Networks Lecture 9

Paolo Ciancarini

Innopolis University

February 14, 2022

Source of the material

- This lecture is based on the following resources
  - Chapter 4 of Computer Networking: A Top Down Approach
    (8th edition) by Jim Kurose and Keith Ross
  - The material is aligned and add/deleted according to the need of the students.

Topic of the lecture

- VLAN
- Routing algorithms
  - Link state
  - Distance vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
- Summary

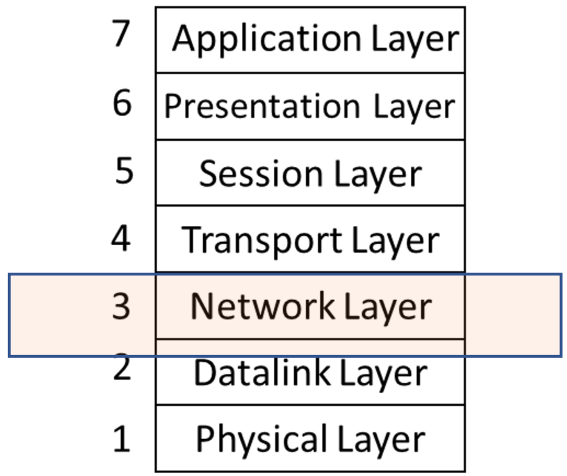
Topic of the tutorial

- Discussion about the lecture topics

- Practice examples about the today lecture

Recap!

- We have introduced the Network Layer
  - Forwarding and routing
  - IP: Internet Protocol
  - What's inside a router



OSI Model

Topic of the lecture

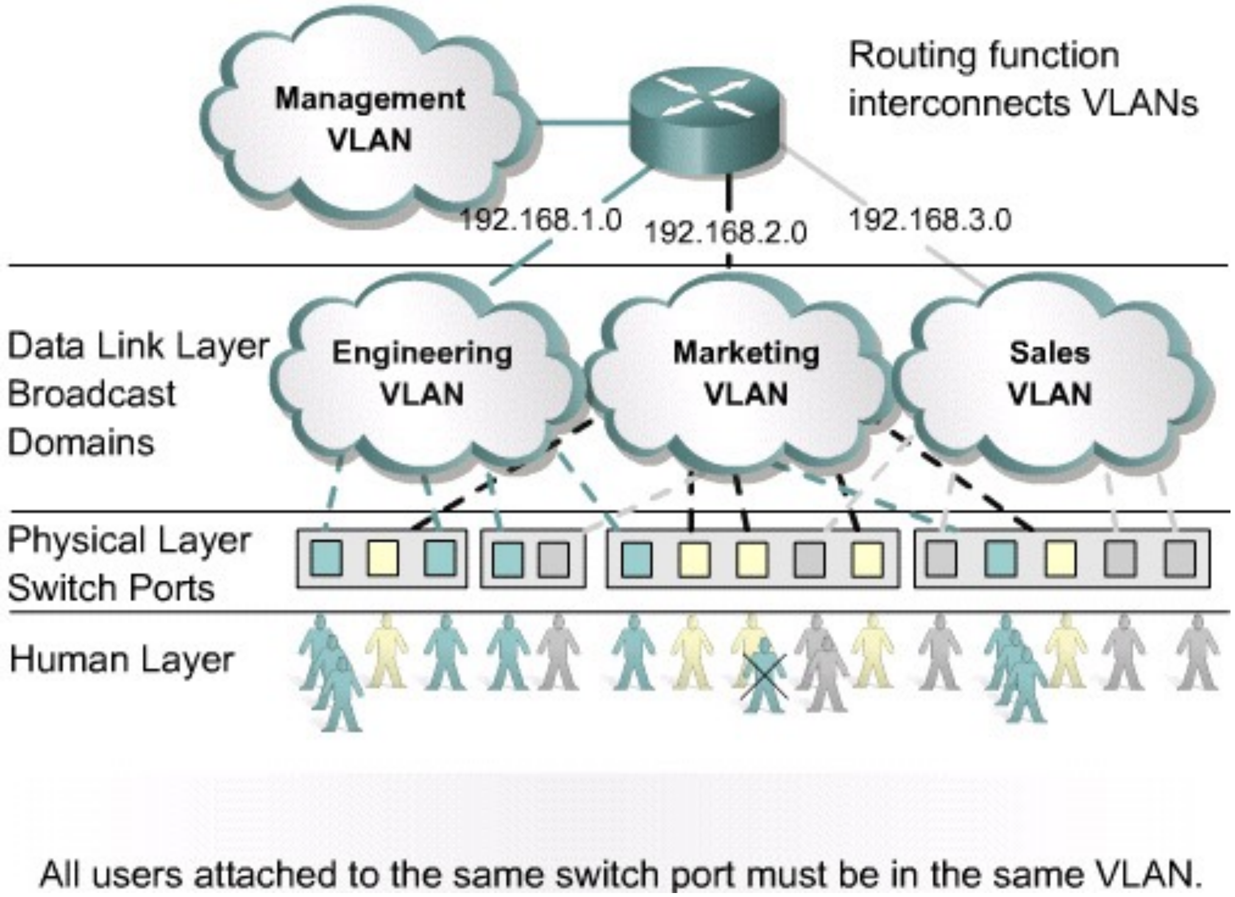
- Virtual LANs
- Routing algorithms
  - Link state
  - Distance vector
  - Hierarchical routing
- Routing in the Internet
  - RIP routing information protocol
  - OSPF open shortest path first
  - BGP border gateway protocol
- Broadcast and multicast routing

Virtual LANs (VLANs)

- Allows us to split switches into separate (virtual) switches
- VLANs logically segment switched networks based on the
  - functions, project teams, or applications of the organization, regardless of the physical location or connections to the network.
- Only members of a VLAN can see that VLAN’s traffic
- VLAN traffic is not encrypted

Benefits of VLANs

- The key benefit of VLANs is that they permit the network administrator to organize the LAN logically instead of physically.

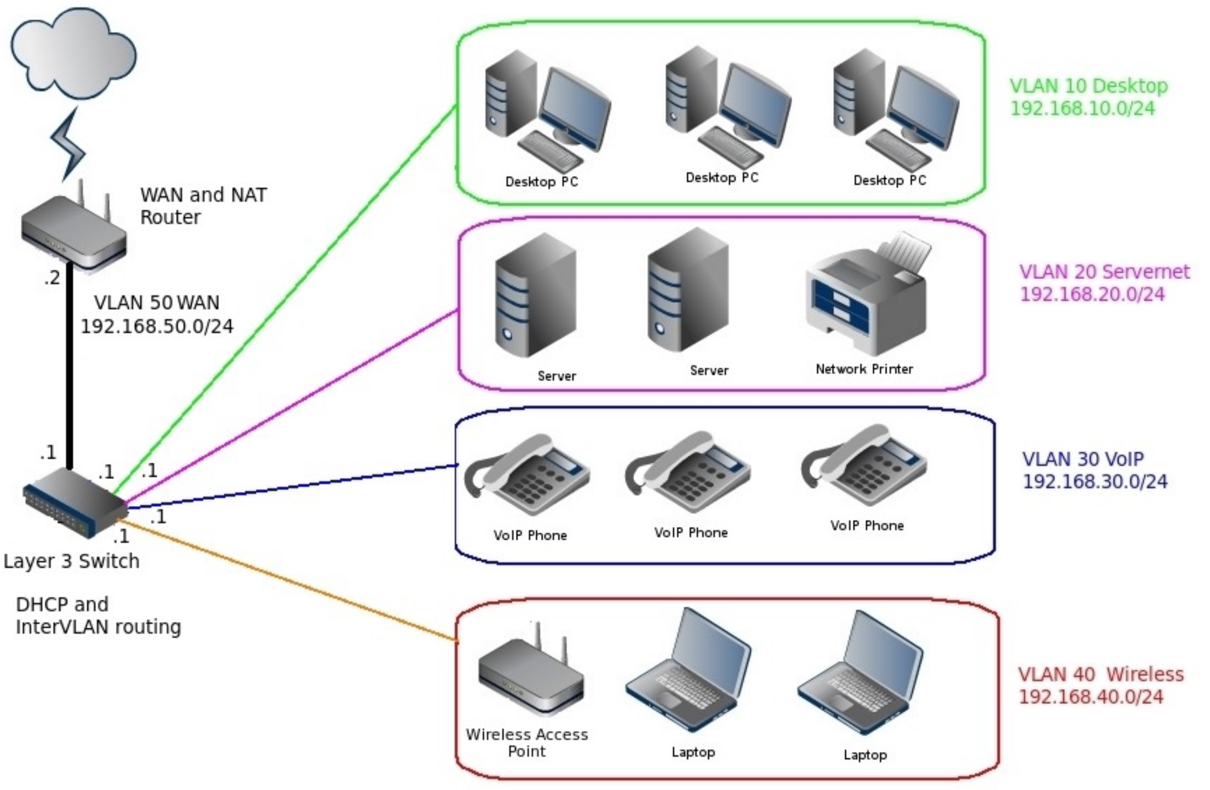


VLAN is Layer 2

VLAN defines broadcast domains in a layer 2 network;

we introduce VLANs here (layer 3) because it is programmable at layer 3

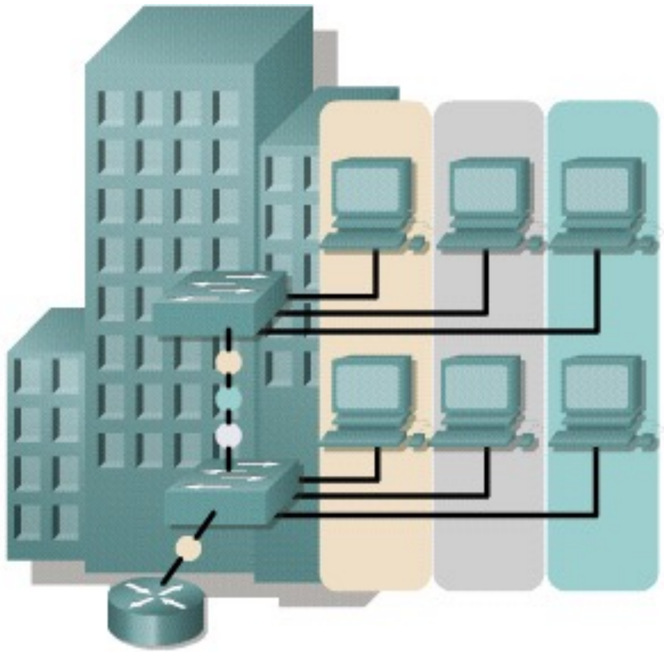
We will explain again VLANs at layer 2 in the related lecture



https://www.fiber-optic-tutorial.com/layer-2-vs-layer-3-switch-for-vlan.html

VLAN structure

- A workstation in a VLAN group is restricted to communicating with file servers in the same VLAN group.



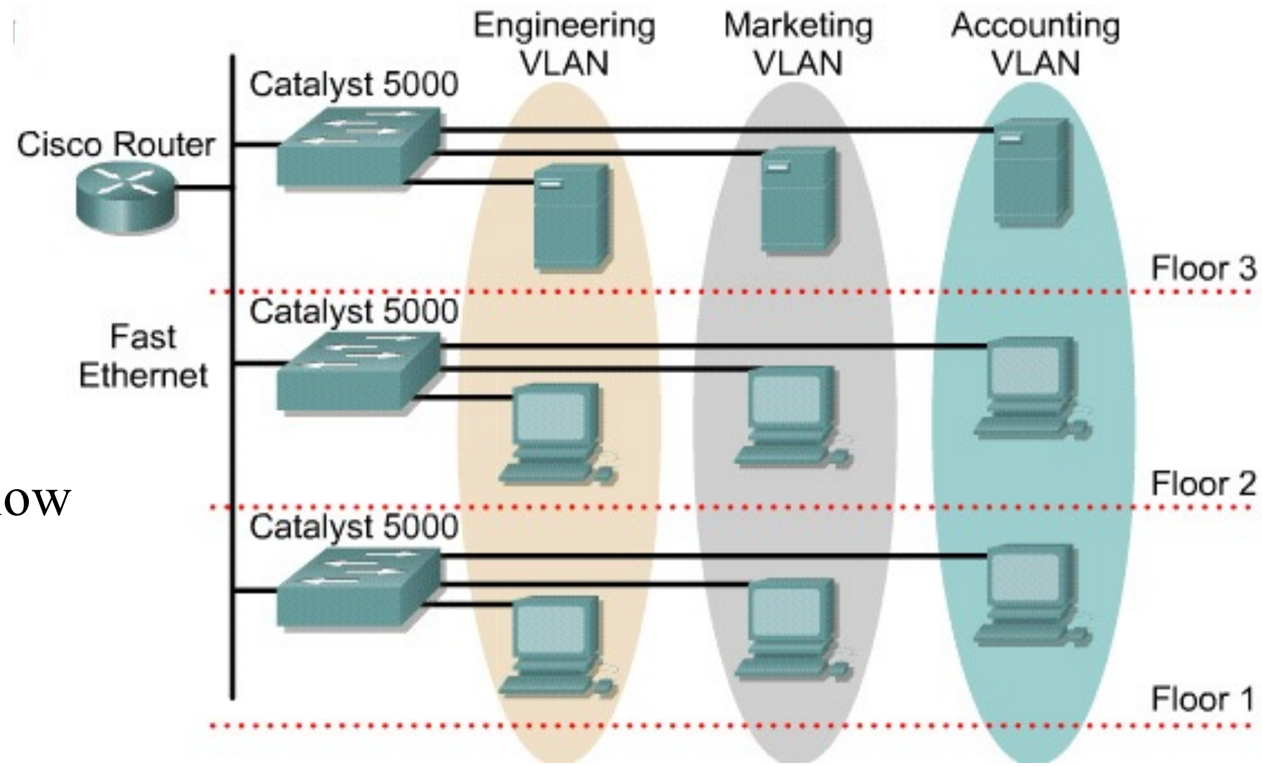
- A group of ports or users in same broadcast domain
- Can be based on port ID, MAC address, protocol or application

- LAN switches and network management software provide a mechanism to create VLANs
- Frame tagged with VLAN ID

VLAN behavior

- VLANs function by logically segmenting the network into different broadcast domains so that packets are only switched between ports that are designated for the same VLAN.

- Routers in VLAN topologies provide broadcast filtering, security, and traffic flow management.



Broadcast Domain

In networking, a broadcast means that we send something that **everyone receives**, whether they need/want it or not

Broadcast domain: a group of devices on a specific network segment that can reach each other with Ethernet broadcasts.

Broadcasts sent by a device in one broadcast domain are **not** forwarded to devices in another broadcast domain.

This improves the performance of the network because not all devices on a network will receive and process broadcasts.

Routers separate a LAN into multiple broadcast domains (every port on a router is in a different broadcast domain).

Switches (by default) flood Ethernet broadcast frames out all ports, just like bridges and hubs. All ports on these devices are in the same broadcast domain.

MAC Addresses

- Some applications and protocols use broadcast traffic: a good example is ARP (Address Resolution Protocol). Switches will recognize it as broadcast traffic by looking at the destination MAC address
- A **MAC address** is a hardware identification number that uniquely identifies each device on a network.

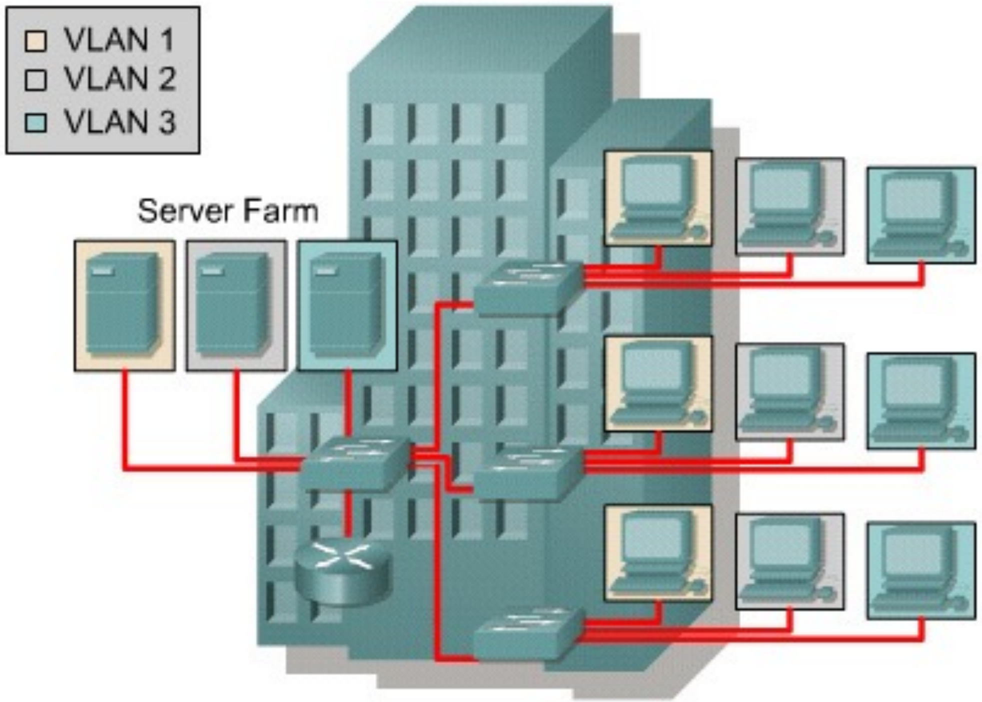
- For example, an Ethernet card may have a MAC address of

  `00:0d:83:b1:c0:8e`

- An organization issues a special number sequence (called the **Organizationally Unique Identifier**) to manufactures.

Broadcast domains with VLANs and routers

- A VLAN is a broadcast domain created by one or more switches.

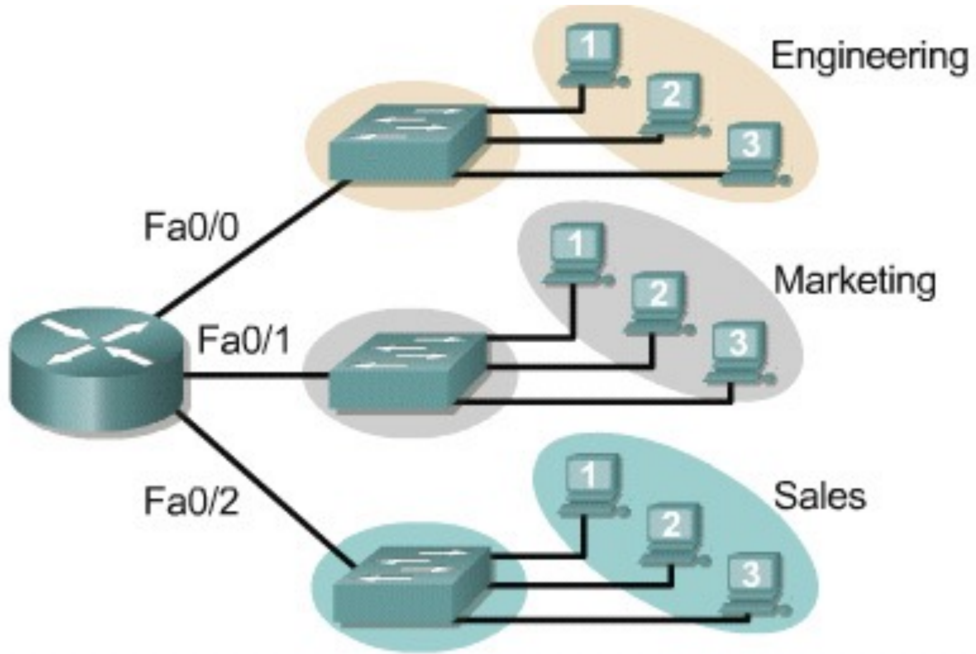


- A switch creates a broadcast domain
- VLANs help to manage broadcast domains
- VLANs can be defined or port groups, users or protocols

- LAN switches and network management software provide a mechanism to create VLANS

Broadcast domains with VLANs and routers

- Layer 3 routing allows the router to send packets to the three different broadcast domains.



- Three switches and one router could be used without VLANs:
  - Switch for Engineering
  - Switch for Sales
  - Switch for Marketing
  - Each switch treats all ports as members of one broadcast domains

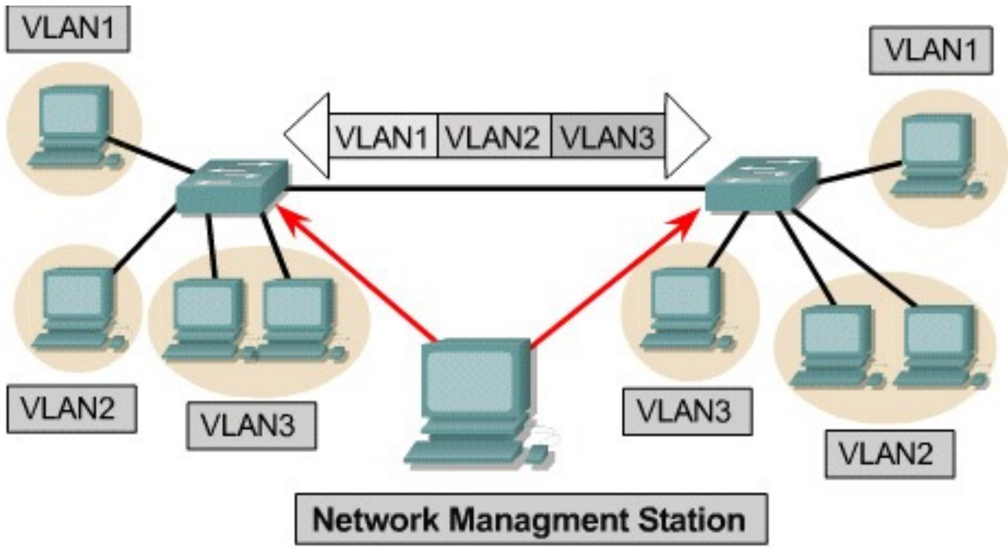
Broadcast domains with VLANs and routers

Implementing VLANs on a switch causes the following to occur:

- The switch maintains a separate bridging table for each VLAN.
- If the frame comes in on a port in VLAN 1, the switch searches the bridging table for VLAN 1.
- When the frame is received, the switch adds the source address to the bridging table if it is currently unknown.
- The destination is checked so a forwarding decision can be made.
- For learning and forwarding the search is made against the address table for that VLAN only.

VLAN operation

- Each switch port could be assigned to a different VLAN.
- Ports assigned to the same VLAN share broadcasts.
- Ports that do not belong to that VLAN do not share these broadcasts.



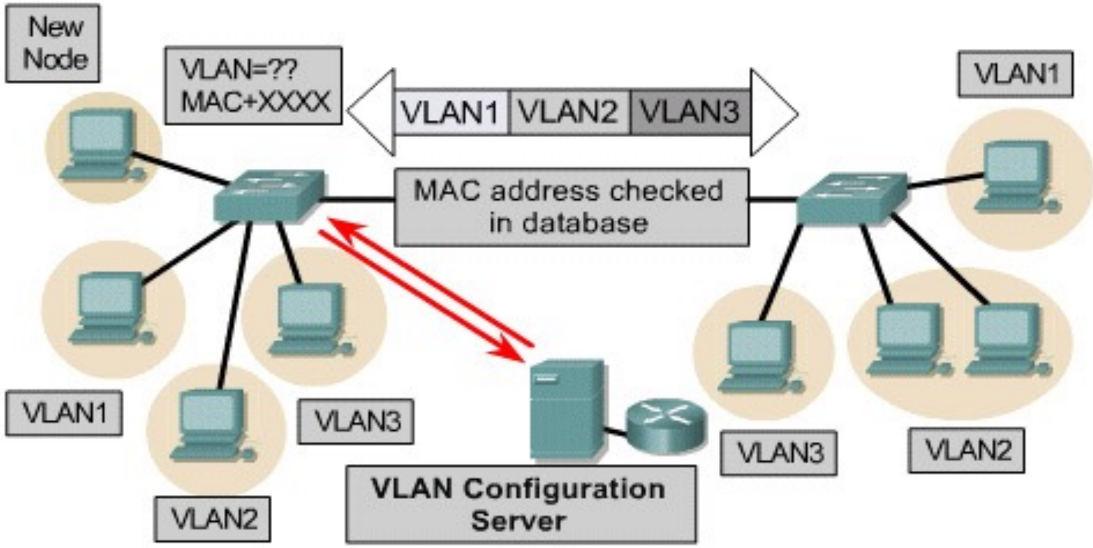
- Assign ports (port-centric)
- Static VLANs are secure, easy to configure and monitor

VLAN operation

- Users attached to the same shared segment, share the bandwidth of that segment.
- Each additional user attached to the shared medium means less bandwidth and deterioration of network performance.
- VLANs offer more bandwidth to users than a shared network.
- The default VLAN for every port in the switch is the management VLAN.
- The management VLAN is always VLAN 1 and may not be deleted. All other ports on the switch may be reassigned to alternate VLANs.

VLAN operation

- Dynamic VLANs allow for membership based on the MAC address of the device connected to the switch port.
- As a device enters the network, it queries a database within the switch for a VLAN membership.



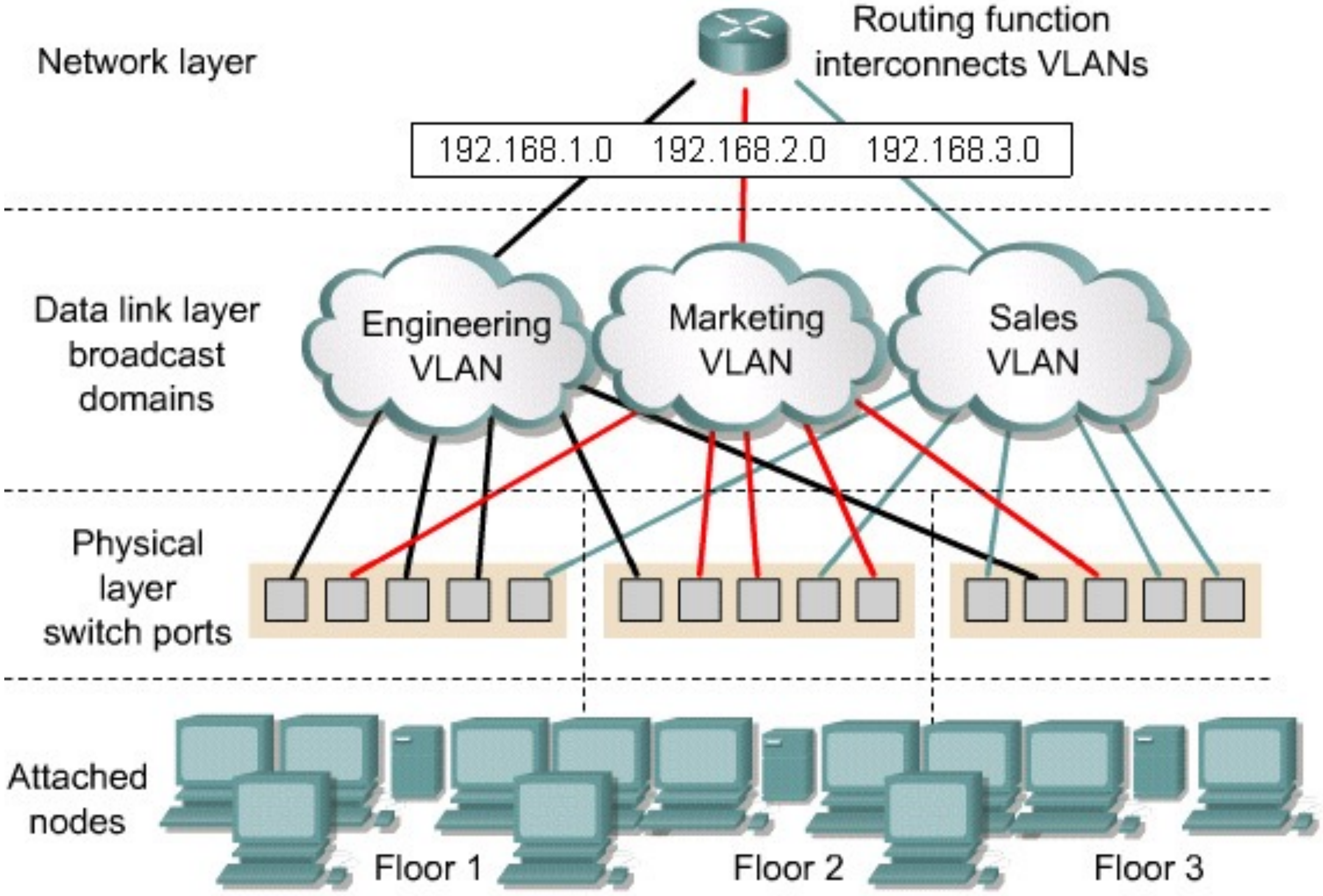
- Less administration in wiring closet
- Notification when unrecognized user is added to network

- Less administration in wiring closet
- Notification when unrecognized user is added to network

VLAN operation

- In port-based or port-centric VLAN membership, the port is assigned to a specific VLAN membership independent of the user or system attached to the port.

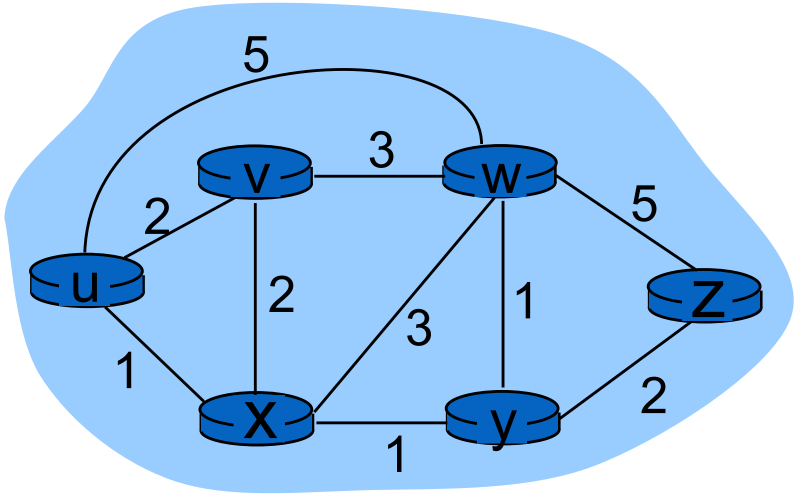
- All users of the same port must be in the same VLAN.



Topic of the lecture

- VLAN
- Routing algorithms
  - Link state
  - Distance vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast routing

Graph Abstraction



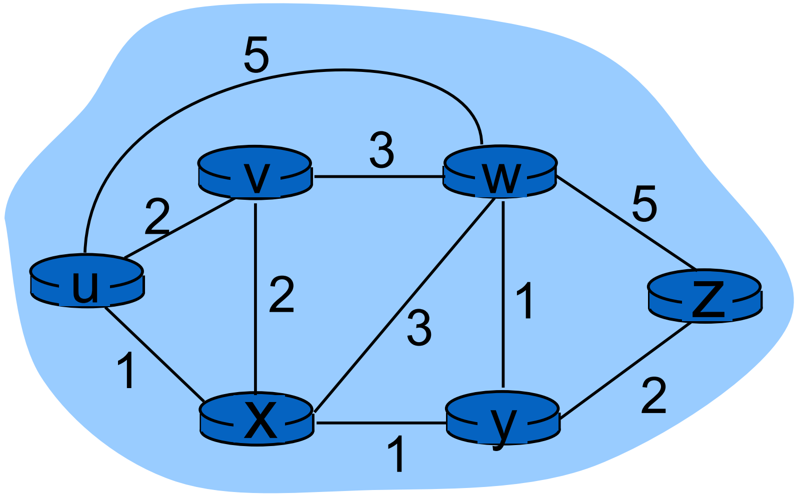
Graph: G = (N,E)

N = set of routers = { u, v, w, x, 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 is directed
- *Aside:* graph abstraction is useful in other network contexts, e.g., P2P, where *N* is set of peers and *E* is set of TCP connections

Graph Abstraction: Costs



$c(x,x')$ = cost of link (x,x')

For Example: $c(w,z) = 5$

cost could be a nominal value (1), or inversely related to bandwidth, or inversely related to congestion or …

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

Key question: What is the least-cost path between u and z ?
Routing algorithm: Algorithm that finds that least cost path

Routing Algorithm Classification

Q: Global or decentralized information?

Global:

- All routers have complete topology, link cost info
- "link state" algorithms

Decentralized:

- Router knows physically-connected neighbors, link costs to neighbors
- Iterative process of computation, exchange of info with neighbors
- "Distance vector" algorithms

Q: Static or dynamic?

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

- Net topology, 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
  - Gives *forwarding table* for that node
- Iterative: after *k* iterations, know least cost path to *k* destination's

Notation:

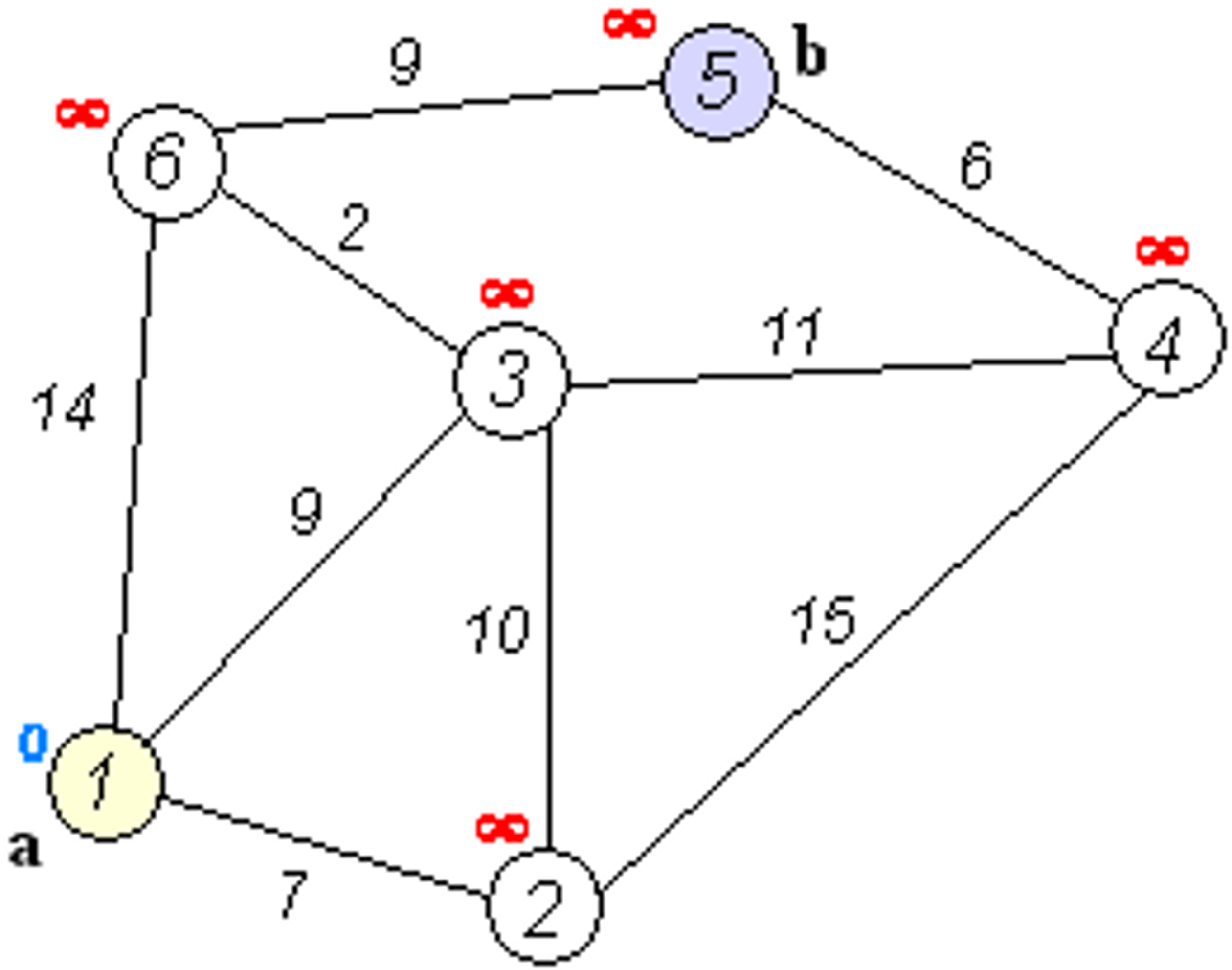
- c(x,y): link cost from node x to y; $= \infty$ if not direct neighbors
- D(v): current value of cost of path from source to destination v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijsktra's algorithm for least cost path

```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
        then D(v) = c(u,v)
    else D(v) = ∞

Loop
  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) )
  /* new cost to v is either old cost to v or known
   shortest path cost to w plus cost from w to v */
until all nodes in N'
```

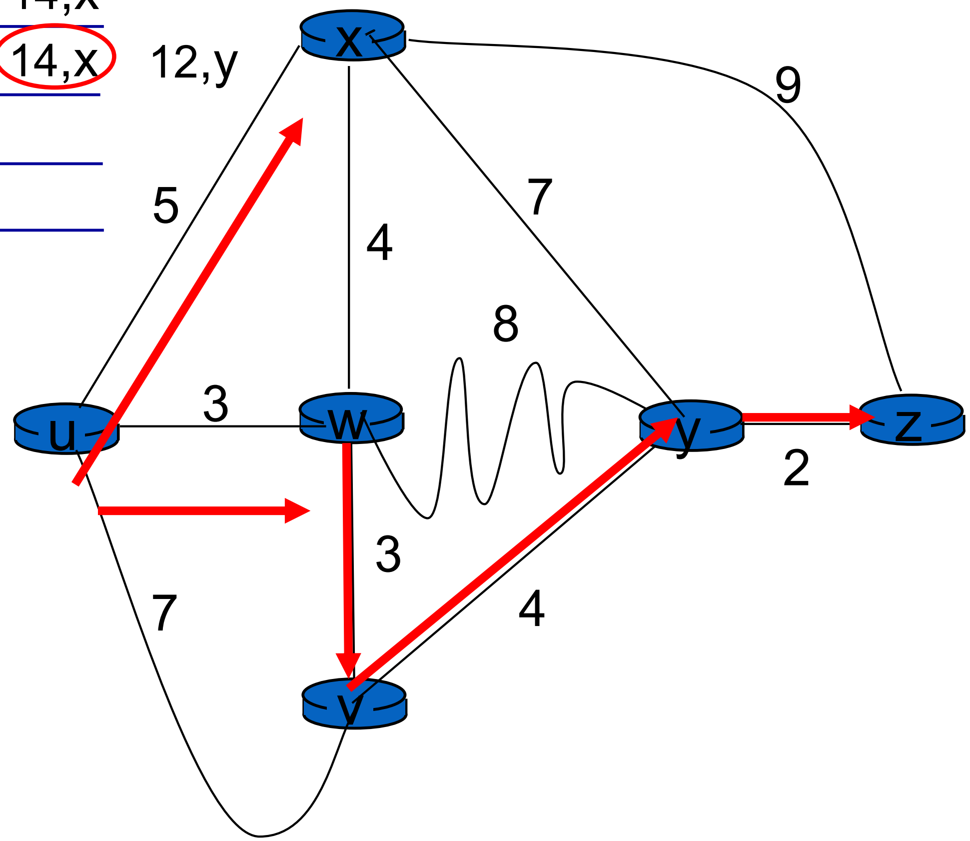
Visual version of Dijsktra's Algorithm



From: https://upload.wikimedia.org/wikipedia/commons/5/57/Dijkstra_Animation.gif

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 12,y
4	uwxvy					
5	uwxvyz					

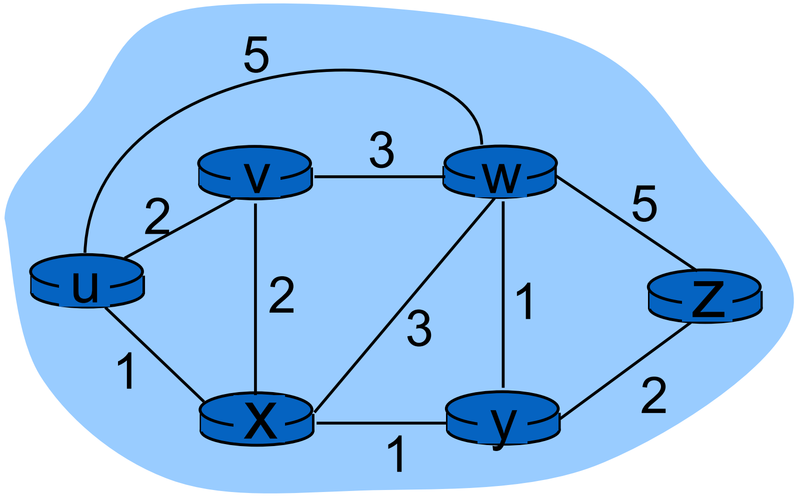


Notes:

- Construct shortest path tree by tracing predecessor nodes
- Ties can exist (can be broken arbitrarily)

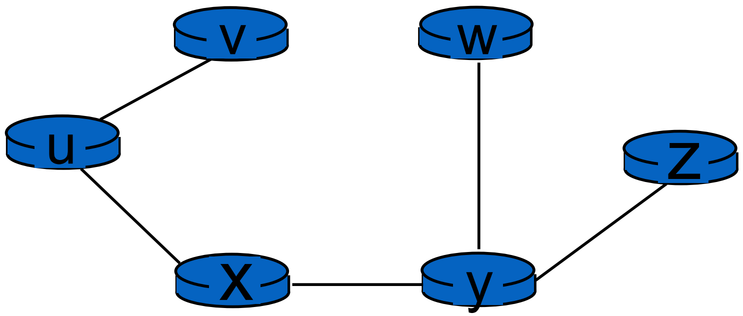
Dijkstra's algorithm: another example

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					



Dijkstra's algorithm: example (2)

Resulting shortest-path tree from u:



Resulting 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

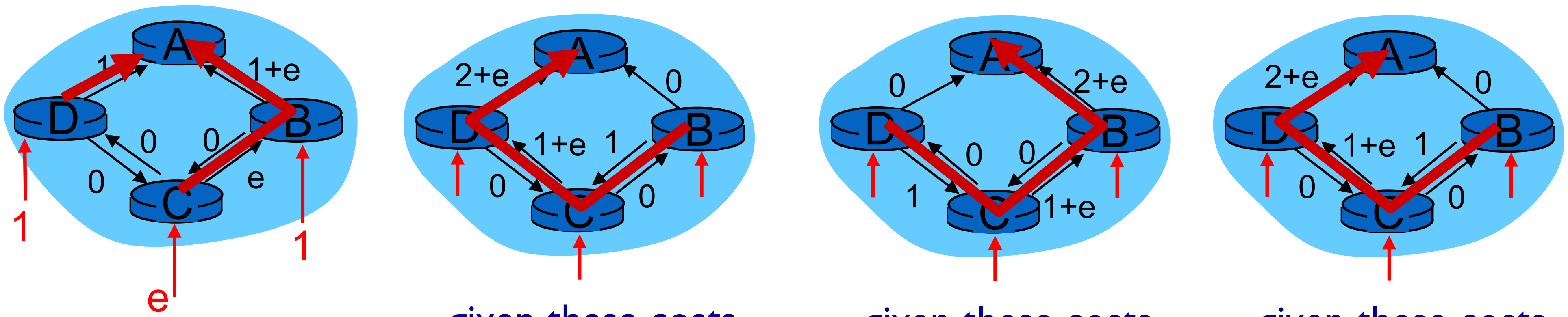
Algorithm complexity: n nodes

- Each iteration: need to check all nodes, w, not in N
- n(n+1)/2 comparisons: O(n^2)
- More efficient implementations possible: O(*nlogn*)

Oscillations possible:

- e.g., support link cost equals amount of carried traffic:

- Example of Oscillations with congestion-sensitive routing



initially

given these costs, find new routing…. resulting in new costs

given these costs, find new routing…. resulting in new costs

given these costs, find new routing…. resulting in new costs

Topic of the lecture

- Routing algorithms
  - Link state
  - Distance vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast routing

Distance Vector Algorithm

Bellman-Ford equation (dynamic programming)

Let

$d_x(y)$:= cost of least-cost path from x to y

then

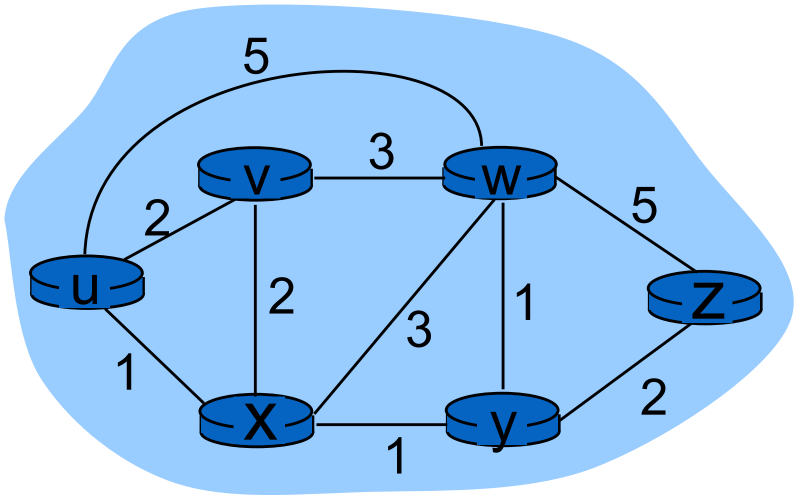
$$d_x(y) = \mathit{min}_v\{c(x,v) + d_v(y)\}$$

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:

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

Node achieving 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 neighbor v: $c(x,v)$
  - maintains its neighbors' distance vectors. For each neighbor v, x maintains
    $\mathbf{D}_v = [D_v(y): y \in N]$

Distance Vector Algorithm

Key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives a new DV estimate from a neighbor, it updates its own DV 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 minor, natural conditions, 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
- DV update message from neighbor

Distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

Each node:

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

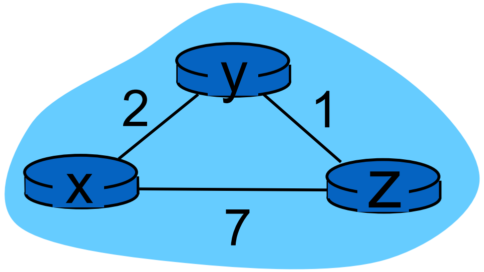
node y table

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

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

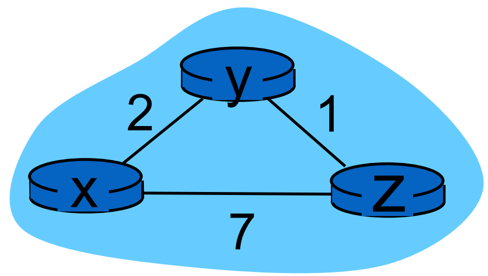
node z table

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

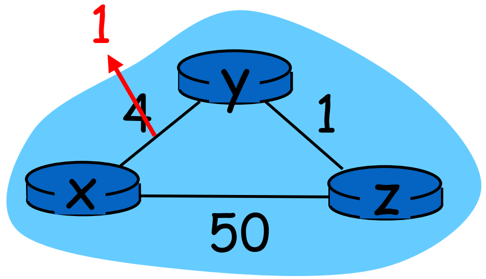
time



Distance Vector: link cost changes

Link cost changes:

- Node detects local link cost change
- Updates routing info, recalculates distance vector
- How to update the routing information?
  - if distance vector changes, notify neighbors



“good news travels fast”

t_0 : *y* detects link-cost change, updates its DV, informs its neighbors.

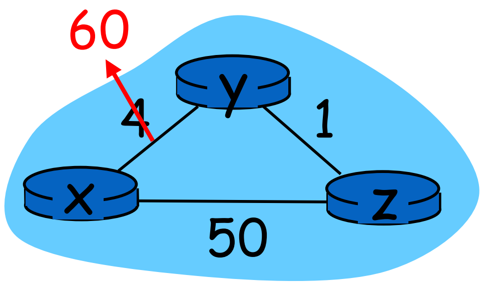
t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.

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

Link cost changes:

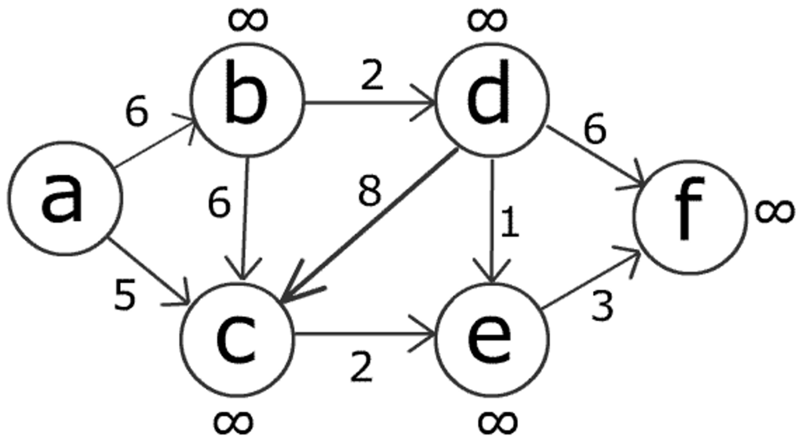
- Node detects local link cost change
- *bad news travels slow* - "count to infinity" problem!
- 44 iterations before algorithm stabilizes: see text-book

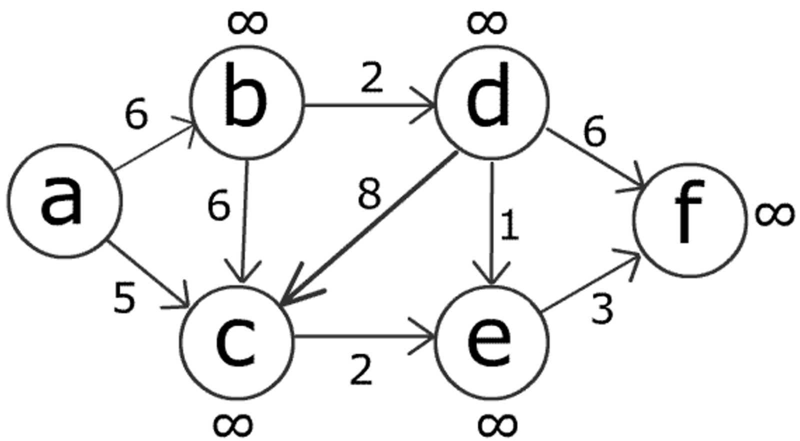


Poisoned reverse:

- 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?

Summary of the two algorithms 1/3





From: https://commons.wikimedia.org/wiki/File:Shortest_path_Dijkstra_vs_BellmanFord.gif

Summary of the two algorithms 2/3

- Bellman-Ford is used for performing distance vector routing whereas Dijsktra is used for performing the link state routing.
- In distance vector routing the routers receive the topological information from the neighbour point of view. On the contrary, in link state routing the routers receive complete information on the network topology.
- Distance vector routing calculates the best route based on the distance (fewest number of hops). Link state routing calculates best route on the basis of least cost.
- Link state routing updates only the link state while Distance vector routing updates full routing table.
- The frequency of update in both routing techniques is different: distance vector updates periodically, whereas link state update frequency employs triggered updates.

Summary of the two algorithms 3/3

- The utilization of CPU and memory in distance vector routing is lower than in the link state routing.
- The distance vector routing is simple to implement and manage. In contrast, the link state routing is complex and requires a trained network administrator.
- The convergence time in distance vector routing is slow, and it usually suffers from count to infinity problem. Conversely, the convergence time in link state routing is fast, and it is more reliable.
- Distance vector does not have hierarchical structure while in link state routing the nodes can have a hierarchical structure.

From: https://techdifferences.com/difference-between-distance-vector-routing-and-link-state-routing.html

Comparison of LS and DV algorithms 1/2

Message complexity

- ***LS:*** with n nodes, E links, O(nE) msgs sent
- ***DV:*** exchange between neighbors only
  - convergence time varies

Speed of convergence

- ***LS:*** $O(n^2)$ algorithm requires O(nE) msgs
  - may have oscillations
- ***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 propagate through network

Comparison of LS and DV algorithms 2/2

Feature	Distance vector routing	Link state routing
Algorithm	Bellman Ford	Dijsktra
Network view	Topology information from the neighbour viewpoint	Complete information on the network topology
Best path calculation	Based on least number of hops	Based on the cost
Updates	Full routing table	Link state updates
Updates frequency	Periodic updates	Triggered updates
CPU and memory	Low utilisation	Intensive
Simplicity	High simplicity	Requires a trained network administrator
Convergence time	Moderate	Fast
Updates	On broadcast	On multicast
Hierarchical structure	No	Yes
Intermediate nodes	No	Yes

From: https://techdifferences.com/difference-between-distance-vector-routing-and-link-state-routing.html

Topic of the lecture

- Routing algorithms
  - Link state
  - Distance vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP
- Broadcast and multicast routing

Hierarchical Routing

our routing study thus far – idealization, as

- all routers assumed identical
- network assumed “flat”

… *not* true in practice

Scale: with billions destinations:

- can’t store all destinations in routing tables!
- Routing table exchange would swamp links!

Administrative autonomy

- Internet = network of networks
- Each network admin may want to control routing in its own network

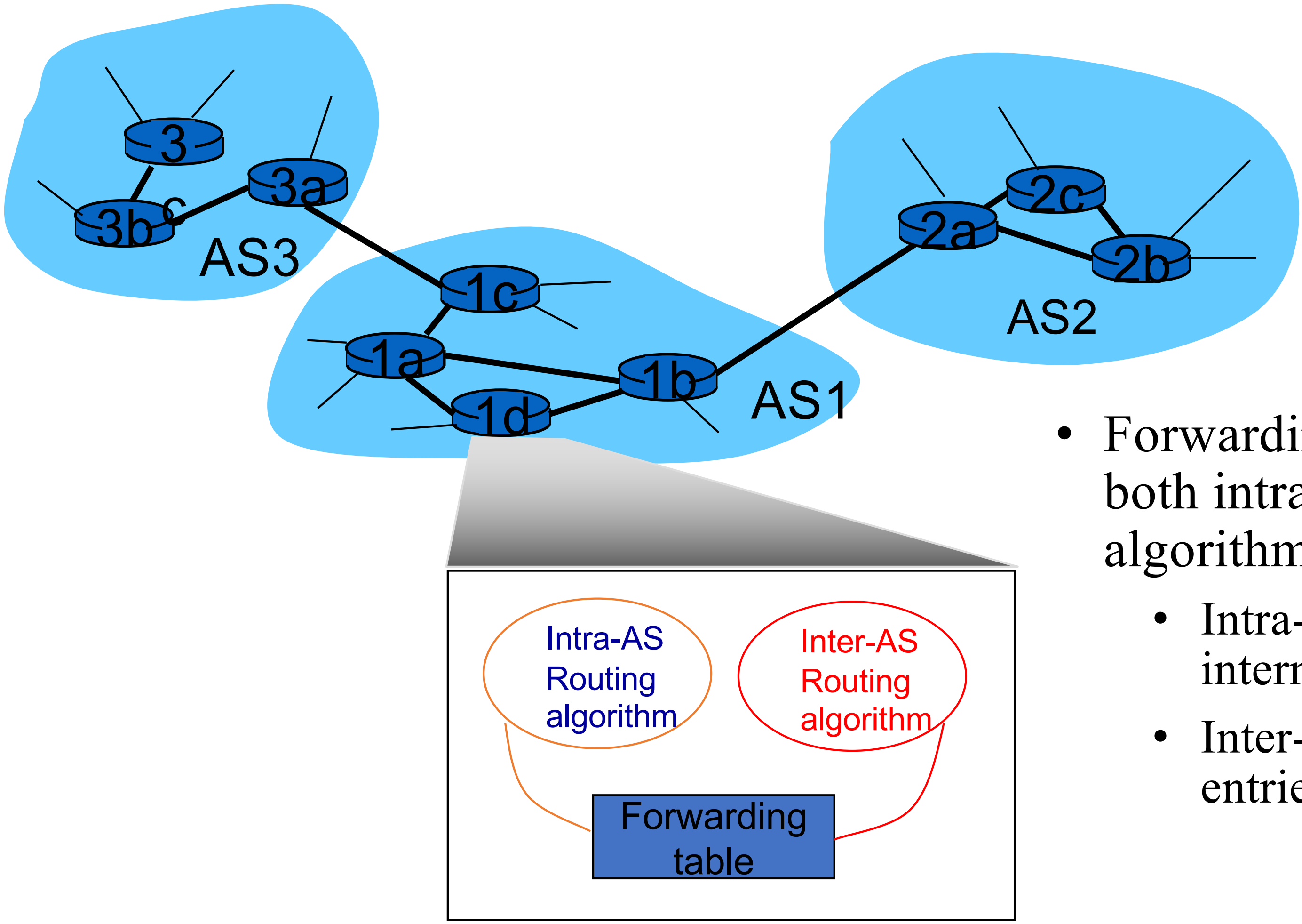
Hierarchical Routing

- Aggregate routers into regions, "Autonomous Systems" (AS)
- Routers in same AS run same routing protocol
  - "Intra-AS" routing protocol
  - Routers in different AS can run different intra-AS routing protocol

Gateway router:

- At "edge" of its own AS
- has link to router in another AS

Interconnected ASes



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

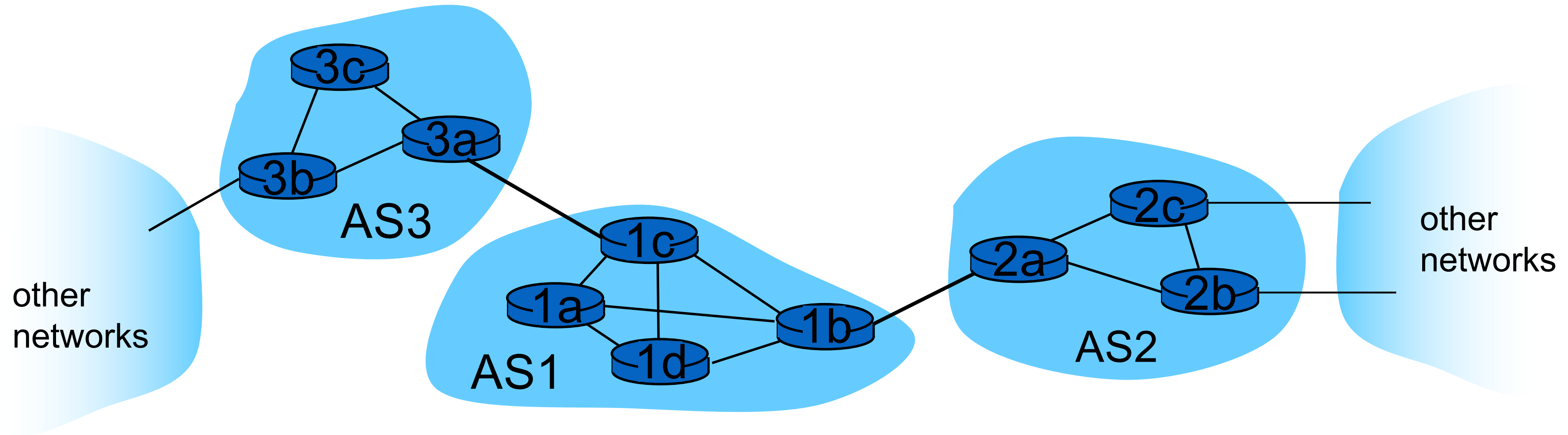
Inter-AS Tasks

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

AS1 must:

1. Learn which destinations are reachable through AS2, which through AS3
2. Propagate this reachability info to all routers in AS1

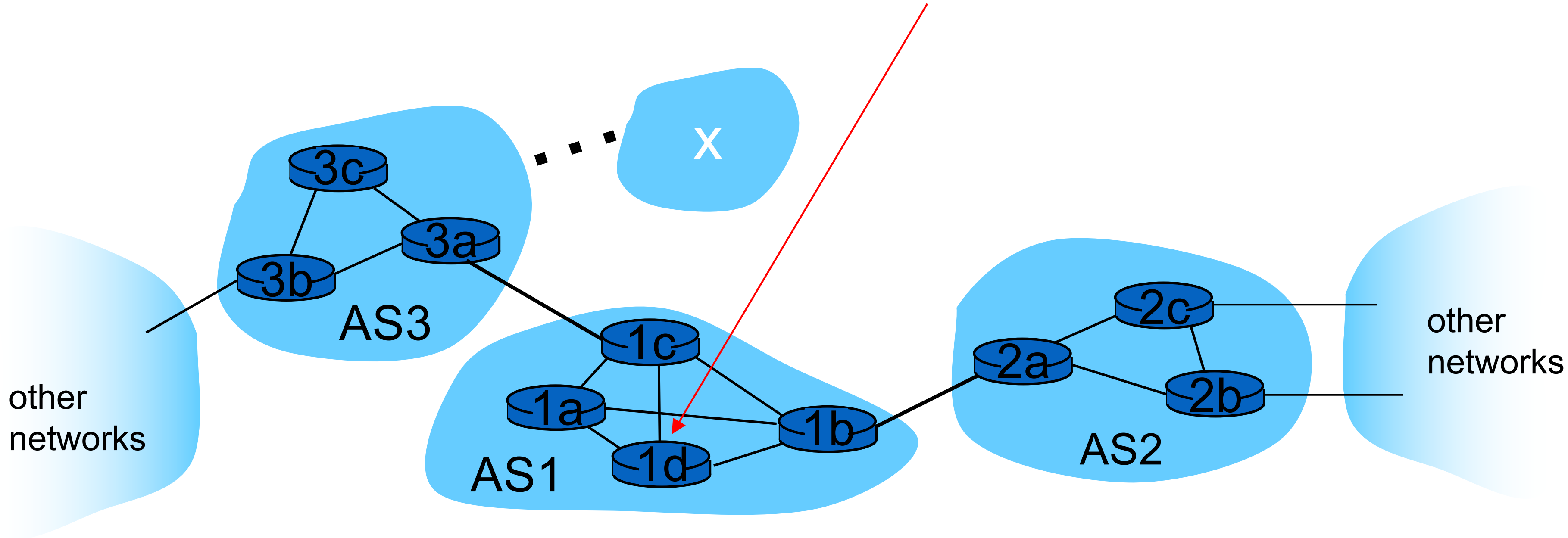
Job of inter-AS routing!



Example: Setting Forwarding Table in Router 1d

- Suppose AS1 learns (via inter-AS protocol) that subnet *x* reachable via AS3 (gateway 1c), but not via AS2
  - Inter-AS protocol propagates reachability info to all internal routers

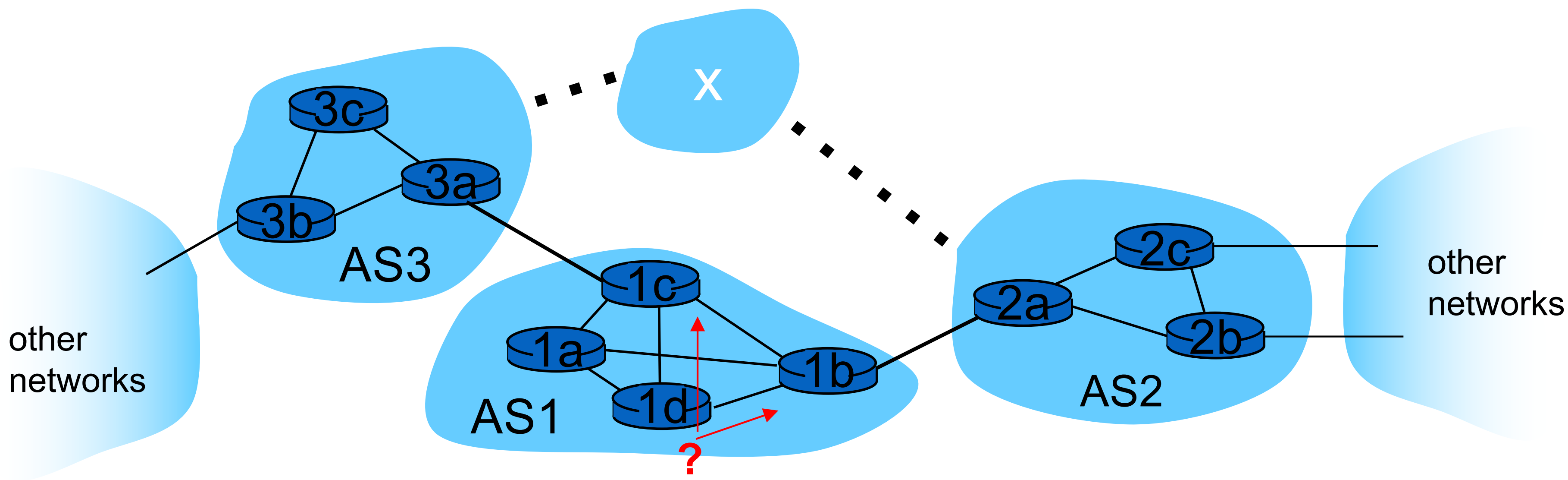
- Router 1d determines from intra-AS routing info that its interface *I* is on the least cost path to 1c
  - Installs forwarding table entry *(x, I)*



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 which gateway it should forward packets towards for destinatio x
  - This is also job of inter-AS routing protocol!



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 dest x
  - this is also job of inter-AS routing protocol!
- *Hot potato routing: send* packet towards closest of two routers.

learn from inter-AS protocol that subnet *x* is reachable via multiple gateways → use routing info from intra-AS protocol to determine costs of least-cost paths to each of the gateways → hot potato routing: choose the gateway that has the smallest least cost → determine from forwarding table the interface *I* that leads to least-cost gateway. Enter *(x,I)* in forwarding table

Topic of the lecture

- Routing algorithms
  - Link state
  - Distance vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF
  - BGP

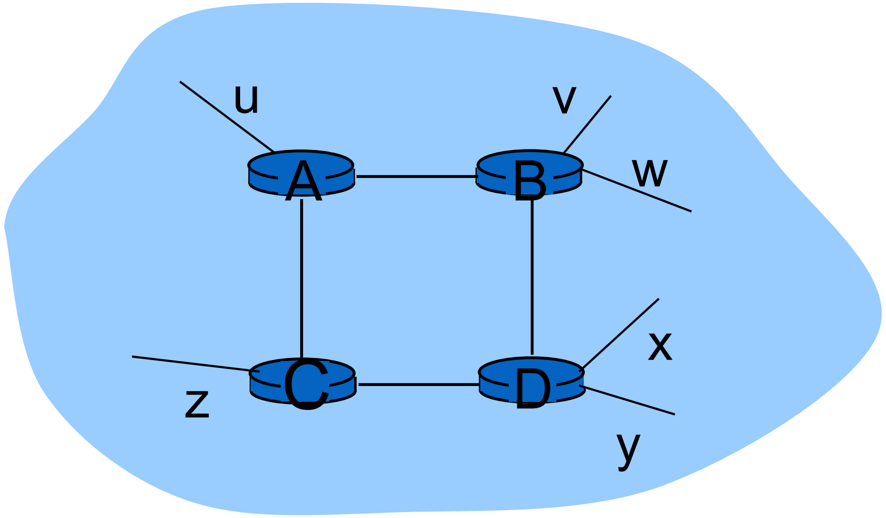
Intra-AS Routing

- Also known as *interior gateway protocols (IGP)*

- Most common intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)

RIP (Routing Information Protocol)

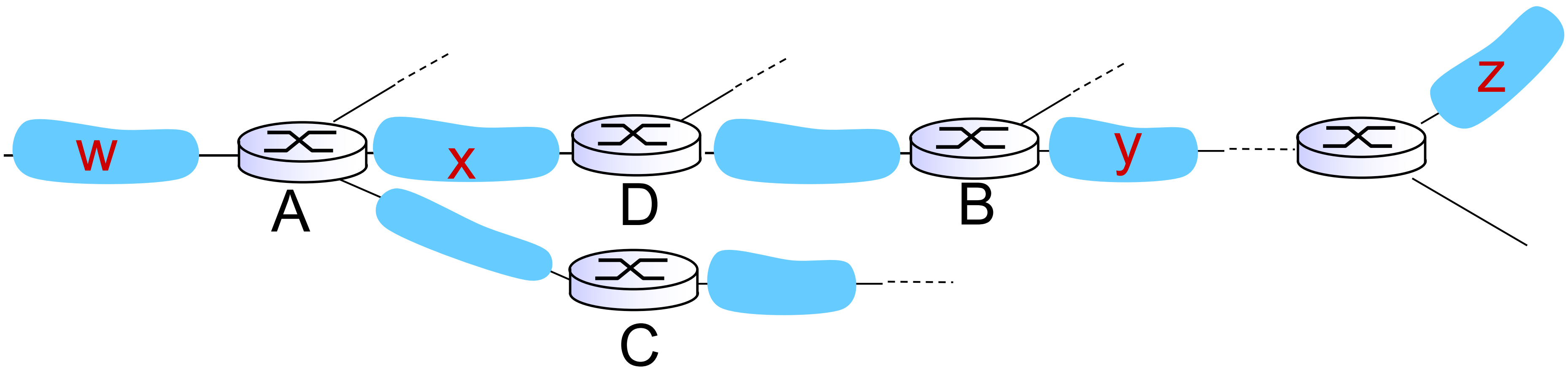
- Included in BSD-UNIX distribution in 1982
- Distance vector algorithm
  - Distance metric: # hops (max = 15 hops), each link has cost 1
  - DVs exchanged with neighbors every 30 sec in response message (aka advertisement)
  - Each advertisement: list of up to 25 destination *subnets (in IP addressing sense)*



from router A to destination *subnets:*

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

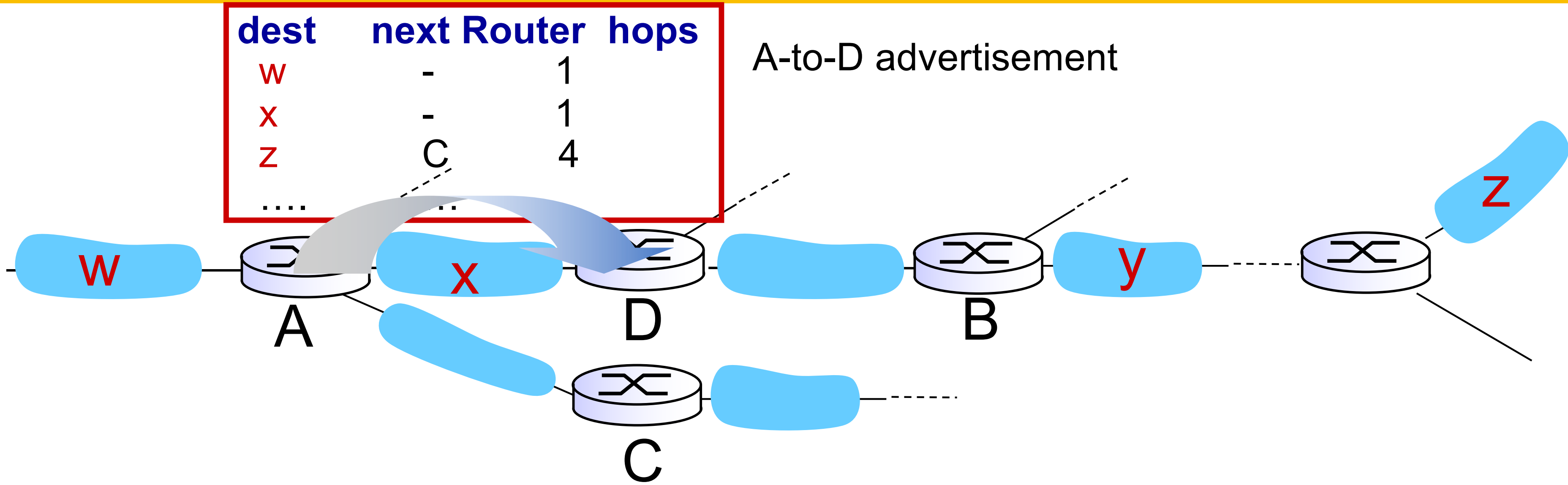
RIP: Example



Routing table in router D

Destination subnet	Next router	# hops to destination
w	A	2
y	B	2
z	B	7
x	--	1
….	….	….

RIP: example



routing table in router D

Destination subnet	Next router	# hops to destination
w	A	2
y	B	1
z	C	4
x	--	1
....	....	....

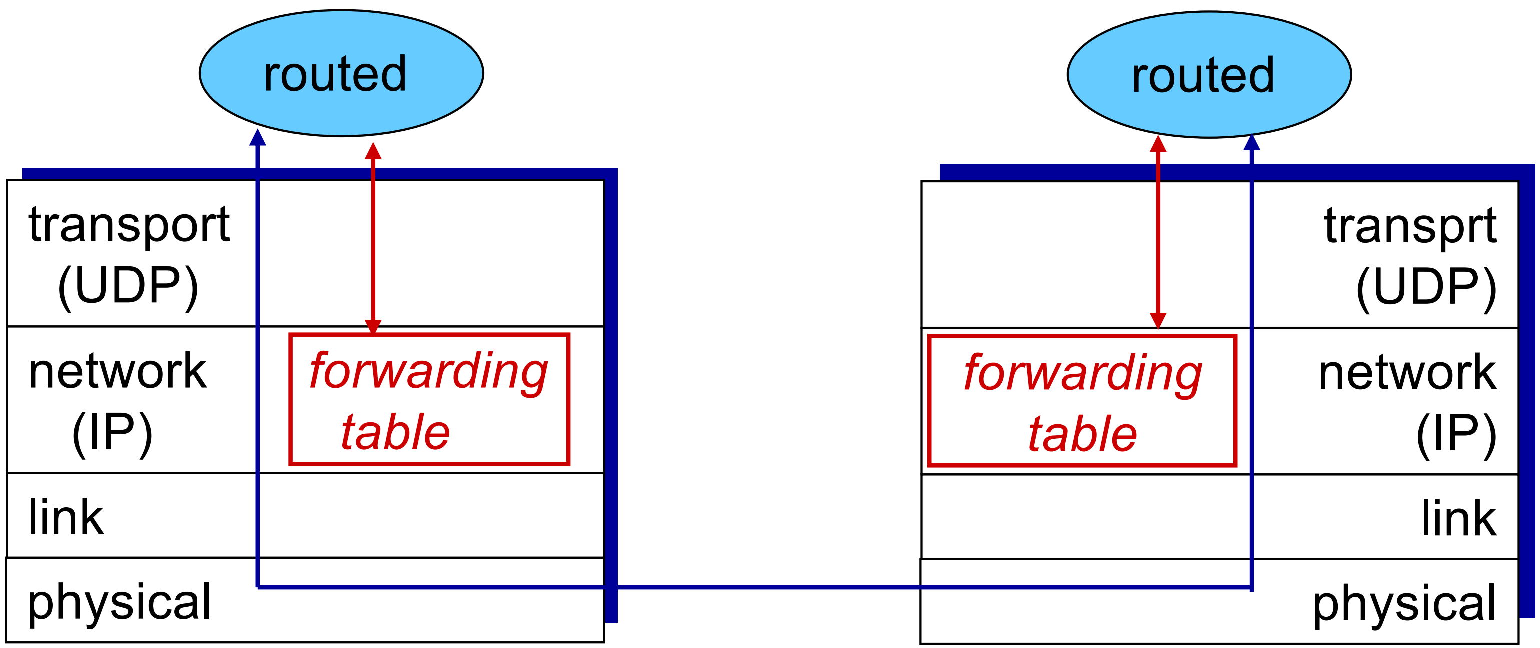
RIP: link failure, recovery

If no advertisement heard after 180 sec → neighbor/link declared dead

- Routes via such neighbor invalidated
- New advertisements sent to neighbors
- Neighbors in turn send out new advertisements (if tables changed)
- Link failure info quickly (?) propagates to entire net
- *Poison reverse* used to prevent ping-pong loops (infinite distance = 16 hops)

RIP Table Processing

- RIP routing tables managed by *application-level* process called route-d (daemon)
- advertisements sent in UDP packets, periodically repeated



OSPF (Open Shortest Path First)

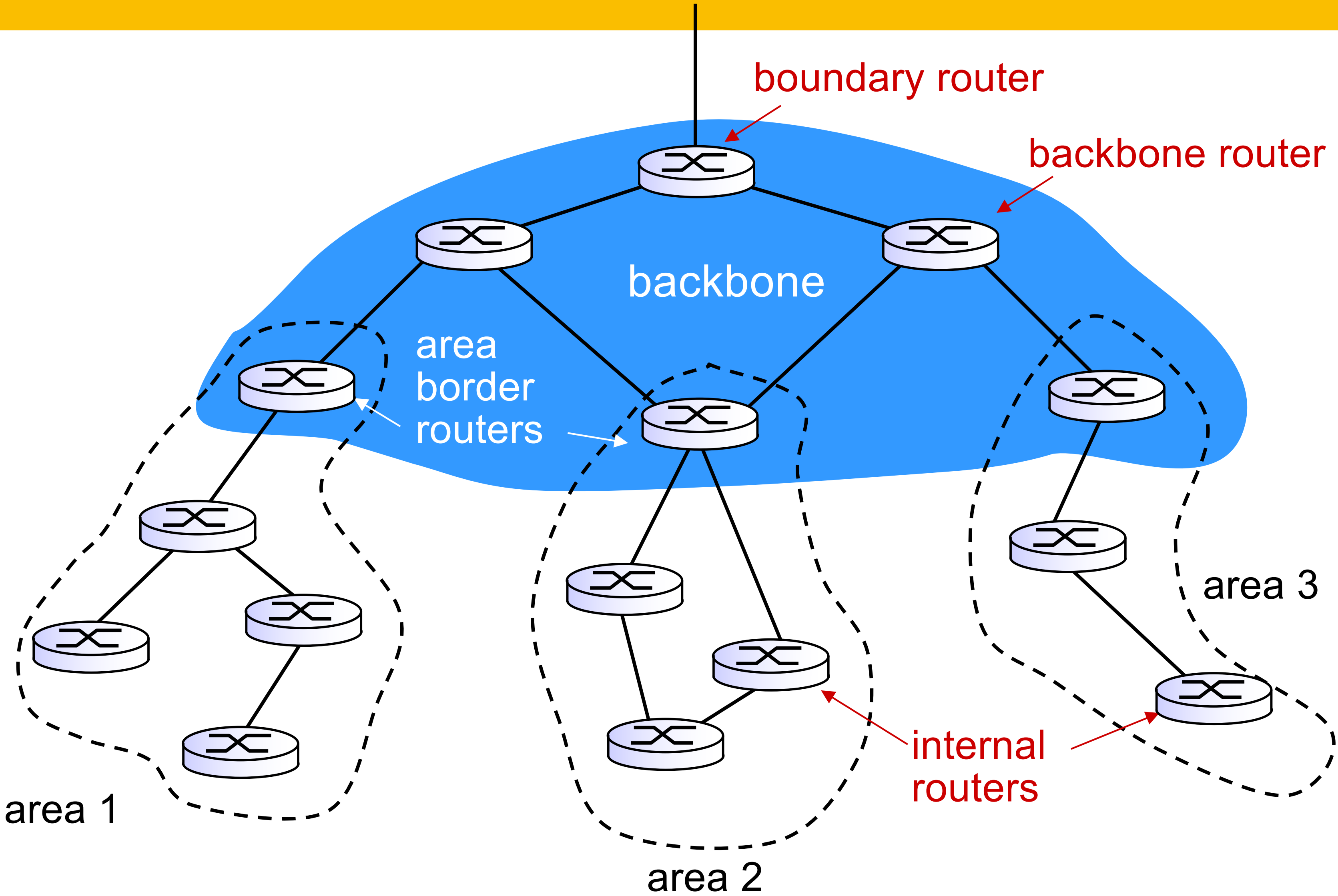
- "open": publicly available
- Uses link state algorithm
  - LS packet dissemination
  - topology map at each node
  - route computation using Dijkstra's algorithm
- OSPF advertisement carries one entry per neighbor
- Advertisements flooded to *entire* AS
  - Carried in OSPF messages directly over IP (rather than TCP or UDP
- *IS-IS routing* protocol: nearly identical to OSPF

OSPF “advanced” features (not in RIP)

- *Security:* all OSPF messages authenticated (to prevent malicious intrusion)
- Multiple same-cost paths allowed (only one path in RIP)
- For each link, multiple cost metrics for different TOS (e.g., satellite link cost set “low” for best effort ToS; high for real time ToS)
- Integrated uni- and multicast support:
  - Multicast OSPF (MOSPF) uses same topology data base as OSPF
- Hierarchical OSPF in large domains.

TOS = Type Of Service

Hierarchical OSPF



Summary

- VLANs
- Routing algorithms
  - Link state
  - Distance vector
  - Hierarchical routing
- Routing in the Internet
  - RIP
  - OSPF