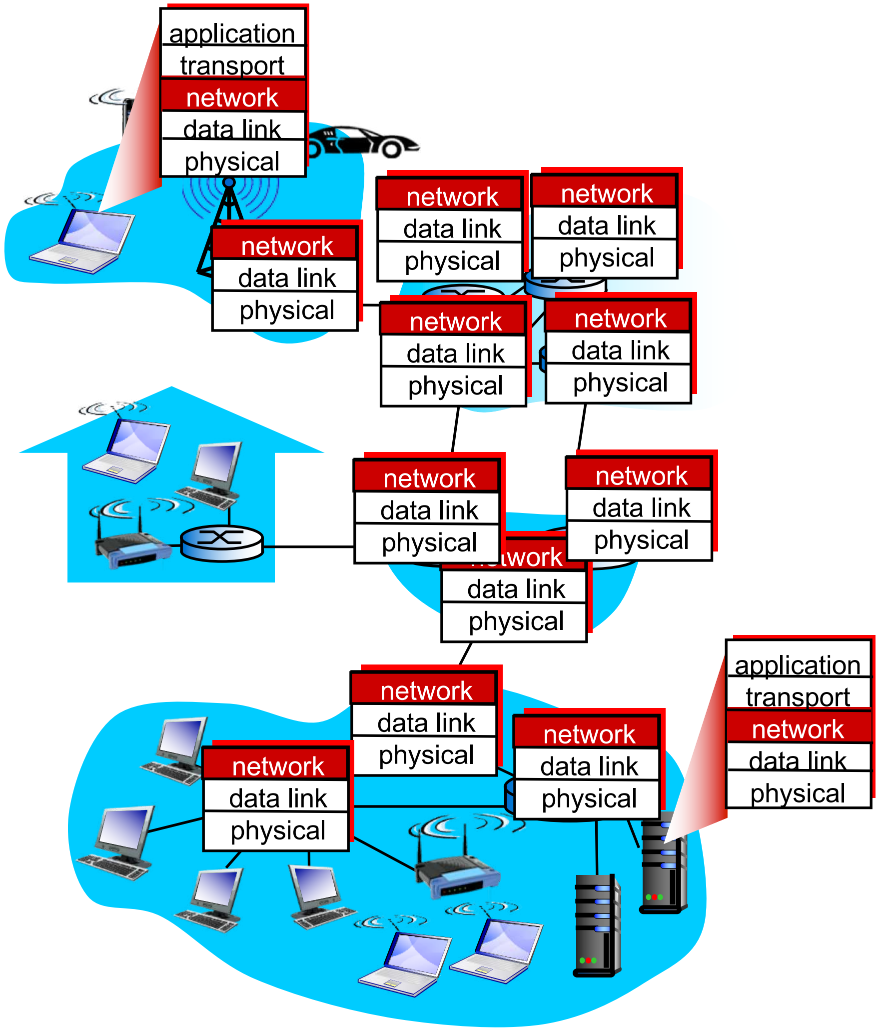
University of Duisburg-Essen
Networked Embedded Systems Group
Institute for Computer Science and
Business Information Systems (ICB)
Schützenbahn 70
D-45127 Essen, Germany

Kommunikationsnetze 2
3 – Routing

Prof. Dr. Pedro José Marrón

Network Layer

- Transport segment from sending to receiving host
- On sending side encapsulates segments into datagrams
- On receiving side, delivers segments to transport layer
- Network layer protocols in every host and router
- Router examines header fields in all IP datagrams passing through it



Two Key Network-Layer Functions

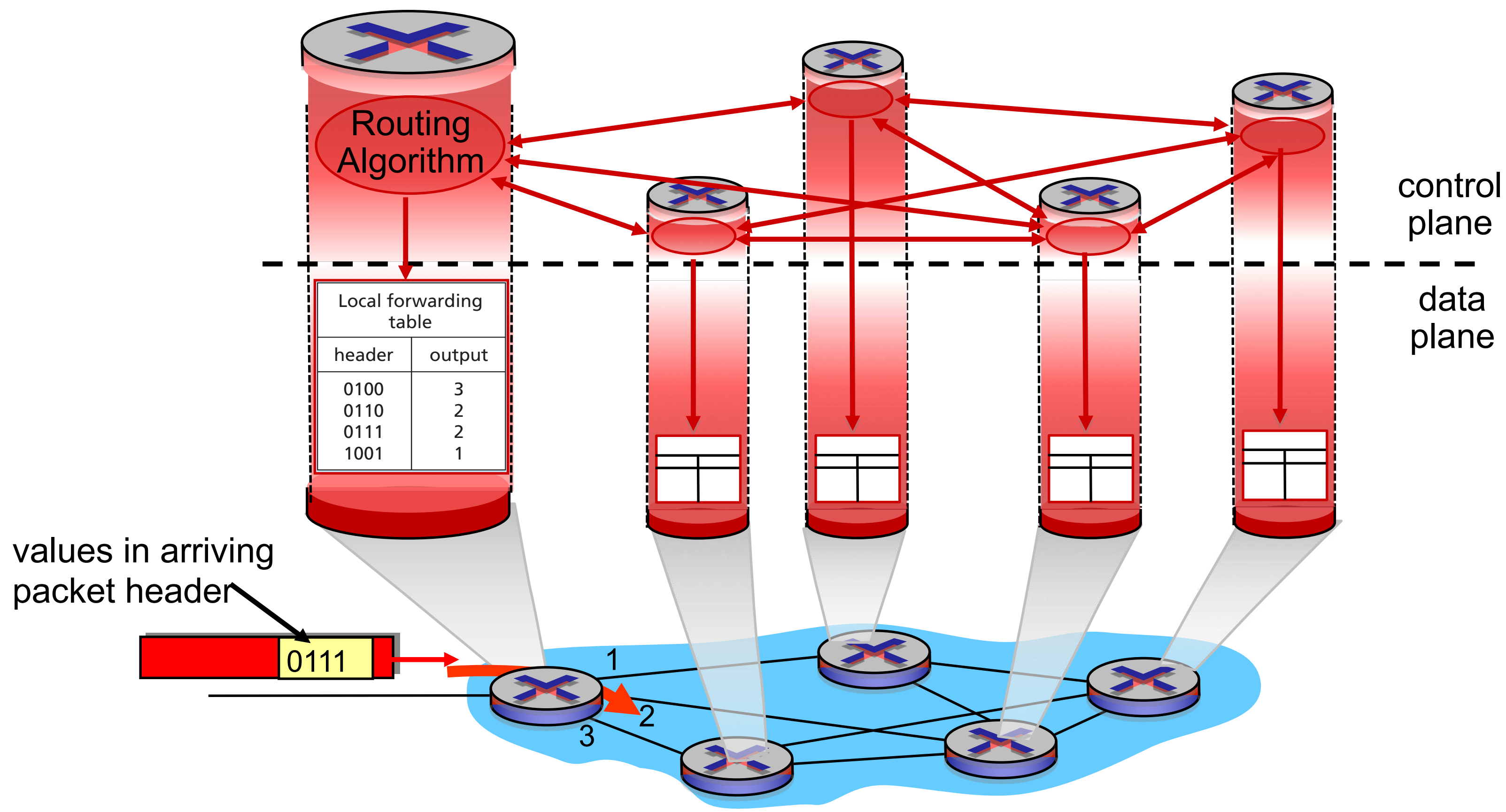
- Forwarding: move packets from router input to appropriate router output
- Routing: determine route taken by packets from source to destination
  - Routing algorithms
- If you are taking a trip,
  - forwarding is the process of getting through a single interchange
  - routing is the planning of the trip from source to destination

Network Layer: Data Plan and Control Plane

- Data plane
  - Local, per-router function
  - Determines how datagram arriving on router input port is forwarded to router output port
  - Forwarding function
- Control plane
  - Network-wide logic
  - Determines how datagram is routed among routers along end-end path from source host to destination host
  - Two control-plane approaches:
    - Traditional routing algorithms implemented in router
    - Software-defined networking (SDN) implemented in (remote) servers

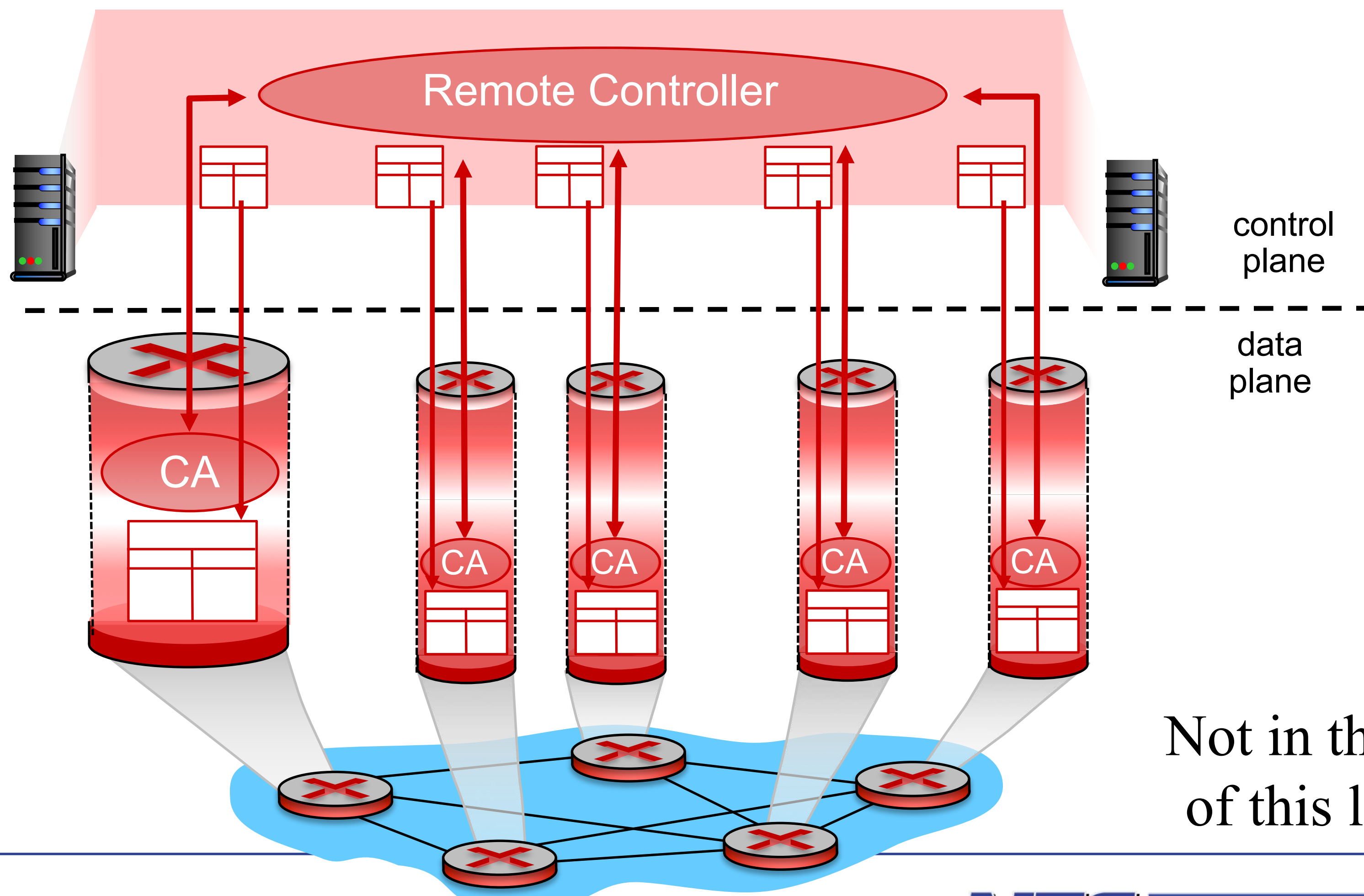
Per-Router Control Plane

- Individual routing algorithm components in each and every router interact in the control plane to compute forwarding tables



Logically Centralized Control Plane (SDN)

- A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables



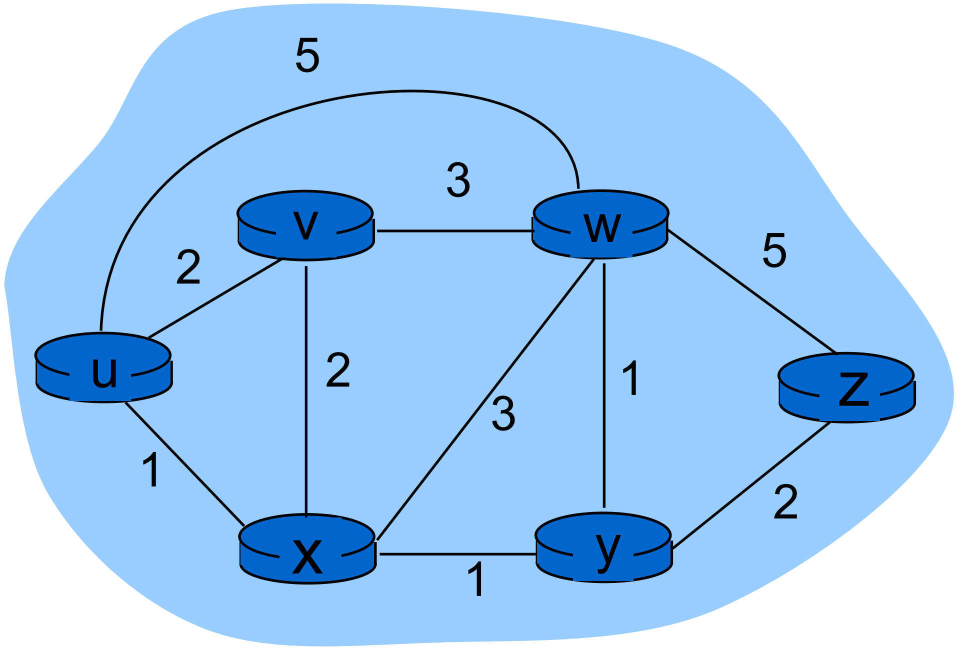
Not in the scope of this lecture!

Routing Protocols

- Routing protocol goal: determine “good” paths (routes) from sending hosts to receiving host through a network of routers
  - Path:
    - sequence of routers that
    - packets will traverse
    - in going from given initial source host to given final destination host
  - Examples of “good”:
    - “least cost”
    - “fastest”
    - “least congested”

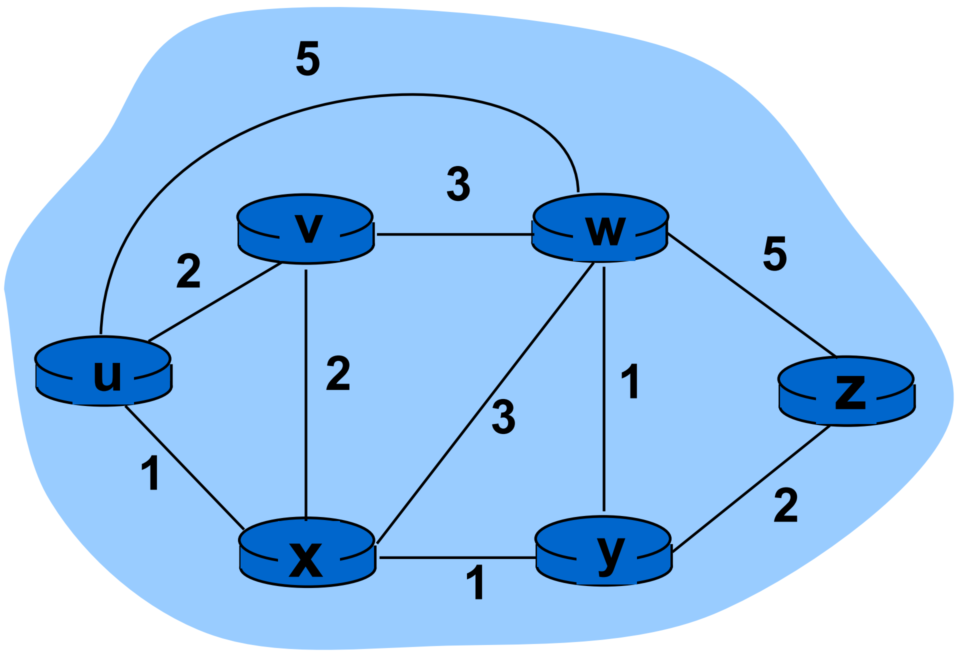
Graph Abstraction of the Network

- Graph: G = (N,E)
- N = set of routers = {u, v, w, y, z}
- E = set of links = {(u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z)}



Graph Abstraction: Costs

- $c(x, x')$ = cost of link (x, x')
  - E.g., $c(w, z) = 5$
  - Could be always 1 or related to bandwidth or related to congestion
- Cost of path $(x_1, x_2, x_3, \ldots, x_p) = c(x_1, x_2) + c(x_2, x_3) + \ldots + c(x_{p-1}, x_p)$
- Routing algorithm question
  - What is the least cost path between, e.g., u and z?



Routing algorithm classification

- Information
  - Global: All routers have complete topology and link costs information
    - “Link state” algorithms
  - Decentralized: Router knows physically-connected neighbours and link costs to neighbours
    - Iterative process of computation and exchange of information with neighbours
    - “Distance vector” algorithms
- Updates
  - Static: Routes change slowly over time
  - Dynamic: Routes change more quickly
    - Periodic update
    - In response to link cost changes

A Link-State Routing Algorithm

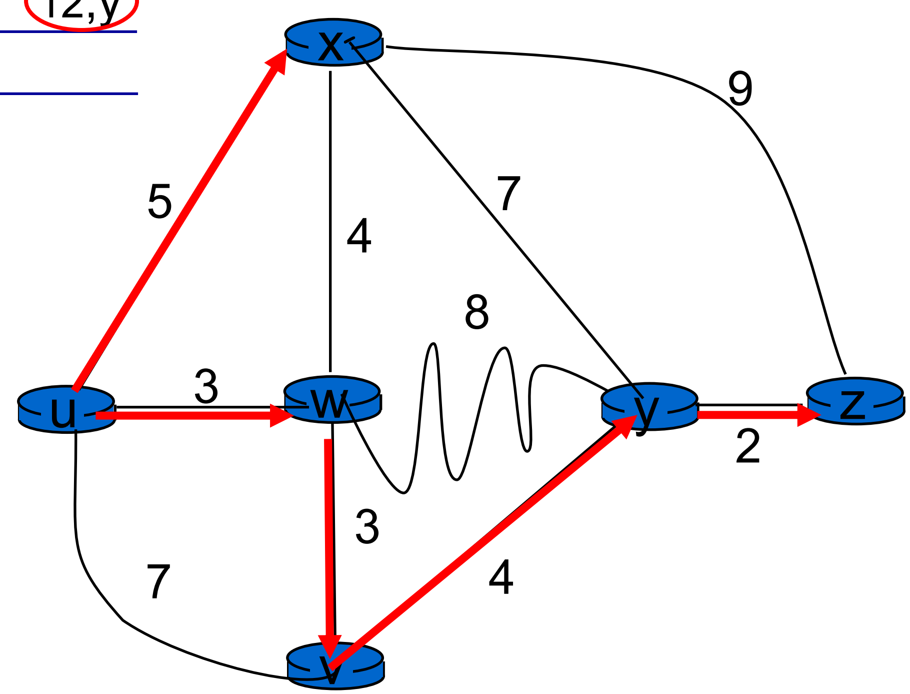
- Dijkstra's algorithm
  - Network topology and link costs known to all nodes
    - Accomplished via "link state broadcast"
    - All nodes have same info
  - Computes least cost paths from one node ("source") to all other nodes
    - Defines forwarding table for that node
  - Iterative: after k iterations, know least cost path to k destinations
- Notation:
  - c(x, y): link cost from node x to y; ∞ if not direct neighbours
  - D(v): current value of cost of path from source to destination v
  - p(v): predecessor node along path from source to v
  - N': set of nodes whose least cost path definitely known

Dijsktra's Algorithm

1. Initialization
2. N' = {u}
3. for all nodes v
4. if v adjacent to u
5. then D(v) = c(u, v)
6. else D(v) = ∞
7. Loop
8. find w not in N' such that D(w) is minimum
9. add w to N'
10. update D(v) for all v adjacent to w and not in N':
11. D(v) = min (D(v), D(w) + c(w, v))
12. /* new cost to v is either old cost to v or known shortest path cost to w plus cost from w to v */
13. until all nodes in N'

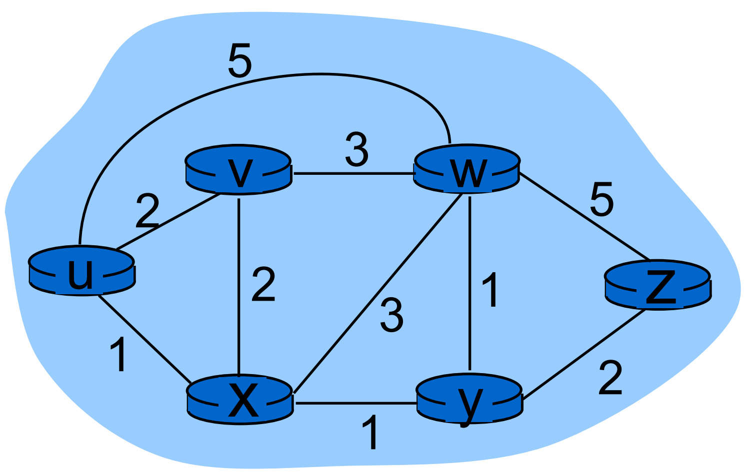
Dijkstra’s Algorithm: Example

Step	N'	D(v) p(v)	D(w) p(w)	D(x) p(x)	D(y) p(y)	D(z) p(z)
0	u	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					

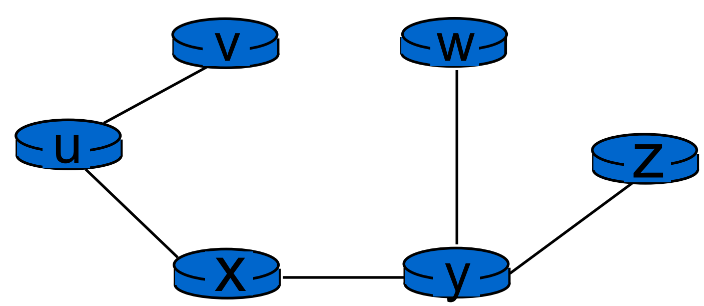


Dijkstra's Algorithm: Another Example

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Resulting shortest-path three (from u) and forwarding table (in u)



destination	link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

Dijkstra's Algorithm: Discussion

- Algorithm complexity in n nodes
  - Each iteration: need to check all nodes, w, not in N
  - n * (n+1)/2 comparisons: $O(n^2)$
  - More efficient implementations are possible
- Oscillations possible:
  - Depending on link cost metric
    - E.g., link cost equals amount of carried traffic

Distance Vector Algorithm

- Bellman-Ford equation
  - Let
    - $d_x(y)$: cost of least cost path from x to y
  - Then

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

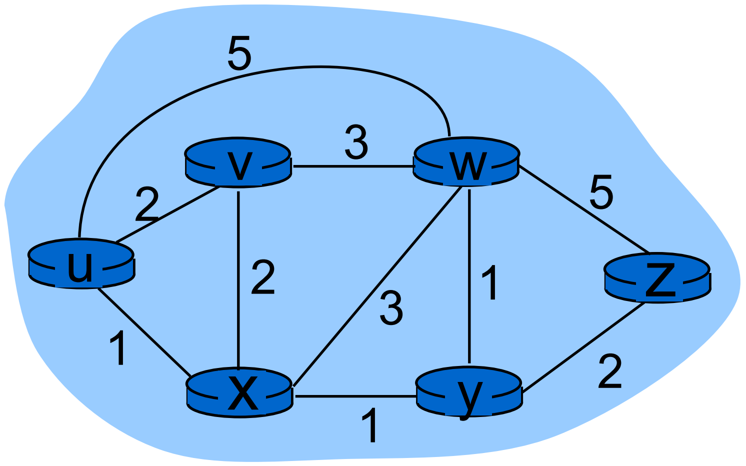
cost from neighbor v to destination y

cost to neighbor v

min taken over all neighbors v of x

Bellman-Ford Example

- Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$
- B-F equation says:
  - $$d_u(z) = \min \{c(u,v) + d_v(z),\ c(u,x) + d_x(z),\ c(u,w) + d_w(z)\}$$
    $$= \min \{2 + 5,\ 1 + 3,\ 5 + 3\} = 4$$
- Node with minimum is next
- Hop in shortest path used in forwarding table



Distance Vector Algorithm

- $D_x(y)$ = estimate of least cost from x to y
  - x maintains distance vector $\mathbf{D_x} = [D_x(y): y \in N]$
- Node x
  - Knows cost to each neighbour v: c(x, v)
  - For each neighbour v, maintains its neighbours' distance vectors $\mathbf{D_v} = [D_v(y): y \in N]$
- Each node sends its own distance vector estimate to neighbours
- When x receives new estimate from neighbour, own distance vector estimate is updated:
  - $D_x(y) = \min_v \{c(x, v) + D_v(y)\}$ for each node $y \in N$
- The estimate $D_x(y)$ converges to the actual least cost $d_x(y)$

Distance Vector Algorithm

- Iterative, asynchronous
  - Each local iteration caused by
    - Local link cost change
    - Distance vector update message from neighbour
- Distributed
  - Each node notifies neighbours only when its distance vector changes
    - Neighbours then notify their neighbours if necessary

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

recompute estimates

if DV to any dest has changed, *notify* neighbors

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

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

node x table

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

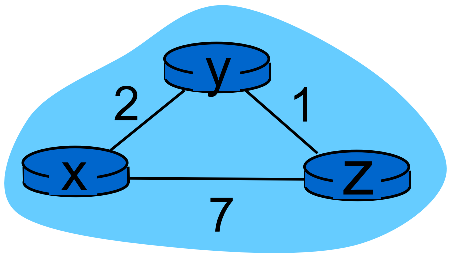
from \ *cost to*	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

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

node z table

from \ *cost to*	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

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

cost to

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

node y table

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

cost to

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

cost to

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

node z table

cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

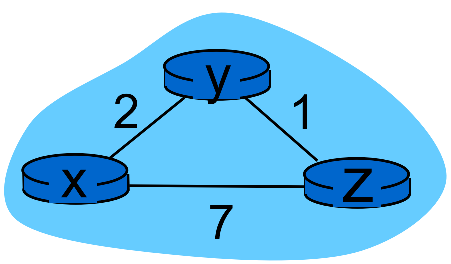
cost to

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

cost to

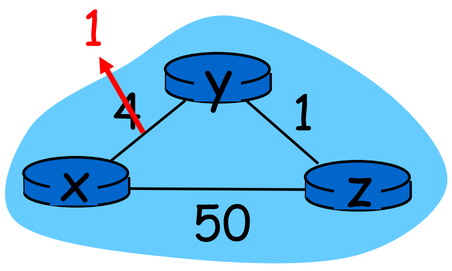
from	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

time



Distance Vector: Link Cost Changes

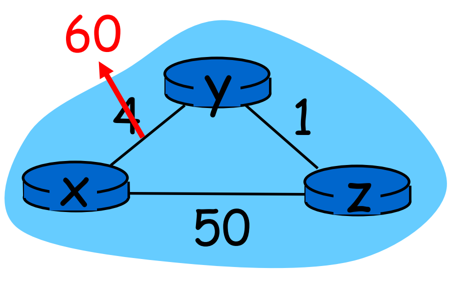
- If link cost changes
  - Node detects local link cost change
  - Updates routing info, recalculates distance vector
  - If distance vector changes, notifies neighbours



- Good news travels fast
  - t_0: y detects link-cost change, updates its distance vector, informs its neighbours
  - t_1: z receives update from y, updates its table, computes new least cost to x, sends its neighbours its distance vector
  - t_2: 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

Distance Vector: Link Cost Changes

- If link cost changes
  - Node detects local link cost change
  - Bad news travels slow – "count to infinity" problem!
  - In example, 44 iterations before algorithm stabilizes



- Poisoned reverse:
  - 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
  - Will this completely solve count to infinity problem?

Link State versus Distance Vector Algorithms

- Message complexity
  - Link State: with n nodes and E links, O(nE) messages sent
  - Distance Vector: exchange between neighbours only
- Convergence speed
  - Link State: $O(n^2)$ algorithm with O(nE) messages
  - Distance Vector: convergence time varies
    - Routing loops, count to infinity problem
- Robustness: what happens if router malfunctions?
  - Link State: node can advertise incorrect link cost; each node computes only its own table
  - Distance Vector: node can advertise incorrect path cost; each node's table used by others
    - Error propagates through network

Making Routing Scalable

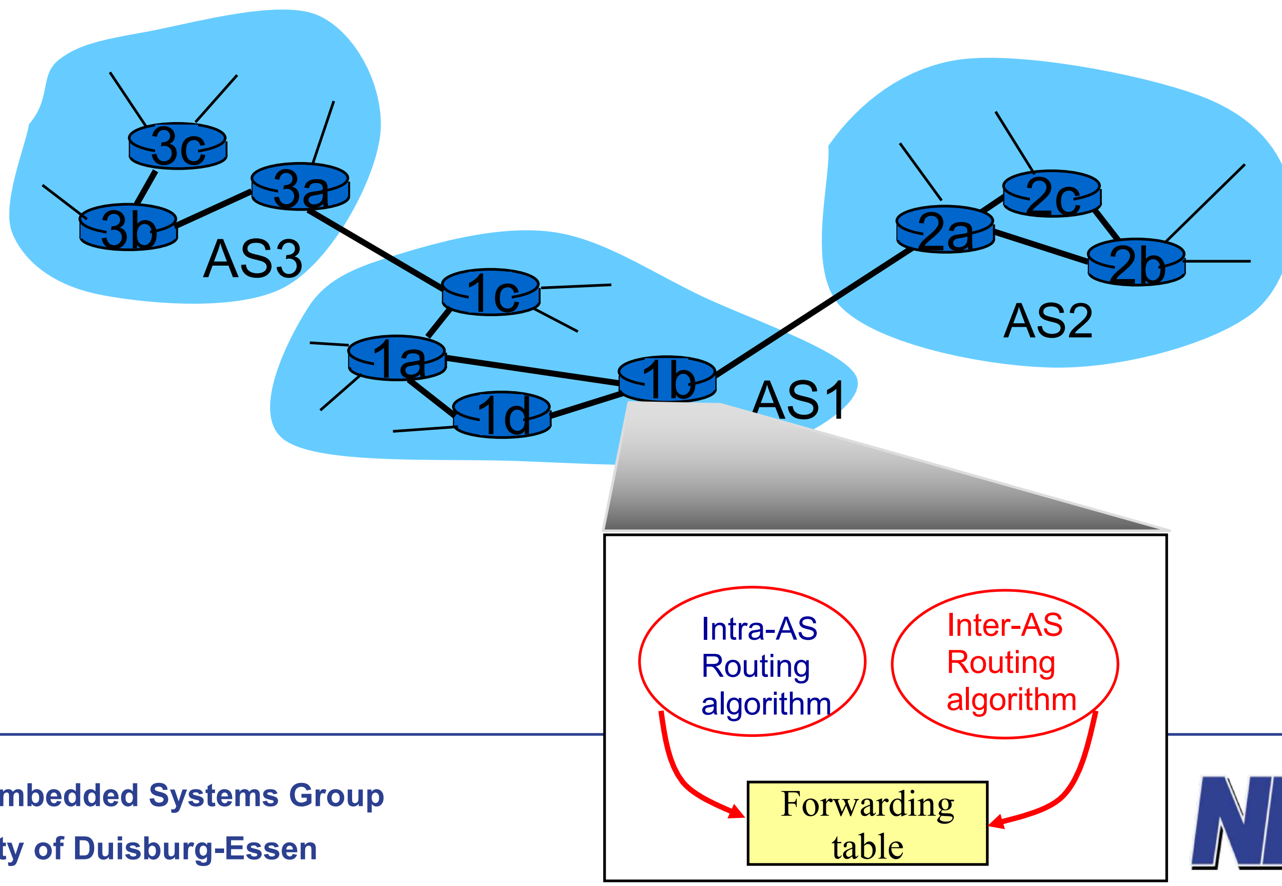
- Our routing thus far is idealized
  - All routers identical
  - Network “flat”
- Scalability with billions of destinations
  - Cannot store all destinations in routing tables
  - Routing table exchange would overload links
- Administrative autonomy
  - Internet = network of networks
  - Each network administrator may want to control routing in its own network

Internet Approach to Scalable Routing

- Aggregate routers into regions known as “autonomous systems” (AS)
- Intra-AS routing
  - Routing among hosts and routers in the same AS
  - All routers in AS must run same intra-domain protocol
  - Routers in different AS can run different intra-domain routing protocol
- Inter-AS routing
  - Routing among different AS
- Gateway router
  - at “edge” of its own AS with link(s) to router(s) in other AS
  - performs inter-domain routing as well as intra-domain routing

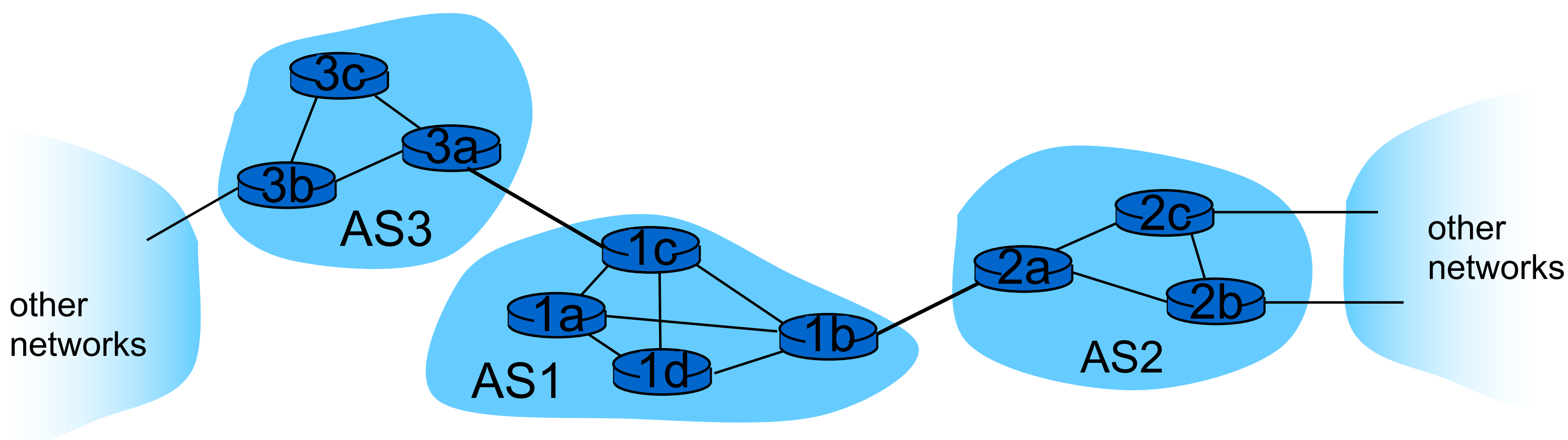
Interconnected AS

- Forwarding table configured by both intra- and inter-AS routing algorithms
  - Intra-AS routing determines entries for destinations within AS
  - Inter-AS and intra-AS determine 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 which one?
- AS1 must
  - Learn which destinations are reachable through AS2, which through AS3
  - Propagate this reachability information to all routers in AS1



Intra-AS Routing

- Also known as Interior Gateway Protocols (IGPs)
- Most common intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol

RIP: Routing Information Protocol

- Distance vector algorithm
  - In Unix since 1982 with the ’routed’ daemon
- Distance metric: number of hops
  - Max number of hops 15, 16 means infinity
- RIP advertisements
  - Contains hop count from the advertising router to destination networks
  - UDP datagrams
  - Size limited to 512 bytes (max 25 routes)
- RIP commands
  - Request or reply
  - Response: solicited or unsolicited

RIP: Routing Information Protocol

1. receive a RIP message
2. for each advertised destination
3. increase hop count of one
4. if destination not in routing table
5. add the information to the table
6. else
7. if next-hop is the same
8. replace entry
9. else
10. if hop count smaller
11. replace entry

RIP Timers

- Periodic timer
  - Controls advertising of regular update messages (25-35 sec)
- Expiration timer
  - Controls the validity of a route (180 sec)
  - Every time on update is received (30 sec average) the timer is reset
  - If no update received within this timer the metric is set to 16
- Garbage timer
  - Purges route after being advertised for 120 sec as invalid (metric = 16)
  - Allows neighbours to know invalid routes ”soon”

RIP: Problems

- Infinite path is defined as of length 16
  - RIP cannot be used in networks with routes longer than 15 hops
  - If the value for infinity is increased, RIP can converge slowly due to count-to-infinity problem
- Default metric is number of hops
  - Relevant metrics can be many: bandwidth, delay, …

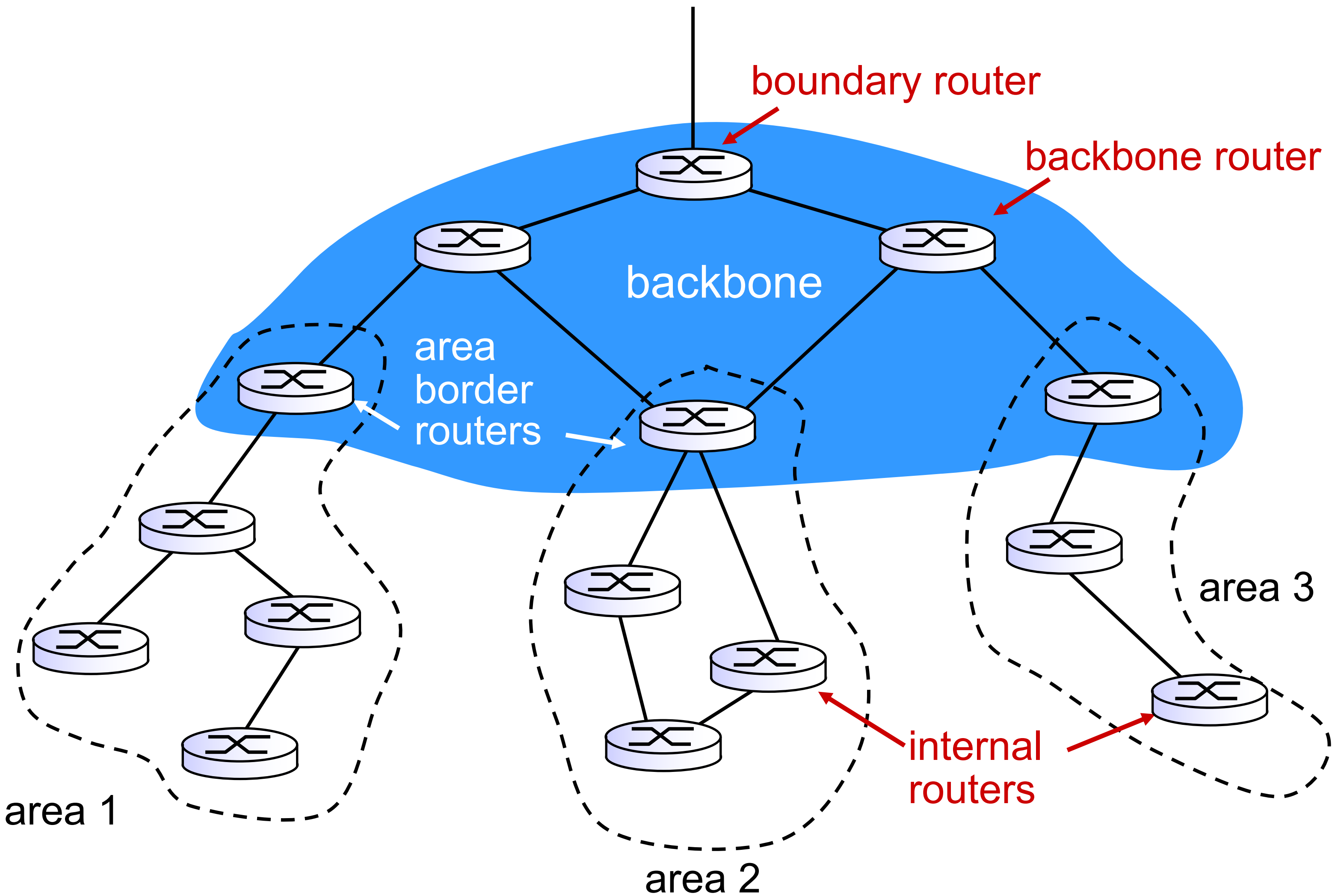
OSPF: Open Shortest Path First

- "open" -> publicly available
- Uses link-state algorithm
  - Link state packet dissemination
  - Topology map at each node
  - Route computation using Dijkstra's algorithm
- Router floods OSPF link-state advertisements to all other routers in entire AS
  - Carried in OSPF messages directly over IP
  - Link state for each attached link

OSPF Features

- Security: all OSPF messages authenticated
- Multiple same-cost paths allowed
  - Only one path in RIP
- For each link, multiple cost metrics for different types of services
  - E.g., satellite link cost set low for best effort and high for real-time
- Integrated uni- and multi-cast support
  - Multicast OSPF (MOSPF) uses same topology data as OSPF
- Hierarchical OSPF in large domains

Hierarchical OSPF



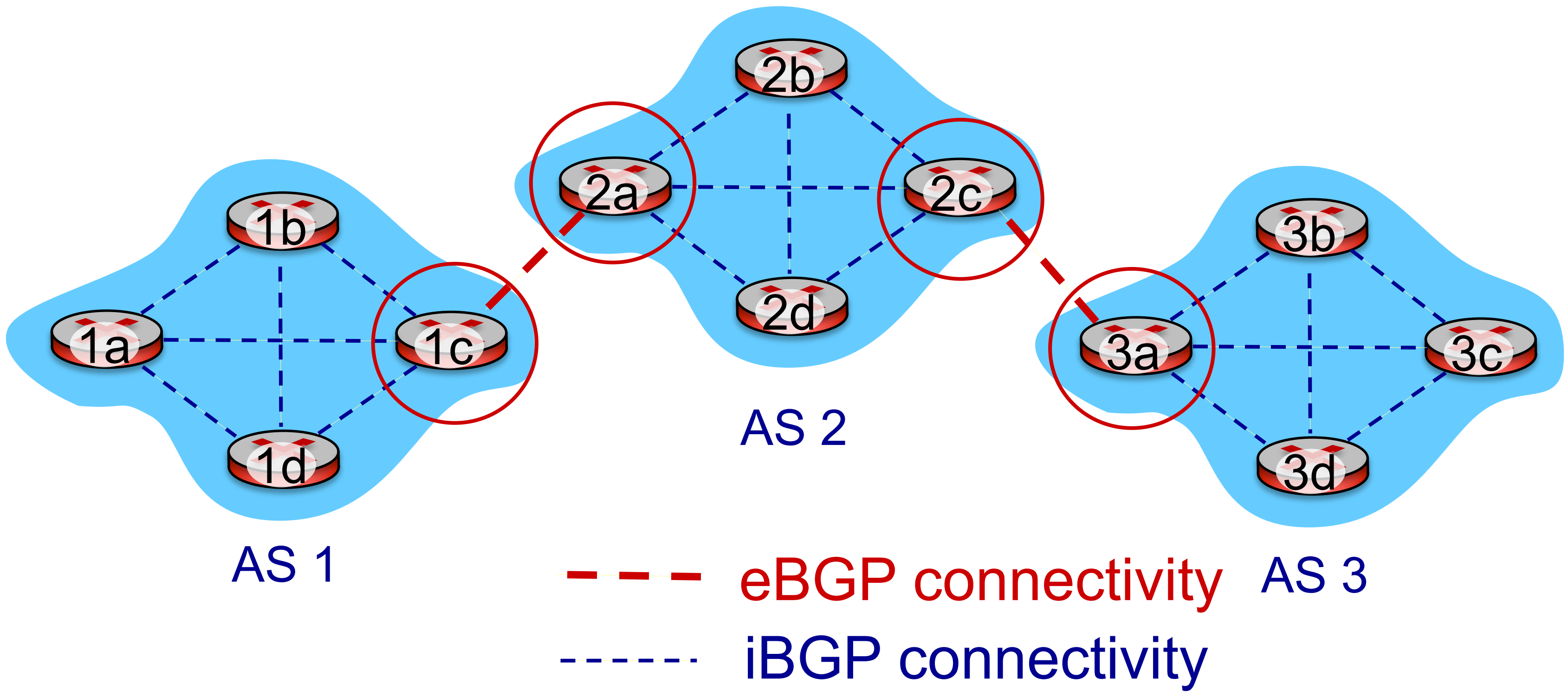
Hierarchical OSPF

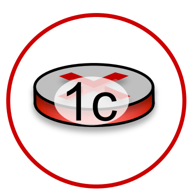
- Two-level hierarchy: local area and backbone
  - Link-state advertisements only in area
  - Each node has detailed area topology and only knows direction (shortest path) to nets in other areas
- Area border routers
  - "Summarize" distances to networks in own area and advertise to other area border routers
- Backbone routers
  - Run OSPF routing limited to backbone
- Boundary routers
  - Connect to other AS

Inter-AS Routing: BGP

- BGP (Border Gateway Protocol): the de facto inter-domain routing protocol
  - “Glue that holds the Internet together”
- BGP provides each AS a means to:
  - eBGP: obtain subnet reachability information from neighbouring AS
  - iBGP: propagate reachability information to all AS-internal routers
  - determine “good” routes to other networks based on reachability information and policies
  - allow a subnet to advertise its existence to the rest of the Internet

eBGP, iBGP Connections

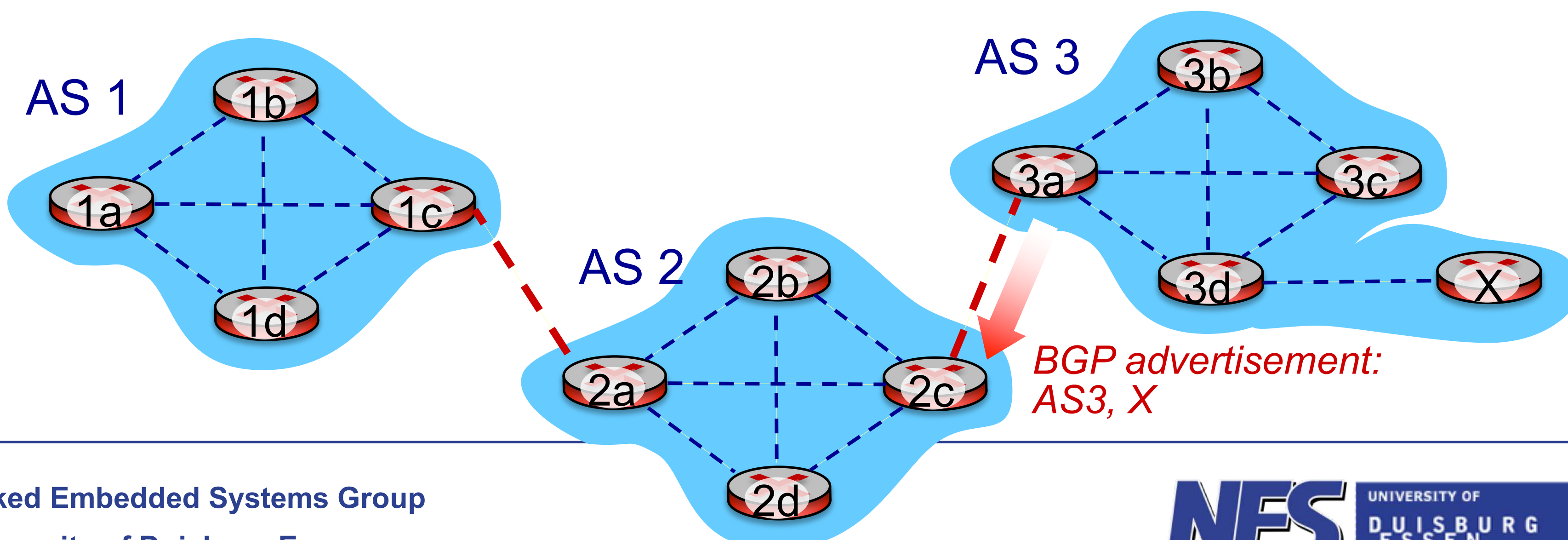




gateway routers run both eBGP and iBGP protocols

BGP Basics

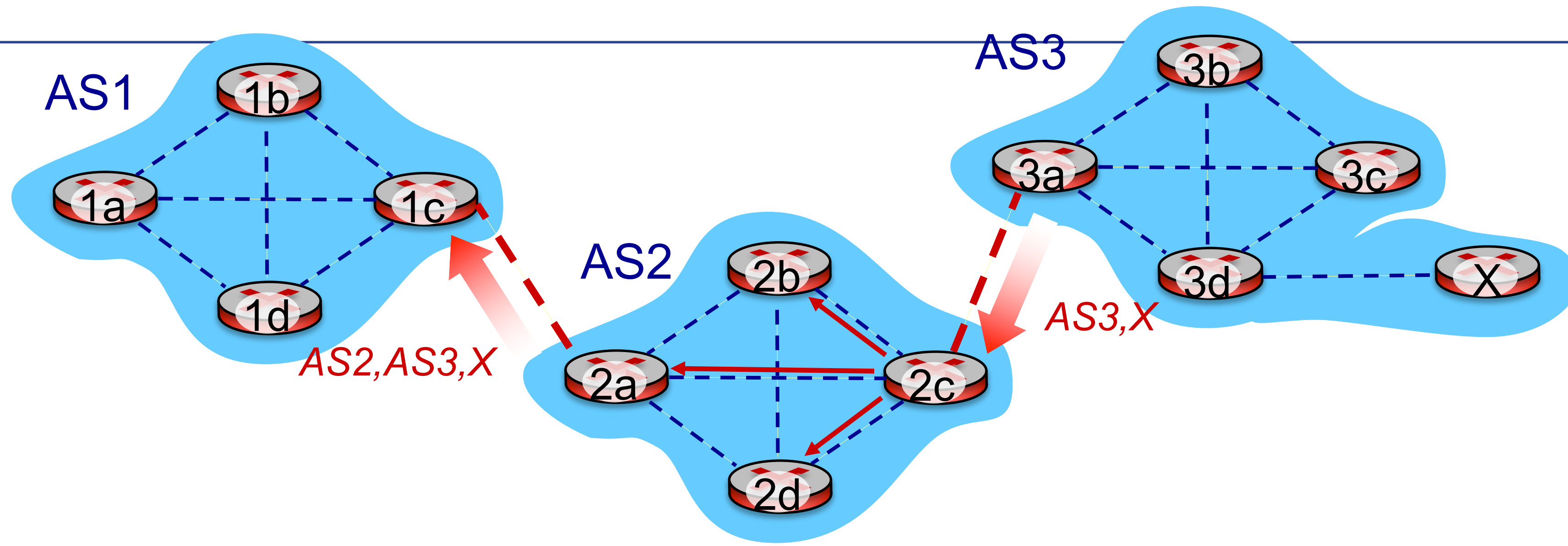
- BGP Session
  - Two BGP routers exchange BGP messages over semi-permanent TCP connection
  - Advertising paths to different destination network prefixes
- When AS3 gateway router 3a advertises path AS3,X to AS2 gateway router 2c, AS3 promises to AS2 that it will forward datagrams towards X



Path Attributes and BGP Routes

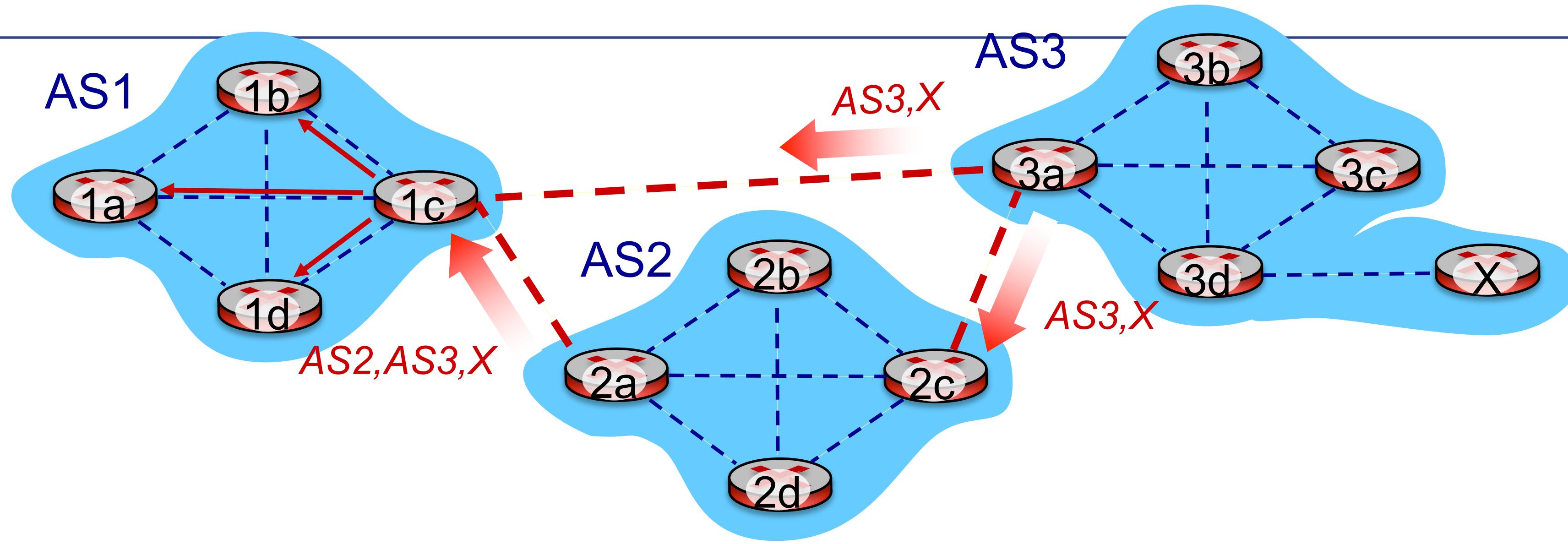
- Advertised prefix includes BGP attributes
  - prefix + attributes = “route”
- Two important attributes
  - AS-PATH: list of AS through which prefix advertisement has passed
  - NEXT-HOP: indicates specific internal AS router to next-hop AS
- Policy-based routing
  - Gateway receiving route advertisement uses import policy to accept/decline path (e.g., never route through AS Y)
  - AS policy also determines whether to advertise path to other neighbouring AS

BGP Path Advertisement



- 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 and 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 Advertisement



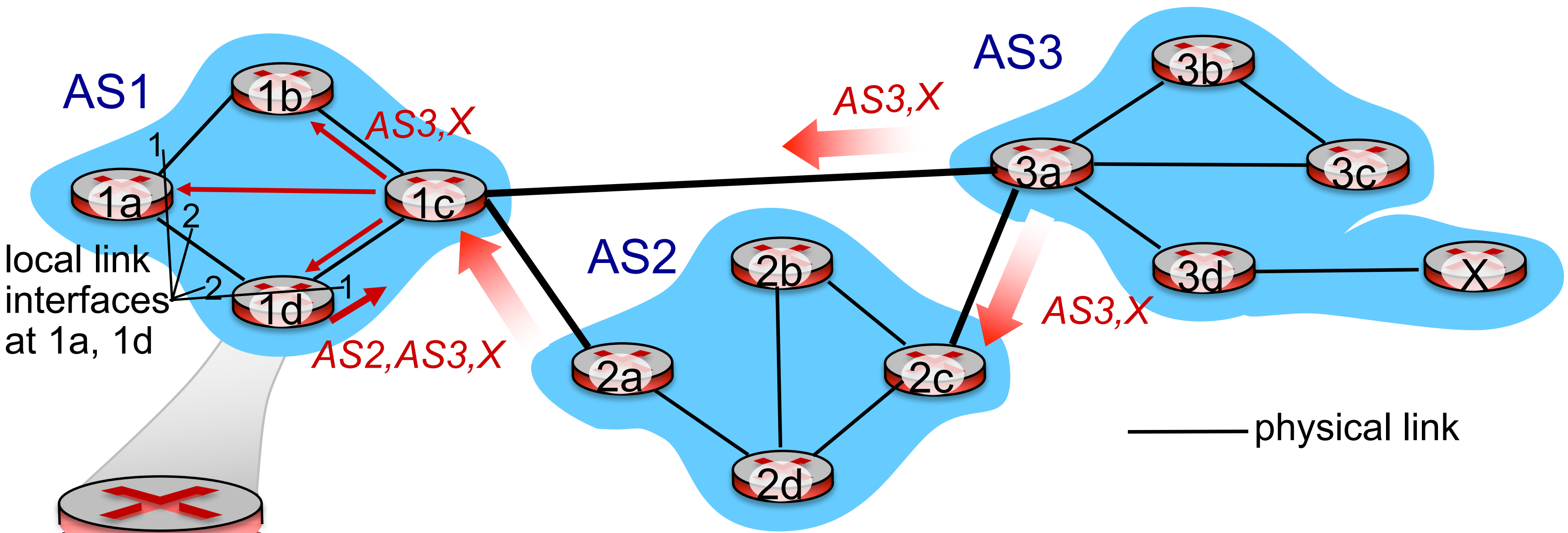
- Gateway router may learn about multiple paths to destination
  - AS1 gateway router 1c learns path AS2,AS3,X from 2a
  - AS1 gateway router 1c learns path AS3,X from 3a
  - Based on policy, AS1 gateway router 1c chooses path AS3,X and advertises path within AS1 via iBGP

BGP Messages

- BGP messages exchanged between peers over TCP connection
- BGP messages
  - OPEN: opens TCP connection to remote BGP peer and authenticates the sending BGP peer
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE: keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION: reports errors in previous messages; also used to close connection

BGP, OSPF, Forwarding Table Entries

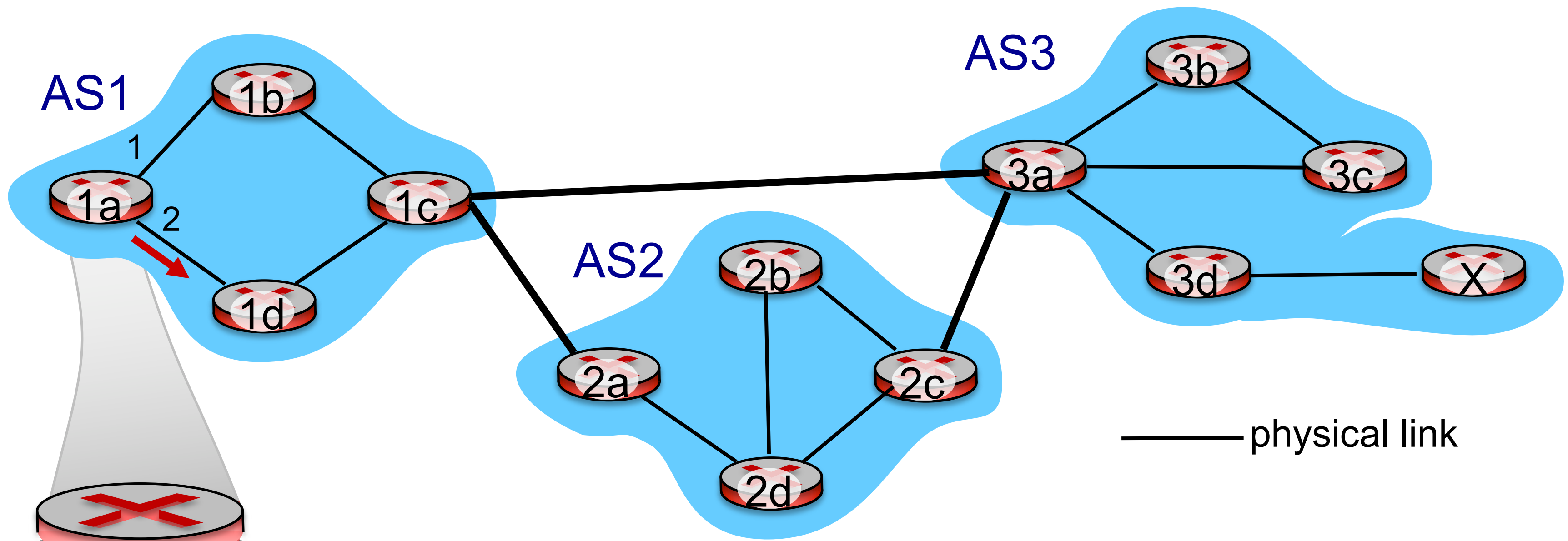
- How does router set forwarding table entry to distant prefix?



dest	interface
…	…
X	1
…	…

- recall: 1a, 1b, 1c learn about dest X via iBGP from 1c: "path to X goes through 1c"
- 1d: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 1

BGP, OSPF, Forwarding Table Entries



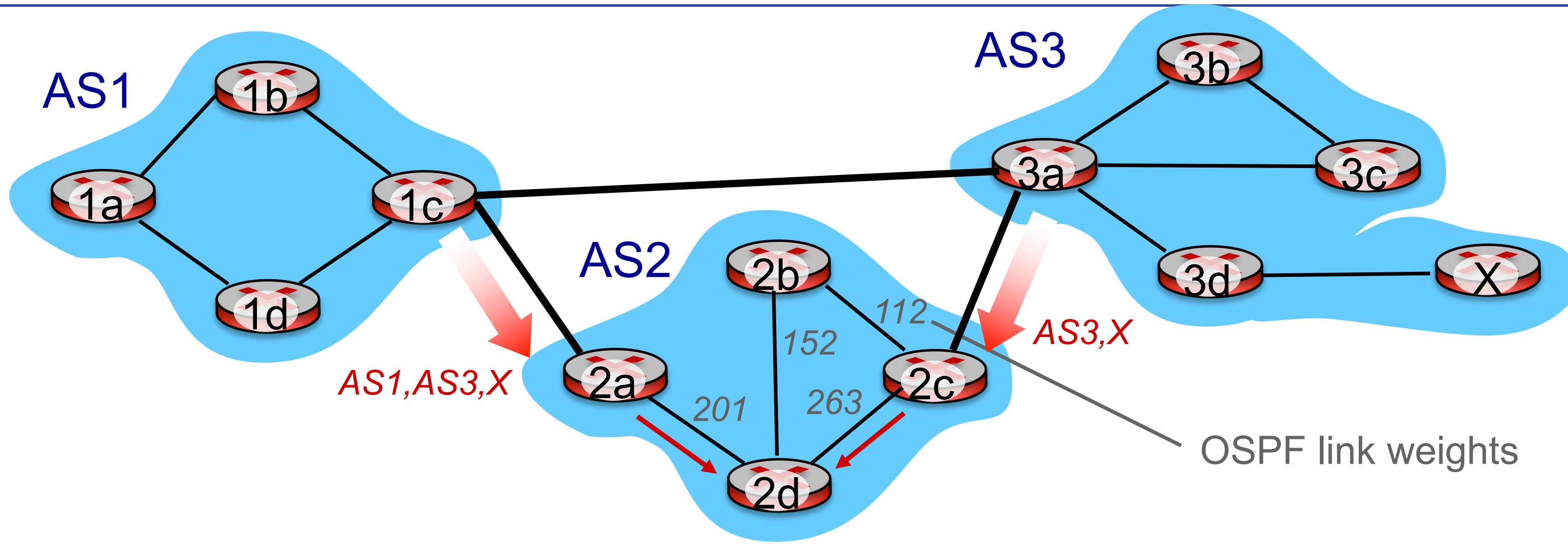
dest	interface
…	…
X	2
…	…

- recall: 1a, 1b, 1c learn about dest X via iBGP from 1c: “path to X goes through 1c”
- 1d: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 1
- 1a: OSPF intra-domain routing: to get to 1c, forward over outgoing local interface 2

BGP Route Selection

- Router may learn about more than one route to destination AS, selects route based on
  - Local preference value attribute: policy decision
  - Shortest AS-PATH
  - Closest NEXT-HOP router: hot potato routing
  - Additional criteria

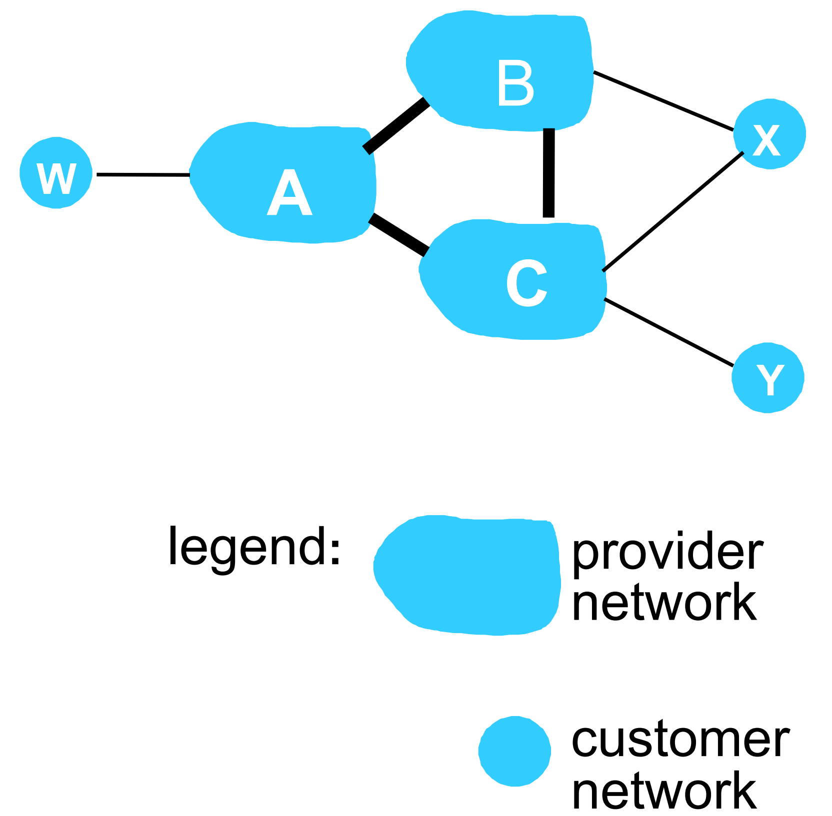
Hot Potato Routing



- 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 (e.g., 2d chooses 2a, even though more AS hops to X): do not worry about inter-domain cost!

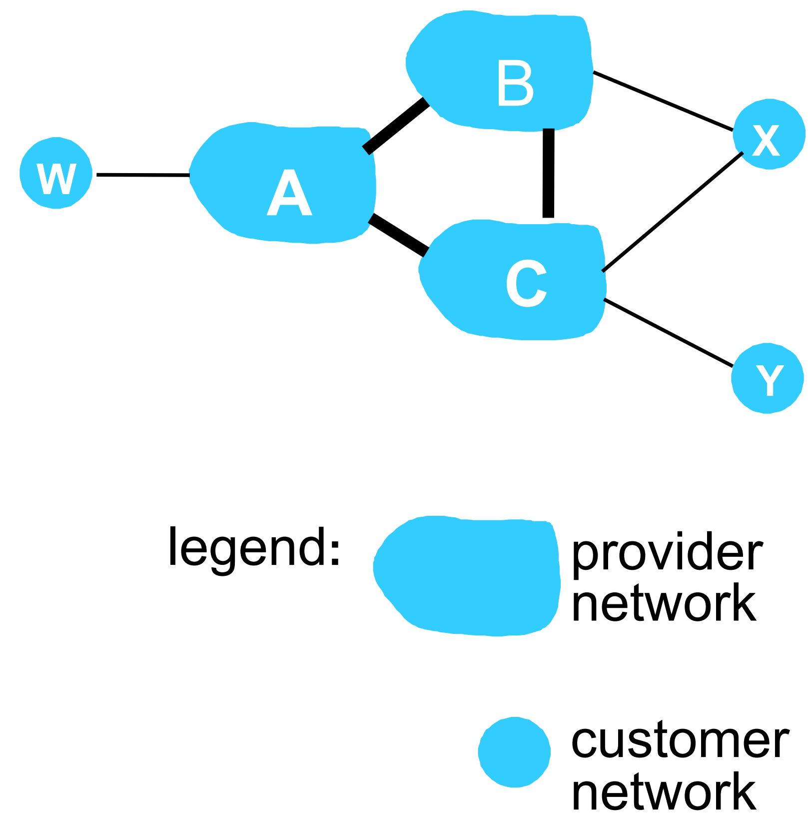
BGP: Achieving Policy via Advertisements

- Suppose an ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs)
- A advertises path Aw to B and to C
- B chooses not to advertise BAw to C
  - B gets no “revenue” for routing CBAw since none of C, A, w are B’s customers
  - C does not learn about CBAw path
- C will route Caw (not using B) to get to w



BGP: Achieving Policy via Advertisements

- A, B, C are providers
- X, W, Y are customer (of provider networks)
- X is dual-homed: attached to two networks
- Policy to enforce: X does not want to route from B to C via X
  - So X will not advertise to B a route to C



Why Different Intra-, Inter-AS Routing?

- Policy
  - Inter-AS: administrator wants to control over how its traffic is routed and who routes through its network
  - Intra-AS: single administrator so no policy decisions needed
- Scale
  - Hierarchical routing saves table size and reduces update traffic
- Goal
  - Intra-AS: can focus on performance
  - Inter-AS: policy may dominate over performance

Additional Material

- BGP: Putting the “Inter” in Internet. Professor Jennifer Rexford, Princeton
  - https://www.youtube.com/watch?v=HAhzj1E1ejI