The Network Layer: NAT, IPv6, Routing Algorithms

CS 352, Lecture 15, Spring 2020

http://www.cs.rutgers.edu/~sn624/352

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Course announcements

- Take-home mid-term this weekend
  - Practice problems and review mid-term released soon
- Open-book and open-notes, but not open-Internet
  - Not permitted to search the Internet for answers
  - No collaboration permitted; all answers and work must be your own
  - Calculators are allowed
- This class will follow absolute grading with generous thresholds
  - i.e., no curve, you're only competing with yourself
  - We trust you to do the right thing

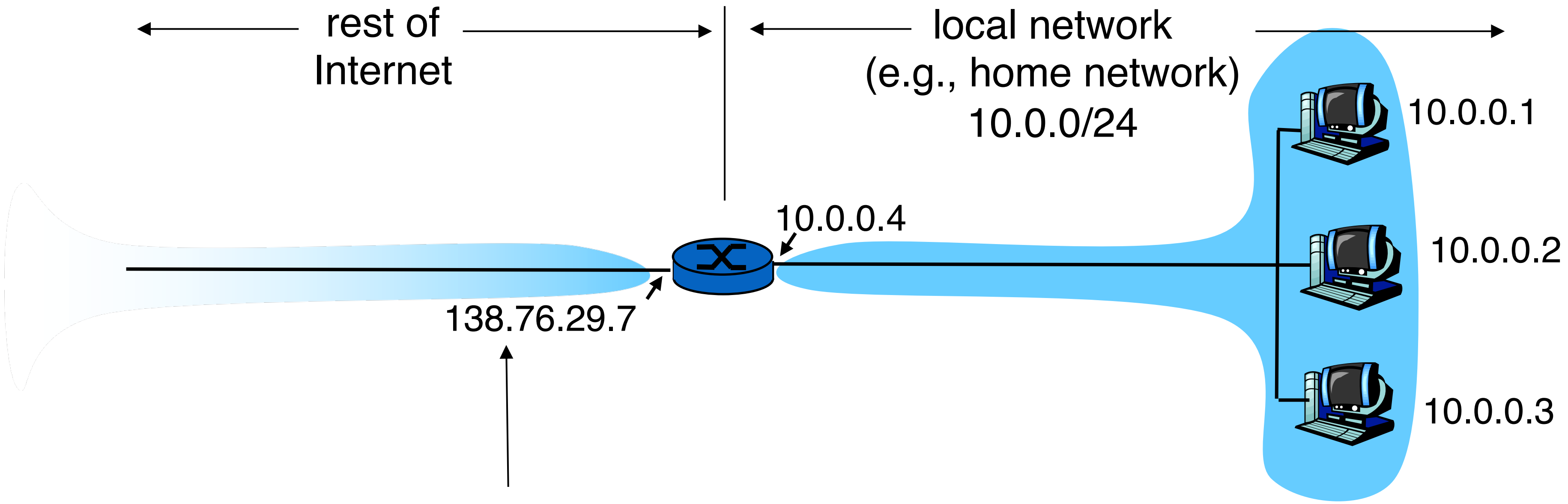
Review of concepts

- Internet Protocol:
  - Headers: src/dst, upper layer, fragmentation, time to live
- Dynamic host configuration protocol (DHCP):
  - How does an endpoint get its IP address?
  - Broadcast-based: endpoint asks entire network for answers
  - DHCP server returns IP address, subnet mask, local DNS server address, gateway router address
- Internet Control Message Protocol (ICMP):
  - Network troubleshooting protocol
  - Ping: reachability using ICMP echo request and reply
  - Traceroute: router-level path using ICMP time-exceeded messages and incrementing TTL

Network Address Translation (NAT)

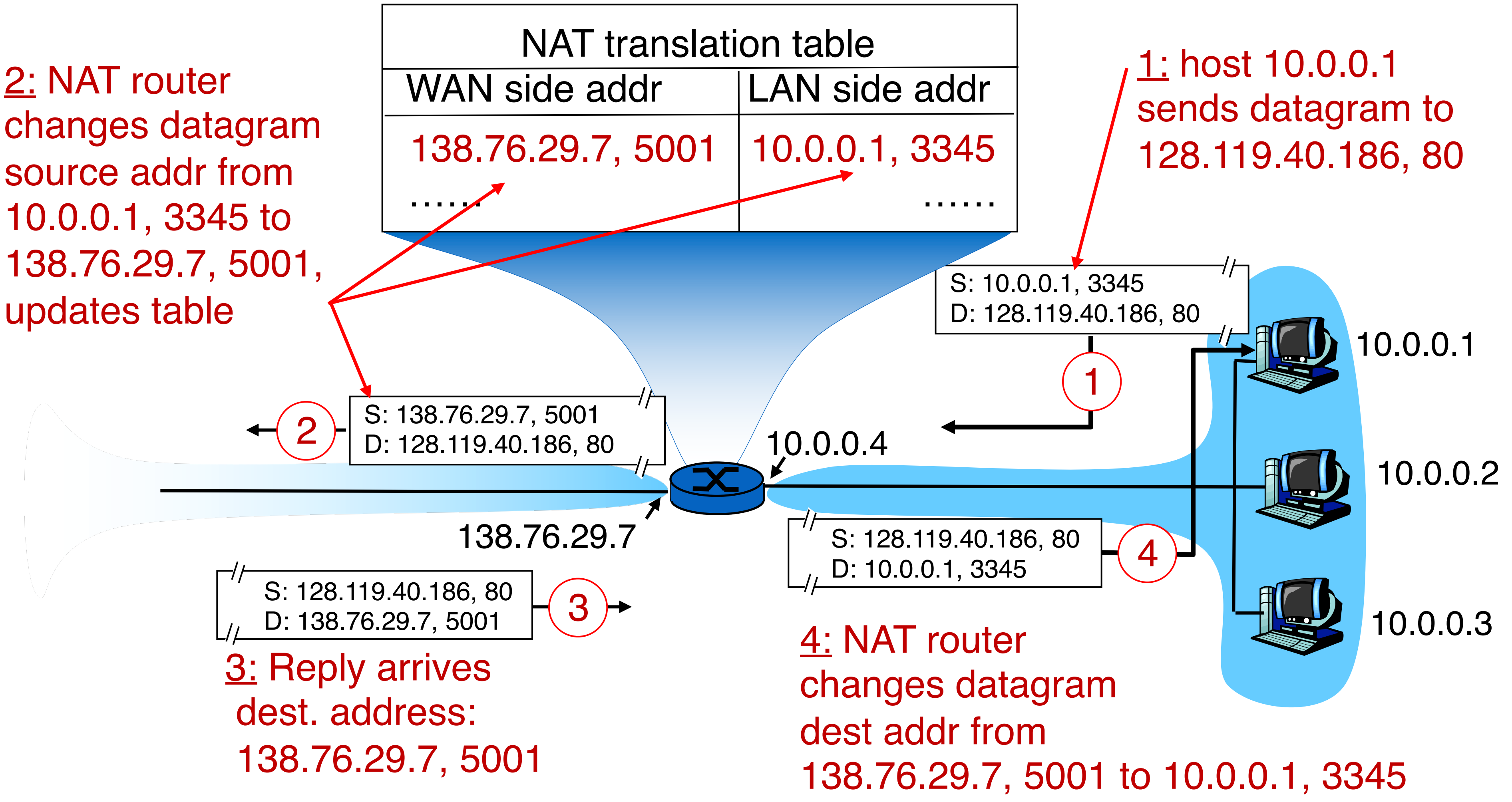
How do you survive in a world where names are scarce?

NAT: Network Address Translation



All datagrams leaving local network have same source IP address: 138.76.29.7, with different source IP port numbers

NAT: Network Address Translation



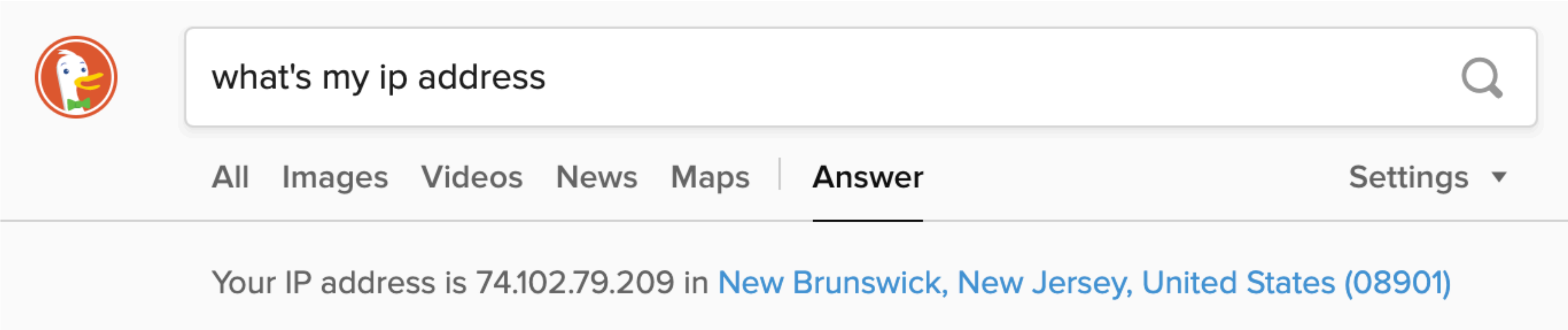
NAT: Network Address Translation

- Features: local network uses just one IP address as far as outside world is concerned:
  - range of addresses not needed from ISP: just one IP address for all devices
  - can change addresses of devices in local network without notifying outside world
  - can change ISP without changing addresses of devices in local network
  - Devices inside local network not explicitly addressable
  - Devices inside local network invisible to the outside world (a security plus) unless the device inside connects first.

Your home WiFi router implements NAT-ting.
If you're at home, you're almost surely behind a NAT gateway right now.

The impact of NATs

```
[flow:352-S20]$  ifconfig en0
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether f0:18:98:1c:fc:36
        inet6 fe80::1036:7dea:82ee:e868%en0 prefixlen 64 secured scopeid 0xa
        inet 192.168.1.151 netmask 0xffffff00 broadcast 192.168.1.255
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
[flow:352-S20]$
```



NAT: Network Address Translation

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!

- NAT is controversial:
  - Routers should only work upto the network layer, not transport ports!

  - violates “end-to-end argument”
    - NAT must be taken into account by app designers
    - e.g., P2P applications like skype

  - Purists: address shortage should instead be solved by IPv6

Think about…

- How do the hosts inside the home network get their IP addresses?

- How does your home router get its externally visible IP address?

Poll #1

- The length of the TCP port field is 16 bits. Assume that a NAT router has a single external IP address. In principle, how many TCP connections can the router support between the NAT and the outside world?
  - (a) 1
  - (b) 2^16
  - (c) none of the above

Internet Protocol v6 (IPv6)

Recent Developments: IPv6

- IPv4 has limited address space (32 bits) and is running out of addresses. 32 bits are not enough!

- More devices: phones, watches, your refrigerator(!), …

- Real-time traffic and mobile users are also becoming more common

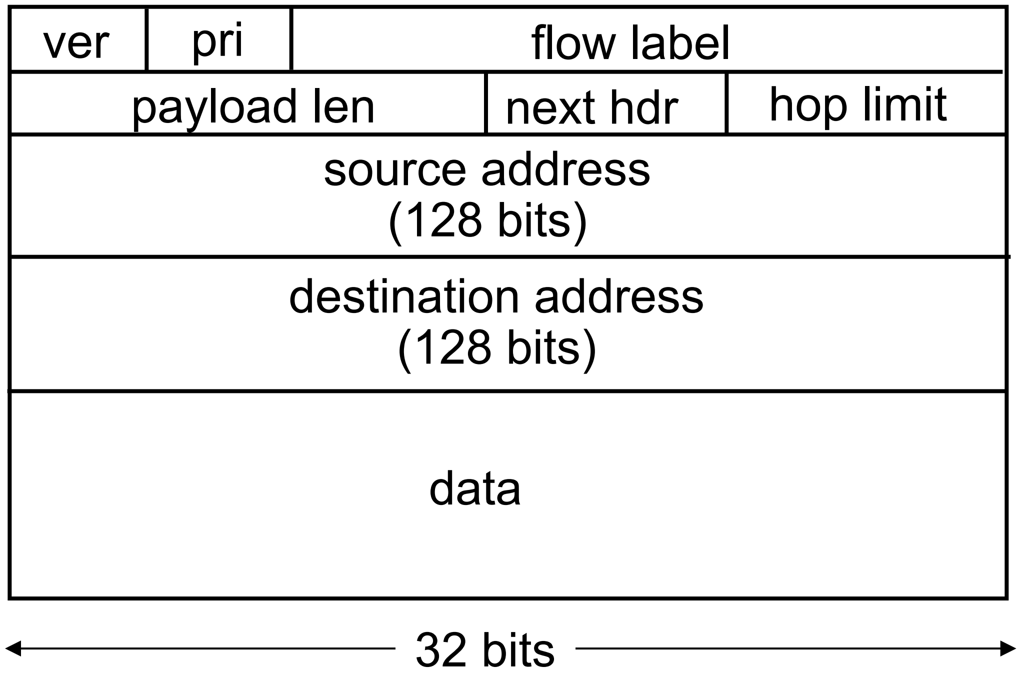
IP version 6

IPv6: Main changes from IPv4

- Large address space:
  - 128-bit addresses (16 bytes)
  - Allows up to 340,282,366,920,938,463,463,374,607,431,768,211,456 unique addresses (3.4×10^{38})
- Fixed length headers (40 bytes)
  - Improves the speed of packet processing in routers
- IPv6 “options” processing happens through a separate mechanism

IPv6 datagram format

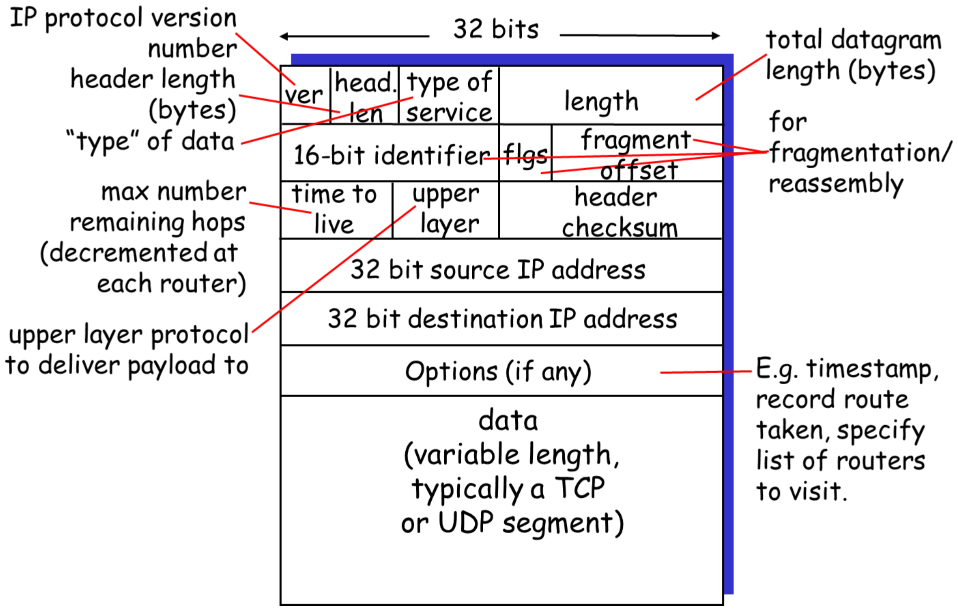
priority: identify priority among datagrams in flow
flow Label: identify datagrams in same "flow"
(concept of "flow" left undefined)
next header: identify upper layer protocol for data

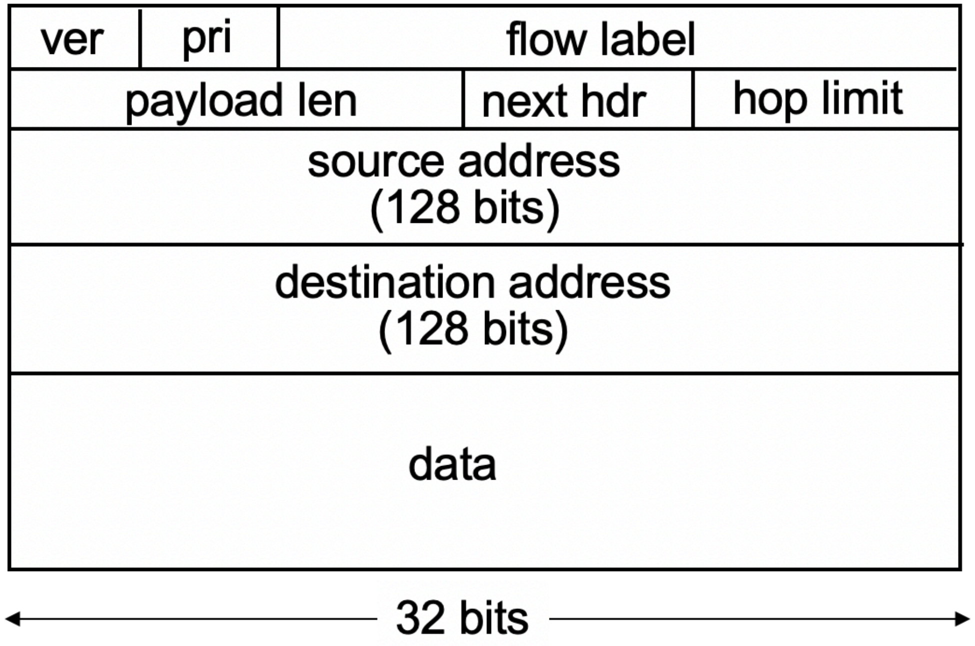


Other changes from IPv4

- *checksum*: removed entirely to reduce processing time at each hop
- *options:* allowed, but outside of header, indicated by “Next Header” field
- *ICMPv6:* new version of ICMP
  - additional message types, e.g. “Packet Too Big”
  - multicast group management functions

IPv4 vs IPv6: Can you tell the differences?





IPv6 Flows

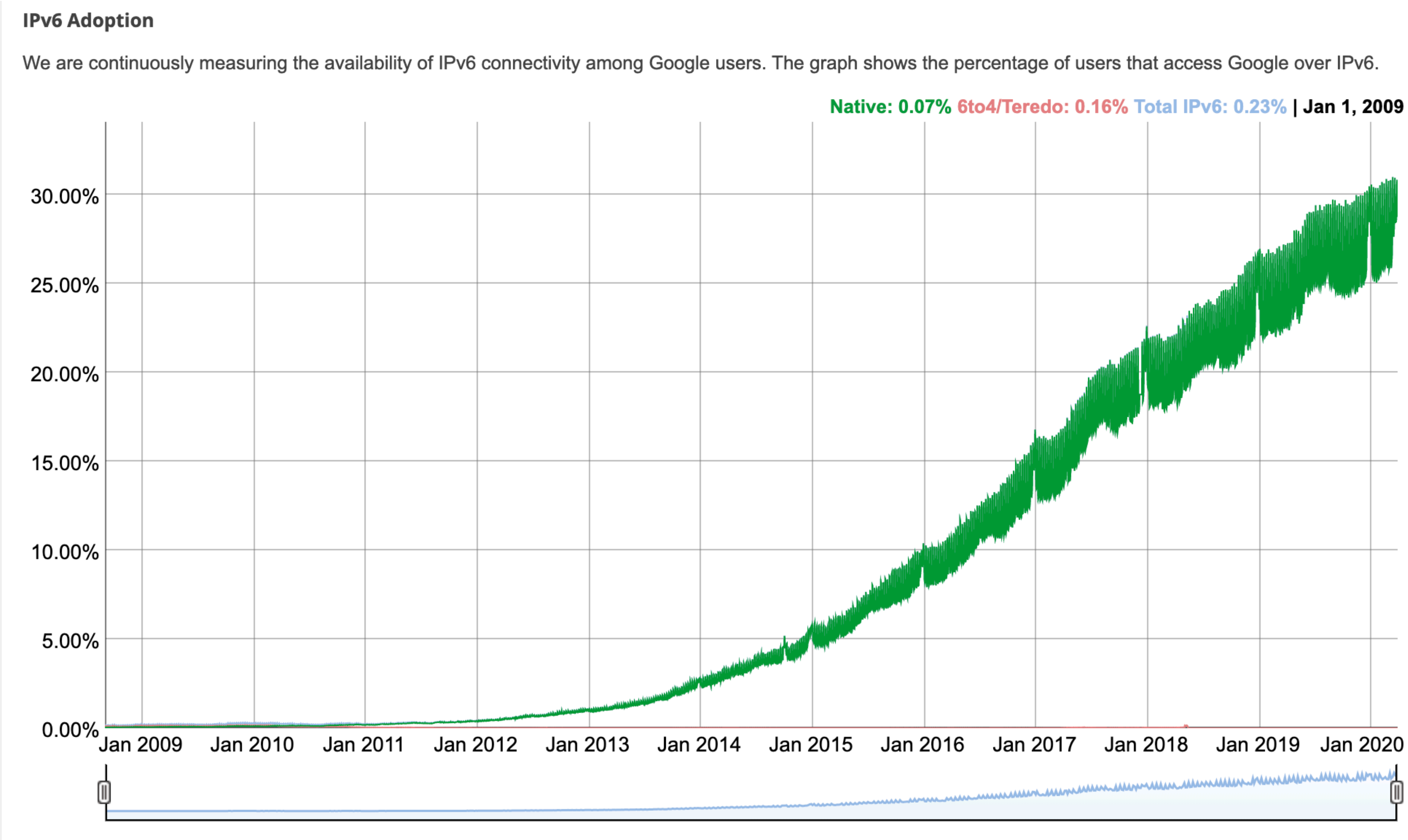
- Support for "flows"
  - Flows help support real-time service in the Internet
  - A "flow" is a number in the IPv6 header that can be used by routers to see which packets belong to the same stream
  - Guarantees can then be assigned to certain flows
  - Example:
    - Packets from flow 10 (e.g., audio call) should receive rapid delivery
    - Packets from flow 12 (e.g., web traffic) should receive reliable delivery

IPv6 Addresses

- Classless addressing/routing (similar to CIDR)

- Notation: xx:xx:xx:xx:xx:xx:xx:xx
  - x = 4-bit hex number
  - contiguous 0s are compressed: 47CD::A456:0124
  - IPv6 compatible IPv4 address: ::128.64.18.87
    - First 96 bits are 0
  - Global unicast addresses start with 001….
  - 2000::/3 prefix

IPv6 adoption



IPv6: Adoption

- Google: ~1/3 of clients access services via IPv6 (Apr 2020)
- *Long (long!) time for deployment, use*
  - 20 years and counting!
- Think of application-level changes in last 20 years: WWW, Facebook, Skype, video streaming, AR, telesurgery,…
  - *Why?*

Poll #2

- What’s the primary difference between IPv4 and IPv6?
  - (a) Larger address space
  - (b) Fixed-length, fewer headers by default
  - (c) No fragmentation
  - (d) All of the above

Routing Protocols

How do we design a “Google Maps” navigator for the Internet?

Network-layer functions

Recall: two network-layer functions:

- *forwarding:* move packets from router's input to appropriate router output

data plane

- *routing:* determine route taken by packets from source to destination

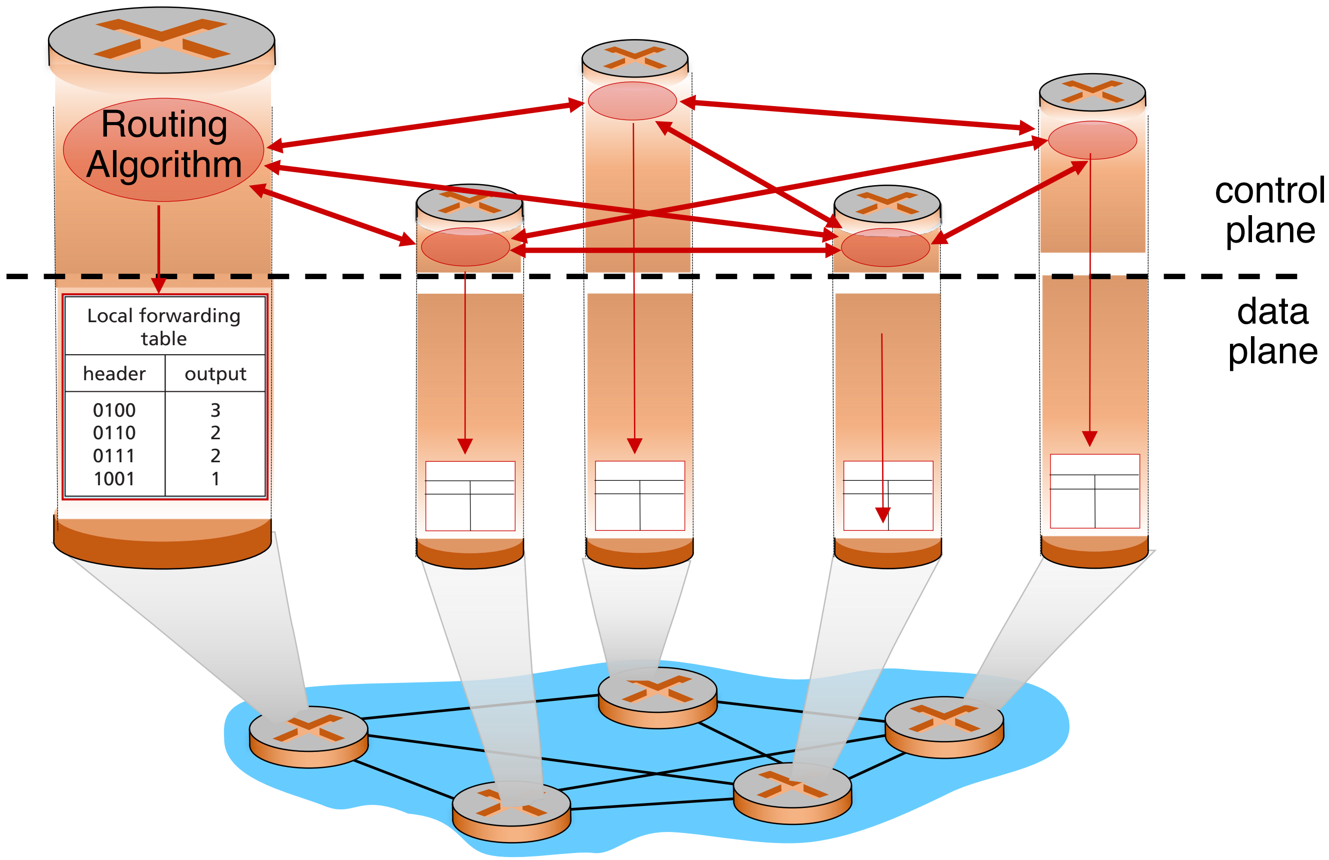
control plane

Two approaches to structuring network control plane:

- per-router control (traditional)
- logically centralized control (software defined networking)

Per-router control plane

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

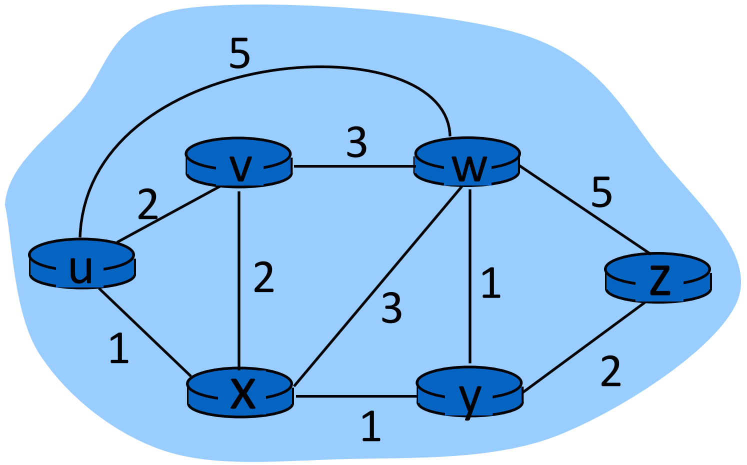


Routing protocols

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

- path: sequence of routers packets will traverse in going from given initial source host to given final destination host
- "good": least "cost", "fastest", "least congested"
- routing: a "top-10" networking challenge!

Graph abstraction



Graph: G = (N,E)

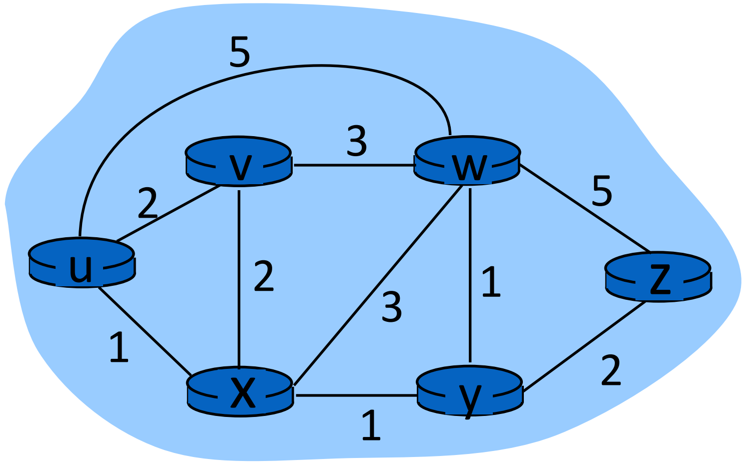
N = set of routers = { u, v, w, x, y, z }

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

Remark: Graph abstraction is useful in other network contexts

Example: 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')

  - e.g., $c(w,z) = 5$

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

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

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

Routing algorithm: find "good" paths from source to destination router.

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

Poll #3

- What is a good candidate to use to fix edge weights in a network graph representation?
  - (a) 1/link rate
  - (b) round-trip time
  - (c) congestion
  - (d) any of the above

Link State Algorithms

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 dest.'s

Notation:

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

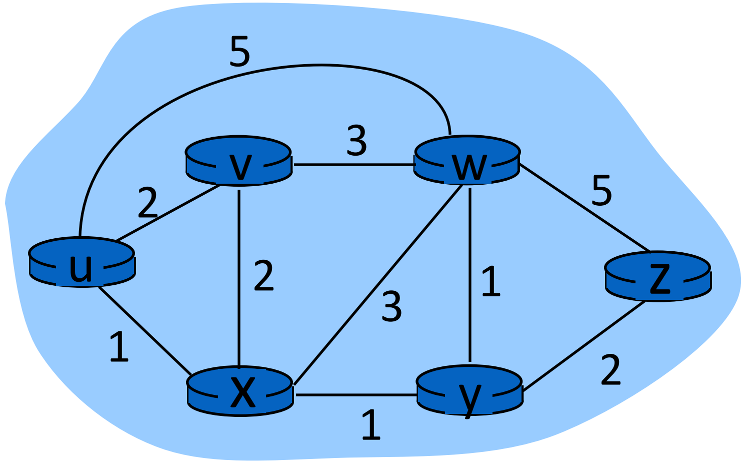
Dijsktra’s Algorithm

```
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'
```

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	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					



Poll #4

- Link-state information of a router is sent to all routers before computing shortest paths in a link-state protocol.
  - (a) true
  - (b) false

Poll #5

- Link-state protocols don’t compute shortest paths when there are cycles in the network topology.
  - (a) true
  - (b) false

Distance Vector Algorithms

Distance Vector Algorithm

- $D_x(y)$ = estimate of least cost from x to y
- Distance vector: $\mathbf{D_x} = [D_x(y): y \in N]$
- Node x knows cost to each neighbor v: $c(x,v)$
- Node x maintains $\mathbf{D_x}$
- Node x also maintains its neighbors' distance vectors
  - For each neighbor v, x maintains
    $\mathbf{D_v} = [D_v(y): y \in N]$

Distance vector algorithm

Bellman-Ford equation (dynamic programming)

let

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

then

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

cost from neighbor v to destination y

cost to neighbor v

min taken over all neighbors v of x

Distance vector algorithm

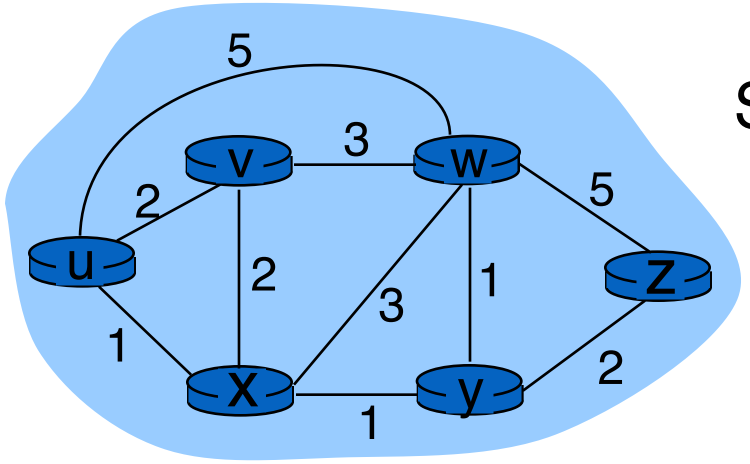
Basic idea:

- Each node periodically sends its own distance vector estimate to neighbors
- When node a node x receives new DV estimate from neighbor, it updates its own DV using Bellman-Ford equation:

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

- Under some conditions, the estimate *$D_x(y)$ converge the actual least cost* $d_x(y)$

Distance vector: example



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

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

Node that achieves minimum is next hop in shortest path ➜ forwarding table

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

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

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

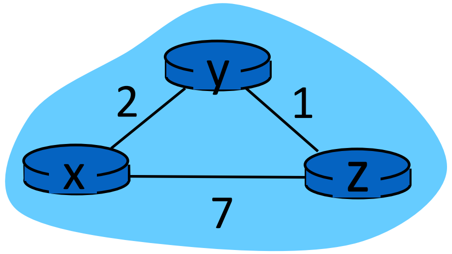
node z table

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

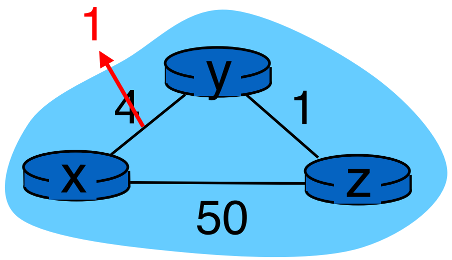
time



Distance vector: link cost changes

link cost changes:

- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors



"good news travels fast"

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

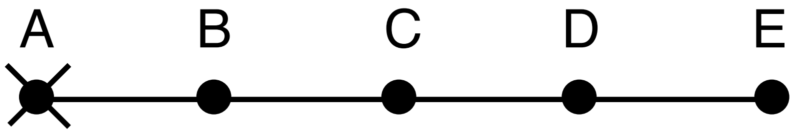
t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.

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*.

Problem: Count-to-Infinity

- With distance vector routing, good news travels fast, but bad news travels slowly
- When a router goes down, it takes can take a really long time before all the other routers become aware of it

Count-to-Infinity



1	2	3	4	Initially
3	2	3	4	After 1 exchange
3	4	3	4	After 2 exchanges
5	4	5	4	After 3 exchanges
5	6	5	6	After 4 exchanges
7	6	7	6	After 5 exchanges

etc… to infinity

Count-to-infinity

"Bad news travels slowly"

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?

Comparison of LS and DV algorithms

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
- ***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 thru network

Poll #6

- Which routing protocol(s) ensure that routers have a view of the network that is consistent with each other at all times?
  - (a) Link-state protocols
  - (b) Distance-vector protocols
  - (c) All of the above
  - (d) None of the above

Routing protocols are widely deployed

- OSPF: a link-state protocol
  - “Open Shortest Path First”
  - IS-IS, nearly identical to OSPF
- RIP: a distance-vector protocol
- LS and DV deployed inside an autonomous system
- Additional tricks to scale the protocols with network size:
  - Areas; Hierarchy
  - “Border routers” that summarize each area
- Next lecture: Routing across autonomous systems