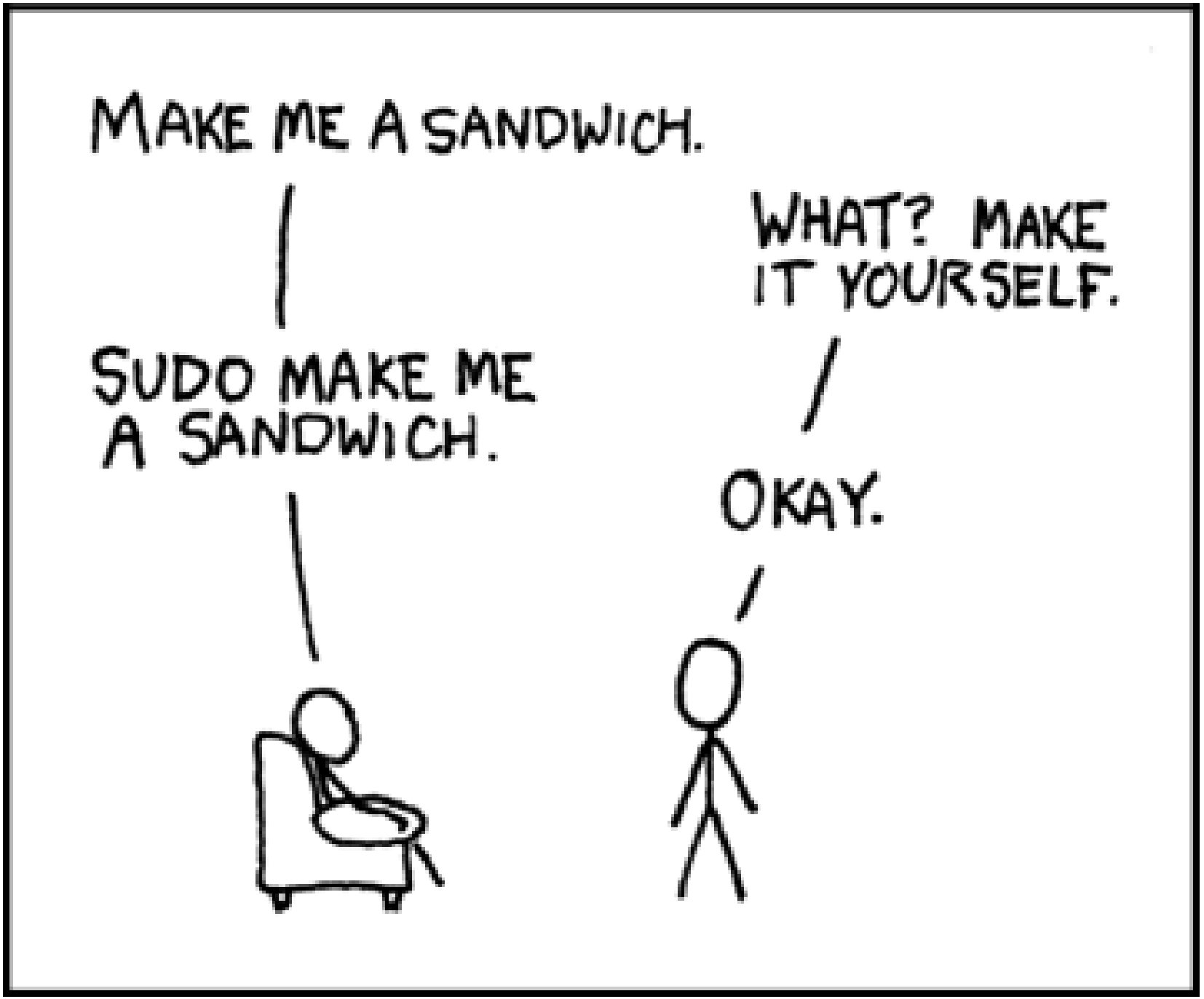
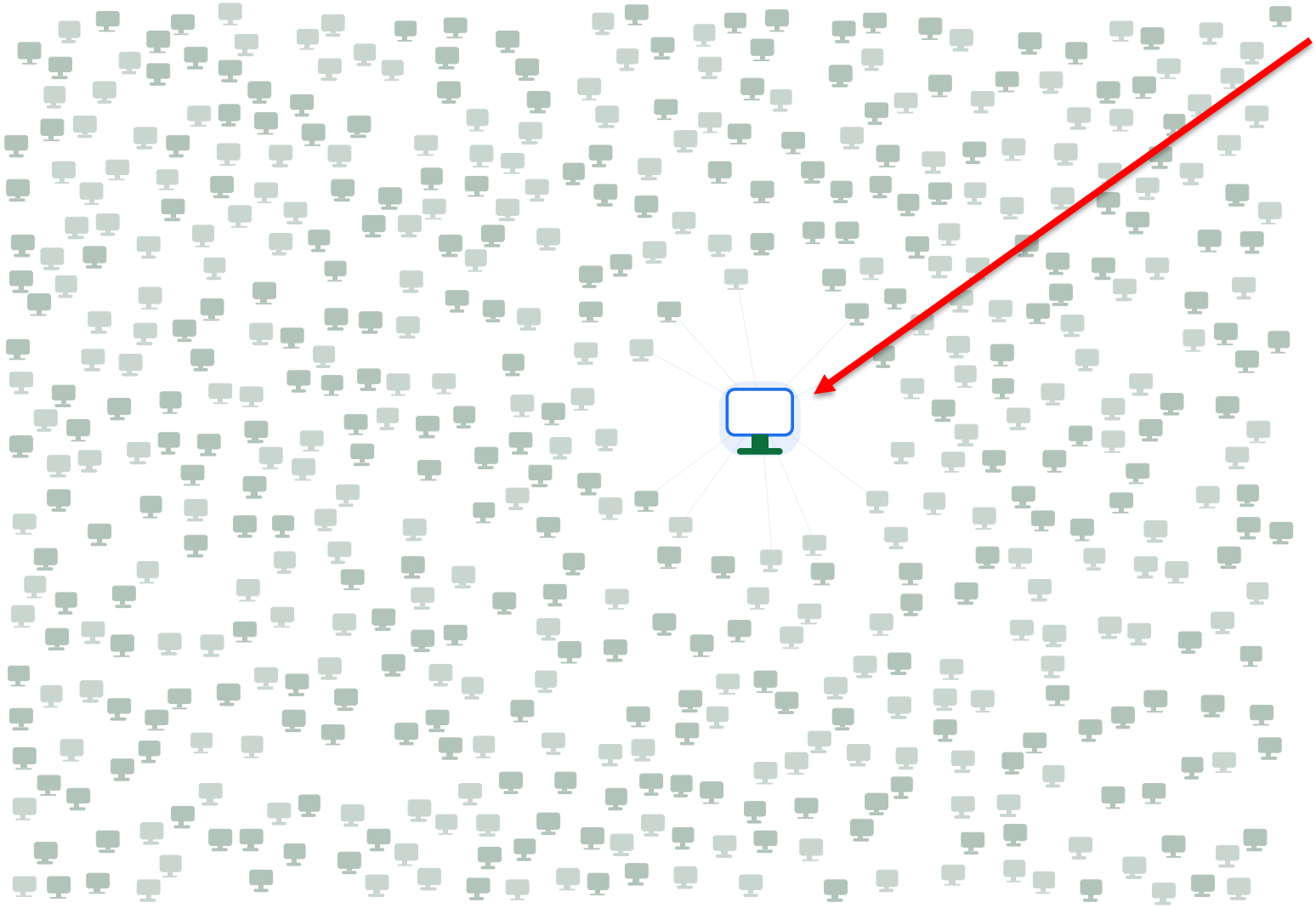


https://xkcd.com/149/

CS 60:
Computer Networks

Network layer: Control plane

Your packet must find *one* host among around 20 billion hosts on the Internet



The one you want

Among around 20 billion hosts

And you get to it in speed of light travel time plus a few tens of ms!

Today we see how it works

Image: Claude Opus 5.5

20 billion hosts down to one



20 billion hosts on the Internet
You don't route to hosts, you route to networks
CIDR collapses billions of addresses into about 1 million IP address ranges

IP Range: 8.8.8.0/24

~1 million IP address ranges controlled by ~120,000 organizations called Autonomous Systems (AS)

8.8.8.8

To find the host you want:

- Navigate your own AS to a gateway to the Internet
- Pass between other ASes on the Internet toward destination AS
- Navigate destination AS network to the specific host

No router knows the whole internet
Each one only needs the next hop toward a destination IP
Two big questions:

- **How to hop *within* an AS (OSPF)**
- **How to hop *between* ASes (BGP)**

Review from last class

We carefully reviewed the bytes produced by each layer of the network stack

Layer 3: routers have two primary functions

1. Forwarding – move packets from input port to output port (last class)
2. Routing – select the best path from source IP to destination IP (today)

Routers maintain a forwarding table with destination IP addresses/subnet masks and the output port that can reach those networks

There are two routing paradigms

1. Traditional – each router makes its own decision about selecting routes
2. Software Defined Network – a control plane monitors the network and updates the routing table on each router

We didn't talk about IPv6 vs IPv4 last class, but I will fix that now

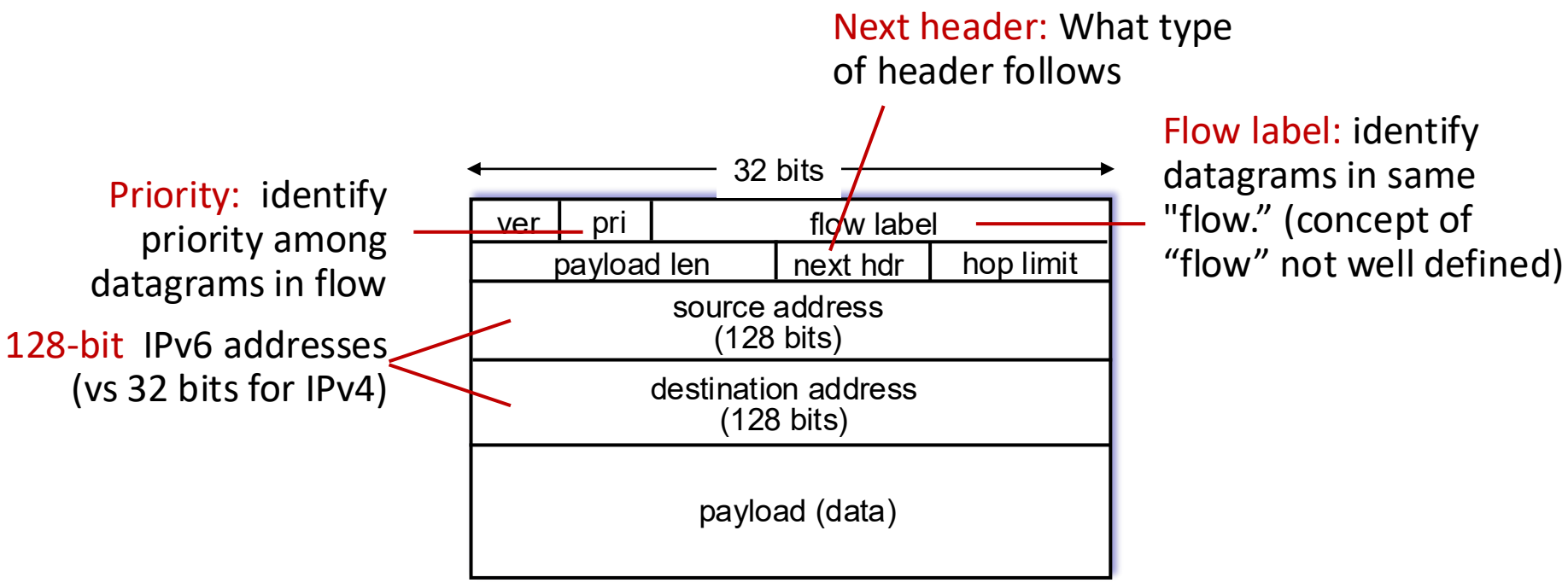
IPv6: motivation

- Initial motivation: 32-bit IPv4 address space has been completely allocated

- Additional motivation:
  - Speed: processing/forwarding: 40-byte fixed length header
  - Enable different network-layer treatment of “flows”
  - Learn lessons from IPv4

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

IPv6 Layer 3 datagram format



What's missing (compared with IPv4):

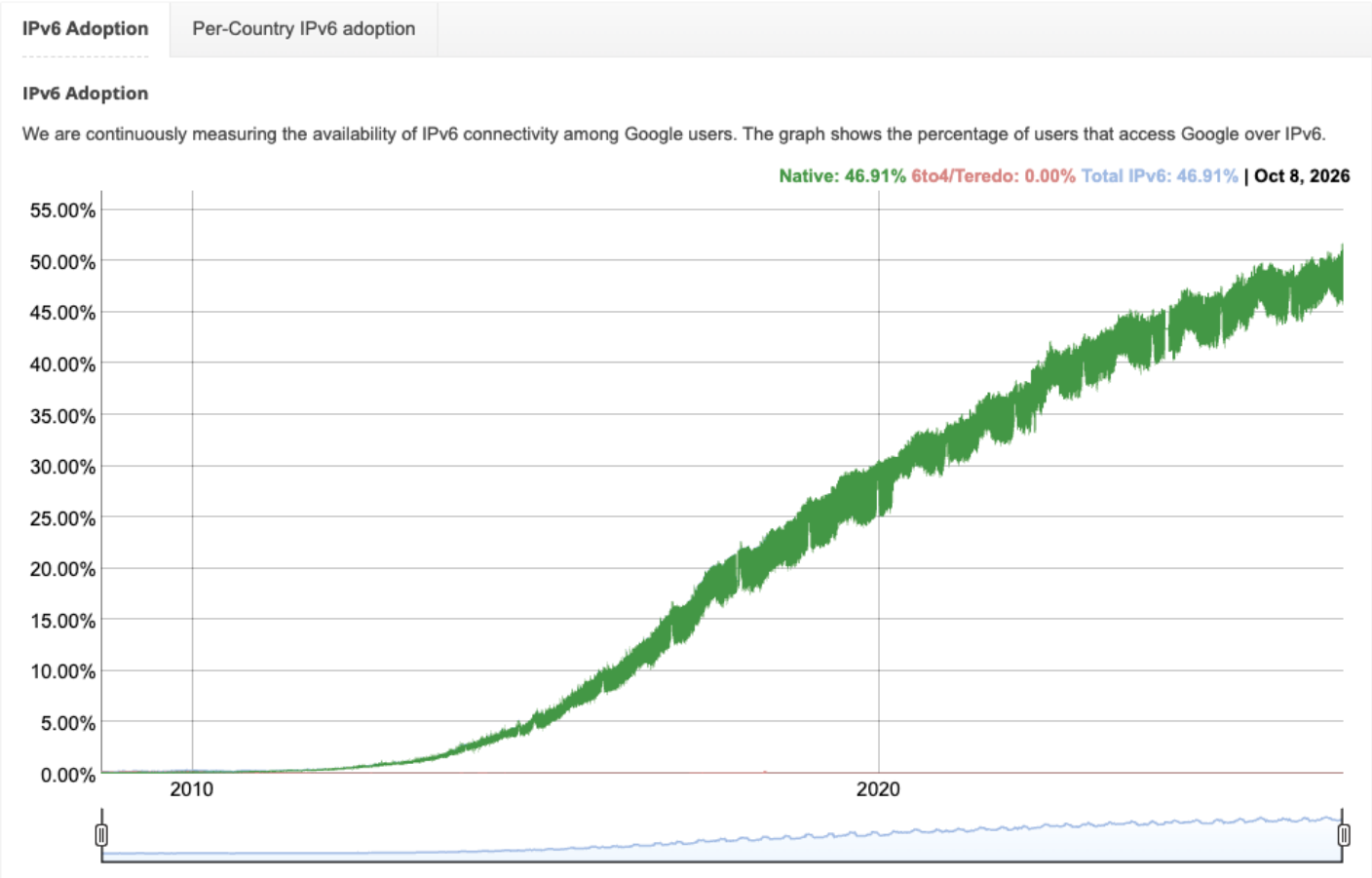
- No checksum (to speed processing at routers, besides, checksum is already at Layer 4)
- No fragmentation/reassembly

Header is always 40 bytes fixed length

- No in-header options — moved to extension headers that chain after the base header
- If Next Header is 6 (TCP) or 17 (UDP), Layer 4 starts at byte 40
- Otherwise follow the Next Header chain — extension headers come first
- Routers only have to read base header; the rest are for destination (this is the speed win)

IPv6 is about 50% of Google's traffic today

- Google[1]: ~ 50% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable



[1] https://www.google.com/intl/en/ipv6/statistics.html

IPv6 is about 50% of Google's traffic today

- Google[1]: ~ 50% of clients access services via IPv6
- NIST: 1/3 of all US government domains are IPv6 capable
- Long (long!) time for deployment, use
  - More than 25 years and counting!
  - Think of application-level changes in last 25 years: WWW, social media, streaming media, gaming, telepresence, …
  - *Why?*

[1] https://www.google.com/intl/en/ipv6/statistics.html

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Review: Transport layer moves packets across a network

Conceptual network layers

7) Application	Interacts with application programs to send ***messages*** Applications assigned a port, multiple instances can run (many browser pages) Examples: HTTP, SSH, FTP, SMTP, DNS
4) Transport	Moves ***segments or datagrams*** May provide error control, flow control, application addressing (ports) Examples: TCP (connection-oriented), UDP (connectionless) TCP provides sequencing, dropped packet resend, traffic congestion routing
3) Network (IP)	Moves ***packets*** between local area networks (routing) Each computer on the Internet identified by an IP address (IP v4 or v6) **Also called Layer 3 or IP layer (ICMP Ping is here)**
2) Link (MAC)	Moves ***frames*** within a local area network (switching) Each computer identified by a MAC address on its Network Interface Card (NIC) Also called Layer 2, MAC layer, Data Link layer, or Ethernet layer
1) Physical	How data is physically transmitted • Transmitter converts logical 1 and 0 ***bits*** to electrical/light pulses or phase/amplitude of radio frequency (RF) and sends down wire or over air • Receiver converts electrical/light or RF back to logical 1 and 0 bits

https://www.ibm.com/think/topics/osi-model

Agenda



1. Routing
2. OSPF (routing within an AS)
3. BGP (routing between AS's)
4. SDN control plane
5. Exercise

Big picture: we want to send packets from one subnet to another

We will consider two cases:

1. The source and destination belong to the same organization
   - By "organization" I mean one group controls the routing of the network
   - An organization may be an ISP, a large company, or college
   - For example, Dartmouth runs a large network consisting of many routers connecting many groups across campus (CS, Thayer, Biology, Chemistry, Government, …)
   - These organization-controlled networks called an **Autonomous System (AS)**
   - Each AS is identified by a unique number (ASN)
   - We will use **Open Shortest Path First (OSPF)** (other options RIP, EIGRP, …) to compute a route between a source and any destination ***within*** **the AS**
2. The source and destination do not belong to the same organization
   - We will use **Border Gateway Protocol (BGP)** to route ***between*** **Autonomous Systems**
   - There are about 120,000 AS'es in the world[1]
   - There were only about 3,000 AS'es in 1995!

[1] https://www-public.telecom-sudparis.eu/~maigron/rir-stats/rir-delegations/world/world-asn-by-number.html

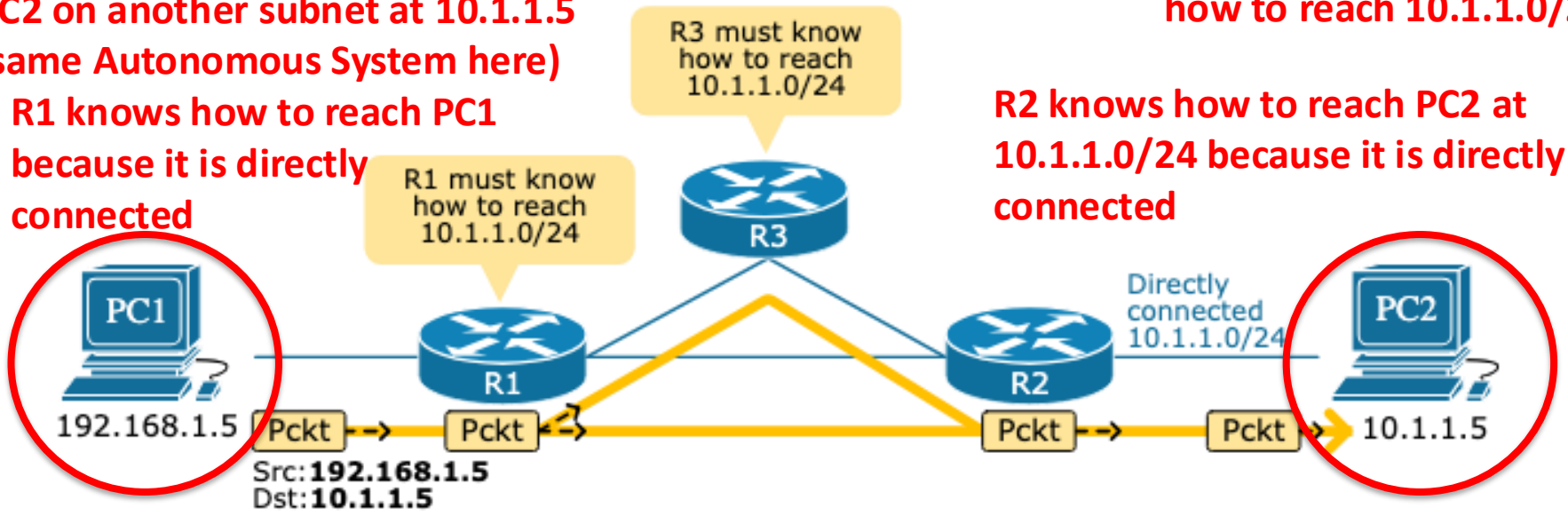
Big picture: we want to send packets from one subnet to another in the same AS

PC1 at 192.168.1.5 wants to send to PC2 on another subnet at 10.1.1.5 (same Autonomous System here)

R1 and R3 need to learn how to reach 10.1.1.0/24

R1 knows how to reach PC1 because it is directly connected

R2 knows how to reach PC2 at 10.1.1.0/24 because it is directly connected



Assume PC1 got DHCP from router R1

- **R1 assigned PC1 an IP address and subnet mask on 192.168.1.0/24**
- **R1 gives itself as the default gateway (router to use when leaving the local subnet)**
- **PC1 can tell if it is sending outside the local network if destination IP is not on the 192.168.1.0/24 subnet**
- **PC1 sends traffic to R1 if destination IP is outside local network**
- **Otherwise, no routing needed if not leaving local subnet, send to MAC address**

PC2 got DHCP from R2 on subnet 10.1.1.0/24

Adapted from: https://www.networkacademy.io/ccna/ospf/why-do-we-need-ospf

Network-layer functions: forwarding and routing

- Forwarding: move packets from router's input port to appropriate router output port — *data plane*
- Routing: determine route taken by packets from source to destination — **Today** *control plane*

Two approaches to structuring network control plane:

1. Per-router control (traditional)
2. Logically centralized control (software defined networking)

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

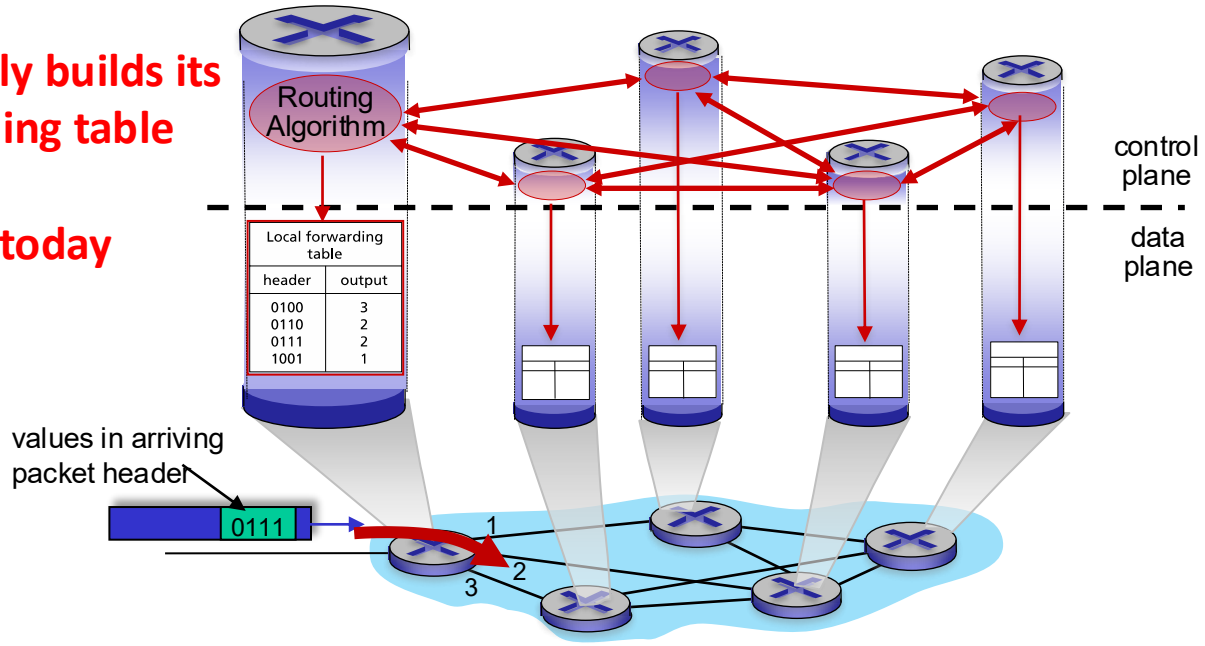
Traditional per-router control plane

Traditional routing

Individual routing algorithm components *in each and every router* interact in the control plane

Each router independently builds its own forwarding table

We see how today



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Software-Defined Networking (SDN) control plane

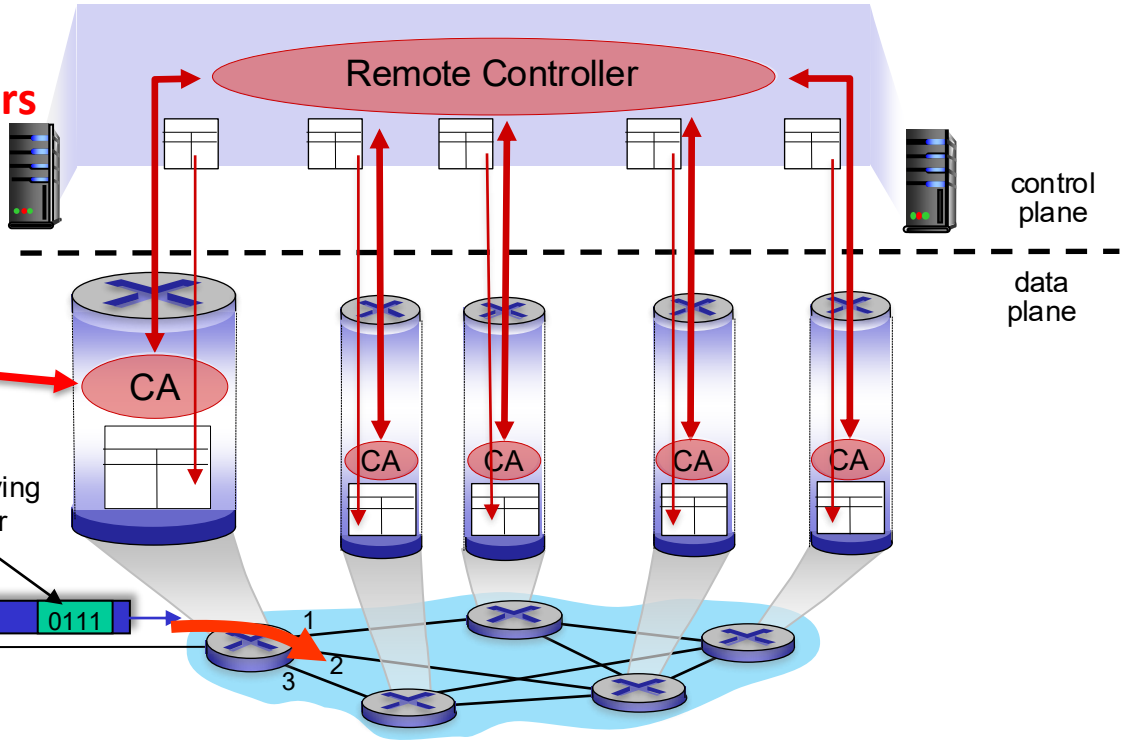
Software Defined Networking (SDN)

Remote controller computes, installs forwarding tables in routers

Controller monitors entire network

Gets inputs from routers observing traffic

Updates forwarding tables on each router



CA = Control Agent communicates with Controller

- **Limited functionality**
- **Does not take part in computing forwarding table**
- **Just communicates with Controller**

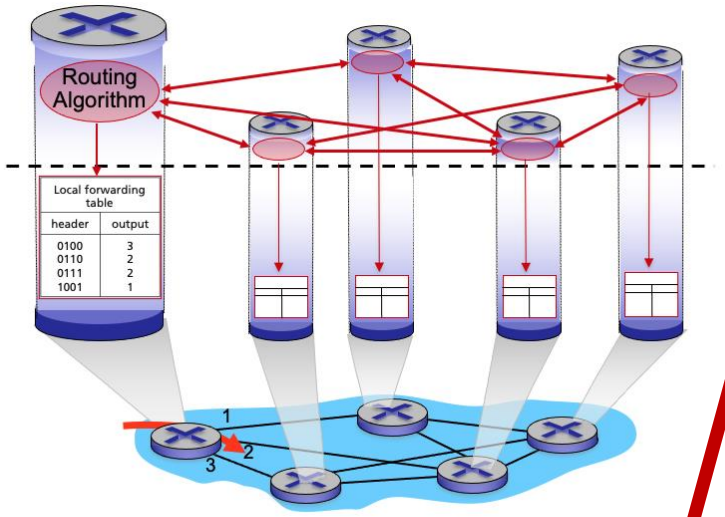
Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Per-router control plane vs SDN control plane

Data plane is the same in both cases

Given a destination address, forward packet to the port that leads to the destination using forwarding table

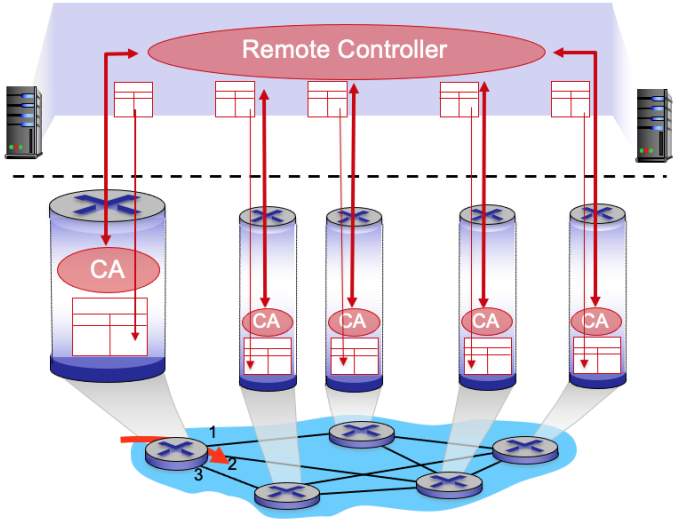
Per-router control plane



Routers communicate among themselves

Each router builds a graph of the network and computes its own forwarding table

SDN control plane



Logically separate Remote Controller computes forwarding tables

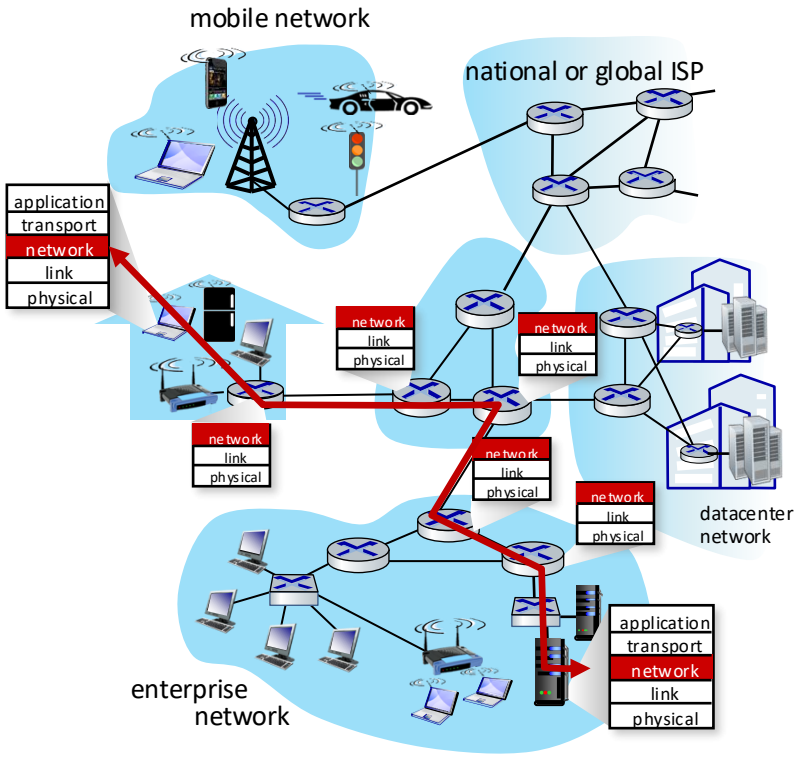
Passes forwarding tables to each router

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Routing protocols

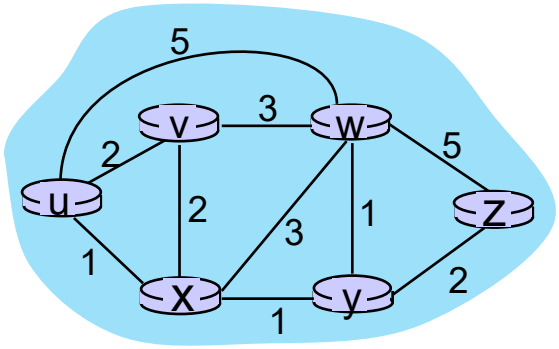
Routing protocol goal: determine "good" paths (equivalently, routes), from sending hosts to receiving host, through network of routers

- Path: sequence of routers packets traverse from given initial source host to final destination host
- "Good": least "cost", "fastest", "least congested"
- Routing: a "top-10" networking challenge!



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Graph abstraction: link costs



$c_{a,b}$: cost of *direct* link connecting *a* and *b*

e.g., $c_{w,z} = 5$, $c_{u,z} = \infty$

Cost defined by network operator could be:

- Always be 1
- Inversely related to bandwidth
- Inversely related to congestion

Undirected 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*) }

Each router provides DHCP to hosts on its local subnet
So, if you want to reach a host on a subnet, send traffic to that subnet's router
The router in turn sends the traffic to the host on its subnet

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Dijkstra's link-state routing algorithm

Compute link costs
Reference bandwidth/Link bandwidth

- Network topology, link costs known to *all* nodes
  - Accomplished via "link state broadcast"
  - All nodes have same info
- Each node computes least cost paths from itself ("source") to all other nodes
  - Gives *forwarding table* for that node
- Iterative: after *k* iterations, know least cost path to *k* destinations

notation

- $c_{x,y}$: direct link cost from node *x* to *y*; = ∞ if not direct neighbors
- *D(v): current* estimate of cost of least-cost-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, initially just starting node

If all edges have same weight, which algorithm can we use to compute shortest path?
BFS!

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Dijkstra's link-state routing algorithm computes shortest path from source

1 *Initialization:*
2 $N' = \{u\}$ // compute least cost path from u to all other nodes
3 for all nodes v
4 if v adjacent to u // u initially knows direct-path-cost only to direct neighbors
5 then $D(v) = c_{u,v}$ // but may not be *minimum* cost!
6 else $D(v) = \infty$
7
8 *Loop*
9 find node 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 least-path-cost to v is either old least-cost-path to v or known
14 least-cost-path to w plus direct-cost from w to v */
15 *until all nodes in N'*

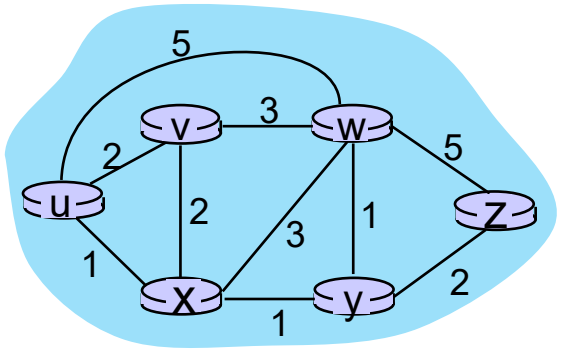
This approach should be very familiar from CS10
Shortest path from u to all other reachable nodes (routers)

How to find node with min D(w)? PriorityQueue!

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Dijkstra's algorithm: an example

		Distance, predecessor				
		v	**w**	**x**	**y**	**z**
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						
2						
3						
4						
5						



Initialization (step 0):

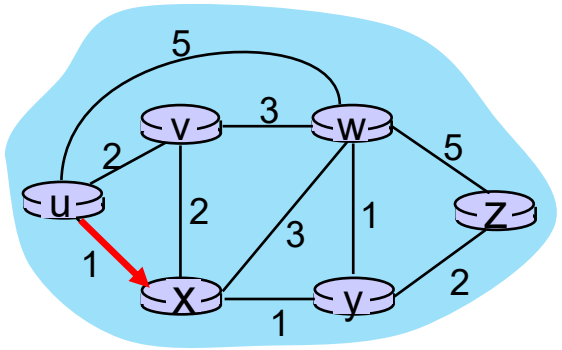
For all *a*: if *a* adjacent to *u* then $D(a) = c_{u,a}$

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Dijkstra's algorithm: an example

Distance, predecessor

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



8 Loop

9 find *a* not in *N'* such that *D(a)* is a minimum

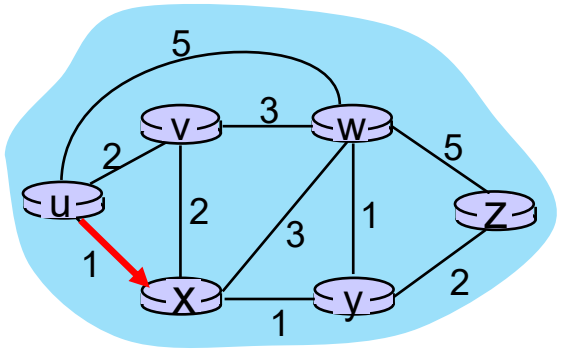
10 add *a* to *N'*

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Dijkstra's algorithm: an example

Distance, predecessor

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



8 *Loop*

9 find *a* not in *N'* such that *D(a)* is a minimum

10 add *a* to *N'*

11 update *D(b)* for all *b* adjacent to *a* and not in *N'* :

$$D(b) = \min (D(b), D(a) + c_{a,b})$$

$D(v) = \min (D(v), D(x) + c_{x,v}) = \min(2, 1+2) = 2$

$D(w) = \min (D(w), D(x) + c_{x,w}) = \min (5, 1+3) = 4$

$D(y) = \min (D(y), D(x) + c_{x,y}) = \min(\inf, 1+1) = 2$

NEW!

NEW!

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

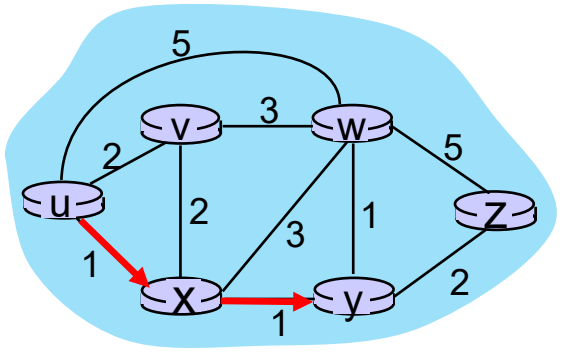
Dijkstra's algorithm: an example

Distance, predecessor

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

Tie between v and y

Choose arbitrarily



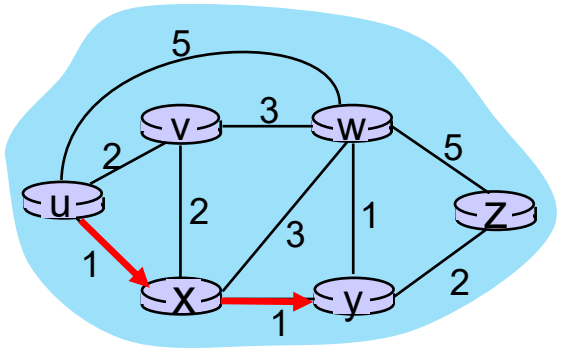
8 *Loop*
9 find *a* not in *N'* such that *D(a)* is a minimum
10 add *a* to *N'*

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Dijkstra's algorithm: an example

Distance, predecessor

Step	N'	v D(v),p(v)	w D(w),p(w)	x D(x),p(x)	y D(y),p(y)	z 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						
4						
5						



8 *Loop*

9 find *a* not in *N'* such that *D(a)* is a minimum

10 add *a* to *N'*

11 update *D(b)* for all *b* adjacent to *a* and not in *N'* :

$D(b) = \min (D(b), D(a) + c_{a,b})$

$D(w) = \min (D(w), D(y) + c_{y,w}) = \min (4, 2+1) = 3$ NEW!

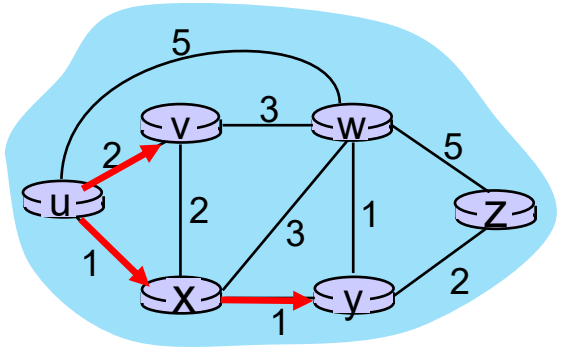
$D(z) = \min (D(z), D(y) + c_{y,z}) = \min(\inf, 2+2) = 4$ NEW!

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Dijkstra's algorithm: an example

Distance, predecessor

Step	N'	v D(v),p(v)	w D(w),p(w)	x D(x),p(x)	y D(y),p(y)	z 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					
4						
5						



8 Loop

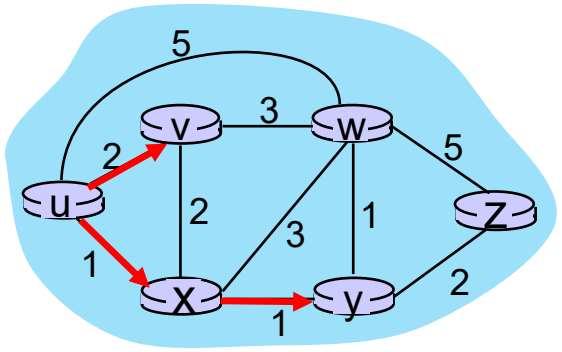
9 find *a* not in *N'* such that *D(a)* is a minimum

10 add *a* to *N'*

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Dijkstra's algorithm: an example

Step	N'	v D(v),p(v)	w D(w),p(w)	x D(x),p(x)	y D(y),p(y)	z D(z),p(z)
		Distance, predecessor				
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						
5						



8 *Loop*

9 find *a* not in *N'* such that *D(a)* is a minimum

10 add *a* to *N'*

11 update *D(b)* for all *b* adjacent to *a* and not in *N'* :

$$D(b) = \min (D(b), D(a) + c_{a,b})$$

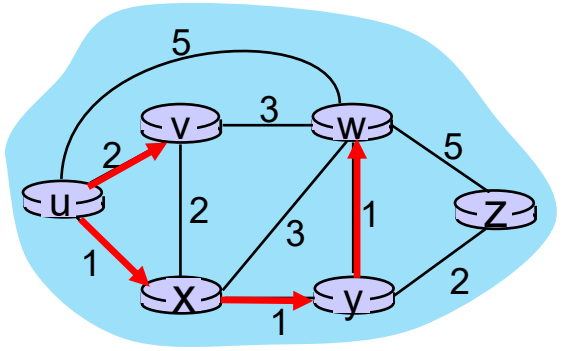
$$D(w) = \min (D(w), D(v) + c_{v,w}) = \min (3, 2+3) = 3$$

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Dijkstra's algorithm: an example

Distance, predecessor

Step	N'	v D(v),p(v)	w D(w),p(w)	x D(x),p(x)	y D(y),p(y)	z 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					
5						

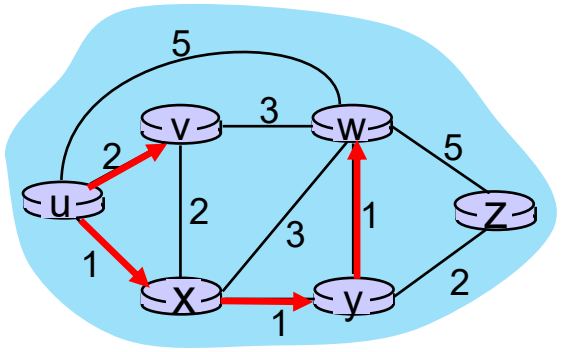


8 *Loop*
9 find *a* not in *N'* such that *D(a)* is a minimum
10 add *a* to *N'*

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Dijkstra's algorithm: an example

Step	N'	v D(v),p(v)	w D(w),p(w)	x D(x),p(x)	y D(y),p(y)	z D(z),p(z)
		Distance, predecessor				
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						



8 *Loop*
9 find *a* not in *N'* such that *D(a)* is a minimum
10 add *a* to *N'*
11 update *D(b)* for all *b* adjacent to *a* and not in *N'* :

$$D(b) = \min (D(b), D(a) + c_{a,b})$$

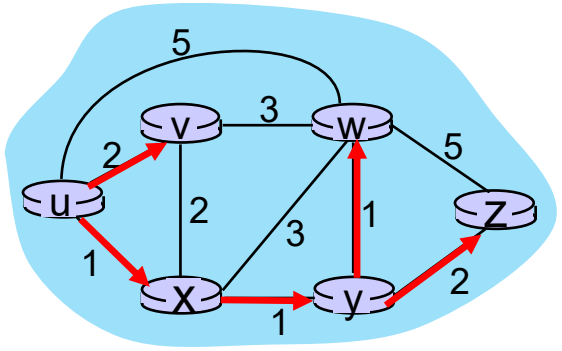
$D(z) = \min (D(z), D(w) + c_{w,z}) = \min (4, 3+5) = 4$

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Dijkstra's algorithm: an example

Distance, predecessor

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



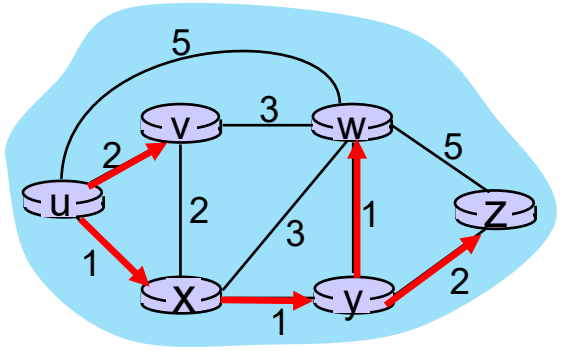

```
8  Loop
9     find a not in N' such that D(a) is a minimum
10    add a to N'
```


Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Dijkstra's algorithm: an example

Distance, predecessor

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



8 Loop

9 find *a* not in *N'* such that *D(a)* is a minimum

10 add *a* to *N'*

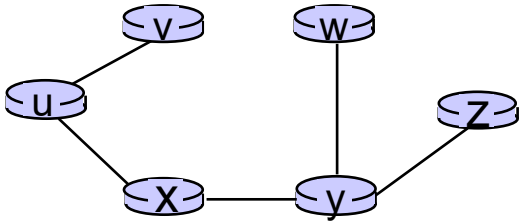
11 update *D(b)* for all *b* adjacent to *a* and not in *N'* :

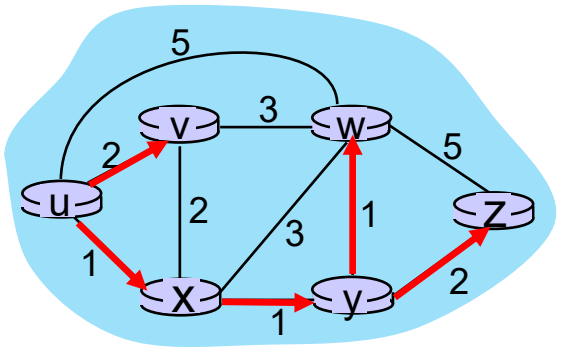
$D(b) = \min (D(b), D(a) + c_{a,b})$

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Dijkstra's algorithm: an example

Resulting least-cost-path tree from u:





Here traffic to v goes out port 2

Traffic everywhere else goes out port 1

All routers keep a similar forwarding table

Resulting forwarding table in u:

destination	outgoing link
v	2: (u,v)
x	1: (u,x)
y	1: (u,x)
w	1: (u,x)
z	1: (u,x)

route from *u* to *v* directly

route from u to all other destinations via *x*

Does the router need to keep the entire tree?

No

It just needs to store the next hop in the forwarding table

This router doesn't care about what other routers do!

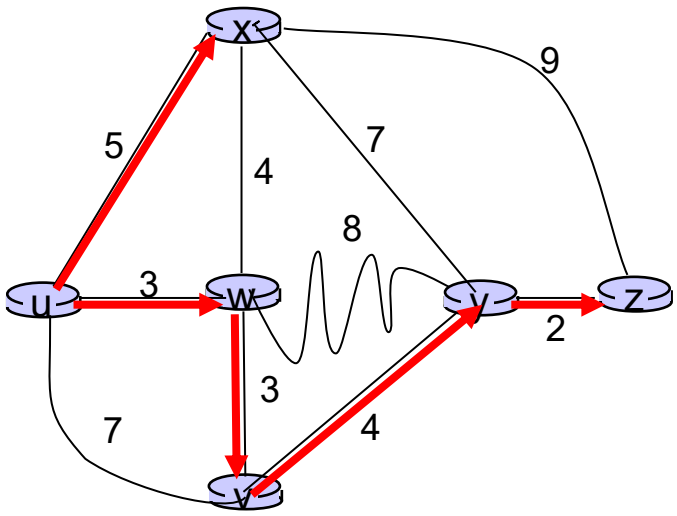
At the control plane, all this router wants to do is send the packet to the next hop quickly

(Storing only the next hop is an important concept! We will see it again)

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Dijkstra's algorithm: another example

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



Resulting forwarding table in u:

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

Notes:

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

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Dijkstra's algorithm complexity

Algorithm complexity: n nodes

- Each of n iteration: need to check all nodes, w, not in N
- $n(n+1)/2$ comparisons: $O(n^2)$ complexity
- More efficient implementations possible: PQ heap = $O(e \log n)$

Message complexity:

- Each router must *broadcast* its link state information to other n routers
- Efficient (and interesting!) broadcast algorithms: $O(n)$ link crossings to disseminate a broadcast message from one source
- Each router's message crosses $O(n)$ links: overall message complexity: $O(n^2)$

Distance Vector (DV) algorithms avoid the $O(n^2)$ flood — routers talk only to neighbors, never network-wide, and never hold the full topology
Tradeoff: slower convergence, possible routing loops

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Agenda

1. Routing
2. OSPF (routing within an AS)
3. BGP (routing between AS's)
4. SDN control plane
5. Exercise

Making routing scalable

Our routing study thus far has been idealized

- all routers identical
- network “flat”

… not true in practice

Scale: billions of destinations:

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

Administrative autonomy:

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

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Internet approach to scalable routing

Aggregate routers into regions known as "autonomous systems" (AS) (a.k.a. "domains")

Intra-AS (aka "intra-domain"): Routing among routers *within same AS ("network")*

- All routers in AS must run same intra-domain protocol and have information about each other
- Routers in different AS can run different intra-domain routing protocols
- Gateway router: at "edge" of its own AS, has link(s) to router(s) in other AS'es
- Each AS has its own number assigned by IANA (via Regional Internet Registries)
- **Use Open Shortest Path First (OSPF)**

Inter-AS (aka "inter-domain"):

Routing *among* AS'es

- Gateways perform inter-domain routing (as well as intra-domain routing)
- **Use Border Gateway Protocol (BGP)**

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Intra-AS routing: Routing within an AS

Most common intra-AS routing protocols:

- RIP: Routing Information Protocol [RFC 1723]
  - Classic DV: DVs exchanged every 30 secs
  - No longer widely used
- EIGRP: Enhanced Interior Gateway Routing Protocol
  - DV based
  - Formerly Cisco-proprietary for decades (became open in 2013 [RFC 7868])
- OSPF: Open Shortest Path First [RFC 2328]
  - Link-state routing
  - IS-IS protocol (pronounced “eye ess, eye ess”, ISO standard, not RFC standard) essentially same as OSPF

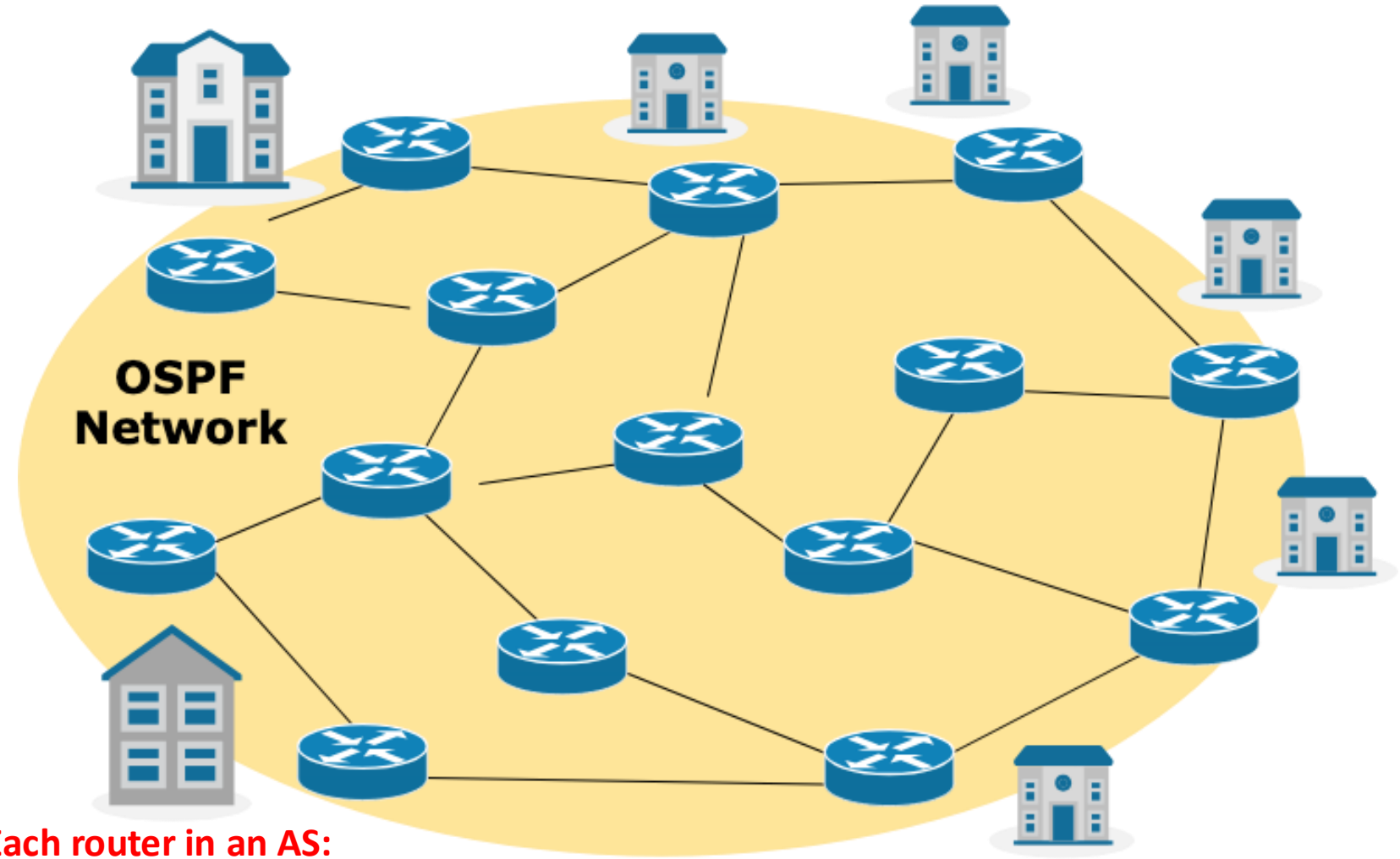
Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

OSPF (Open Shortest Path First) routing

- "Open": publicly available
- Classic link-state algorithm
  - Each router floods OSPF Link-State Advertisements (LSAs) directly over IP rather than using TCP/UDP, uses protocol 89 (recall TCP=6, UDP=17) to all other routers in entire AS
  - Multiple link costs metrics possible: bandwidth, delay
  - Each router builds full topology of the AS and uses Dijkstra's algorithm to compute forwarding table
- *Security:* all OSPF messages with sequence number (to prevent replay attacks) can be authenticated via MD5(!) hash to prevent malicious intrusion

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

OSPF allows each router to calculate the best route to all other routers in the AS



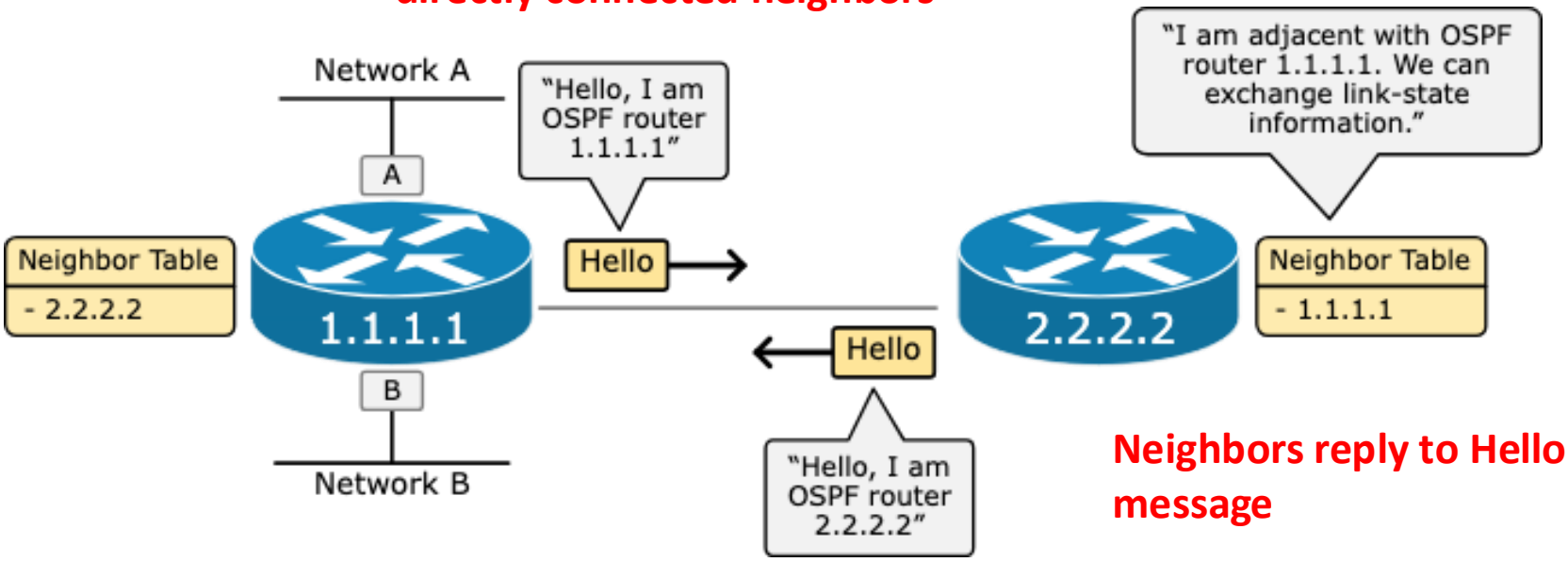
Each router in an AS:

- **Given a unique ID**
- **Runs the OSPF protocol**

Adapted from: https://www.networkacademy.io/ccna/ospf/what-is-ospf

Step 1: become neighbors with directly connected routers

New router sends a Hello message to directly connected neighbors

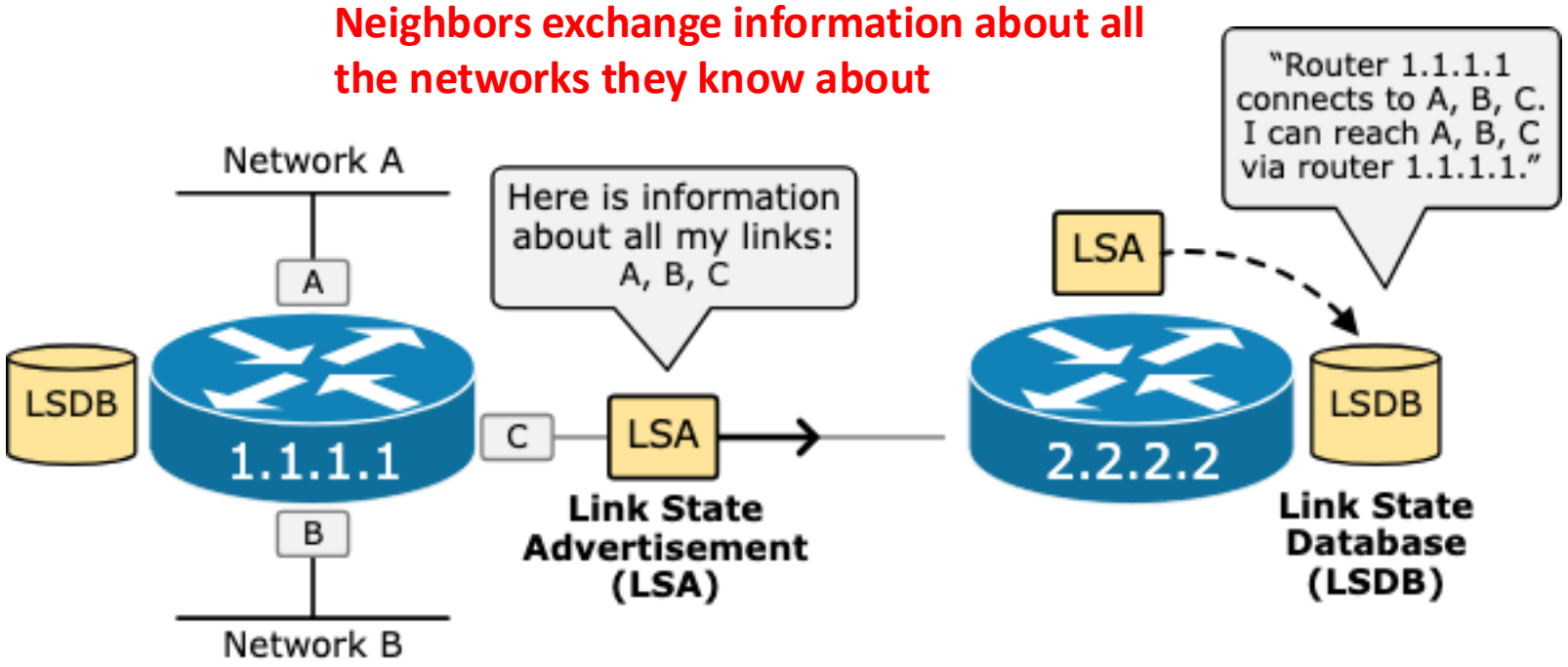


Neighbors reply to Hello message

Neighbors now know about each other

Adapted from: https://www.networkacademy.io/ccna/ospf/what-is-ospf

Step 2: Exchange database information with neighbor

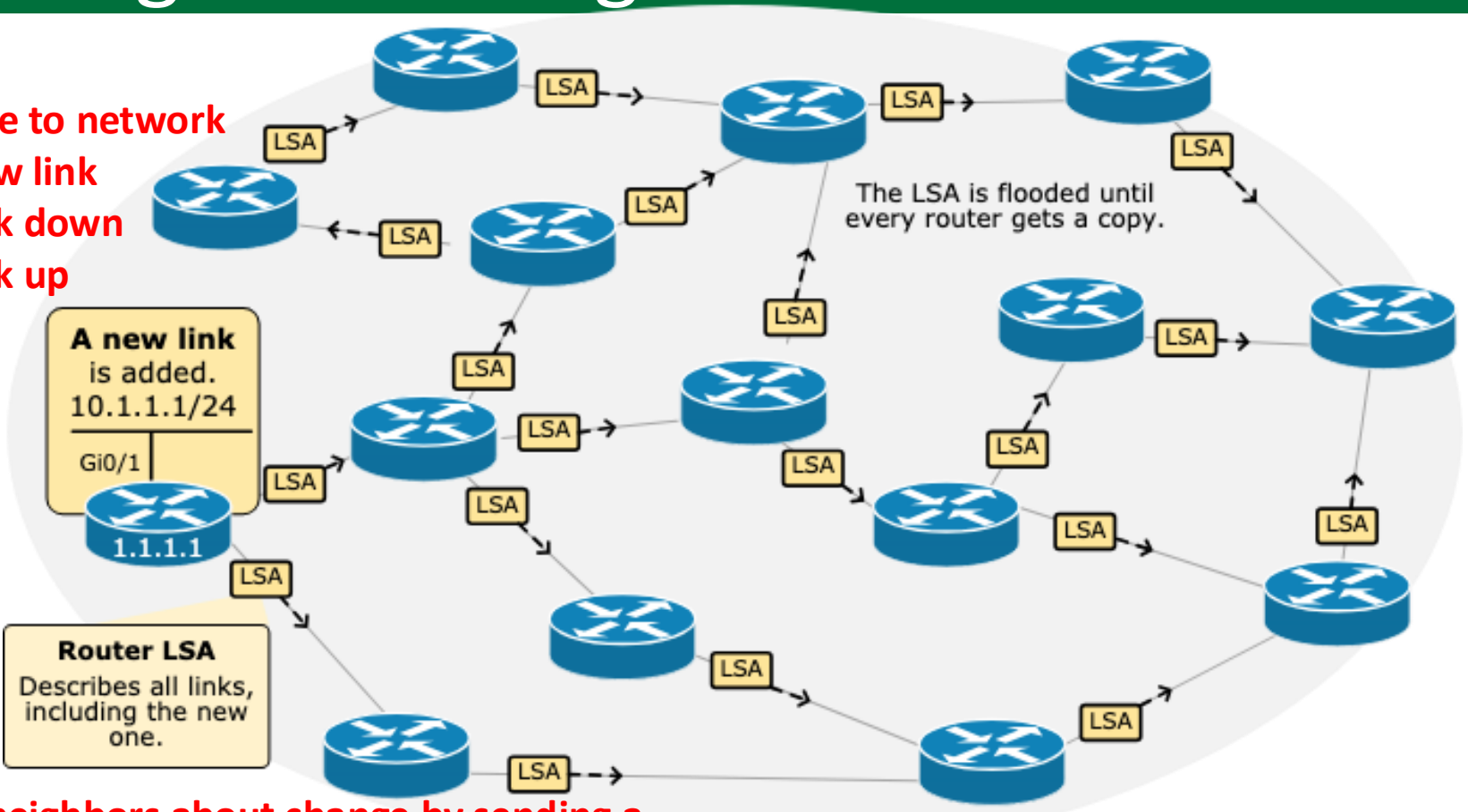


Adapted from: https://www.networkacademy.io/ccna/ospf/what-is-ospf

Step 3: flood the network with LSA messages on changes

Change to network

- **New link**
- **Link down**
- **Link up**



Tell neighbors about change by sending a Link State Advertisement (LSA)

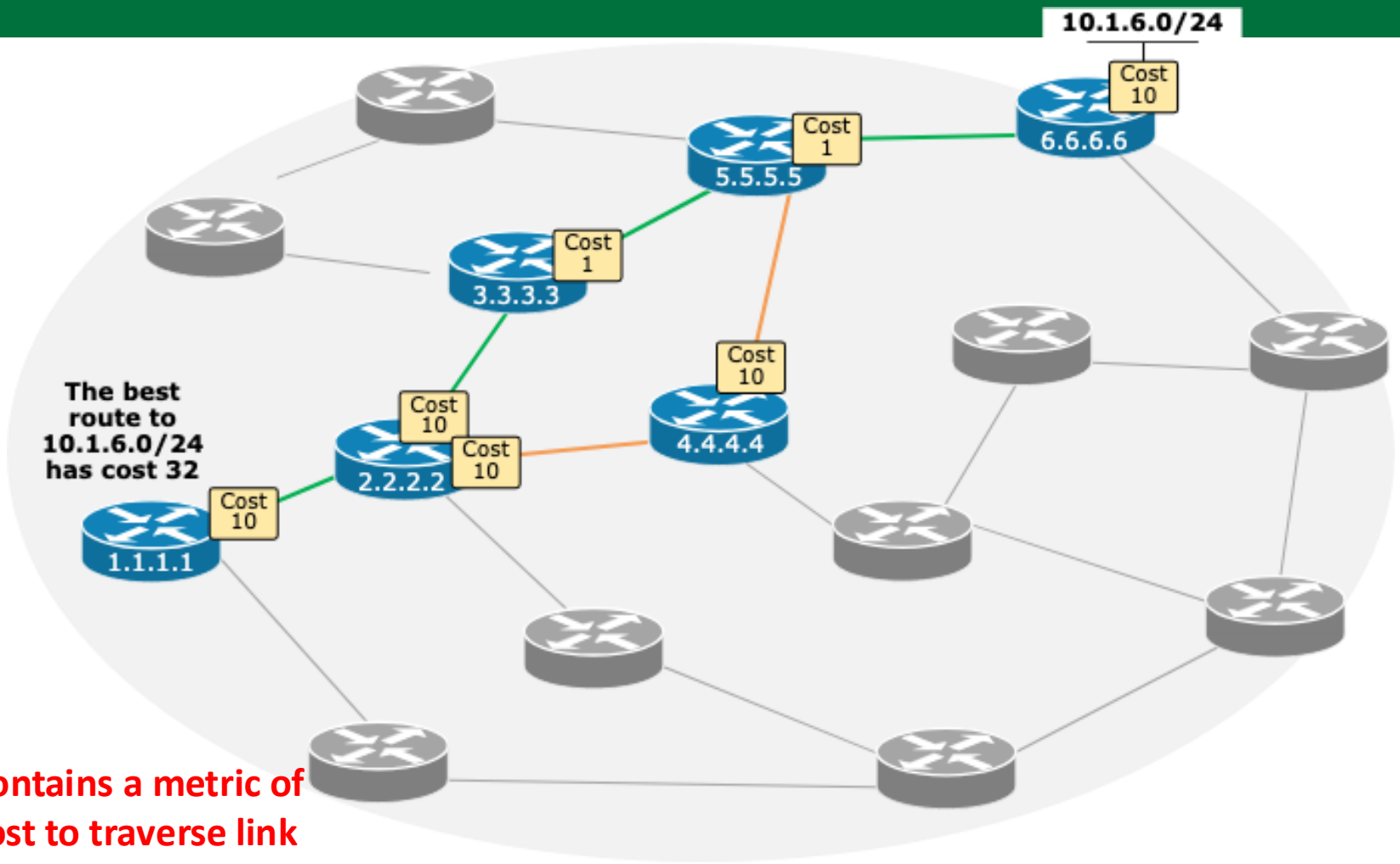
Neighbors tell their neighbors about the change

Who tell their neighbors…

Called LSA flooding

Adapted from: https://www.networkacademy.io/ccna/ospf/what-is-ospf

Step 4: Calculate best routes



LSA contains a metric of the cost to traverse link

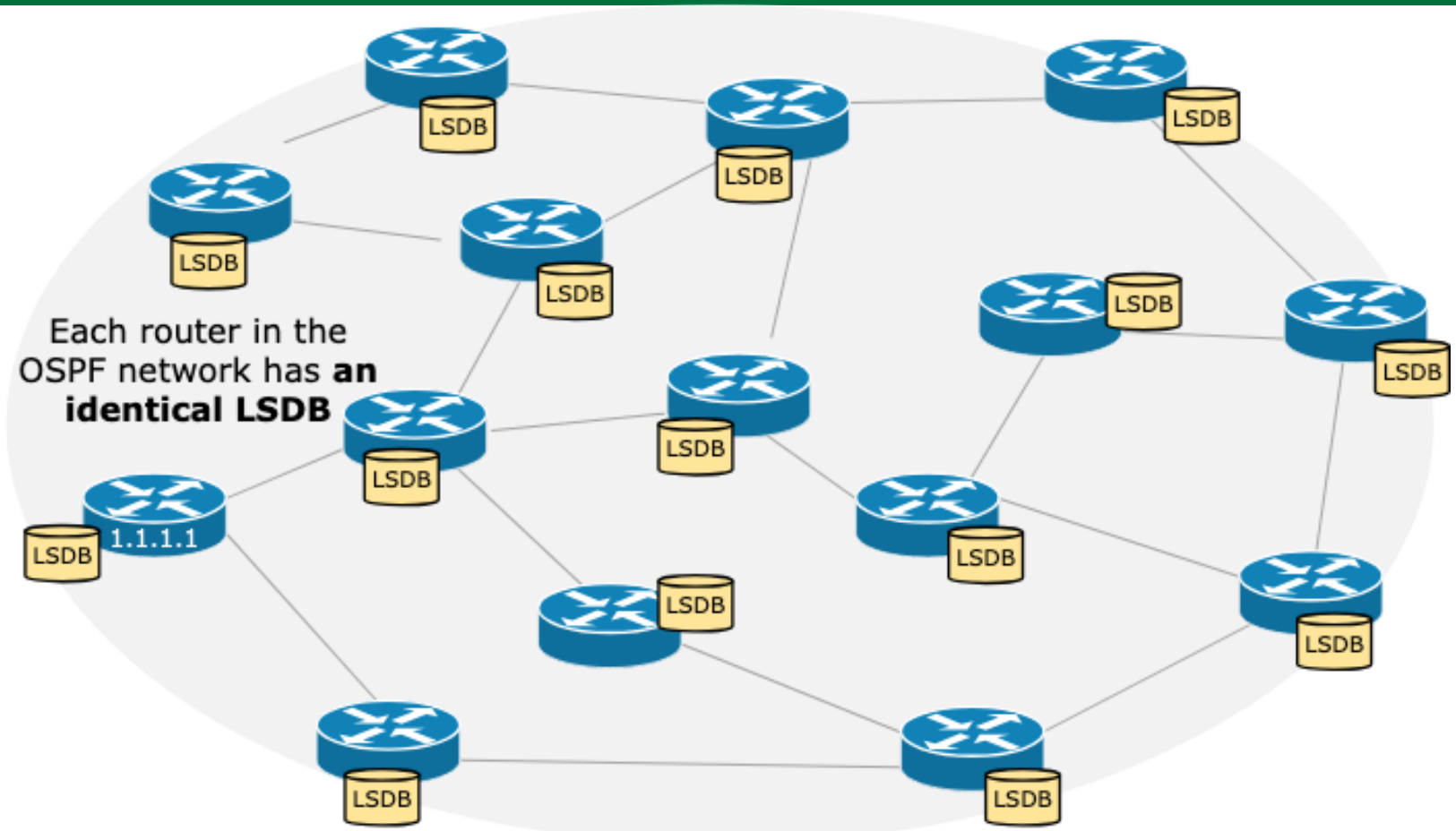
Reference bandwidth
Link bandwidth

Reference often 100 Mbs

- **10 Mbps link has cost of 10**
- **100 Mbps link has cost of 1**

Adapted from: https://www.networkacademy.io/ccna/ospf/what-is-ospf

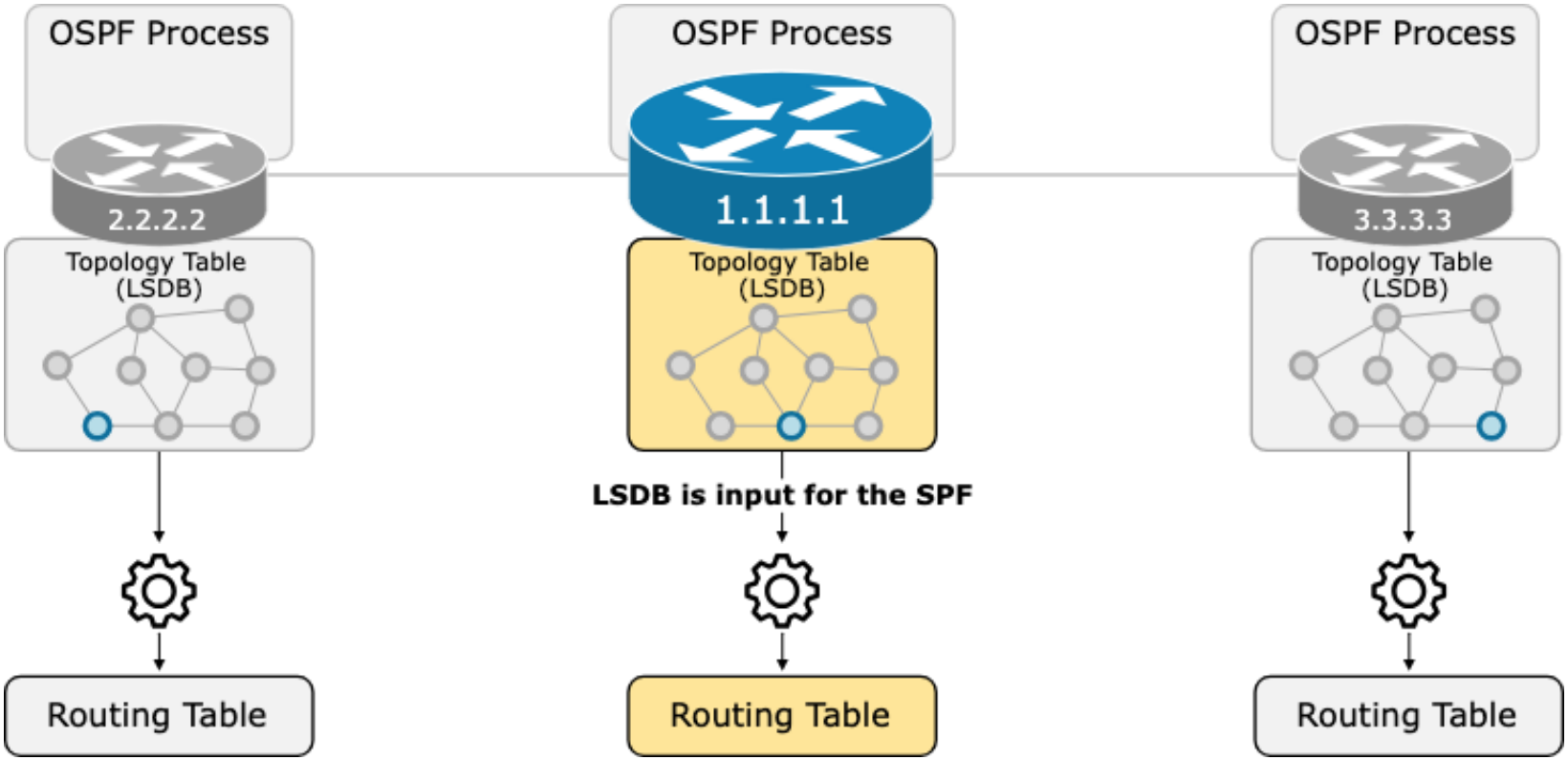
Step 4: Calculate best routes



After flooding, each router has an identical database of links with costs for the entire Autonomous System in a Link State Database (LSDB)

Adapted from: https://www.networkacademy.io/ccna/ospf/what-is-ospf

Step 4: Calculate best route



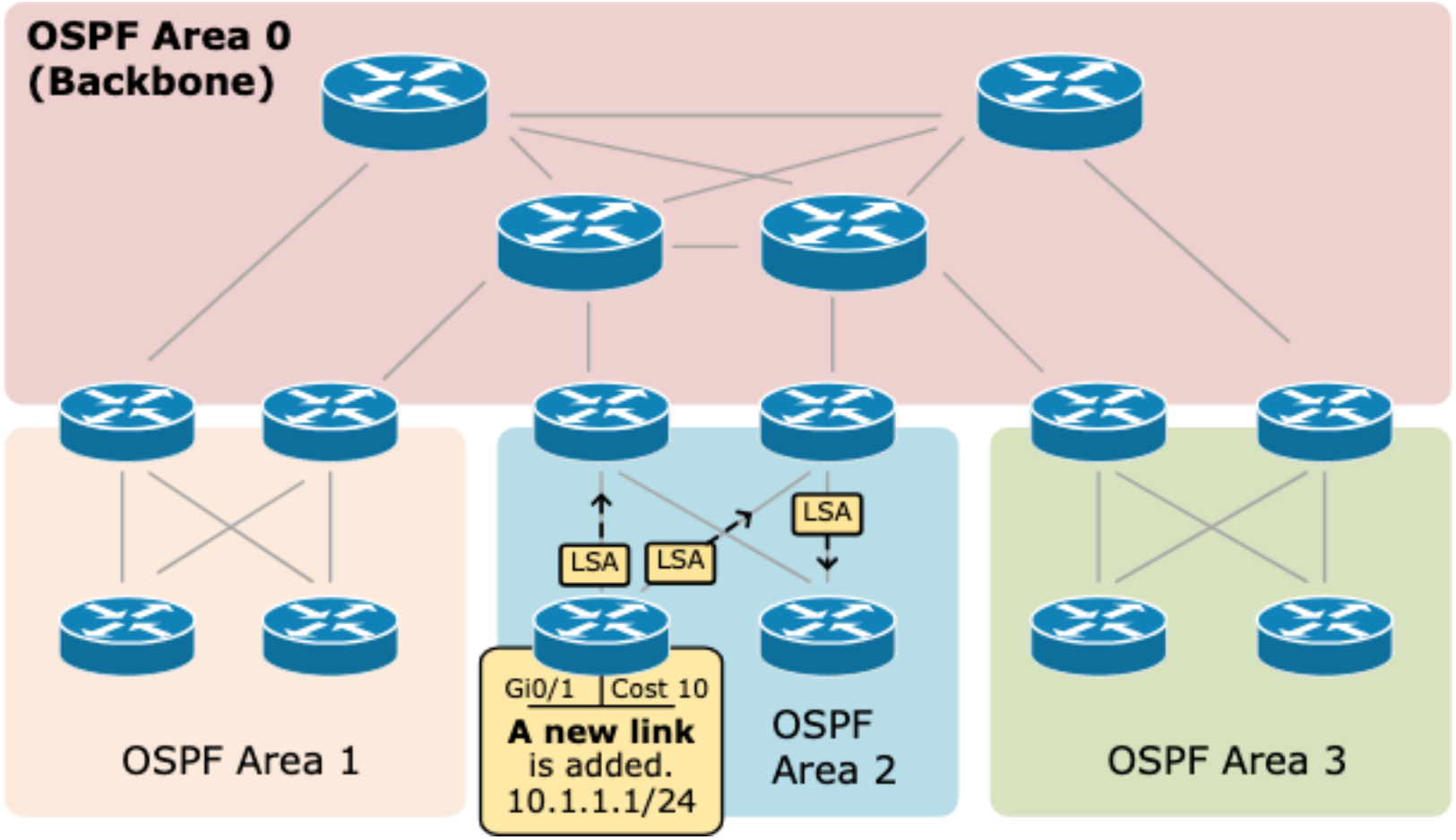
Each router uses its own LSDB to create a graph of the entire network

Then runs Dijkstra's algorithm to find the shortest path between itself and all other routers

Outcome: forwarding table to all intra-AS subnets (keep only next hop in forwarding table)

Adapted from: https://www.networkacademy.io/ccna/ospf/what-is-ospf

OSPF doesn't scale well, use Areas to cut down on message traffic



A large network would generate a lot of message traffic between routers
Routers are assigned to an Area and communicate in their area
Backbone Area (Area 0) connects Areas
Inter Area routing must pass through Area 0

Adapted from: https://www.networkacademy.io/ccna/ospf/what-is-ospf

Agenda

1. Routing
2. OSPF (routing within an AS)
3. BGP (routing between AS's)
4. SDN control plane
5. Exercise

Interconnected ASes

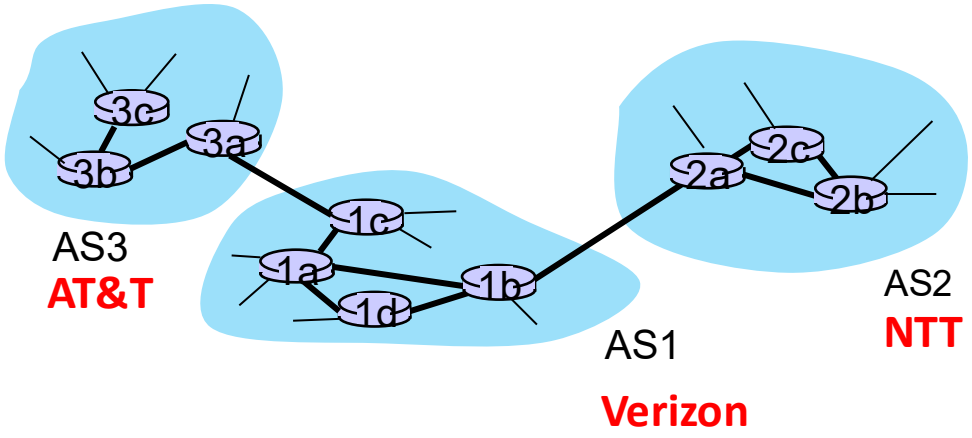
Autonomous Systems (AS)

Large organizations

- ISPs (AT&T, Verizon, NTT …)
- Big companies
- Universities

One organization controls routing within the AS

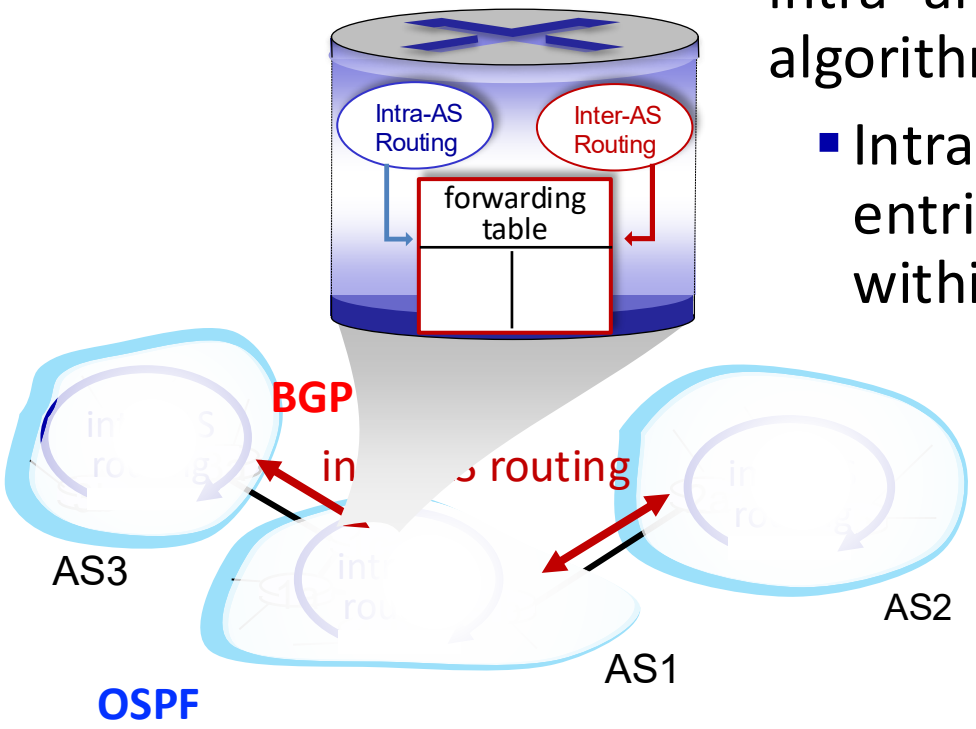
AS'es often have large blocks of IP address space



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Interconnected ASes

Forwarding table entry (x, I)
x = network address/subnet
I = router port number



Forwarding table configured by intra- and inter-AS routing algorithms

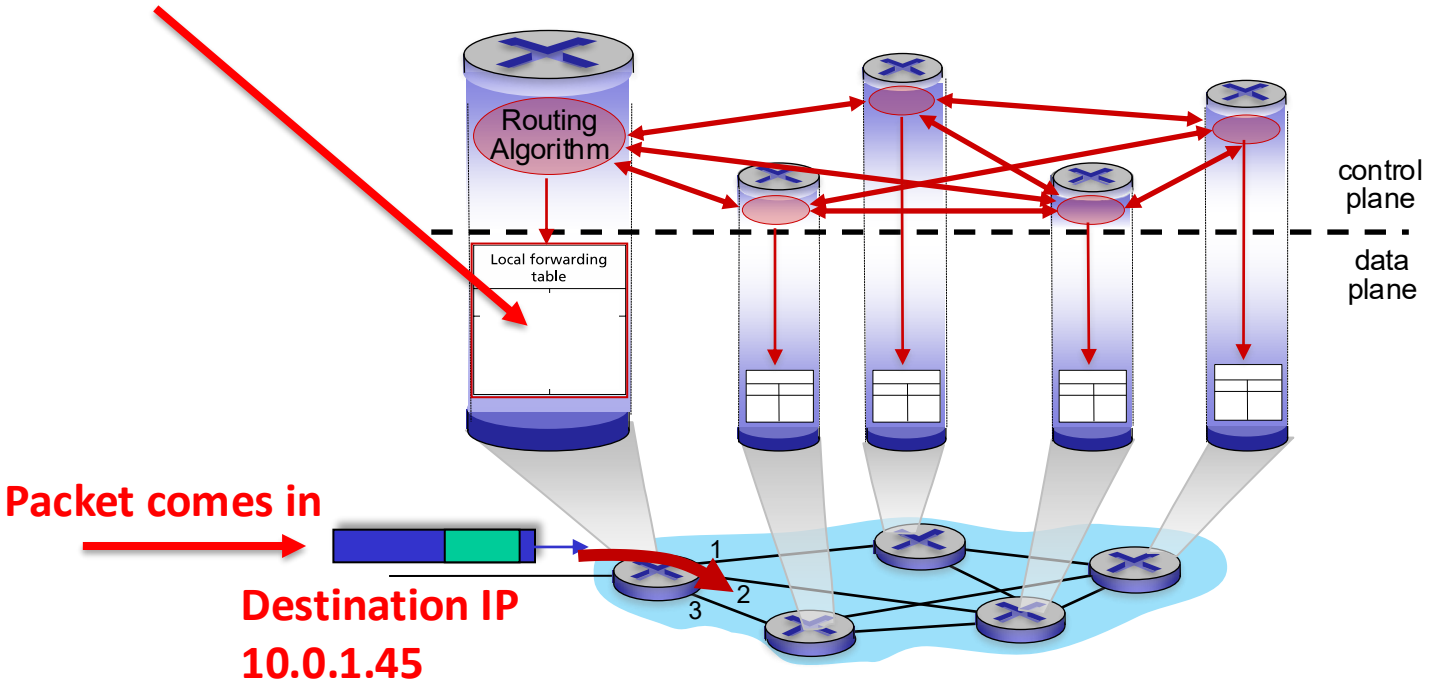
- Intra-AS routing determine entries for destinations within AS
  - Inter-AS determine entries for external destinations

Gateway routers connect AS'es

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Forwarding review: router selects port with the longest match

Network/subnet mask	Port		
10.0.0.0/24	1	covers 10.0.0.0 to 10.0.0.255	**10.0.1.45 does NOT match**
10.0.1.0/24	2	covers 10.0.1.0 to 10.0.1.255	**10.0.1.45 matches**
0.0.0.0/0	3	Default, covers everything else	**Forward packet out port 2**

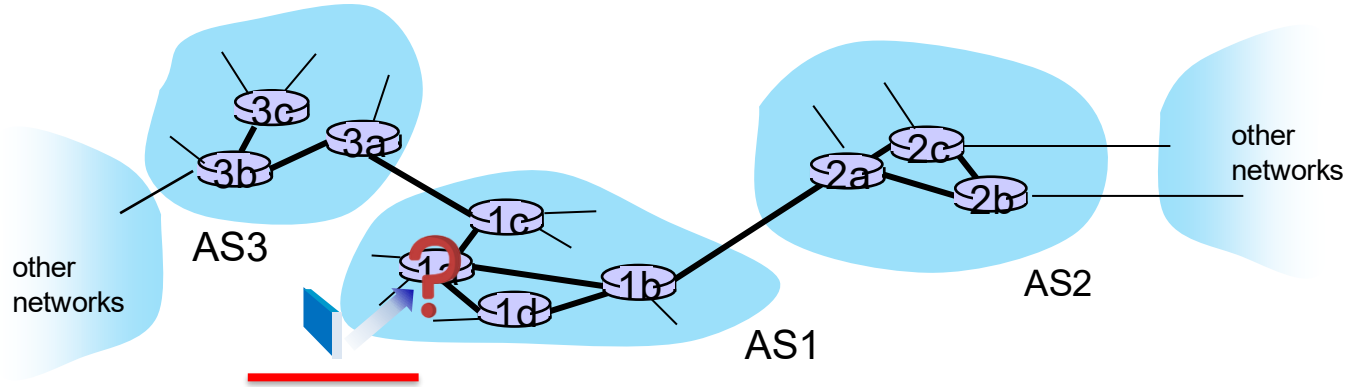


Inter-AS routing: a role in intradomain forwarding

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

AS1 inter-domain routing must:

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



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Border Gateway Protocol is used for Internet inter-AS routing

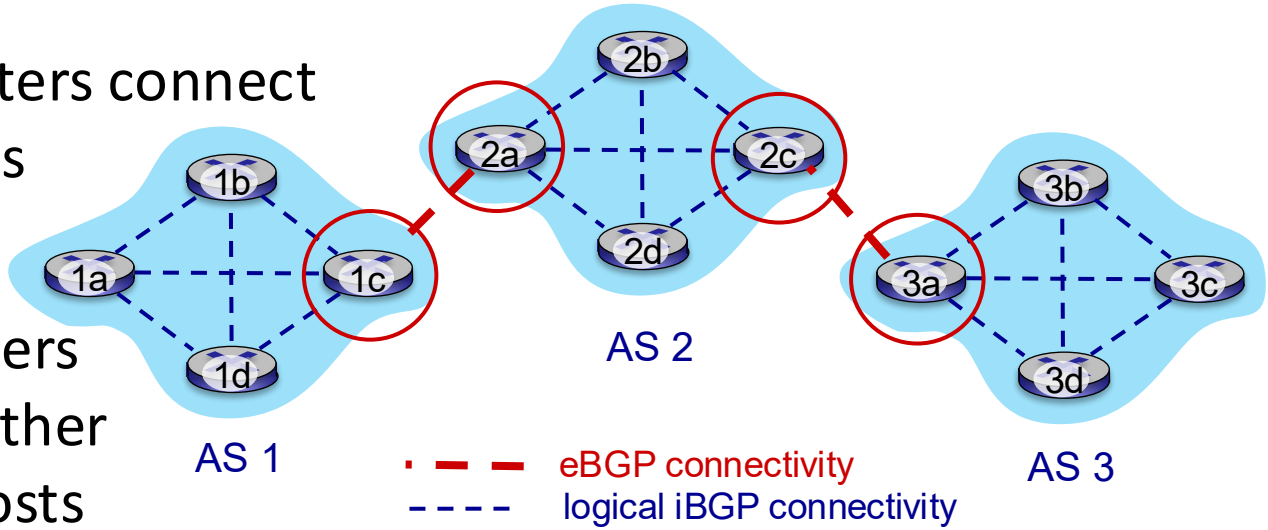
- BGP (Border Gateway Protocol): *the* de facto inter-domain routing protocol
  - "Glue that holds the Internet together"
- Allows subnet to advertise its existence, and the destinations it can reach, to rest of Internet: *"I am here, here is who I can reach, and how"*
- BGP provides each AS a means to:
  - Obtain destination network reachability info from neighboring ASes (eBGP)
  - Determine routes to other networks based on reachability information and *policy*
  - Propagate reachability information to all AS-internal routers (iBGP)
  - Advertise this network to neighboring networks with destination reachability info

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Example: eBGP, iBGP connections

Three interconnected AS'es: AS1, AS2, AS3
Each router is either gateway or internal router

Gateway routers connect to other AS'es



Internal routers connect to other routers or hosts in the same AS



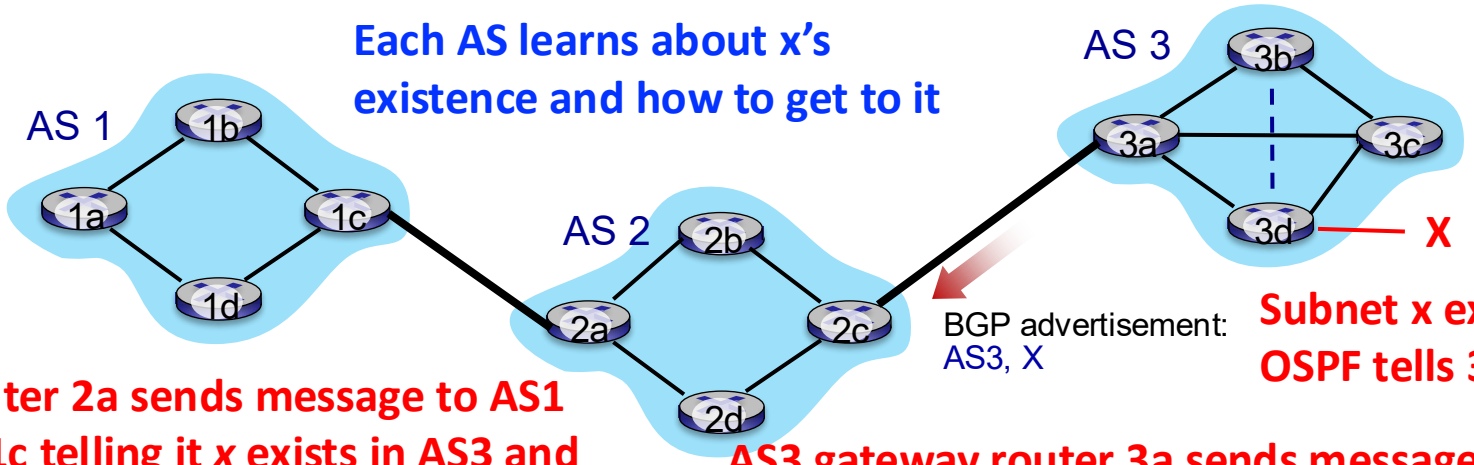
Gateway routers run both eBGP and iBGP protocols

Gateway routers forwarding table entry format: (*x, I*)
x = subnet (138.16.68/22)
I = router interface number

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

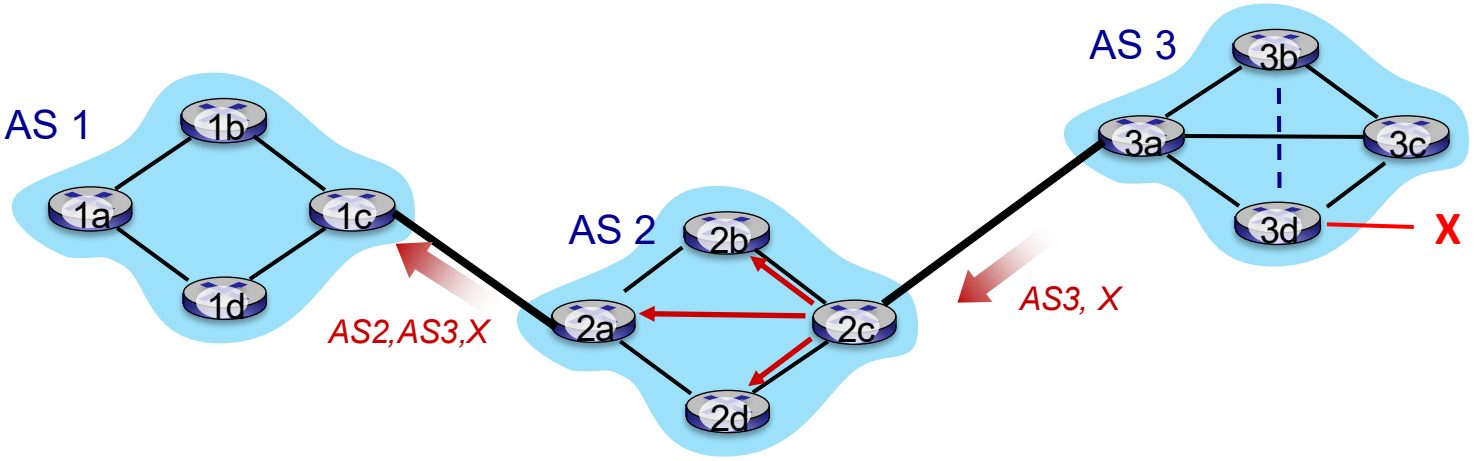
BGP basics

- BGP session: two BGP routers ("peers") exchange BGP messages over semi-permanent TCP connection on port 179:
  - Advertising *paths* to different destination network prefixes (BGP is a "path vector" protocol)
- When AS3 gateway 3a advertises path AS3,X to AS2 gateway 2c:
  - AS3 *promises* to AS2 it will forward datagrams towards *x*



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

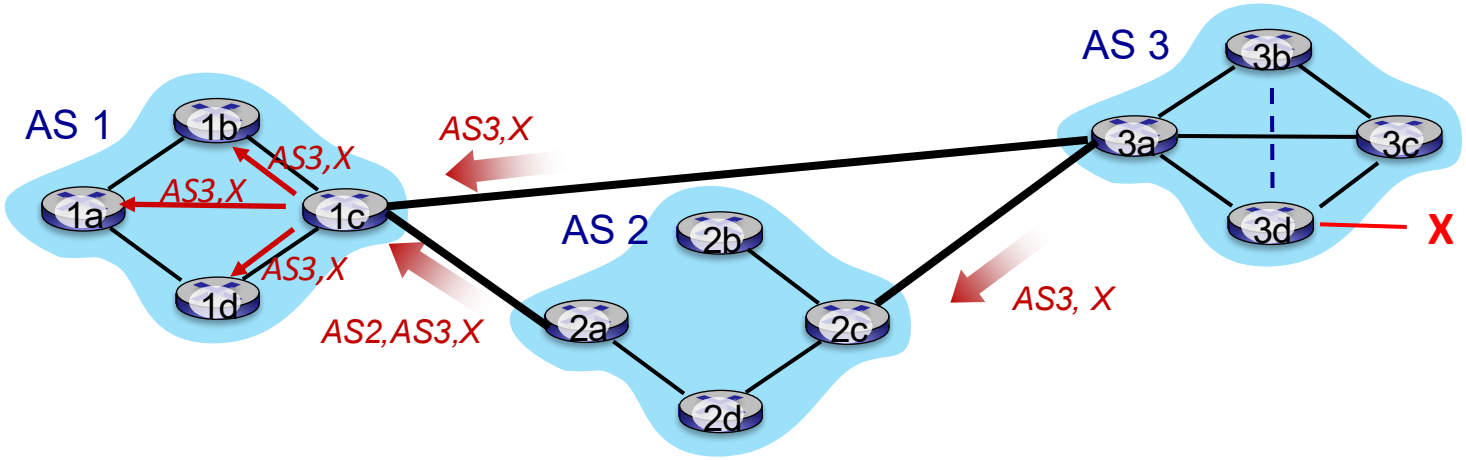
BGP path advertisement



- 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, 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

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

BGP path advertisement: multiple paths

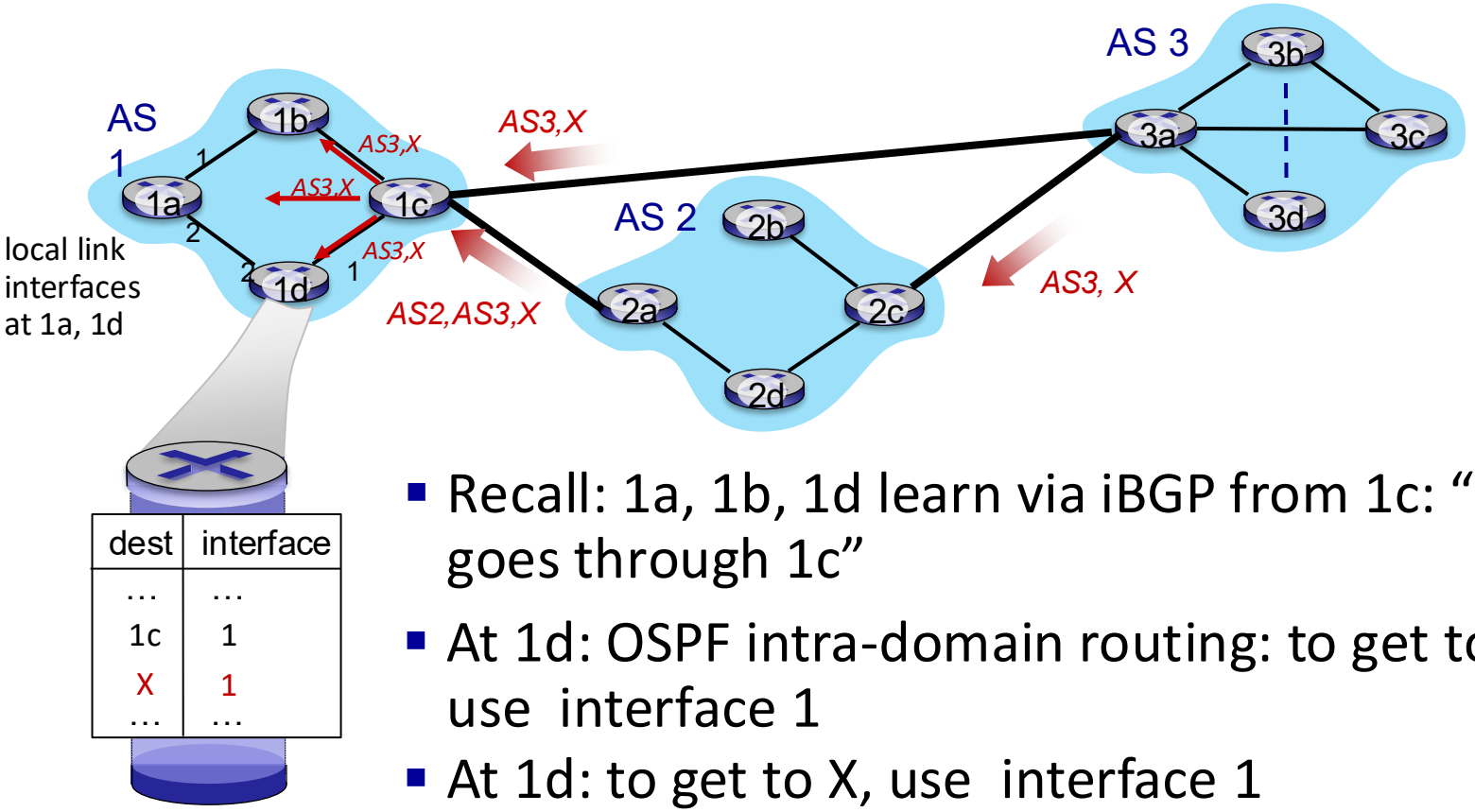


Gateway router may learn about multiple paths to destination:

- 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

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

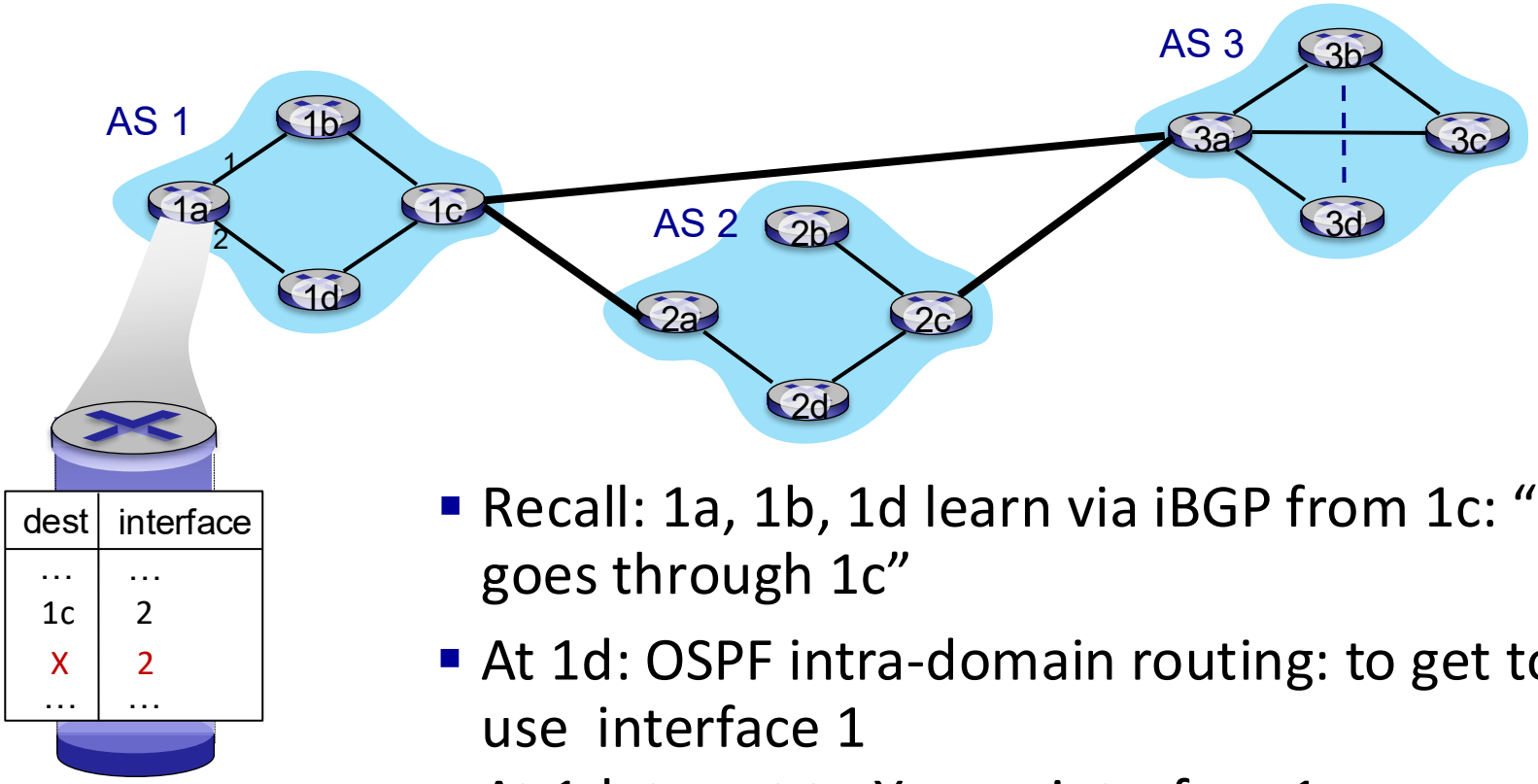
BGP: populating forwarding tables



- Recall: 1a, 1b, 1d learn via iBGP from 1c: “path to X goes through 1c”
- At 1d: OSPF intra-domain routing: to get to 1c, use interface 1
- At 1d: to get to X, use interface 1

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

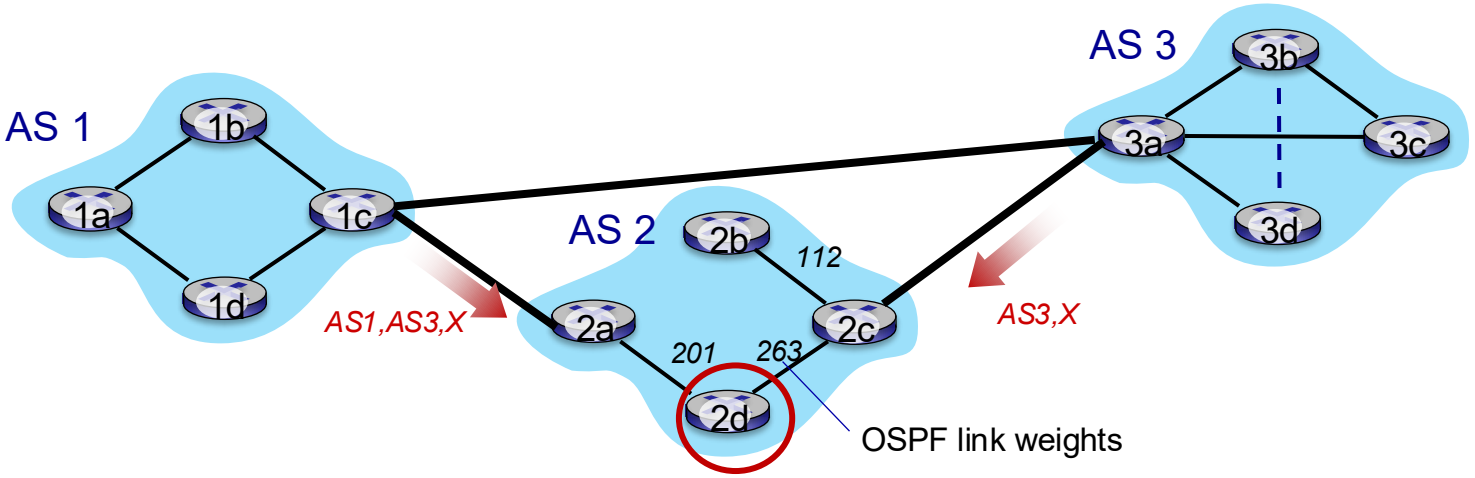
BGP: populating forwarding tables



- Recall: 1a, 1b, 1d learn via iBGP from 1c: “path to X goes through 1c”
- At 1d: OSPF intra-domain routing: to get to 1c, use interface 1
- At 1d: to get to X, use interface 1
- At 1a: OSPF intra-domain routing: to get to 1c, use interface 2
- At 1a: to get to X, use interface 2

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Hot potato routing



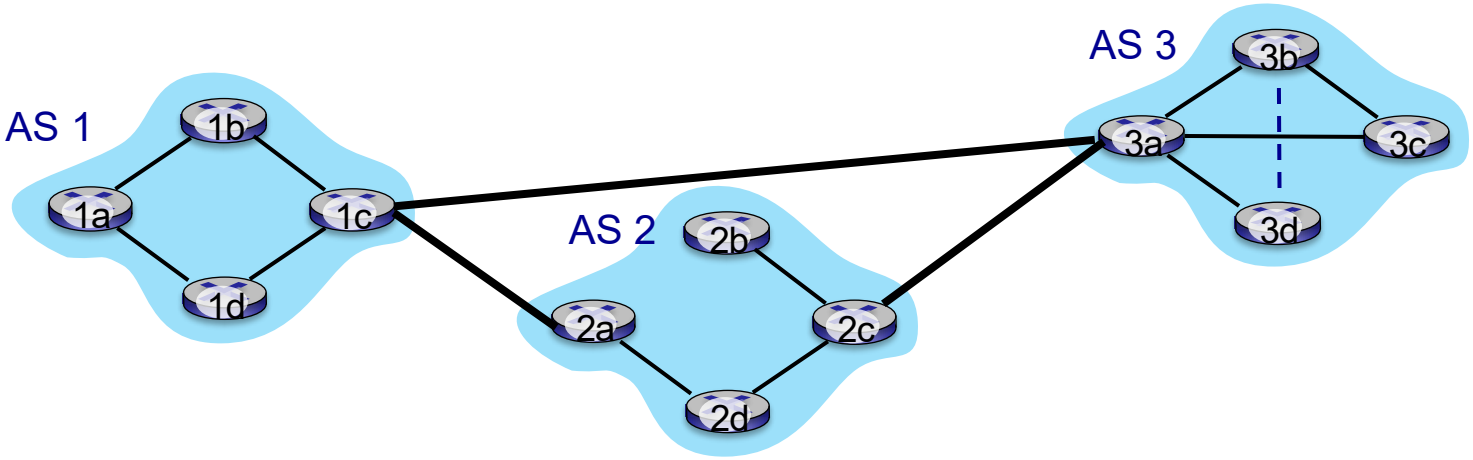
- 2d learns (via iBGP) it can route to X via 2a or 2c
- Hot potato routing: choose local gateway that has least *intra-domain* link weight cost (e.g., 2d chooses 2a (201 vs 263), even though more AS hops to *X*): don't worry about inter-domain cost!

Real BGP routing a little more complex than hot potato (but based on it!)

BGP routing tables in the real world may have over a half million entries

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

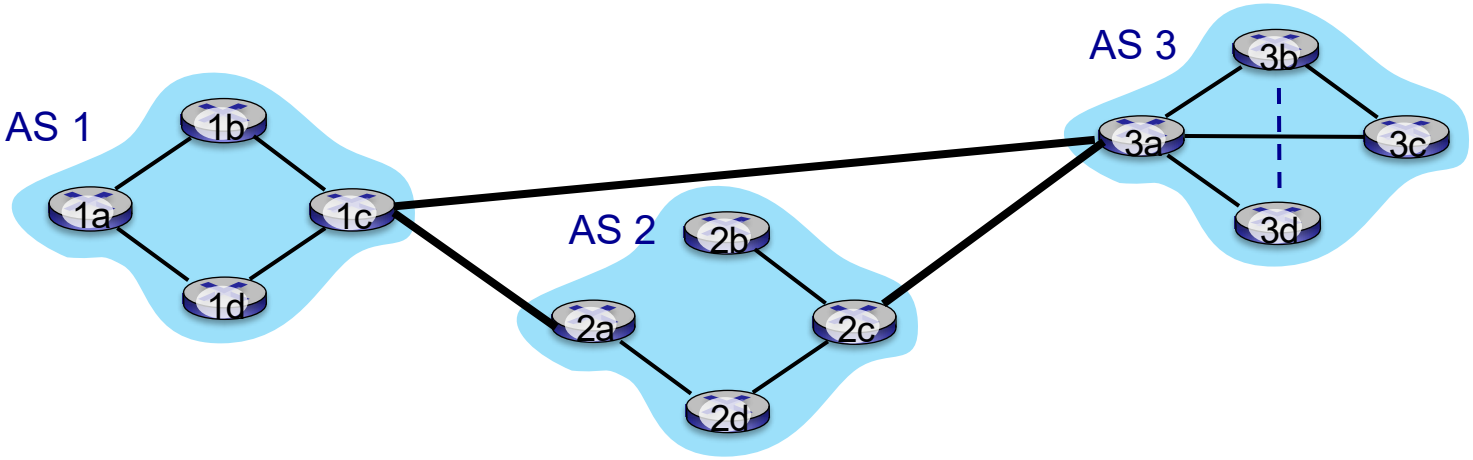
Approximate BGP router entries



Router type	Approx number of forwarding table entries	Notes
Home/Enterprise	10 – 500	Internal home network(s), plus default route to ISP for everything else
Regional ISP	1,000 – 100,000	Routes to each client, plus peer ISPs and Tier 1 ISPs
Tier 1 ISP/Backbone	~1,000,000 (all of them)	Every network prefix in the global routing table

https://bgp.potaroo.net/as2.0/bgp-active.html

The VM has only a few entries




```
$ ip route
default via 192.168.159.2 dev eth0 proto dhcp src 192.168.159.129 metric 100
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
192.168.159.0/24 dev eth0 proto kernel scope link src 192.168.159.129 metric 100
192.168.159.2 dev eth0 proto dhcp scope link src 192.168.159.129 metric 100
```

Everything not bound for 172.17.0.0/16 (docker) or 192.168.159.0/24 (VM network), send to next hop router at 192.168.159.2 (how the VM gets to the Internet)

https://bgp.potaroo.net/as2.0/bgp-active.html

Why use different Intra- and Inter-AS routing?

Policy:

- Inter-AS: admin wants control over how its traffic routed, who routes through its network
- Intra-AS: single admin, so policy less of an issue

Scale:

- Hierarchical routing saves table size, reduced update traffic

Performance:

- Intra-AS: can focus on performance
- Inter-AS: policy dominates over performance

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Putting it all together



Contact your ISP

- Your company will install a gateway router that connects to the ISP's gateway router
- ISP will give you an IP address range (e.g., a /24 range = 254 hosts)
- ISP's AS propagates via BGP that your network exists

- Set up servers, give each an IP from your address range
  - Web server
  - Mail server
  - DNS name server
- Give your gateway router one of your IP addresses
- Give your servers addresses from your address range
- OSPF calculates routes between your hosts (gateway and servers)

Ignoring NAT for simplicity!

Get a domain name

- Give the Registrar the IP address of your web and mail servers
- Registrar will make DNS A and MX entries

Now people can find your web server and email

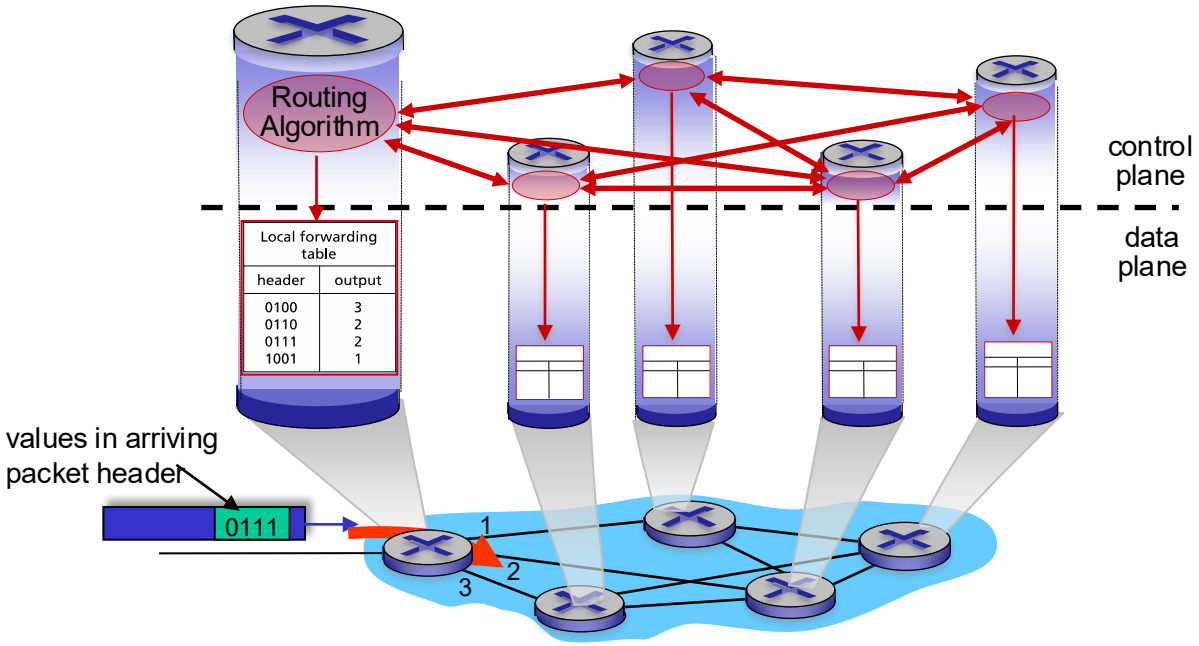
Agenda

1. Routing
2. OSPF (routing within an AS)
3. BGP (routing between AS's)
4. SDN control plane
5. Exercise

Recall: traditional router control plane calculated in each router

Traditional routing

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



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Software-Defined Networking (SDN) control plane

Logically monolithic, usually implemented with many redundant components

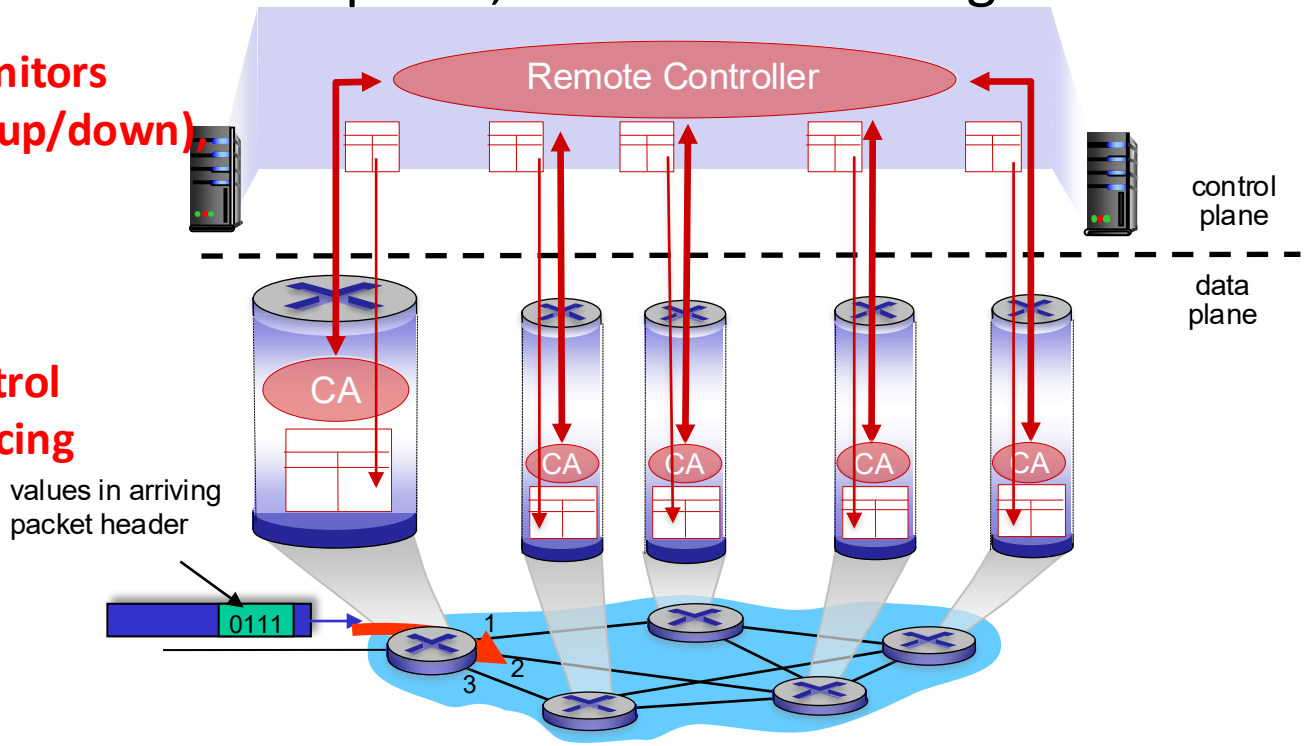
Software Defined Networking (SDN)

Remote controller computes, installs forwarding tables in routers

Controller monitors state of links (up/down) congestion

Can also do:

- **Routing**
- **Access control**
- **Load balancing**



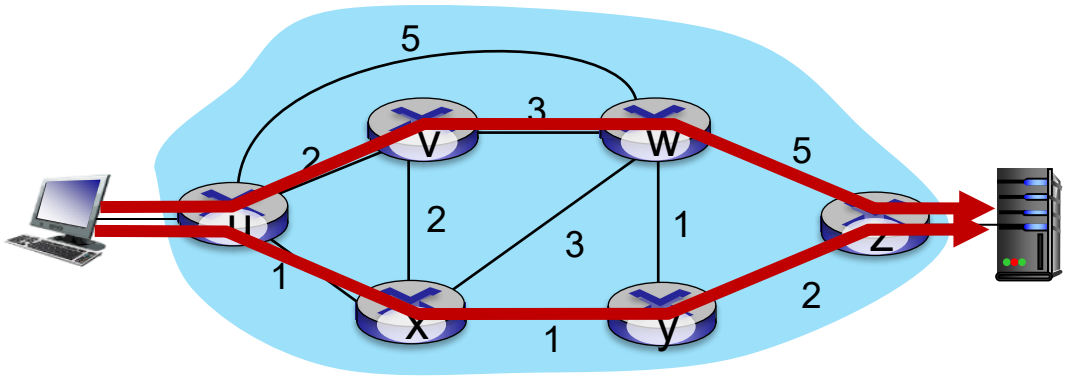
Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Software defined networking (SDN)

***Why* a *logically centralized* control plane?**

- Easier network management: avoid router misconfigurations, greater flexibility of traffic flows
- Table-based forwarding allows "programming" routers
  - Centralized "programming" easier: compute tables centrally and distribute
- Open (non-proprietary) implementation of control plane
  - Foster innovation: let 1000 flowers bloom

Traffic engineering: difficult with traditional routing



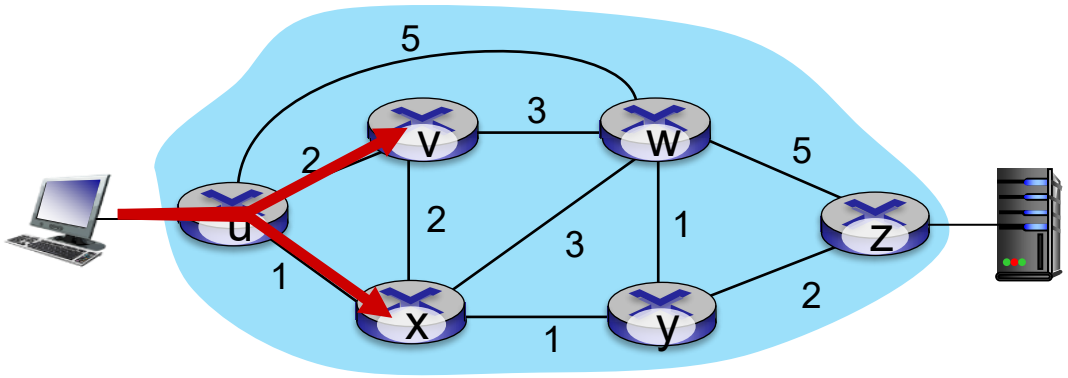
Q: What if network operator wants u-to-z traffic to flow along *uvwz*, rather than *uxyz*?

A: Need to re-define link weights so traffic routing algorithm computes routes accordingly (or need a new routing algorithm)!

Link weights are only control "knobs": not much control!

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Traffic engineering: difficult with traditional routing

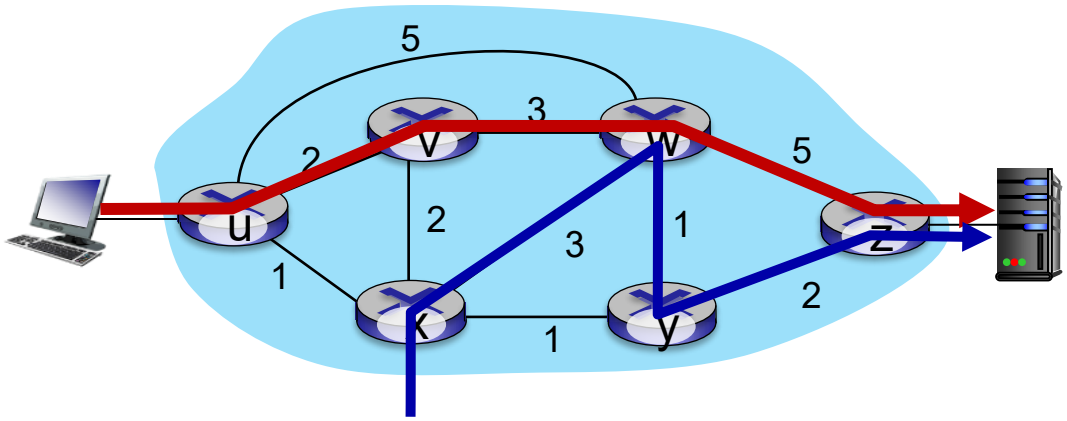


Q: What if network operator wants to split u-to-z traffic along uvwz *and* uxyz (load balancing)?

A: Can't do it (or need a new routing algorithm)

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Traffic engineering: difficult with traditional routing



Q: What if w wants to route blue and red traffic differently from w to z?

A: Can't do it (with destination-based forwarding)

We learned previously that generalized forwarding and SDN can be used to achieve *any* routing desired

Can look at any header field in any layer

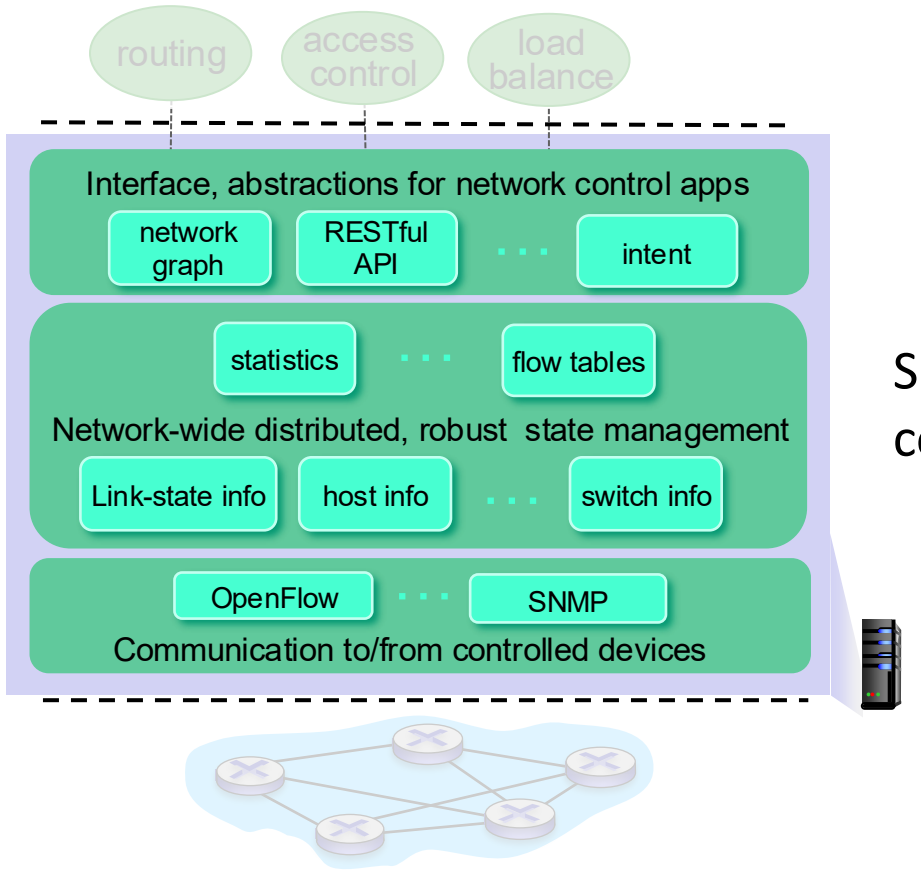
Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Components of SDN controller

Interface layer to network control apps: abstractions API

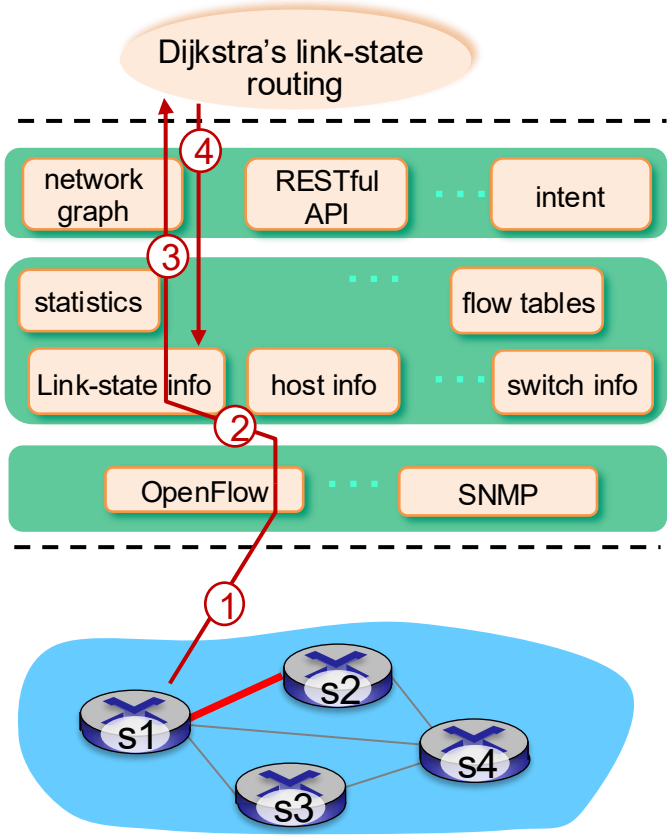
Network-wide state management : state of networks links, switches, services: a *distributed database*

Communication: communicate between SDN controller and controlled switches



Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

SDN: control/data plane interaction example



1 S1, experiencing link failure uses OpenFlow port status message to notify controller

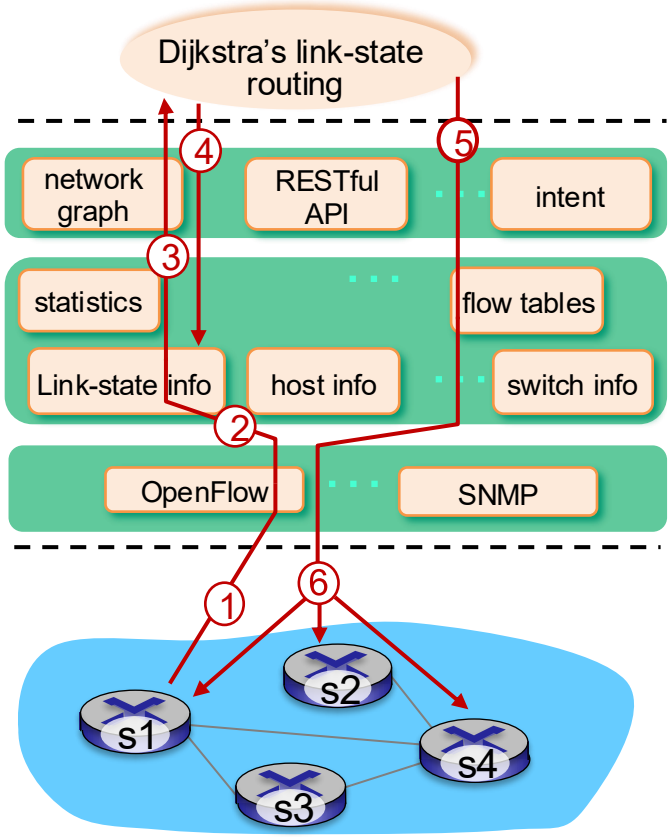
2 SDN controller receives OpenFlow message, updates link status info

3 Dijkstra's routing algorithm application has previously registered to be called whenever link status changes. It is called.

4 Dijkstra's routing algorithm accesses network graph info, link state info in controller, computes new routes

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

SDN: control/data plane interaction example



5 Link state routing app interacts with flow-table-computation component in SDN controller, which computes new flow tables needed

6 Controller uses OpenFlow to install new tables in switches that need updating

Adapted from Kurose and Ross: Computer Networking: A Top-Down Approach

Agenda

1. Routing
2. OSPF (routing within an AS)
3. BGP (routing between AS's)
4. SDN control plane
5. Exercise

Exercise

Download these files from the course web site for today

- as_explorer.py
- [apple | campus | cloudflare | europe | google | japan].txt

The .txt files contain the output from traceroute runs:

traceroute -n 8.8.8.8 > google.txt
traceroute -n 1.1.1.1 > cloudflare.txt
traceroute -n dartmouth.edu > campus.txt
traceroute -n www.u-tokyo.ac.jp > japan.txt
traceroute -n www.apple.com > apple.txt
traceroute -n www.ox.ac.uk > europe.txt
(note: -n prints numeric IP addresses not hostnames)

Steps:

1. Run a traceroute to your favorite website and output to a .txt file
2. Run as_explorer.py which will:
   - Examine all .txt files in its directory
   - Look up the AS that controls each IP address using Team Cymru (falling back RADb registry if not found)
   - Outputs details about the ASes along the route

Exercise

1. **How many router hops did your trace take, and how many ASes did it cross?** Why is the second number so much smaller?

2. **Who owns each AS on your path, and what kind of organization is it?** Label each as campus, research network, ISP/transit provider, or content provider.

3. **Did the path end in the AS of the company that runs your site?** If not, whose AS was it?

4. **Which ASes do all of your group's paths go through?** If one of them went down, what would happen to your access?

5. **Where do your paths split, and which organization makes that choice?** What information does that AS have to make it?

6. **These AS paths depend on agreements between organizations, not on which route is physically shortest. Look for an example in your output that supports that.**