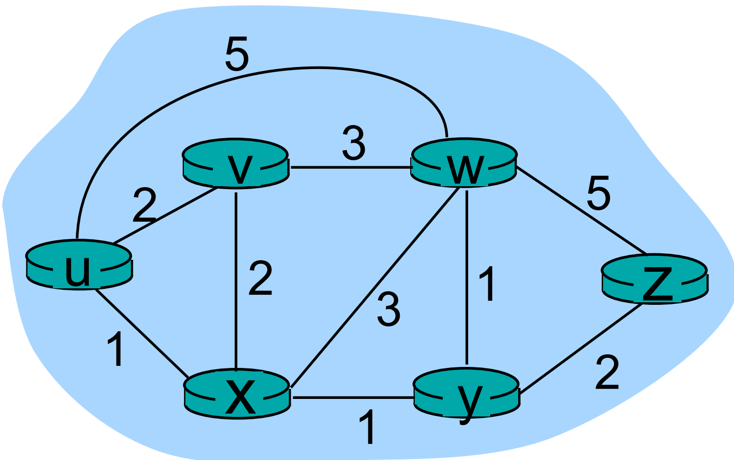
N = nodes = set of routers = { u, v, w, x, y, z }

E = edges = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Remark: Graph abstraction is also useful in other network contexts

Example: P2P network, where N is set of peers and E is set of TCP connections

Graph Abstraction: Costs



- c(x,x') =: cost of link (x,x')
  e.g.: c(w,z) = 5

- cost could always be 1,
- or inversely related to bandwidth,
- or inversely related to congestion

Cost of path $(x_1, x_2, x_3,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Question: What's the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

Routing Algorithm Classification

Global or decentralized information?

Global:

- All routers have complete topology and link cost info
- *link state* algorithms (L-S)

Decentralized:

- Router only knows physically-connected neighbours and link costs to neighbours
- Iterative process of computation = exchange of info with neighbours
- *distance vector* algorithms (D-V)
- Variant: *path vector* algorithms

Static or dynamic?

Static:

- Routes
  - do not change, or
  - change *slowly* over time

Dynamic:

- Routes change more quickly
  - periodic updates
  - in response to topology or link cost changes

A Broader Routing Classification

- Type of algorithm: Link State, Distance Vector, Path Vector, …
- Scope:
  - Intra-domain
  - Inter-domain
  - Special purpose (e.g., sensor network)
- Type of target set: unicast vs. multicast
- Type of reaction: "static" vs. dynamic/adaptive
  - Warning: "Dynamic routing" is a fuzzy term:
    a) Dynamic ≔ reacts to topology changes (state of the art)
    b) Dynamic ≔ reacts to traffic changes (even better, but most protocols don't do that!)
- Trigger type:
  - Permanent routing (standard)
  - On-demand routing: only start routing algorithm if there is traffic to be forwarded (e.g., certain wireless ad-hoc networks)

A Link-State Routing Algorithm

- Network topology and link costs made known to each node
  - Accomplished via *link state broadcasts*
  - All nodes have same information (…after all information has been exchanged)
- Each node independently computes least-cost paths from one node ("source") to all other nodes
  - Usually done using Dijkstra's shortest-path algorithm
    - c.f. algorithms & data structures lecture/textbook
    - n nodes in network ⇒ $O(n^2)$ or $O(n \log n)$
  - Gives forwarding table for that node
- Result:
  - All nodes have the same information,
  - … thus calculate the same shortest paths,
  - … hence obtain consistent forwarding tables

Distance Vector Algorithm

- No node knows entire topology
- Nodes only communicate with neighbours (i.e., no broadcasts)
- Nodes *jointly* calculate shortest paths
  - Iterative process
  - Algorithm == protocol
- Distributed application of Bellman-Ford algorithm
  - c.f. algorithms&data structures lecture/textbook

Distance Vector Algorithm

Bellman-Ford Equation (dynamic programming)

Let

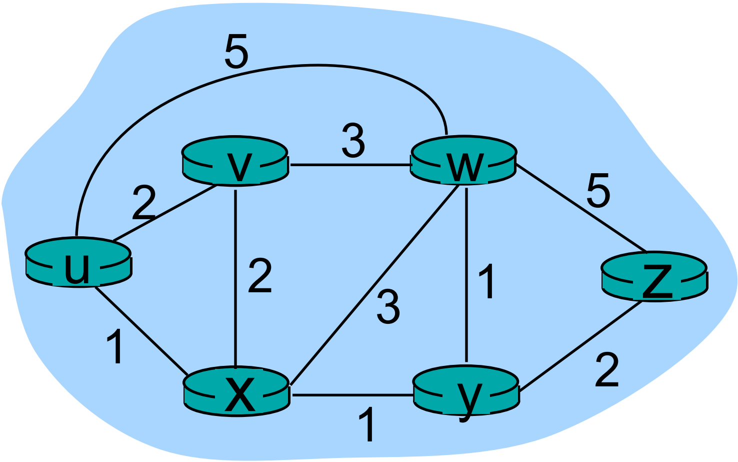
- $c(x,y) \coloneqq$ cost of edge from x to y
- $d_x(y) \coloneqq$ cost (or distance) of least-cost path from x to y
- Set to ∞ if no path / no edge available

Then

$$d_x(y) = \min \{c(x,v) + d_v(y) \}$$

where min is taken over all neighbours v of x

Bellman-Ford Example



We can see that

$d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

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

Node that calculated minimum is next hop in shortest path
→ forwarding table

Distance Vector Algorithm

- Define $D_x(y)$ ≔ estimate of least cost from x to y
- Node x knows cost to each neighbour v: $c(x,v)$
- Node x maintains distance vector $\overrightarrow{D_x}$ ≔ [$D_x(y)$: $y \in N$]
  (N ≔ set of nodes)
- Node x also maintains copies of its neighbours' distance vectors
  - Received via update messages from neighbours
  - For each neighbour v,
    x knows $\overrightarrow{D_v}$ = [$D_v(y)$: $y \in N$]

Distance vector algorithm (4)

Basic idea:

- From time to time, each node sends its own distance vector estimate D to its neighbours
  - Asynchronously
- When a node x receives new distance vector from neighbour, it updates its own distance vector using B-F equation:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- Under good-natured conditions, these estimates $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Distance Vector Algorithm (5)

Iterative, asynchronous:
Each local iteration caused by:
- Local link cost change
- DV update message from neighbour

Distributed:
- Each node notifies neighbours *only* when its DV changes
  - neighbours then notify their neighbours if this caused *their* DV to change
  - etc.

Usually some waiting delay between consecutive updates

Each node:

Forever:

wait for (change in local link cost *or* message arriving from neighbour)

recompute estimates

if (DV to any destination has changed) { *notify* neighbours }

Distance Vector Algorithm (6)

node x table

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

node y table

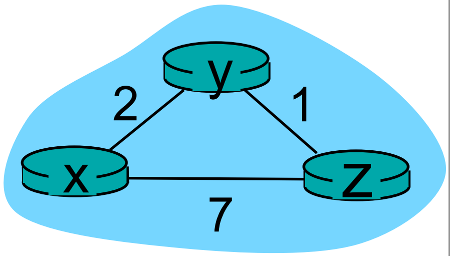
cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0



time

$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\}$
$= \min\{2+1 , 7+0\} = 3$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

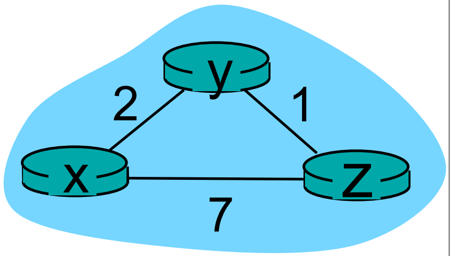
node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

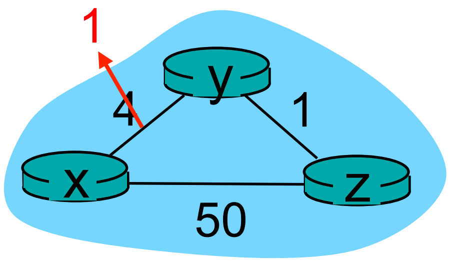
time →



Distance Vector: Link Cost Changes (1)

Link cost changes:

- Node detects local link cost change
- Updates routing info, recalculates distance vector
- If DV changes, notify neighbours



"good news travels fast"

At time t_0, *y* detects the link-cost change, updates its DV, and informs its neighbours.

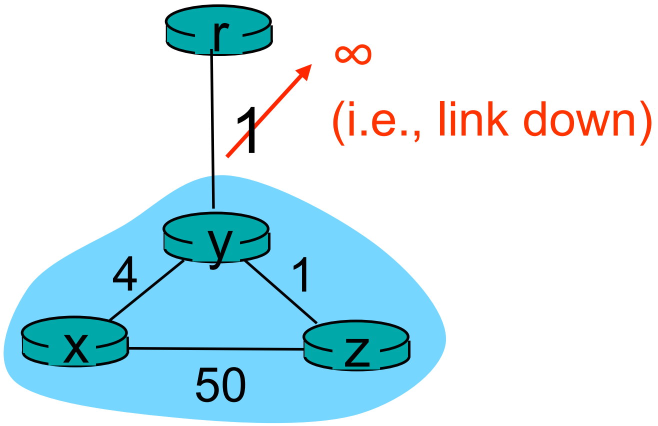
At time t_1, *z* receives the update from *y* and updates its table. It computes a new least cost to *x* and sends its neighbours its new DV.

At time t_2, *y* receives *z*'s update and updates its distance table.

y's least costs do not change and hence *y* does *not* send any message to *z*.

Distance Vector: link cost changes (2)

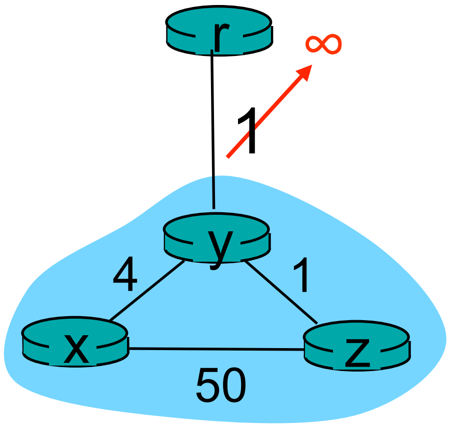
- But: bad news travels slow
- In example: Many iterations before algorithm stabilizes!
  1. Cost increase for *y*→*r*:
     - *y* consults DV,
     - *y* selects "cheaper" route via *z* (cost 2+1 = 3),
     - Sends update to *z* and *x* (cost to *r* now 3 instead of 1)
  2. *z* detects cost increase for path to *r*:
     - was 1+1, is now 3+1
     - Sends update to *y* and *x* (cost to *r* now 4 instead of 2)
  3. *y* detects cost increase, sends update to *z*
  4. *z* detects cost increase, sends update to *y*
  5. ….



- Symptom: "count to infinity" problem

Distance Vector: Problem Solutions…

- Finite infinity: Define some number to be ∞
  - in RIP: ∞ ≔ 16, see in RFC 1058
- Split Horizon:
  - Tell to any neighbour that is part of a best path to a destination that the destination cannot be reached
  - ⇨ If *z* routes through *y* to get to *r* *z* tells *y* that its own (i.e., *y*' s) distance to *r* is infinite (so *y* won' t route to *r* via *z*)
- Poisoned Reverse:
  - In addition, *actively* advertise a route as unreachable to the neighbour from which the route was learned
- (Warning: Terms often used interchangeably!)



…that only half work

- Mechanisms can be combined
- Both mechanisms can significantly increase number of routing messages
- Often help, but cannot solve all problem instances
  - Think yourselves: Come up with a topology where this does not help

Comparison of LS and DV algorithms

Message complexity

- LS: with n nodes, E links, $O(nE)$ messages sent
- DV: exchange between neighbours only
  - convergence time varies

Speed of Convergence

- LS: $O(n^2)$ algorithm requires $O(nE)$ messages
  - may have oscillations if link cost is based on link traffic
- DV: convergence time varies
  - may be routing loops
  - count-to-infinity problem

Robustness: what happens if router malfunctions?

LS:

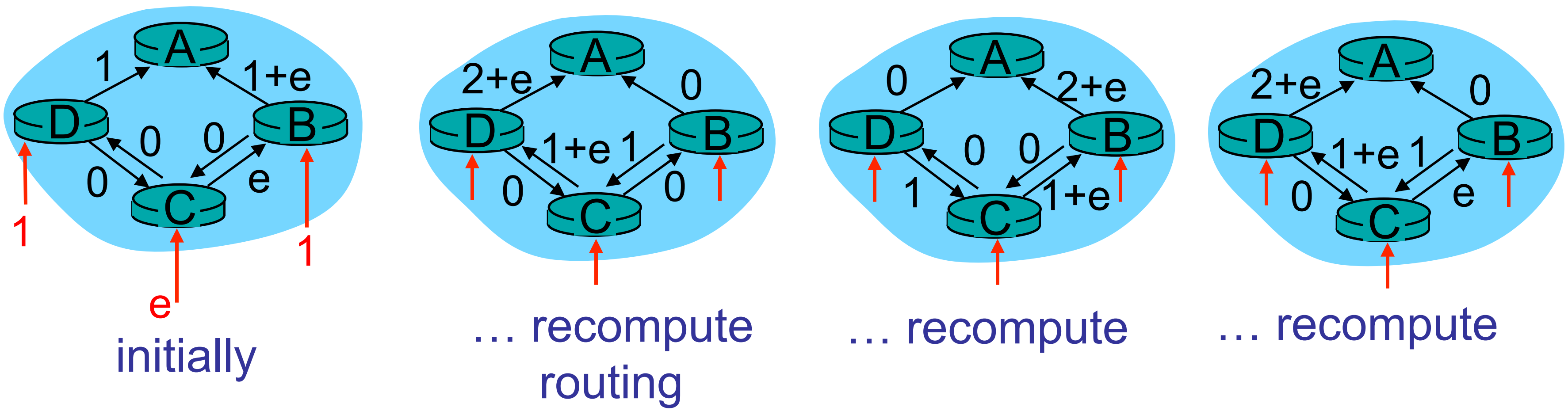
- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagates through network

Dynamic (i.e., traffic-adaptive) routing?

- Dangerous: Oscillations possible!
- e.g., link cost = amount of carried traffic



- Why is this a bad thing?
  - Possibly sub-optimal choice of paths (as in example above)
  - Additional routing protocol traffic in network
  - Increased CPU load on routers
  - Inconsistent topology information during convergence: worst! (why?)

Inconsistent topology information

- Typical causes (not exhaustive)
  - One router finished with calculations, another one not yet
  - Relevant information has not yet reached entire network
    - LS: Broadcasts = fast
    - DV: Receive message, calculate table, inform neighbours: slow
  - DV: Count-to-infinity problem
  - LS: Different algorithm implementations!
  - LS: Problem if there is no clear rule for handling equal-cost routes
- Possible consequences?
  - Erroneously assuming some destination is not reachable
  - Routing loops
    - What happens when there is a routing loop?

Path Vector protocols

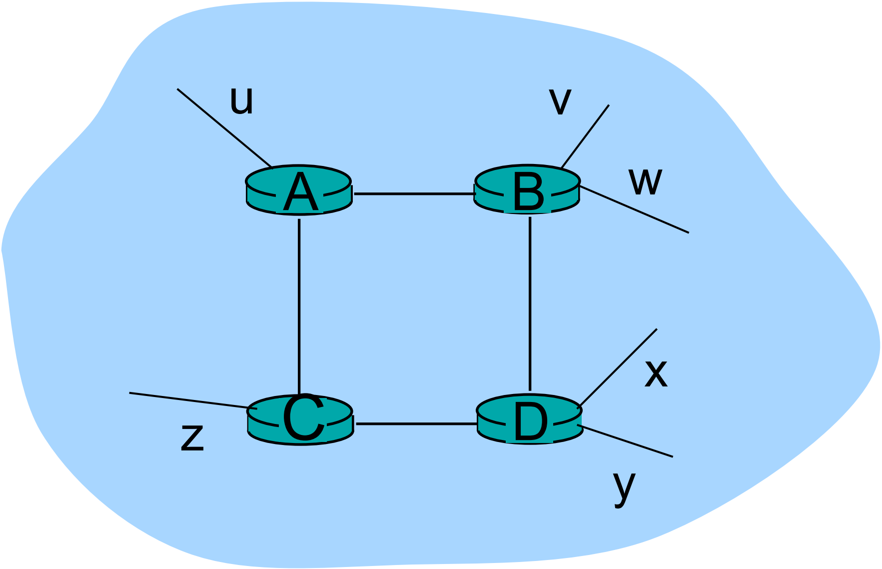
- Problem with D-V protocol:
  Path cost is “anonymous” single number; does not contain any topology information
- Path Vector protocol:
  - For each destination, advertise entire path (=sequence of node identifiers) to neighbours
  - Cost calculation can be done by looking at path
    - E.g., count number of hops on the path
  - Easy loop detection: Does my node ID already appear in the path?
- Not used very often
  - only in BGP …
  - … and BGP is much more complex than just paths

Intra-AS Routing

- Also known as Interior Gateway Protocols (IGP)
- Most common Intra-AS routing protocols:
  - RIP: Routing Information Protocol — DV (typically small systems)
  - OSPF: Open Shortest Path First — hierarchical LS (typically medium to large systems)
  - IS-IS: Intermediate System to Intermediate System — hierarchical LS (typically medium-sized ASes)
  - (E)IGRP: (Enhanced) Interior Gateway Routing Protocol (Cisco proprietary) — hybrid of LS and DV

RIP (Routing Information Protocol)

- Distance vector algorithm
- Included in BSD-UNIX Distribution in 1982
- Distance metric: # of hops (max = 15 hops, ∞ ≔ 16)
- Sometimes still in use by very small ISPs



From router A to subnets:

destination	hops
u	1
v	2
w	2
x	3
y	3
z	2

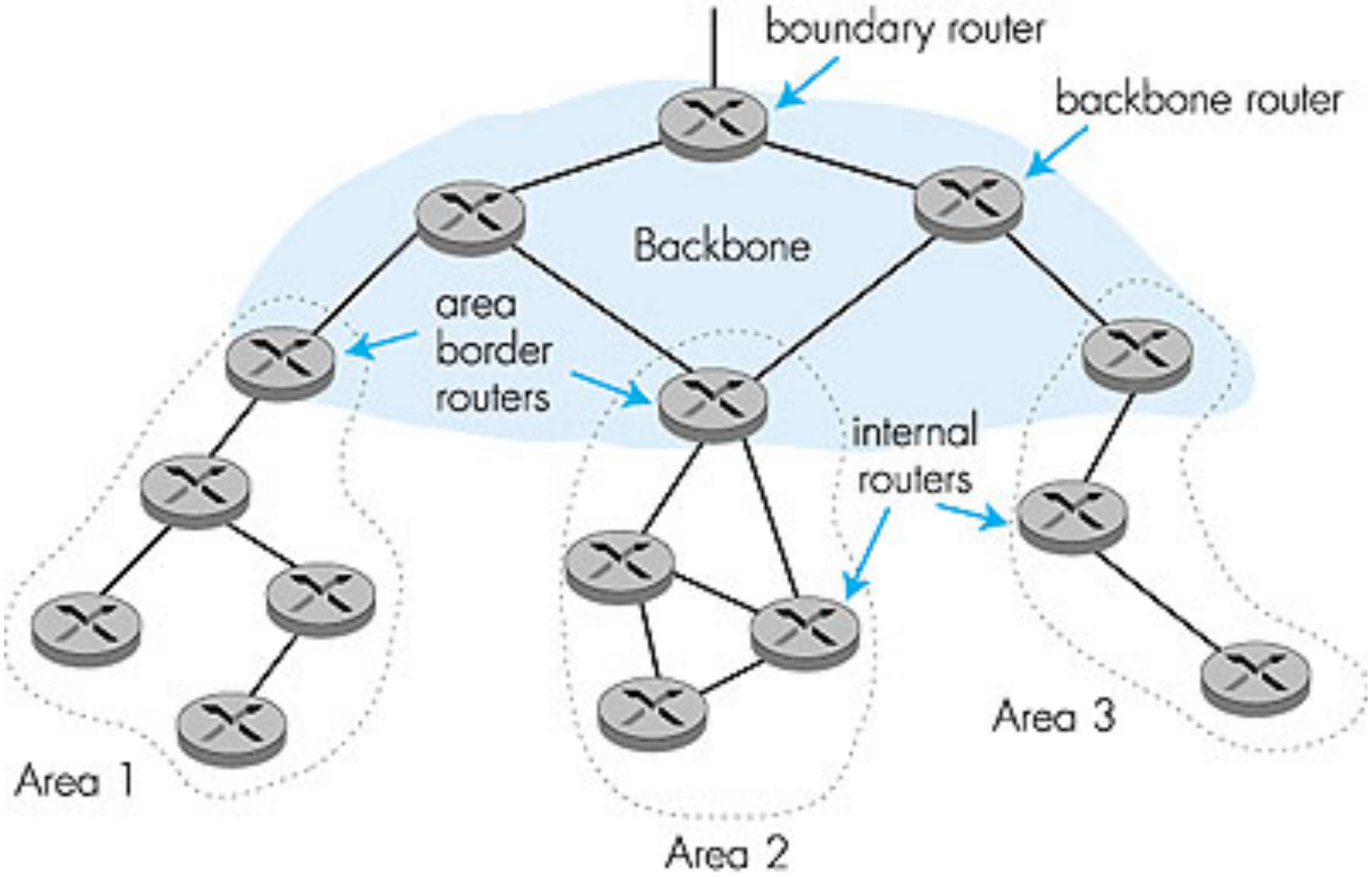
OSPF (Open Shortest Path First)

- "Open": publicly available (vs. vendor-specific, e.g., EIGRP = Cisco-proprietary)
- Uses Link State algorithm
  - LS packet dissemination (broadcasts)
  - Unidirectional edges ($\Rightarrow$costs may differ by direction)
  - Topology map at each node
  - Route computation using Dijkstra's algorithm
- OSPF advertisement carries one entry per neighbour router
- Advertisements disseminated to entire AS (via flooding)
  - (exception: hierarchical OSPF, see next slides)
  - carried in OSPF messages directly over IP (rather than TCP or UDP)

OSPF "Advanced" Features (not in, e.g., RIP)

- Security: all OSPF messages authenticated (to prevent malicious intrusion)
- Multiple same-cost paths allowed (only one path in RIP): *ECMP* (equal-cost multipath) for link load balancing
- For each link, multiple cost metrics for different Type of Service (TOS):
  e.g., satellite link cost set to "low" for best effort, but to "high" for real-time traffic (e.g. for telephony traffic)
- Integrated unicast *and* multicast support:
  - Multicast OSPF (MOSPF)
  - Uses same topology data base as OSPF ➔ less routing protocol traffic
- Hierarchical OSPF in large domains
  - Significantly reduces number of broadcast messages

Hierarchical OSPF



Hierarchical OSPF

- OSPF *can* create a two-level hierarchy
  - (similar, but not identical to to inter-AS and intra-AS routing within an AS)
- Two levels: local *areas* and the *backbone*
  - Link-state advertisements only within local area
  - Each node has detailed area topology;
    but only knows coarse direction to networks in other areas (shortest path to border router)
- *Area border routers:* "summarize" distances to networks in own area; advertise distances to other Area Border and Boundary routers
- *Backbone routers:* run OSPF routing limited to backbone
- *Boundary routers:* connect to other ASes
  - "The outside world" ≈ another area

Hierarchical Routing in the Internet

Our routing study thus far = idealisation

- all routers identical
- network "flat"

… *not* true in practice!

Scale = billions of destinations:

- Cannot store all destinations in routing tables
- Routing table exchange would swamp links
- Thousands of OSPF Areas? Would not scale!

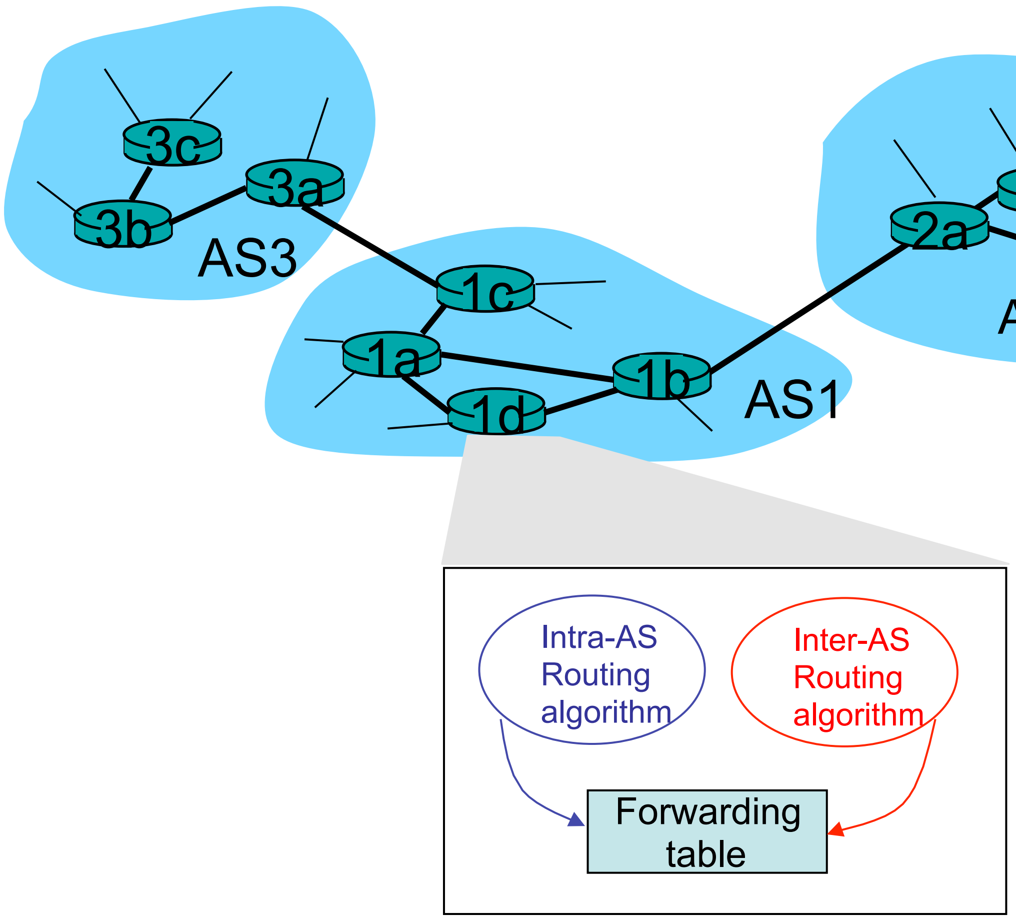
Administrative autonomy

- Internet = network of networks
- Each network admin may want to control routing in its own network — no central administration!

Hierarchical Routing

- Aggregate routers into regions called
  "autonomous systems" (short: AS; plural: ASes)
  - One AS ≈ one ISP / university
- Routers in same AS run same routing protocol
  - = "intra-AS" routing protocol (also called "intra-domain")
  - Routers in different ASes can run different intra-AS routing protocols
- ASes are connected: via gateway routers
  - Direct link to [gateway] router in another AS
    = "inter-AS" routing protocol (also called "inter-domain")
  - Warning: *Non-gateway routers* need to know about *inter-AS* routing as well!

Interconnected ASes



- Forwarding table configured by both intra- *and* inter-AS routing algorithm:
  - Intra-AS routing algorithm
  - sets entries for internal destinations
  - Inter-AS *and* intra-AS routing algorithms set entries for external destinations

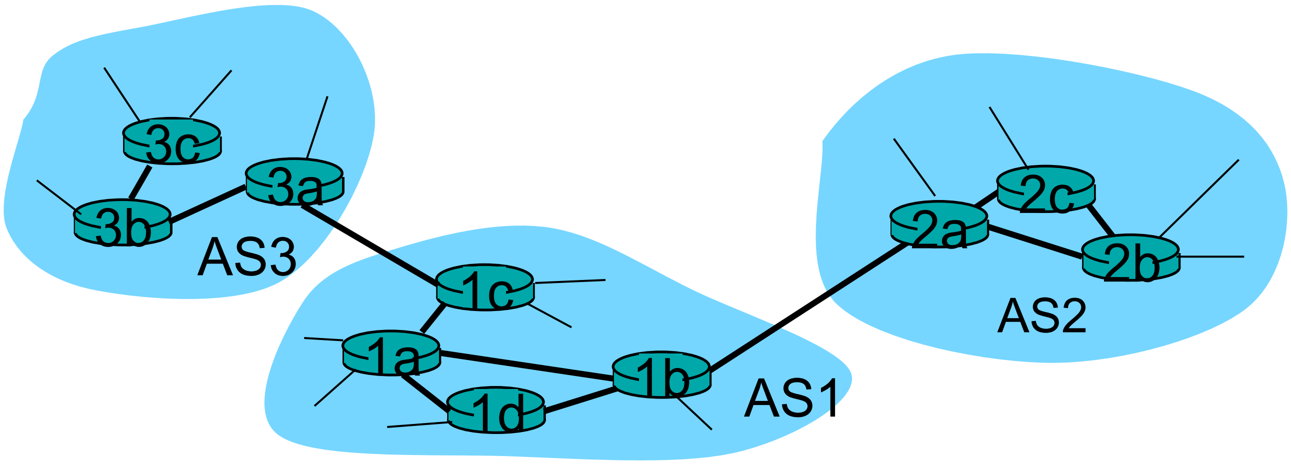
Inter-AS Tasks

- Suppose router in AS1 receives datagram destined outside of AS1:
  - Router should forward packet to gateway router
  - …but to which one?

AS1 must:

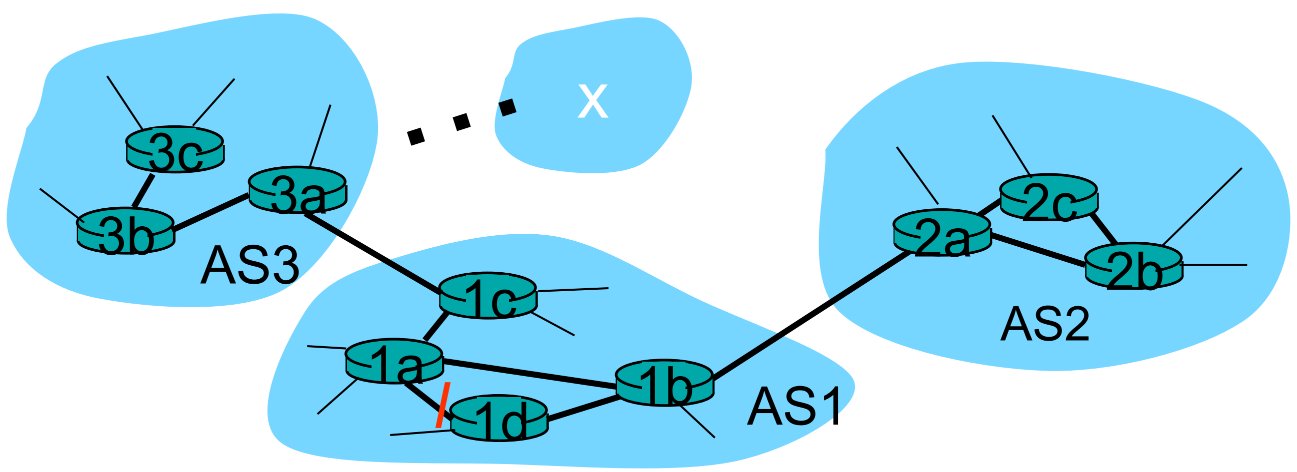
1. learn which destinations are reachable through AS2, which through AS3
2. propagate this reachability info *to all* routers in AS1 (i.e., not just the gateway routers)

Job of inter-AS routing!



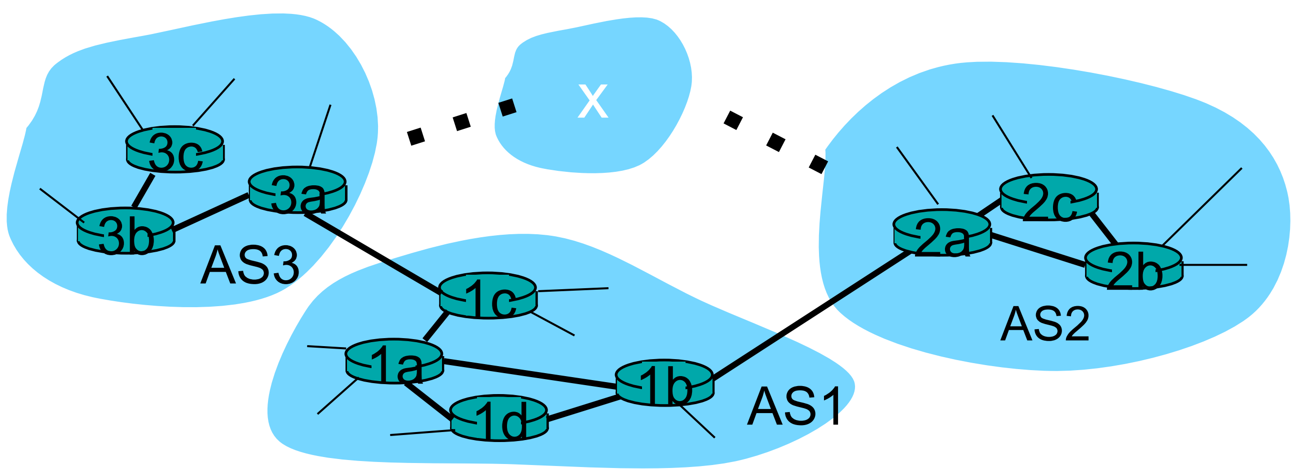
Example: Setting forwarding table in router 1d

- Suppose AS1 learns (via inter-AS protocol) that subnet *x* is reachable via AS3 (gateway 1c) but not via AS2.
- Inter-AS protocol propagates reachability info to all internal routers.
- Router 1d determines from intra-AS routing info that its interface *I* (i.e., interface to 1a) is on the least cost path to 1c.
  - installs forwarding table entry *(x,I)*



Example: Choosing among multiple ASes

- Now suppose AS1 learns from inter-AS protocol that subnet *x* is reachable from AS3 *and* from AS2.
- To configure forwarding table, router 1d must determine towards which gateway it should forward packets for destination x.
  - "Do we like AS2 or AS3 better?"
  - Also the job of inter-AS routing protocol!



Interplay of inter-AS and intra-AS routing

- Inter-AS routing
  - Only for destinations outside of own AS
  - Used to determine gateway router
  - Also: Steers transit traffic
    (from AS *x* to AS *y* via our own AS)
- Intra-AS routing
  - Used for destinations within own AS
  - Used to reach gateway router for destinations outside own AS

⇒ Often, routers need to run *both* types of routing protocols… even if they are not directly connected to other ASes!