Romain Fontugne

2023 Fall Semester

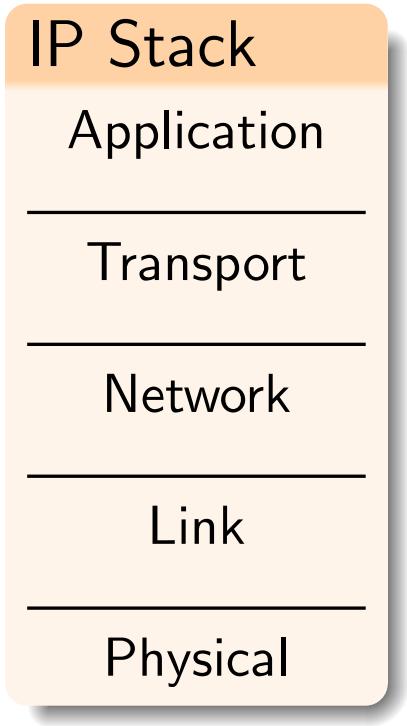
Information Network Systems

The Network Layer (2)

Last lecture:

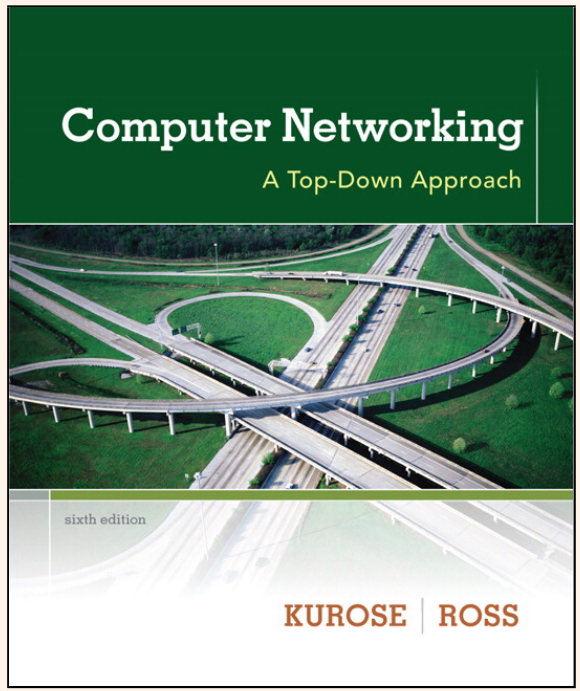
The network layer (1)

- Network layer services
- IP Addressing
- IPv4
- IPv6
- ICMP



Today's Lecture: Network layer (2)

1. Forwarding Table
2. Routing Algorithms
   - Link State
   - Distance Vector
   - Hierarchical Routing
3. Routing in the Internet
   - RIP
   - OSPF
   - BGP
4. Study cases
   - YouTube Hijacking
   - Internet AS graph



- Chapter 4

Packet-Switching: key functions

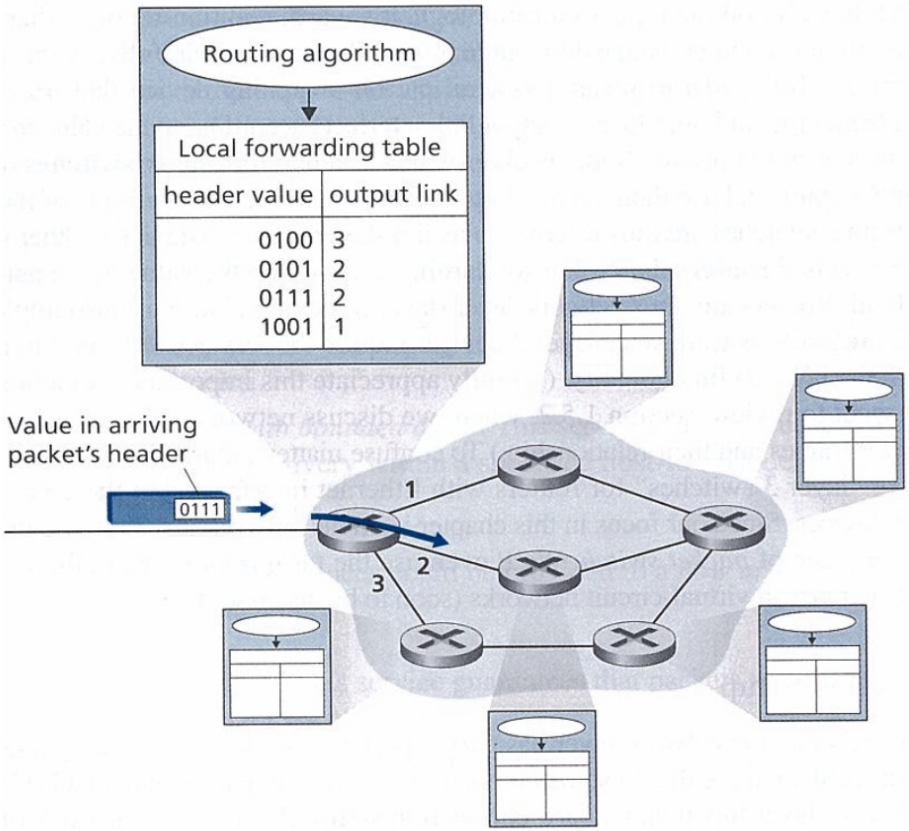
Routing algorithms

- Determines source-destination route taken by packets

Forwarding

- Look at the dest. IP address
- Move packets to the appropriate router's output link

→ In IPv4, 4 billion IP addresses possible! so forwarding tables list range of addresses (subnet)



Forwarding Table: Example

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011000 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Question: but what happens if ranges don't divide up so nicely?

Longest Prefix Matching

When looking for forwarding table entry for given destination address, use **longest** address prefix that matches destination address.

Destination Address Range	Link Interface
11001000 00010111 00010*** ********	0
11001000 00010111 00011000 ********	1
11001000 00010111 00011*** ********	2
otherwise	3

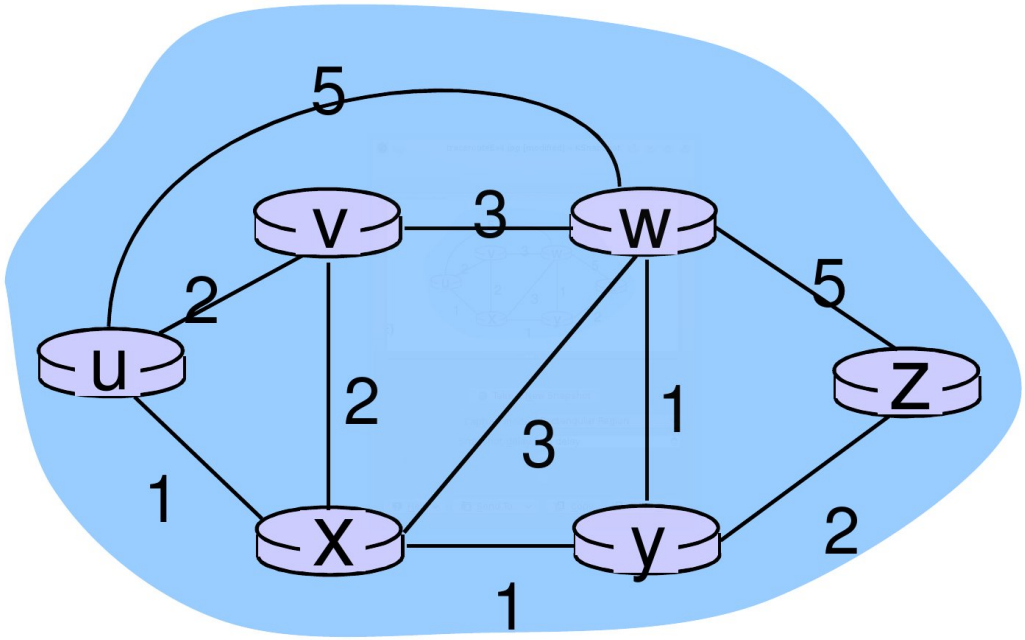
Examples:

- Dest. Addr.: 11001000 00010111 0001**0110 10100001** which interface?
- Dest. Addr.: 11001000 00010111 0001**1000 10101010** which interface?

Routing Algorithms: Graph Abstraction

Graph: Nodes and Edges

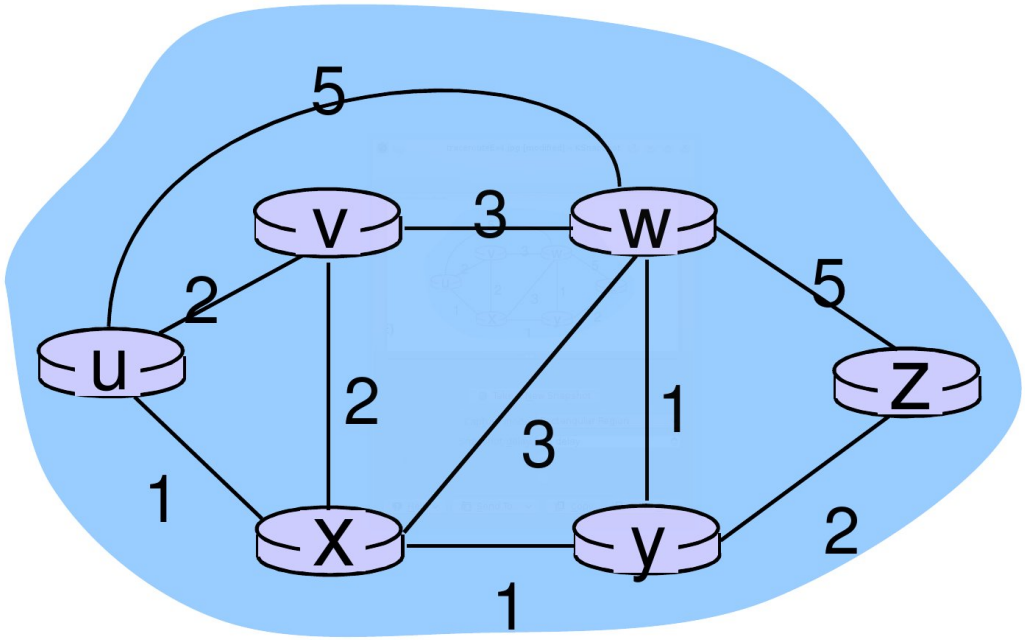
- $G = (N, E)$
- $N = set\ of\ routers = u, v, w, x, y, z$
- $E = set\ of\ links =$
  $(u, v), (u, x), (v, x), (v, w), (x, w), (x, y), (w, y), (w, z), (y, z)$



Routing Algorithms: Graph Abstraction

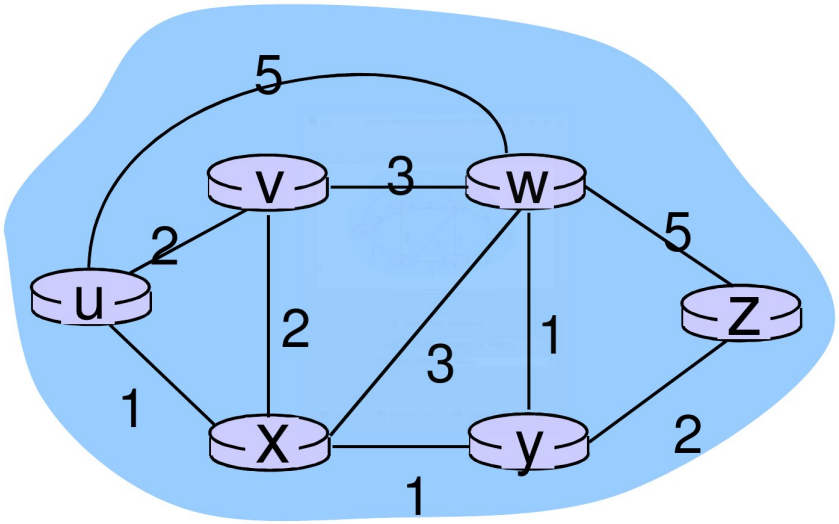
Graph: Nodes and Edges

- $G = (N, E)$
- $N = set\ of\ routers = u, v, w, x, y, z$
- $E = set\ of\ links =$
  $(u, v), (u, x), (v, x), (v, w), (x, w), (x, y), (w, y), (w, z), (y, z)$



aside: graph abstraction is useful in other network contexts, e.g., P2P, where N is set of peers and E is set of connections

Graph Abstraction: Costs



$c(x, y)$ = cost of link between x and y

- e.g., $c(w, z) = 5$
- could be inversely related to bandwidth
- or inversely related to congestion
- $c(x_1, x_2, x_3, ..., x_p) = c(x_1, x_2)+c(x_2, x_3)+...+c(x_{p-1}, x_p)$

- Key question: what is the least-cost path between u and z ?
- Routing algorithm: algorithm that finds that least-cost path (aka shortest path)

Routing Algorithm Classification

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**

Routing Algorithm Classification

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**

Static or dynamic?

- Static:
  - routes change slowly over time
- Dynamic:
  - routes change more quickly
  - periodic update in response to link cost changes

Dijkstra: A Link-State Routing Algorithm

Dijkstra's algorithm

- Network 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 destinations

Dijkstra: A Link-State Routing Algorithm

Dijkstra's algorithm

- Network 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 destinations

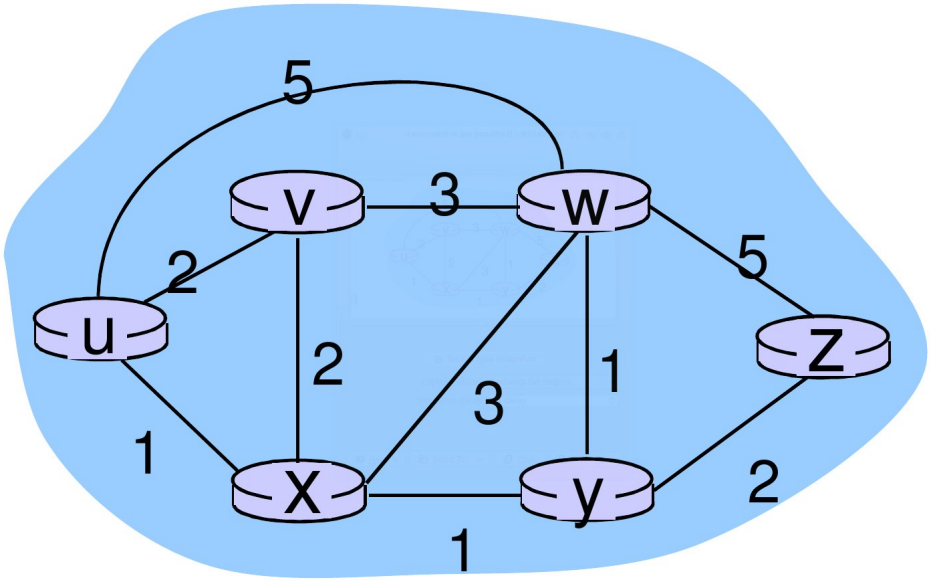
Notation:

- $c(x, y)$: 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

Dijkstra's Algorithm

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

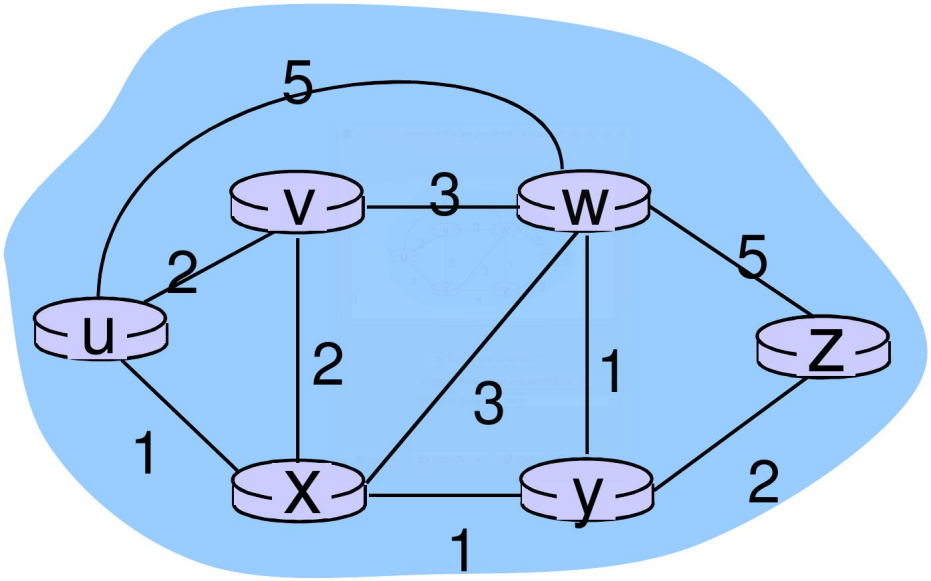
Dijkstra's Algorithm: Example



Notes:

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

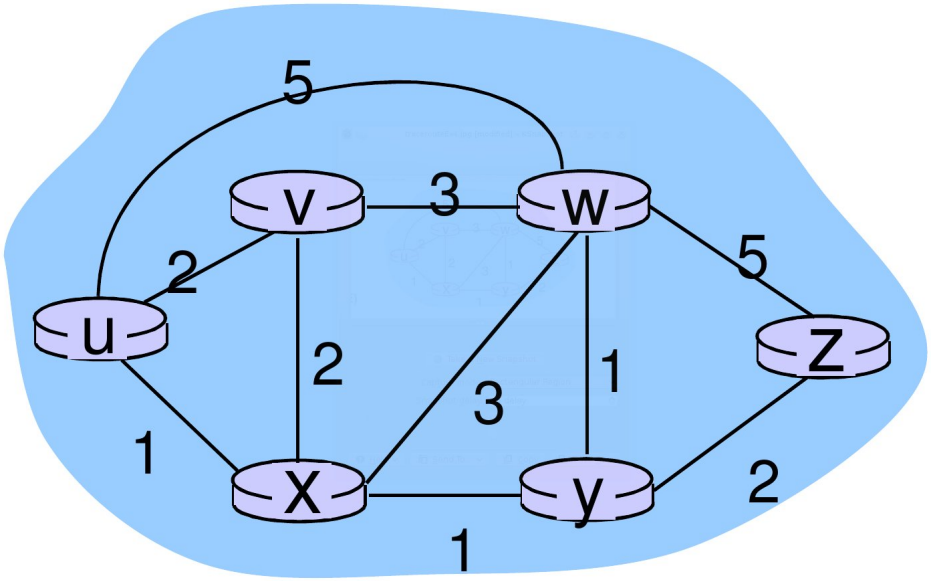
Dijkstra's Algorithm: Example



Notes:

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	∞	∞

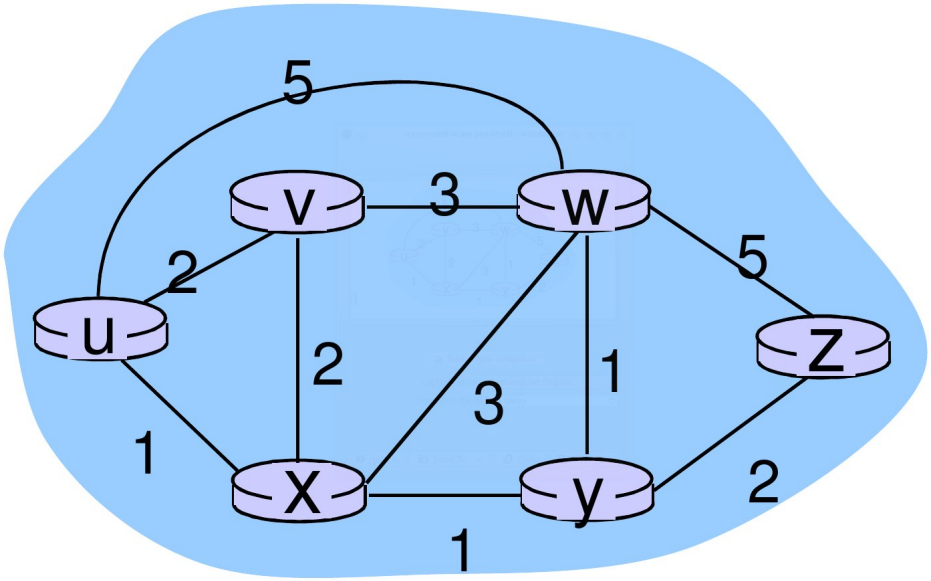
Dijkstra's Algorithm: Example



Notes:

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					

Dijkstra's Algorithm: Example

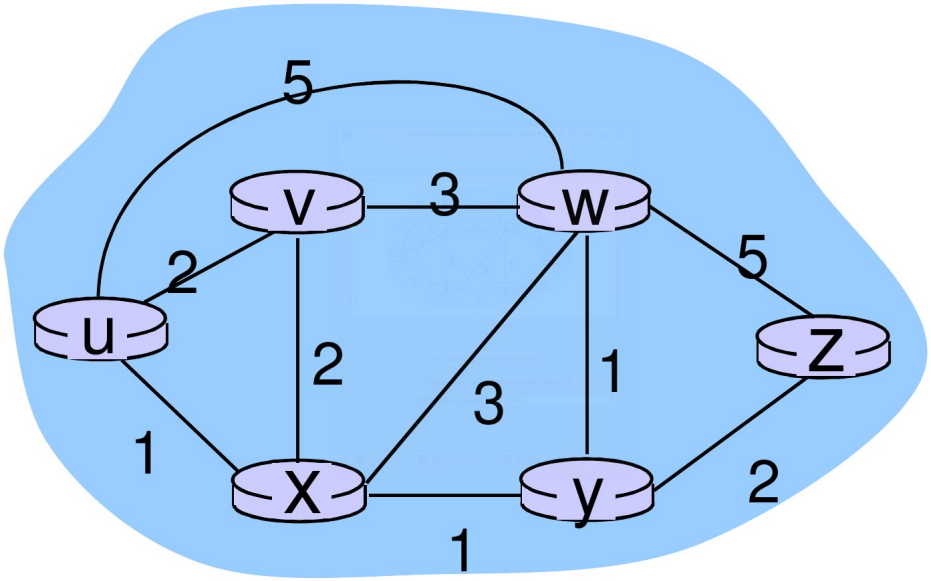


Notes:

- Break the ties arbitrarily

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	∞

Dijkstra's Algorithm: Example

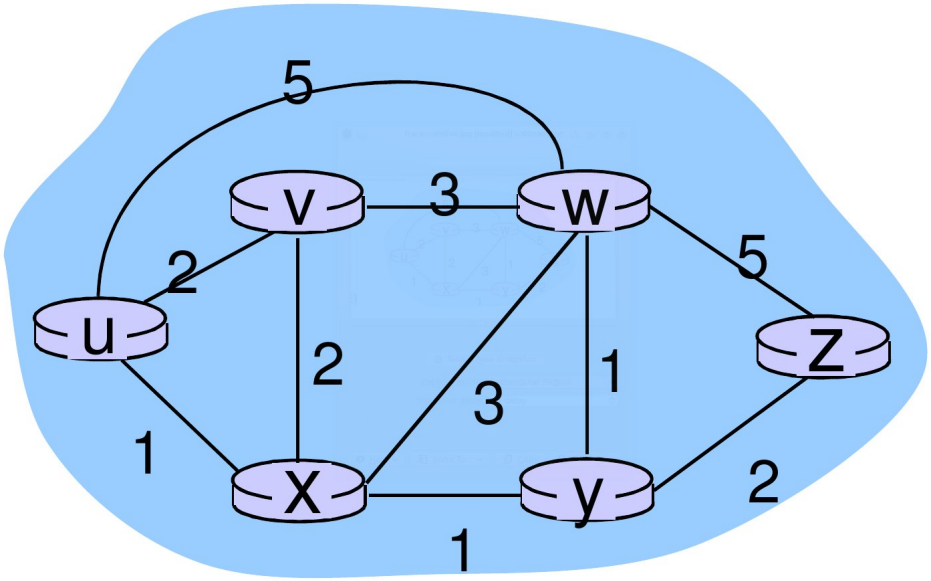


Notes:

- Break the ties arbitrarily

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					

Dijkstra's Algorithm: Example

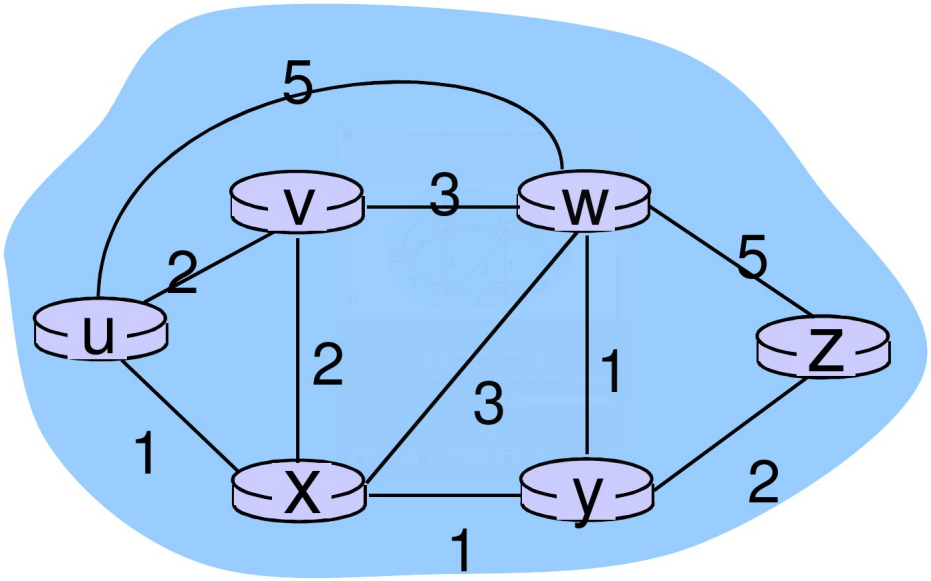


Notes:

- Break the ties arbitrarily

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

Dijkstra's Algorithm: Example

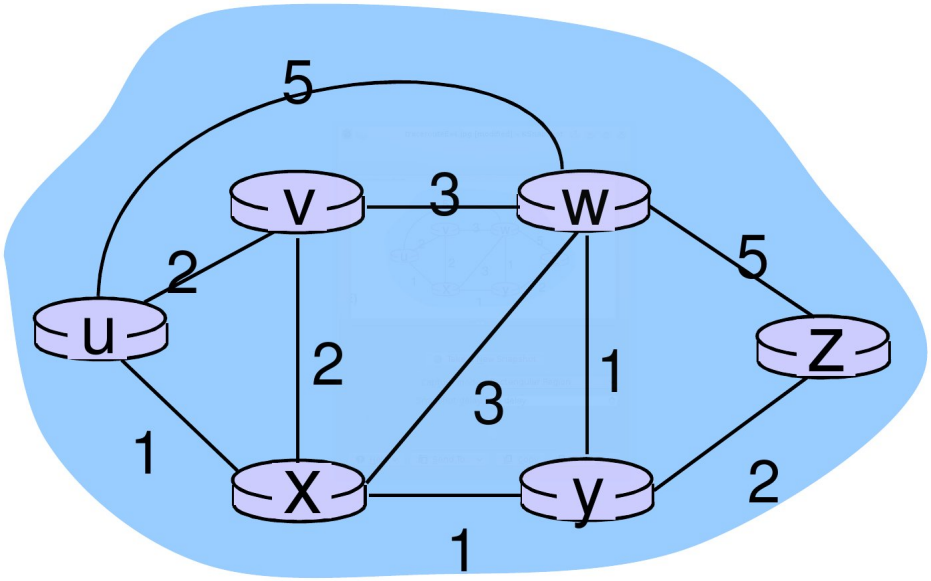


Notes:

- Break the ties arbitrarily

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					

Dijkstra's Algorithm: Example

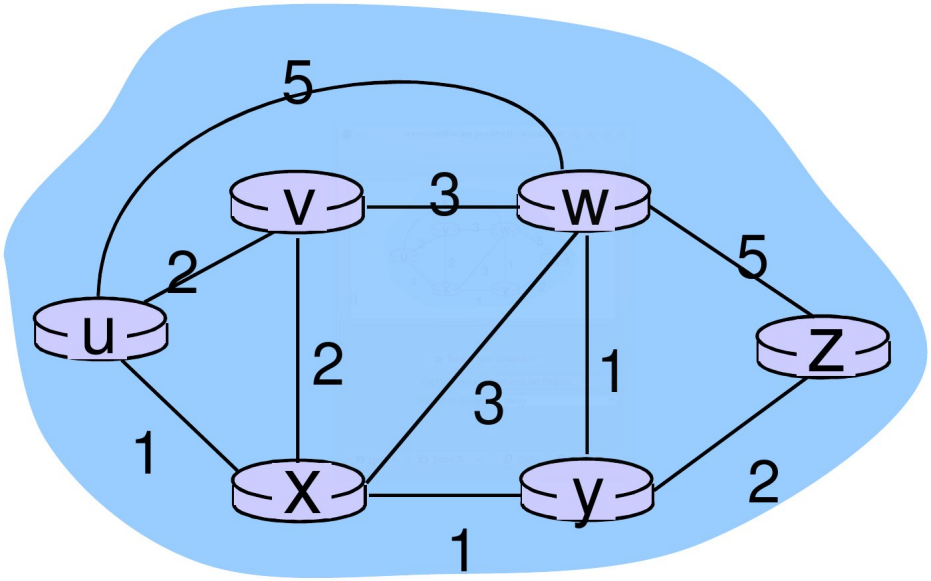


Notes:

- Break the ties arbitrarily

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

Dijkstra's Algorithm: Example

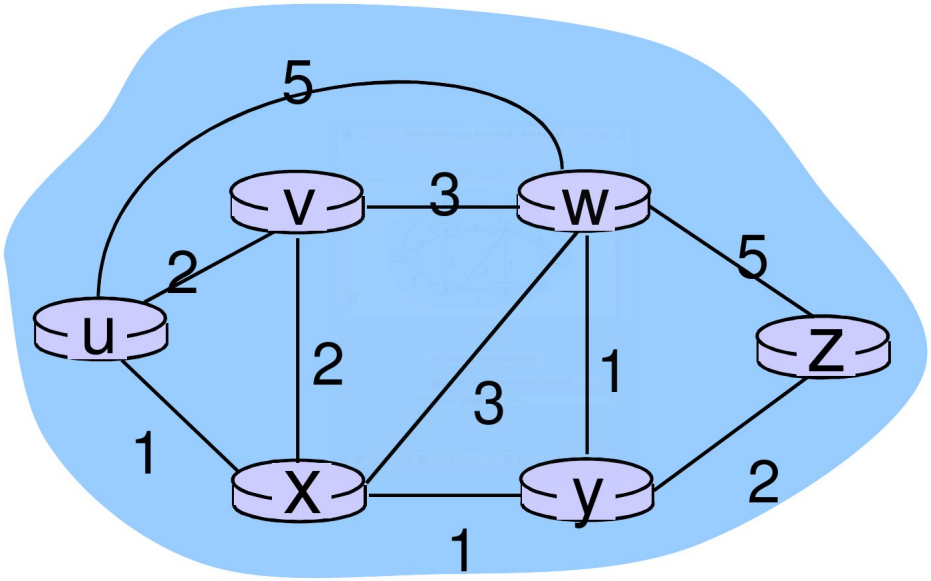


Notes:

- Break the ties arbitrarily

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					

Dijkstra's Algorithm: Example

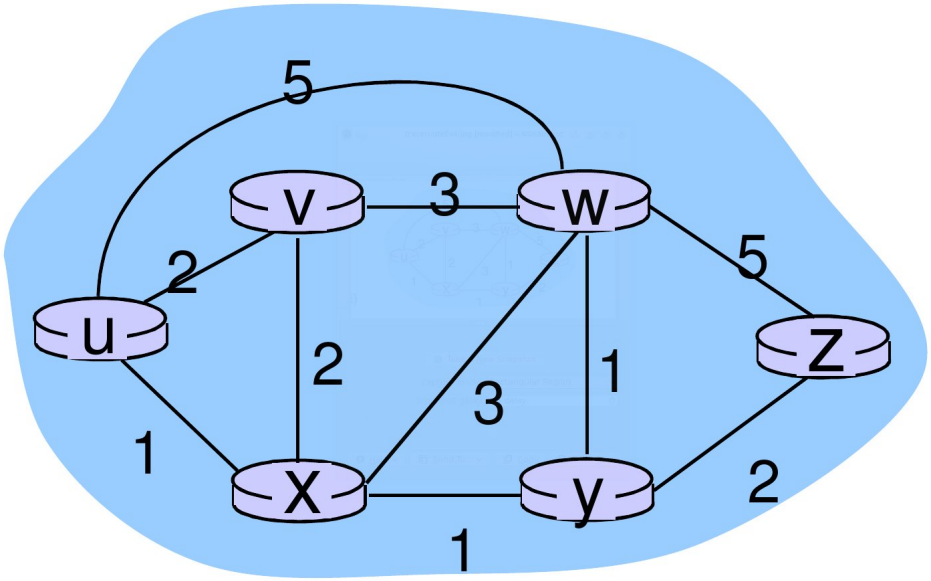


Notes:

- Break the ties arbitrarily

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

Dijkstra's Algorithm: Example

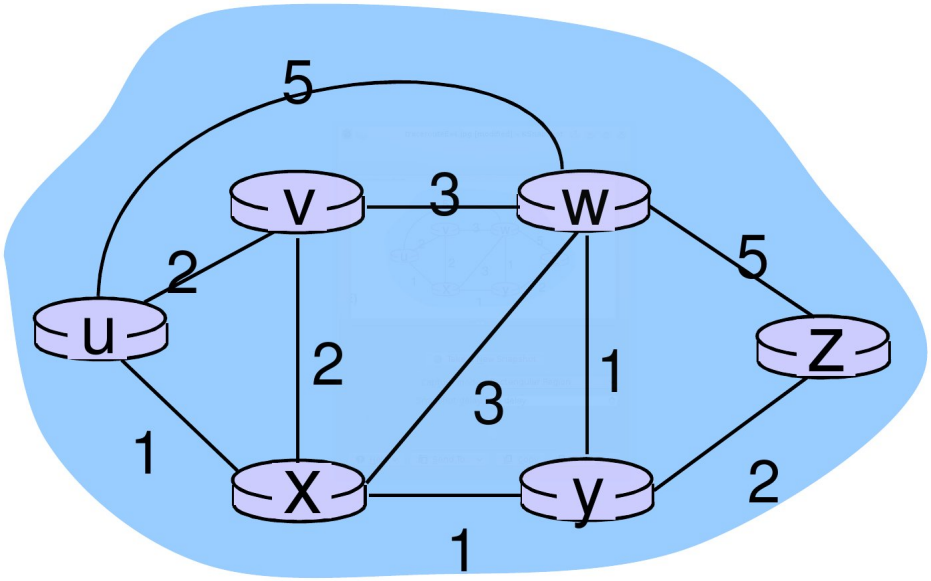


Notes:

- Break the ties arbitrarily

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



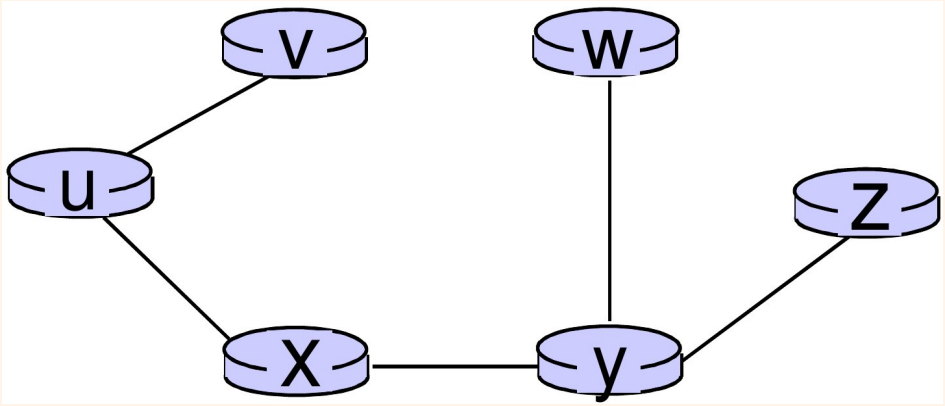
Notes:

- Break the ties arbitrarily
- $\frac{n(n+1)}{2}$ comparisons: $\mathcal{O}(n^2)$ (efficient implem.: $\mathcal{O}(n \log n)$)
- Construct shortest path tree by tracing predecessor nodes

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

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)

Decentralized algorithm

Distance Vector Algorithm

- Distributed: each node exchange information only with its direct neighbors
- Iterative: the process continues on until no more information is exchanged
- Asynchronous: does not require all of the nodes to operate at the same time

Decentralized algorithm

Distance Vector Algorithm

- Distributed: each node exchange information only with its direct neighbors
- Iterative: the process continues on until no more information is exchanged
- Asynchronous: does not require all of the nodes to operate at the same time

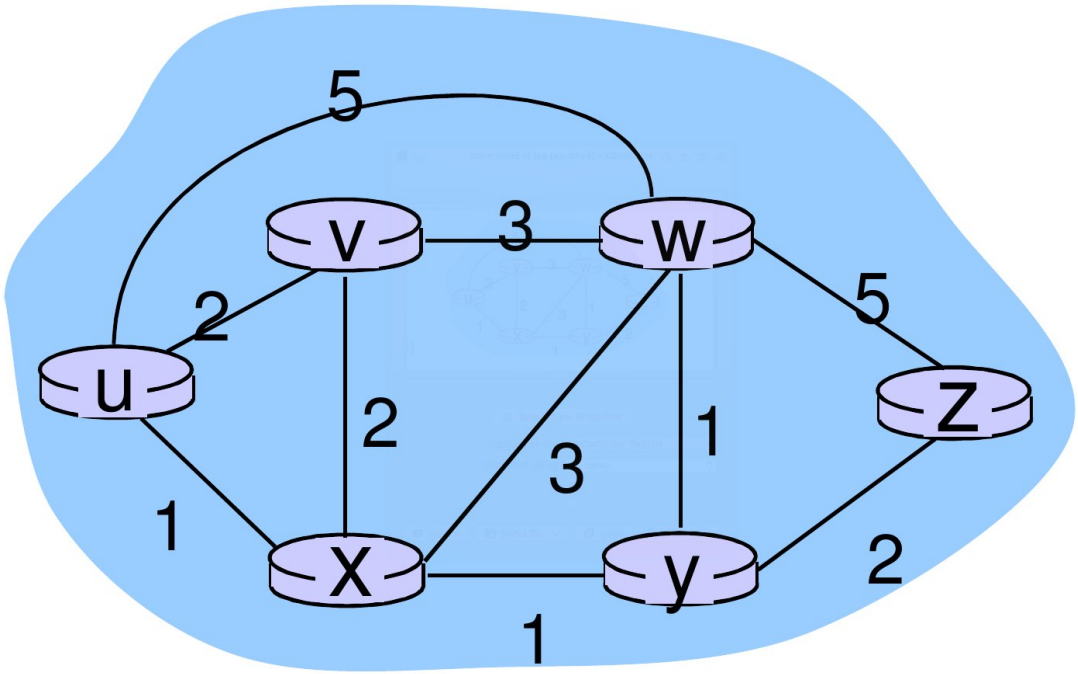
Based on the Bellman-Ford equation

- let $d_x(y)$ be the cost of least-cost path from x to y
- then $d_x(y) = \min_v\{c(x,v) + d_v(y)\}$

Bellman-Ford Example

Example: Least-cost path from u to z

- Knowing that: $d_v(z) = 5, \quad d_x(z) = 3, \quad d_w(z) = 3$

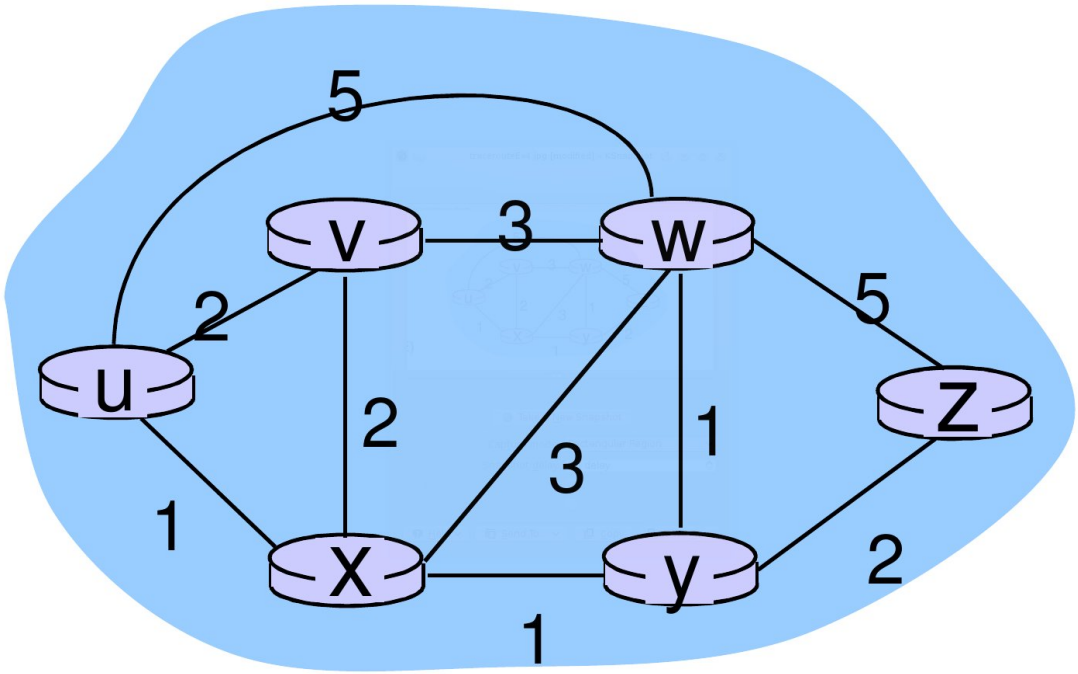


Bellman-Ford Example

Example: Least-cost path from u to z

- Knowing that: $d_v(z) = 5, \quad d_x(z) = 3, \quad d_w(z) = 3$
- Bellman-Ford equation says:

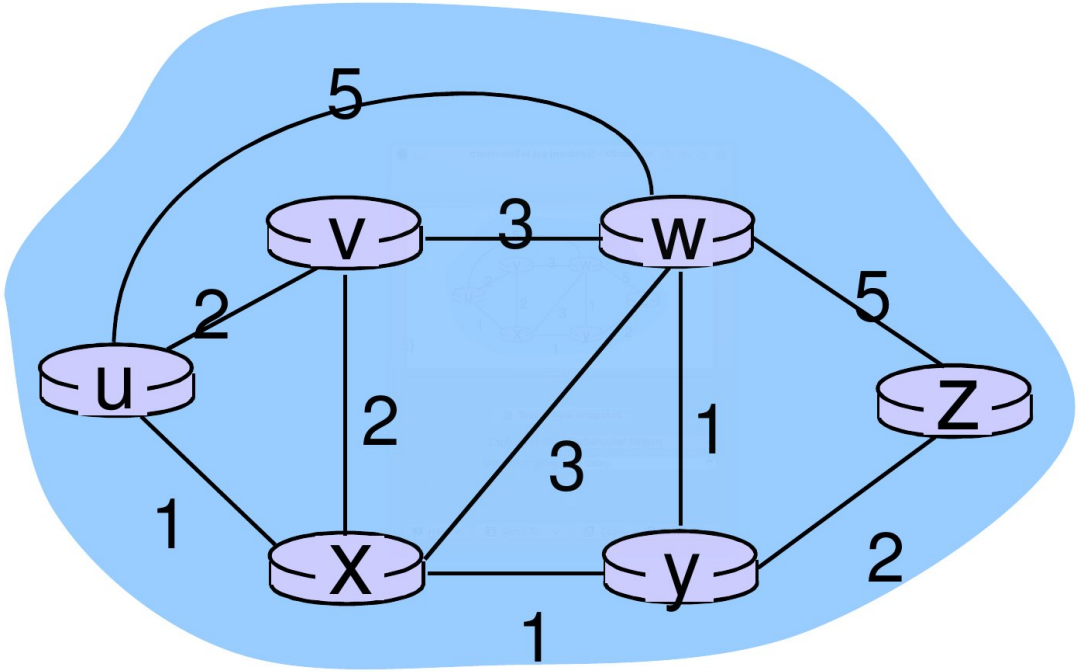
$$d_u(z) = \min\{c(u,v) + d_v(z), \quad c(u,x) + d_x(z), \quad c(u,w) + d_w(z)\}$$
$$= \min\{2 + 5, \quad 1 + 3, \quad 5 + 3\} = 4$$



Bellman-Ford Example

Example: Least-cost path from u to z

- Knowing that: $d_v(z) = 5, \quad d_x(z) = 3, \quad d_w(z) = 3$
- Bellman-Ford equation says:
  $$d_u(z) = \min\{c(u,v) + d_v(z), \quad c(u,x) + d_x(z), \quad c(u,w) + d_w(z)\}$$
  $$= \min\{2 + 5, \quad 1 + 3, \quad 5 + 3\} = 4$$
- x is the next hop on the least-cost path to z



Distance Vector Algorithm

Basic Idea

- Node x maintains
  - distance vector $D_x = [D_x(y) : y \in N]$
  - the cost to each neighbor v: $c(x, v)$
  - the distance vector of each neighbors: $D_v = [D_v(y) : y \in N]$

Distance Vector Algorithm

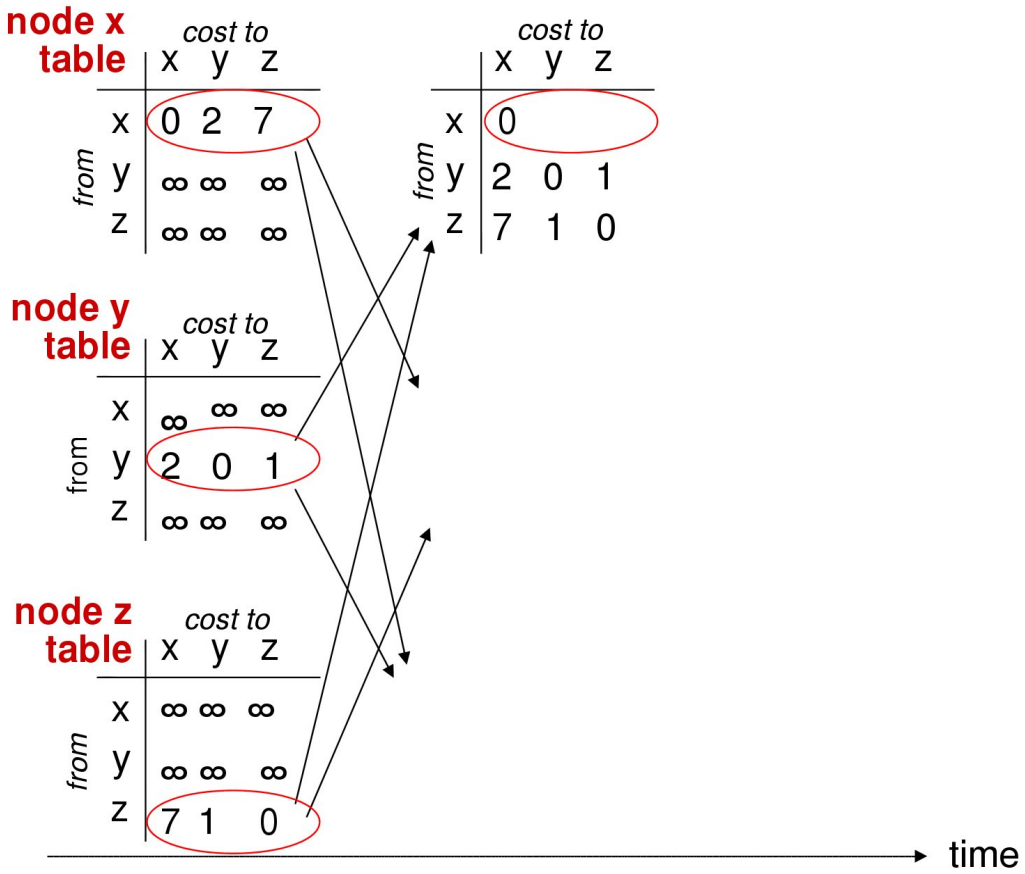
Basic Idea

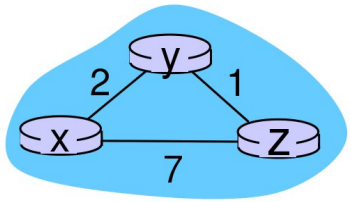
- Node x maintains
  - distance vector $D_x = [D_x(y) : y \in N]$
  - the cost to each neighbor v: $c(x, v)$
  - the distance vector of each neighbors: $D_v = [D_v(y) : y \in N]$

Asynchronous iterations

1. **wait** for (change in local link cost or msg from neighbor)
2. **recompute** estimates using Bellman-Ford equation
3. if DV to any dest has changed, **notify** neighbors
4. go back to 1

Distance Vector Algorithm: Example





Initialization: $D_x(v) = c(x, v)$ for neighbors v, $D_x(y) = \infty$ otherwise

Distance Vector Algorithm: Example

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

node x table

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

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

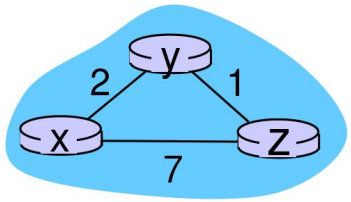
node y table

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

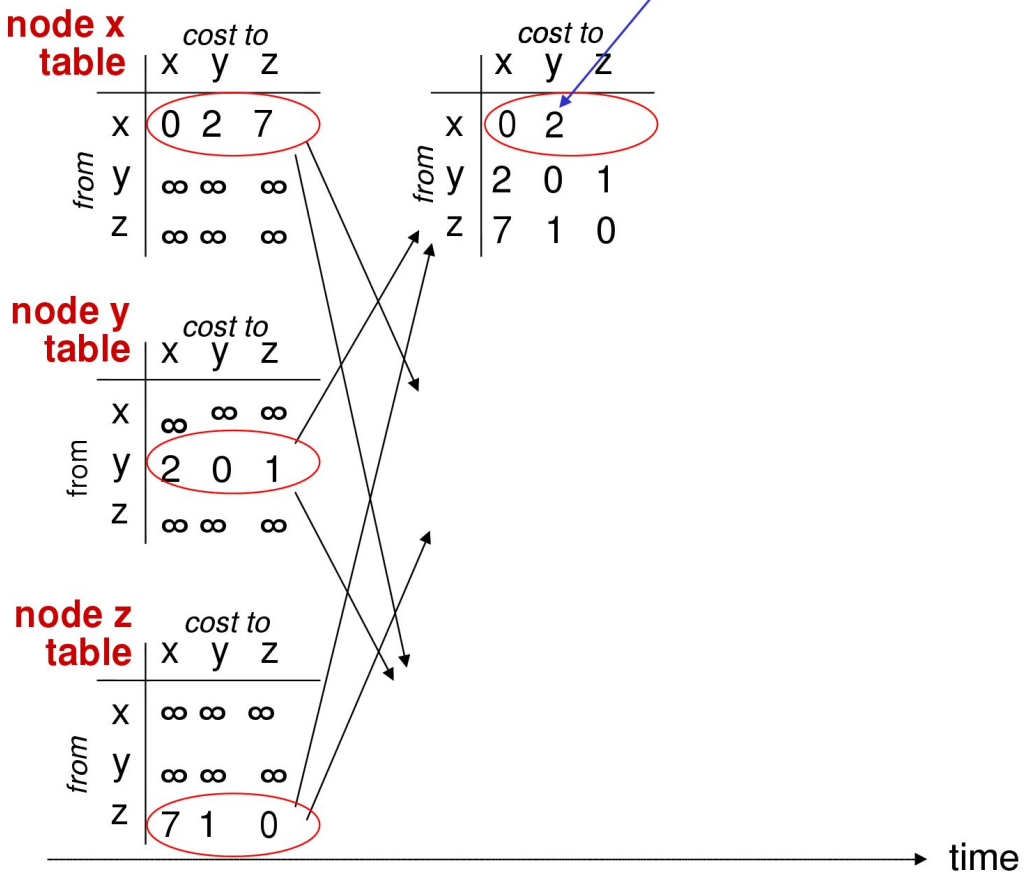
time

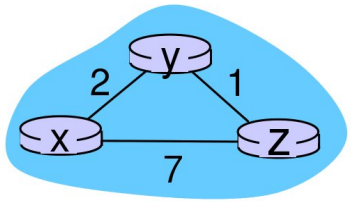


Initialization: $D_x(v) = c(x, v)$ for neighbors v, $D_x(y) = \infty$ otherwise

Distance Vector Algorithm: Example

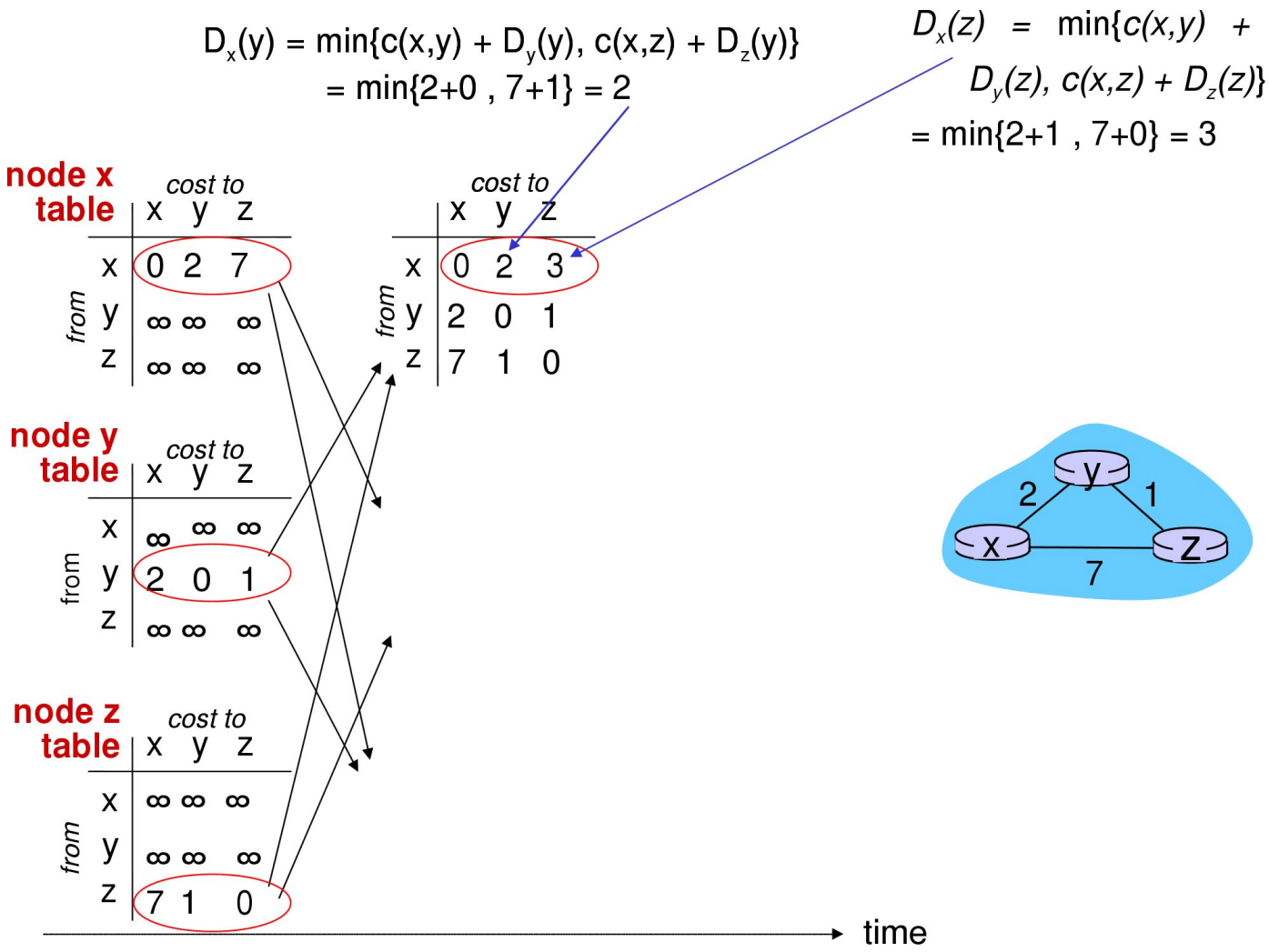
$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\}$
$= \min\{2+0 , 7+1\} = 2$





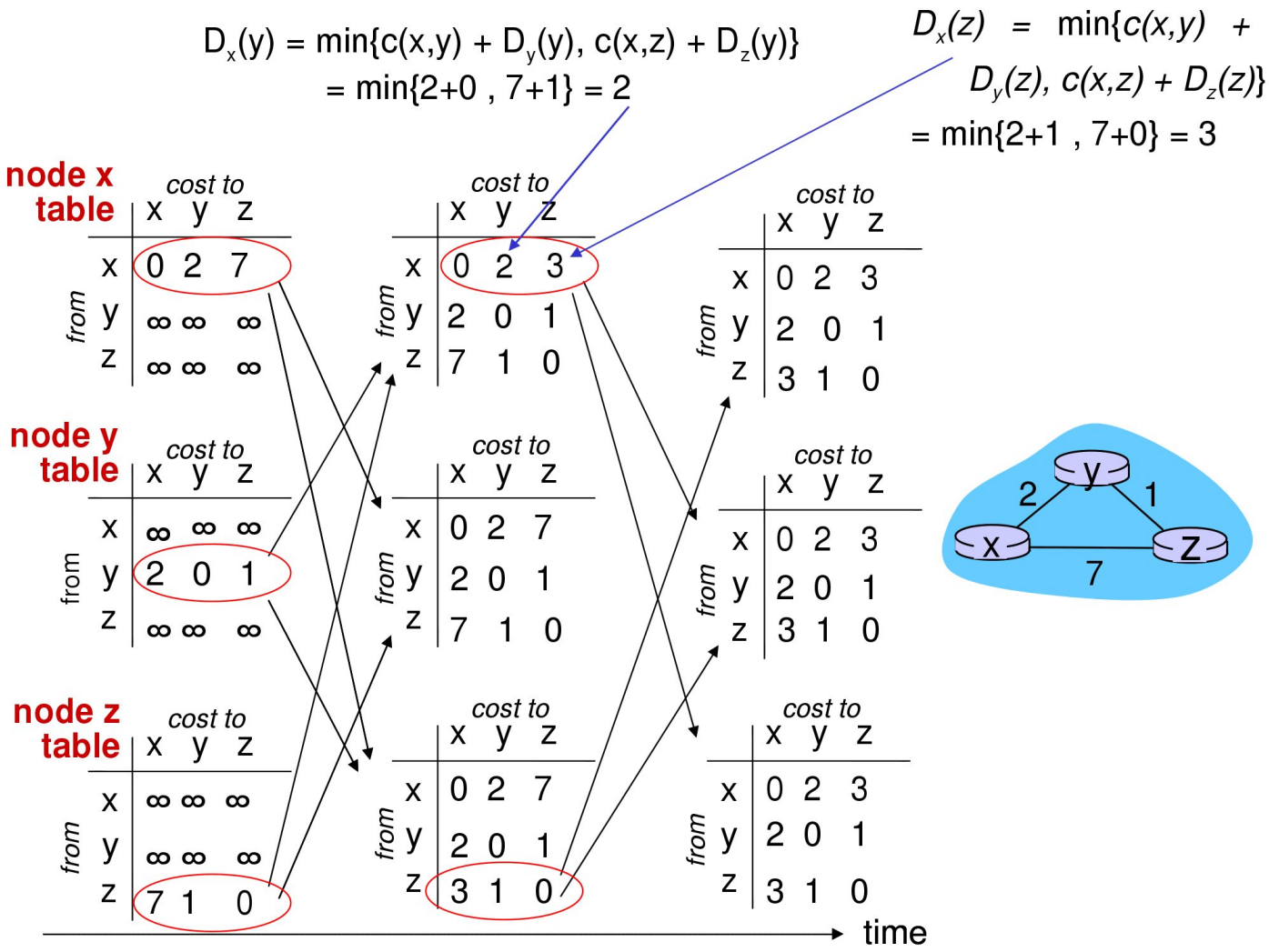
Initialization: $D_x(v) = c(x, v)$ for neighbors v, $D_x(y) = \infty$ otherwise

Distance Vector Algorithm: Example



Initialization: $D_x(v) = c(x, v)$ for neighbors v, $D_x(y) = \infty$ otherwise

Distance Vector Algorithm: Example



Initialization: $D_x(v) = c(x, v)$ for neighbors v, $D_x(y) = \infty$ otherwise

Comparison of Link-State and Distance Vector Algorithms

Message complexity

- LS: with $|N|$ nodes and $|E|$ links, $|N|.|E|$ messages sent
- DV: exchanges between neighbors only

Convergence

- LS: $\mathcal{O}(|N|^2)$ algorithm
- DV: converges slowly and can have routing loops while converging

Comparison of Link-State and Distance Vector Algorithms

Message complexity

- LS: with $|N|$ nodes and $|E|$ links, $|N|.|E|$ messages sent
- DV: exchanges between neighbors only

Convergence

- LS: $\mathcal{O}(|N|^2)$ algorithm
- DV: converges slowly and can have routing loops while converging

Who's the best?

- No winner, both algorithms are used in the Internet:
  - Link-State routing protocols: OSPF, IS-IS
  - Distance Vector routing protocols: RIP, IGRP

Hierarchical Routing

In practice routing is not that simple

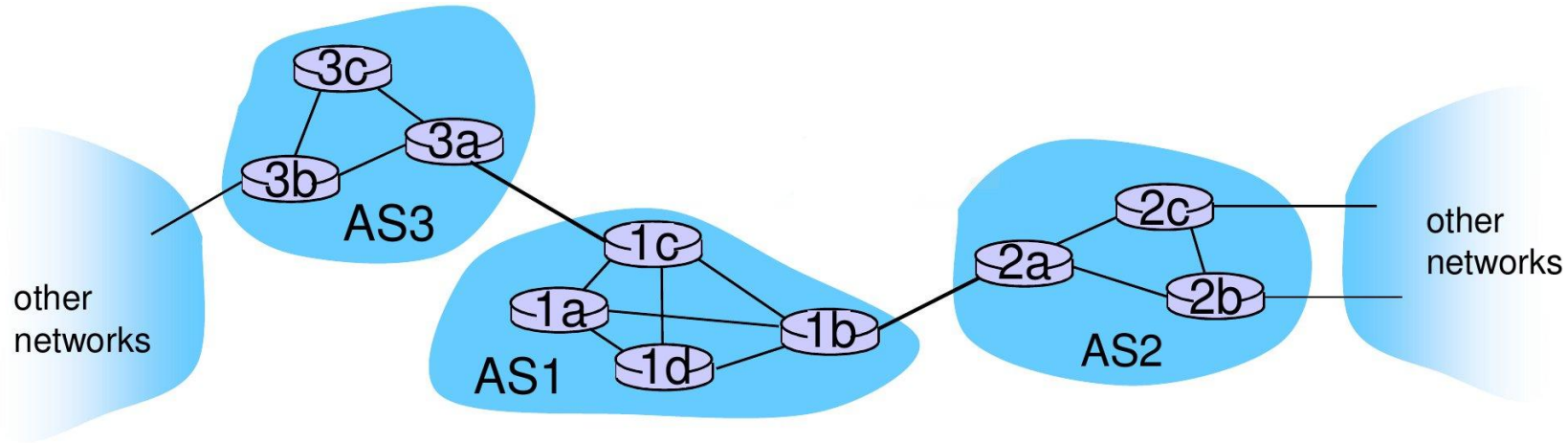
- Scale:
  - cannot store all Internet destinations in routing tables!
  - overhead in computing/communicating routing information
- Administrative autonomy:
  - Internet is a network of networks
  - each network admin want to control routing in its own network
  - a company may want to hide its network's internal organization

Both problems can be solved by organizing routers into **autonomous systems (AS)**

Hierarchical Routing

Autonomous System (AS)

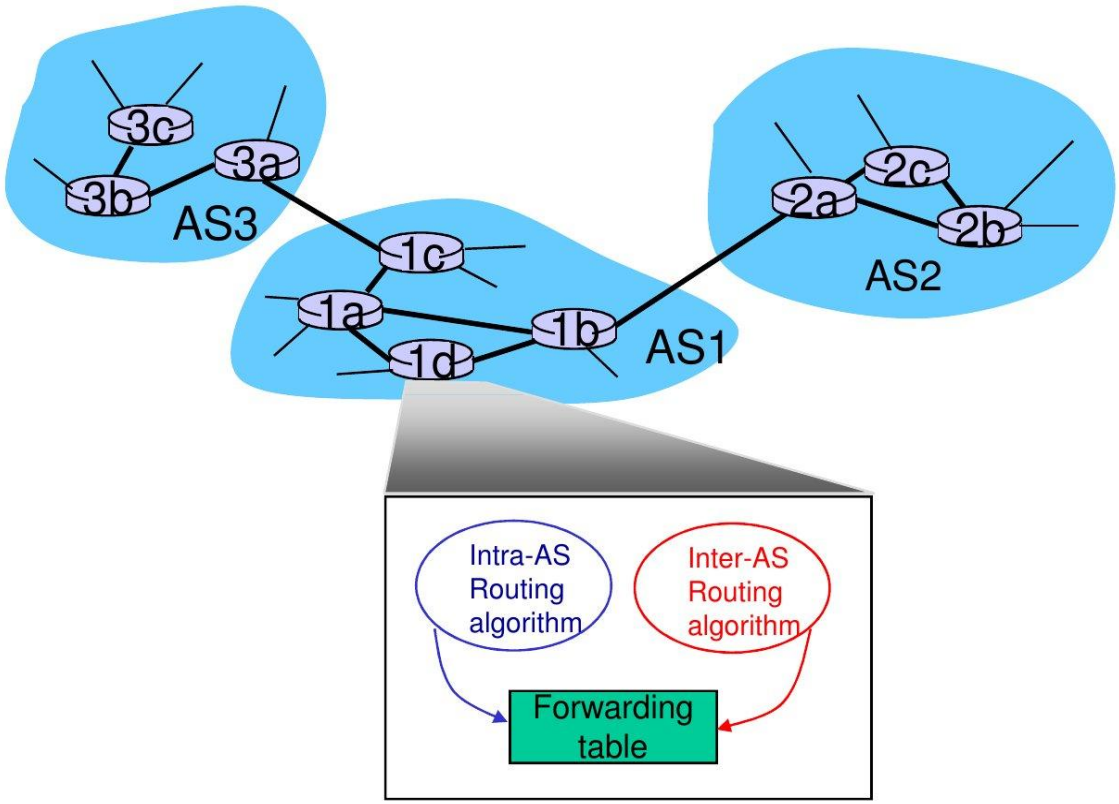
- Group of router under the same administrative control
- Routers in same AS run same routing protocol
  - "intra-AS" routing protocol
  - routers in different AS can run different intra-AS routing protocol
- Gateway router:
  - at "edge" of its own AS
  - has link to router in another AS



Interconnected ASes

Forwarding table configured by both intra- and inter-AS routing algorithm

- intra-AS sets entries for internal destinations
- inter-AS and intra-AS sets entries for external destinations



Routing in the Internet

Intra-AS routing

- also known as interior gateway protocols (IGP)
- most common intra-AS routing protocols:
  - using Distance Vector algorithms:
    - RIP: Routing Information Protocol
    - IGRP: Interior Gateway Routing Protocol (Cisco proprietary)
  - using Link-State algorithm:
    - OSPF: Open Shortest Path First
    - IS-IS: Intermediate System to Intermediate System

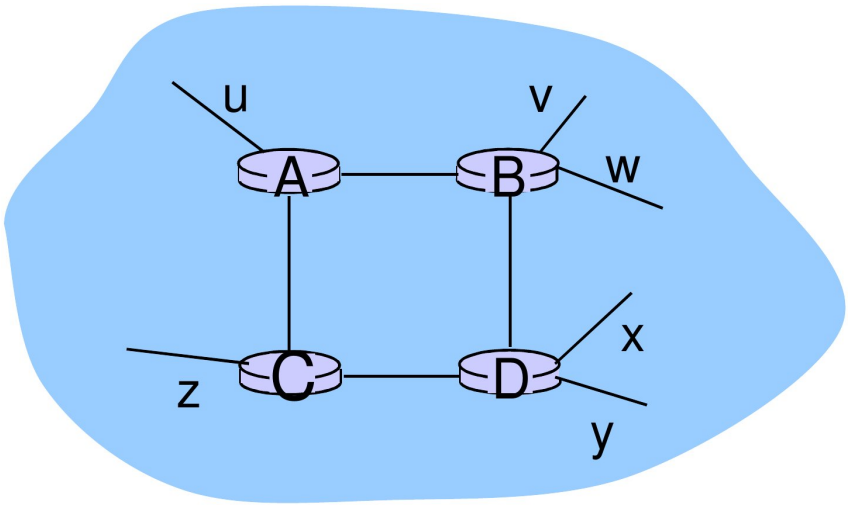
Inter-AS routing

- BGP (Border Gateway Protocol): the de facto inter-domain routing protocol
- $\rightarrow$ "glue that holds the Internet together"

RIP (Routing Information Protocol)

RIP: One of the earliest intra-AS routing protocol

- Included in BSD-UNIX distribution in 1982
- Distance Vector algorithm
  - distance metric: number of hops, each link has cost 1 (max = 15 hops)
  - 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: link failure, recovery

If no advertisement heard after 180 sec.

- neighbor/link declared dead
- routes via 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

OSPF (Open Shortest Path First)

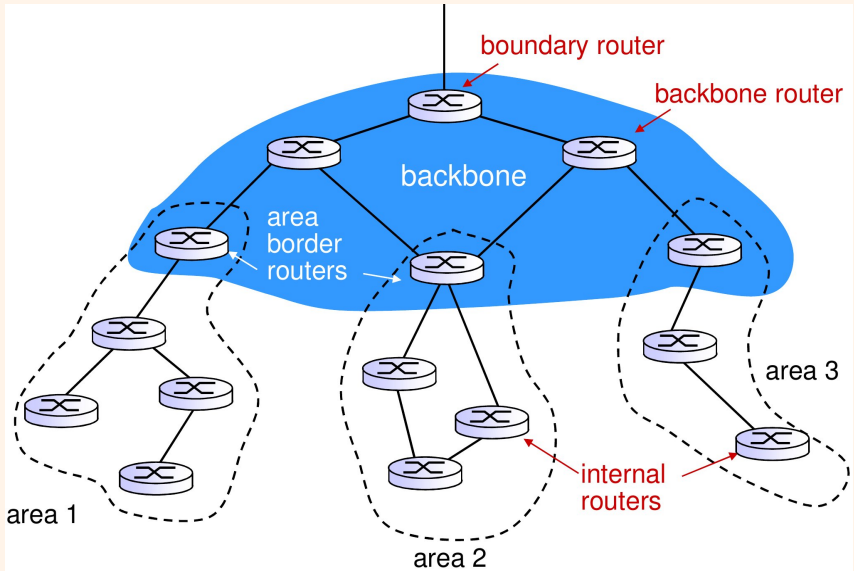
OSPF was conceived as the successor to RIP

- 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

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)
- Link costs are configured by the network administrator
  e.g. 1 for all links (like RIP), or inversely proportional to link capacity
- Hierarchical OSPF in large domains

  - two-level hierarchy: local area, backbone
  - link-state advertisements only in area



Internet Inter-AS Routing: BGP

BGP (Border Gateway Protocol)

- The most widely used inter-AS routing protocol
- BGP provides each AS a means to:
  - eBGP: obtain subnet reachability information from neighboring ASs
  - iBGP: propagate reachability information to all AS-internal routers
  - determine "good" routes to other networks based on reachability information and policy
- Allows subnet to advertise its existence to rest of Internet: *"I am here!"*

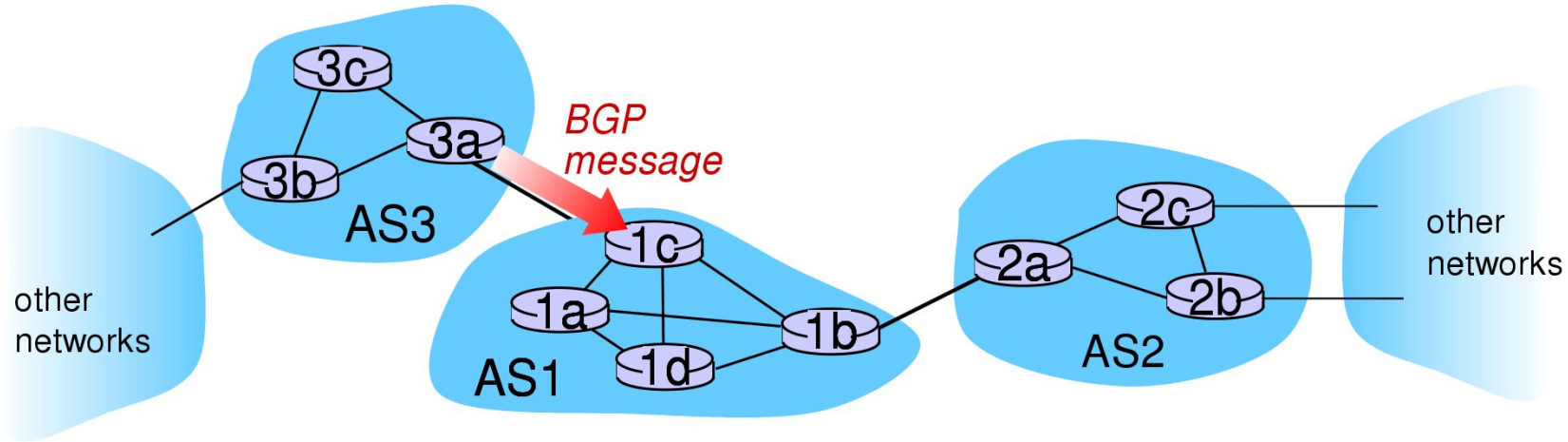
BGP basics

BGP session: two BGP routers ("peers") exchange BGP messages:

- advertising paths to different destination network prefixes (subnets)

when AS3 advertises a prefix to AS1:

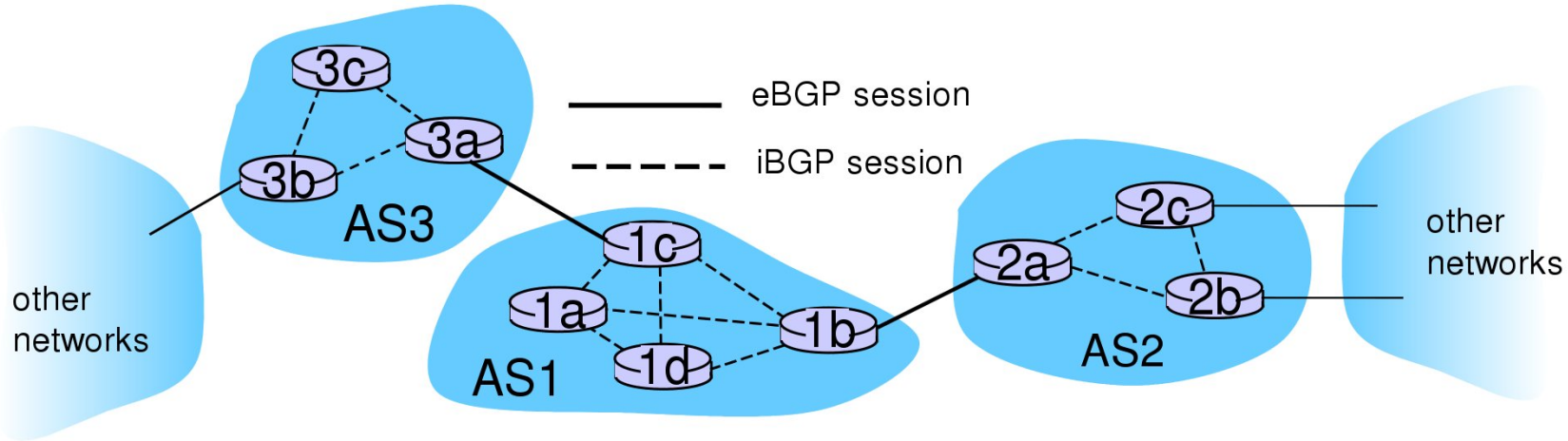
- AS3 promises it will forward datagrams towards that prefix
- AS3 can aggregate prefixes in its advertisement



BGP basics

Distributing path information

- Using eBGP session between 3a and 1c, AS3 sends prefix reachability info to AS1
  - 1c can then use iBGP do distribute new prefix info to all routers in AS1
  - 1b can then re-advertise new reachability info to AS2 over 1b-to-2a eBGP session
- When router learns a new prefix, it creates entry for prefix in its forwarding table



Path Attributes and BGP Routes

Path information and policy

- Advertised prefix includes BGP attributes
  - prefix + attributes = "route"
- Two important attributes:
  - AS-PATH: contains ASs through which prefix advertisement has passed: e.g., AS 67, AS 17
  - NEXT-HOP: indicates specific internal-AS router to next-hop AS. (may be multiple links from current AS to next-hop-AS)
- Gateway router receiving route advertisement uses import policy to accept/decline
  - e.g., never route through AS x
  - policy-based routing

BGP Route Selection

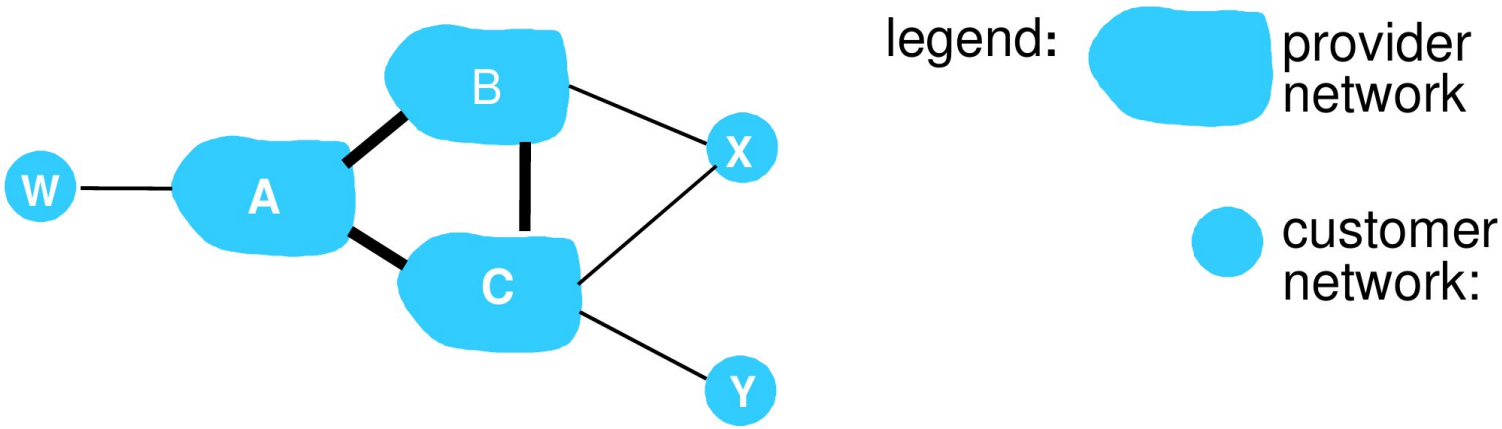
BGP best path selection algorithm

- Router may learn about more than 1 route to destination AS, selects route based on:
  1. local preference value attribute: policy decision
  2. shortest AS-PATH
  3. closest NEXT-HOP router (aka hot potato routing)
  4. additional criteria

BGP routing policy

BGP allows admin to control how the traffic is routed

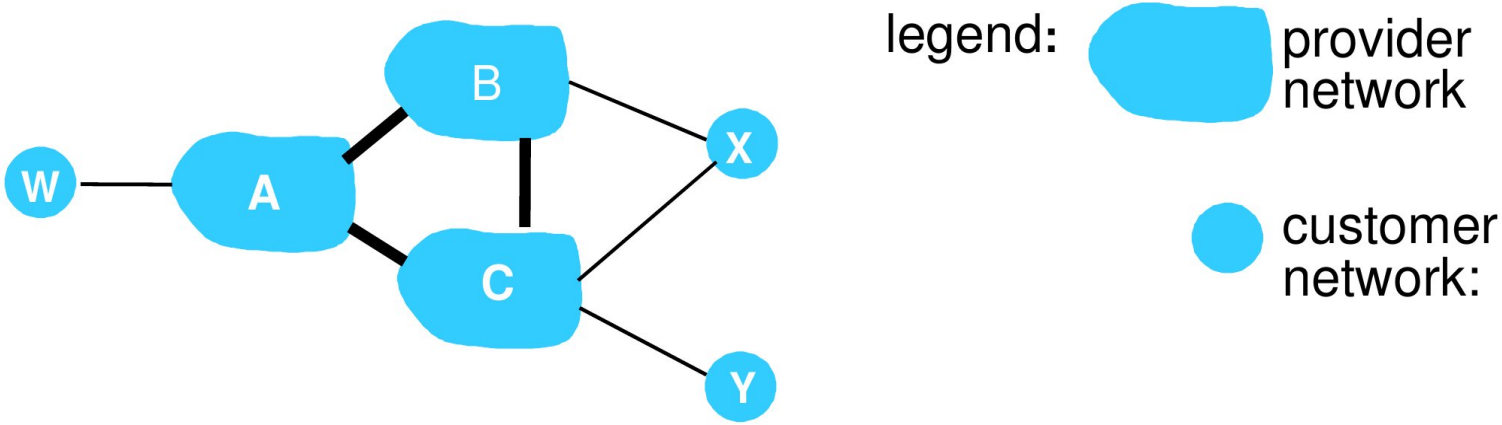
- A,B,C are provider networks
- X,W,Y are customer (of provider networks)
- X is dual-homed: attached to two networks
  - X does not want to route from B via X to C
  - .. so X will not advertise to B a route to C



BGP routing policy

Example for provider networks:

- A advertises path AW to B
- B advertises path BAW to X
- Should B advertise path BAW to C?
  - No way! B gets no "revenue" for routing CBAW since neither W nor C are B's customers
  - B wants to force C to route to w via A
  - B wants to route only to/from its customers!



Why different Intra-, Inter-AS routing?

Policy:

- inter-AS: admin wants control over how its traffic routed, who routes through its net.
- intra-AS: single admin, so no policy decisions needed

Scale:

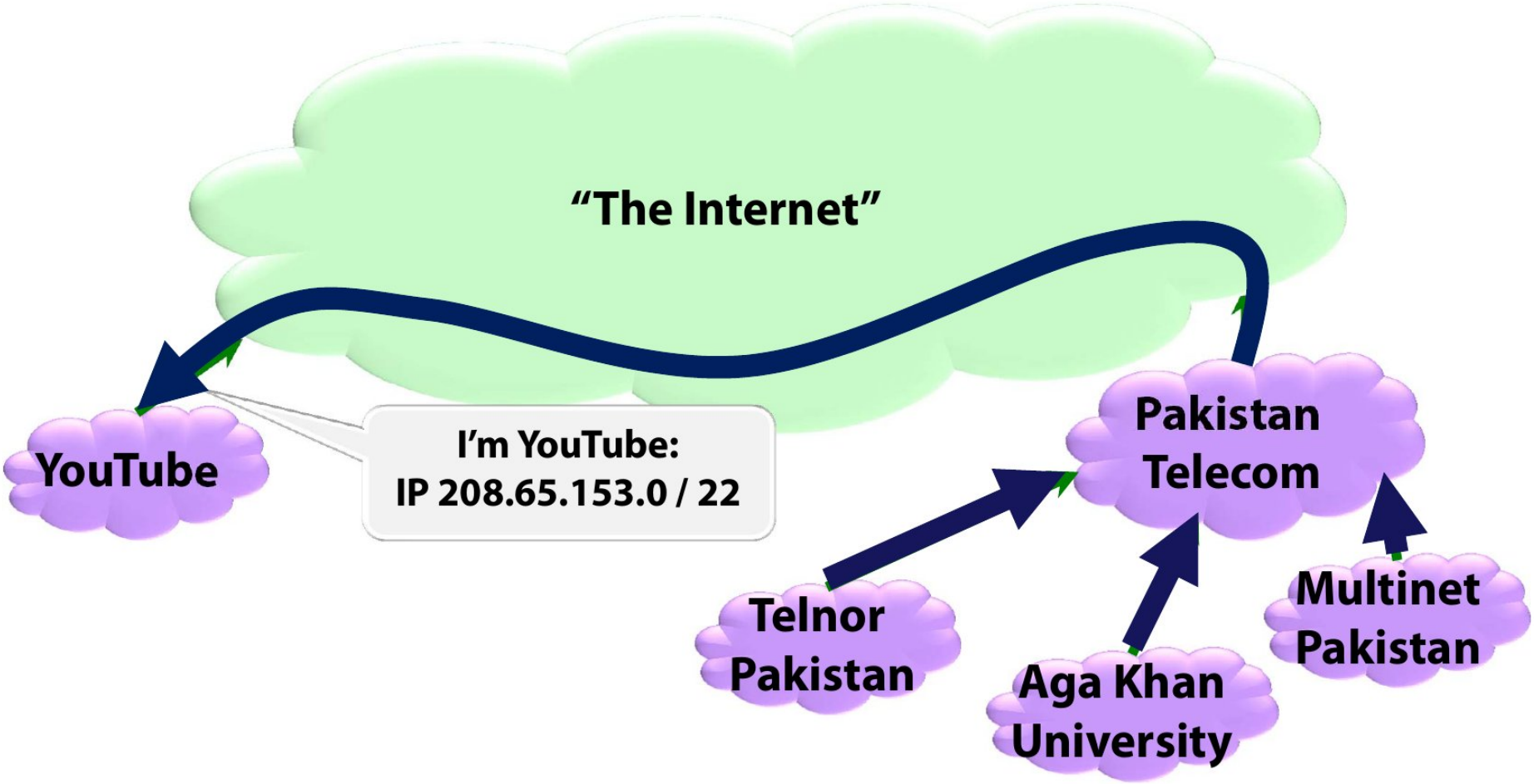
- hierarchical routing saves table size, reduced update traffic, fast convergence

Performance

- intra-AS: can focus on performance
- inter-AS: policy may dominate over performance

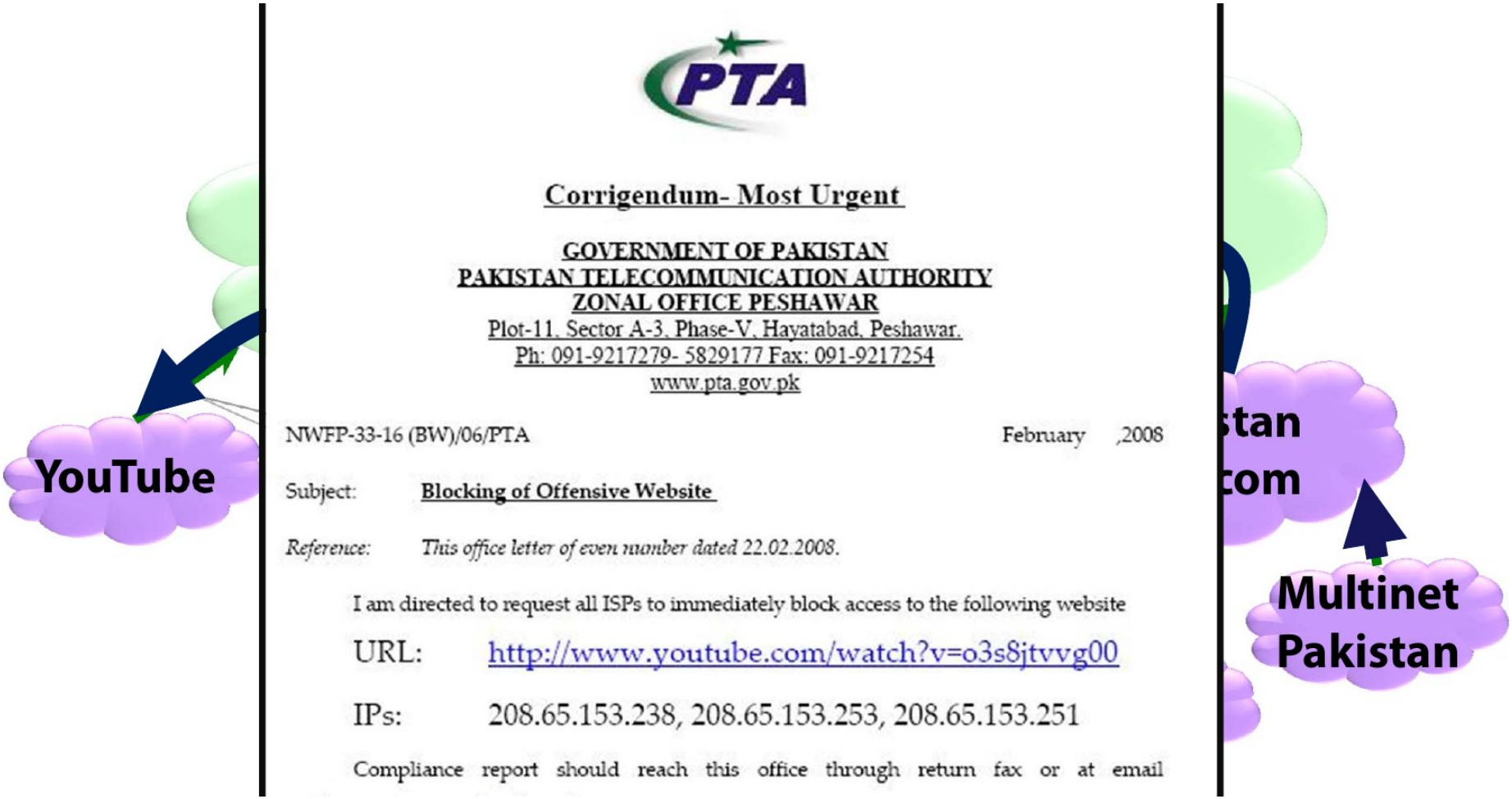
Youtube Hijacking

Usual route to youtube (AS36561)



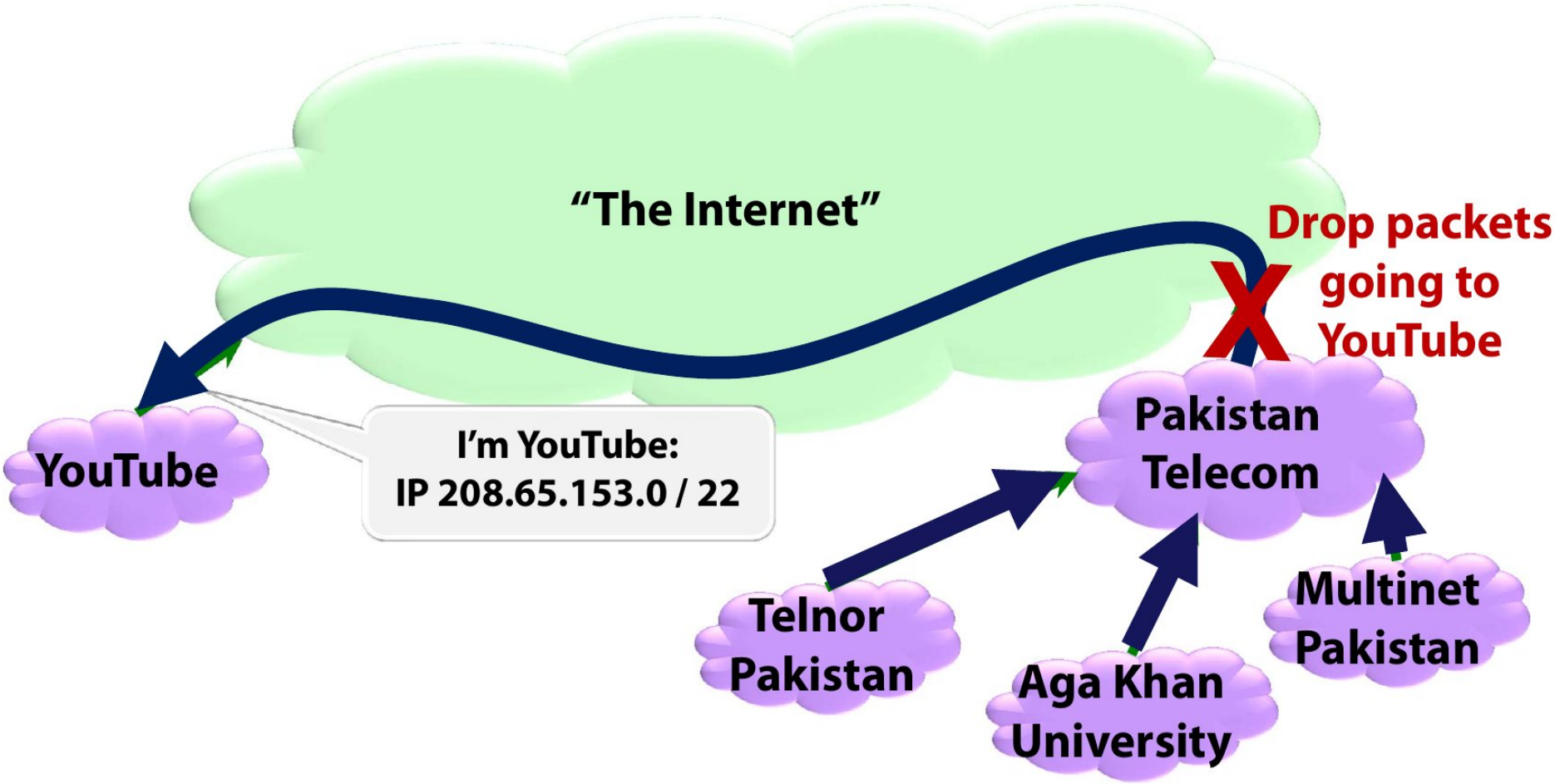
Youtube Hijacking

Pakistan government orders to block access to a youtube video



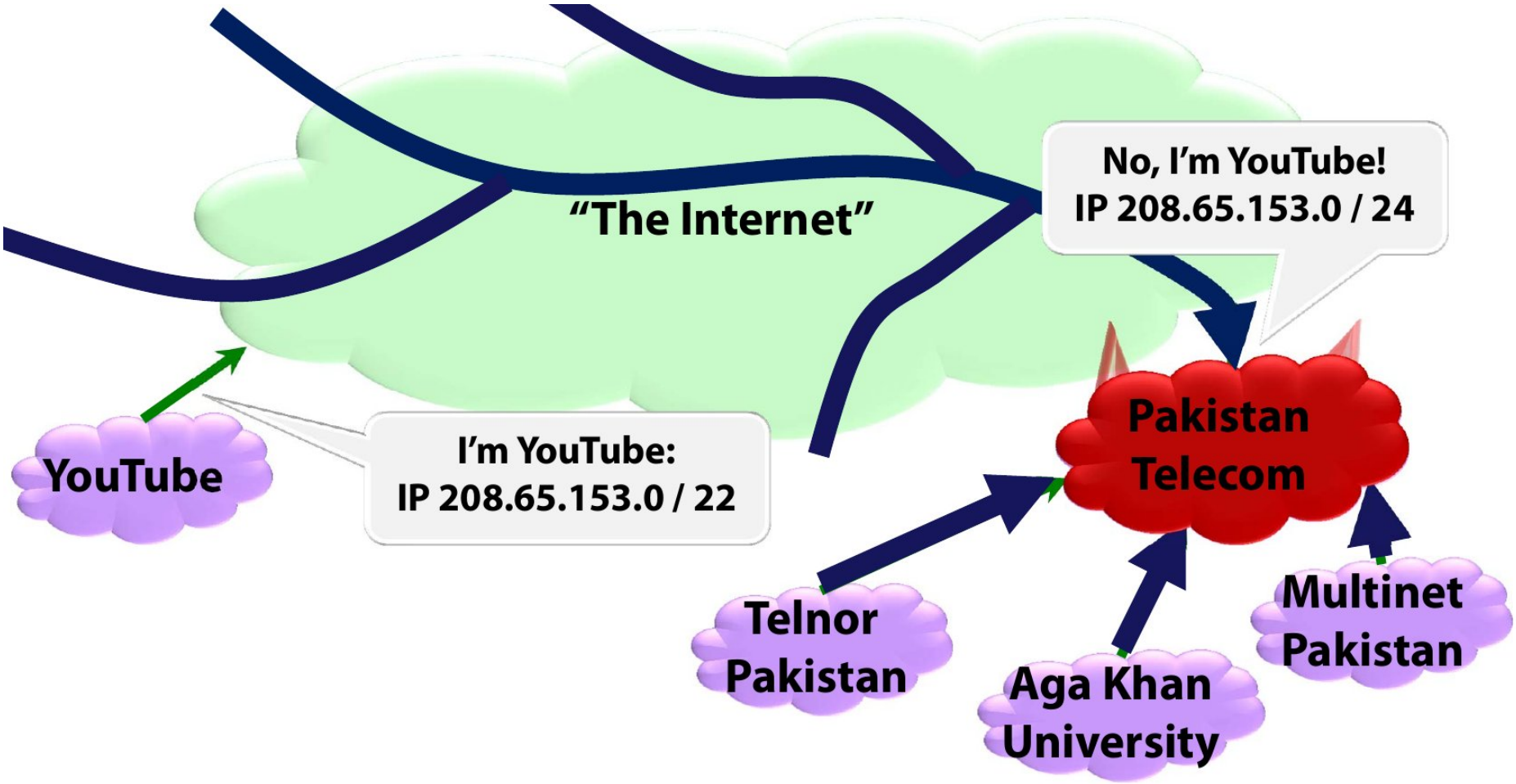
Youtube Hijacking

This is what should have happened



Youtube Hijacking

But this is what Pakistan ended up doing



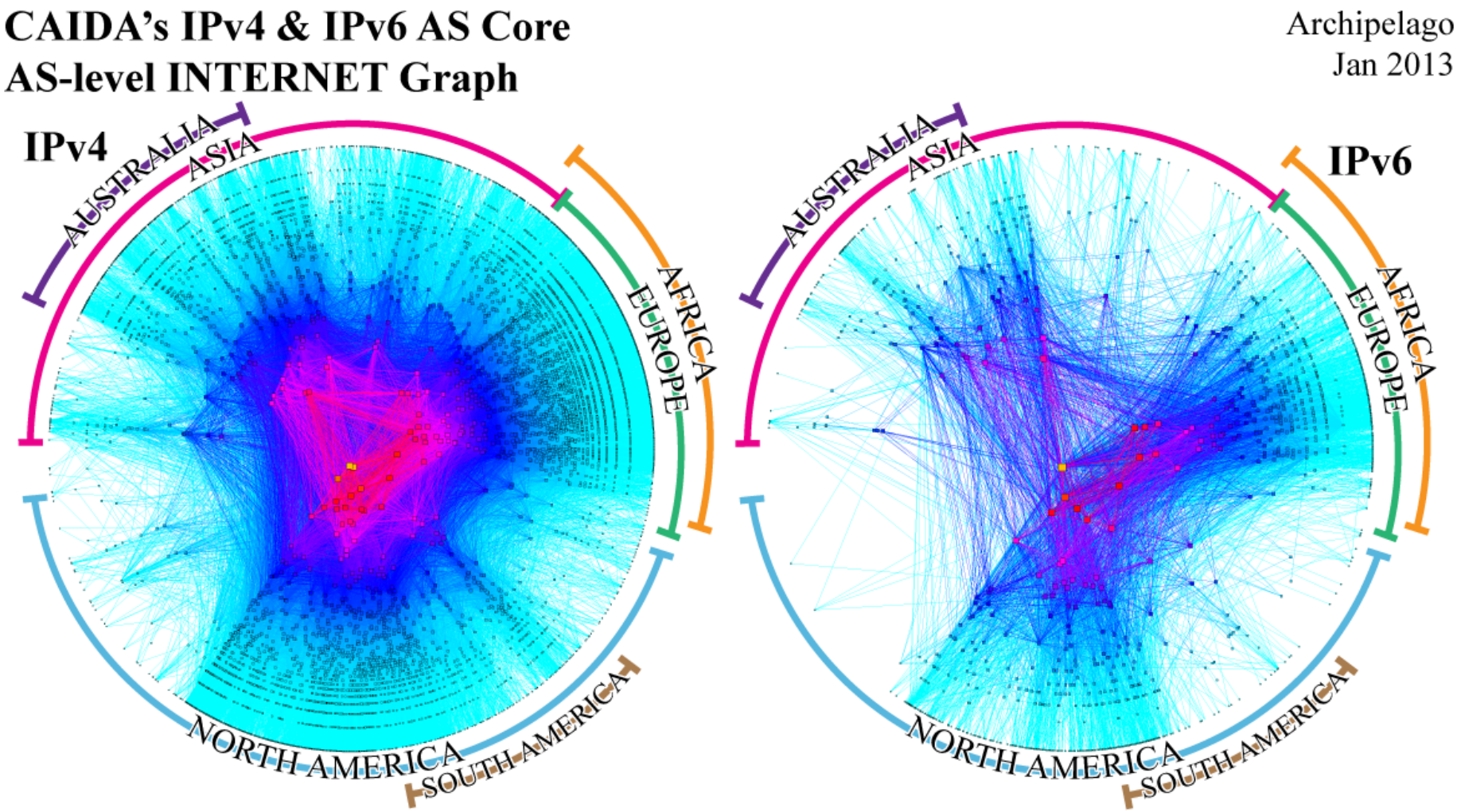
Youtube Hijacking

Timeline: Sunday, 24 February 2008

- Before, during and after the event: AS36561 (YouTube) announces 208.65.152.0/22 and other prefixes
- **18:47** AS17557 (Pakistan Telecom) announces 208.65.153.0/24. Routers around the world redirect YouTube traffic to Pakistan.
- **20:07** (YouTube) announces 208.65.153.0/24. BGP policy rules, such as preferring the shortest AS path, determine which route is chosen. (Pakistan Telecom) continues to attract some of YouTube's traffic.
- **20:18** (YouTube) announces 208.65.153.128/25 and 208.65.153.0/25. Every router that receives these announcements will send the traffic to YouTube.
- **21:01** AS3491 (PCCW Global) withdraws all prefixes originated by AS17557 (Pakistan Telecom), thus completely stopping the hijack of 208.65.153.0/24.

1 hour 30min. of downtime reported by users in Germany, China, US, Russia, the UK, and Australia

CAIDA: AS-level Internet graph



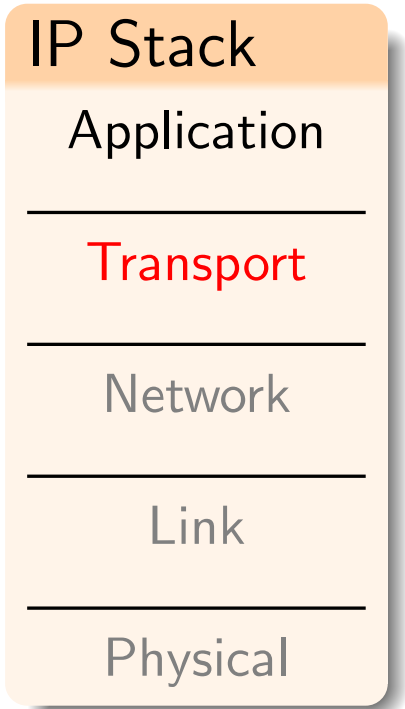
The Network Layer (2): Summary

Today's lecture covered routing algorithms

- Link State algorithm: Dijkstra
- Distance Vector Algorithm
- BGP

In the next lecture

We'll see the transport layer: TCP, UDP



Today's important points

- Longest prefix match
- Dijkstra, Distance Vector
- (BGP)