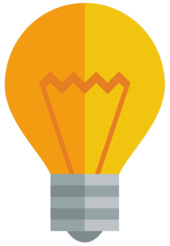
Lesson 06-06: SDN and ICMP

CS 326E Elements of Networking

Mikyung Han

mhan@cs.utexas.edu

Internet Reference Model



Example Protocols		Responsible for
FTP, HTTP, SMTP	Application	application specific needs
TCP, UDP	Transport	process to process data transfer
IP	Network	host to host data transfer across different network
Ethernet, WiFi	Link	data transfer between physically adjacent nodes
802.3 PHY	Physical	bit-by-bit or symbol-by-symbol delivery

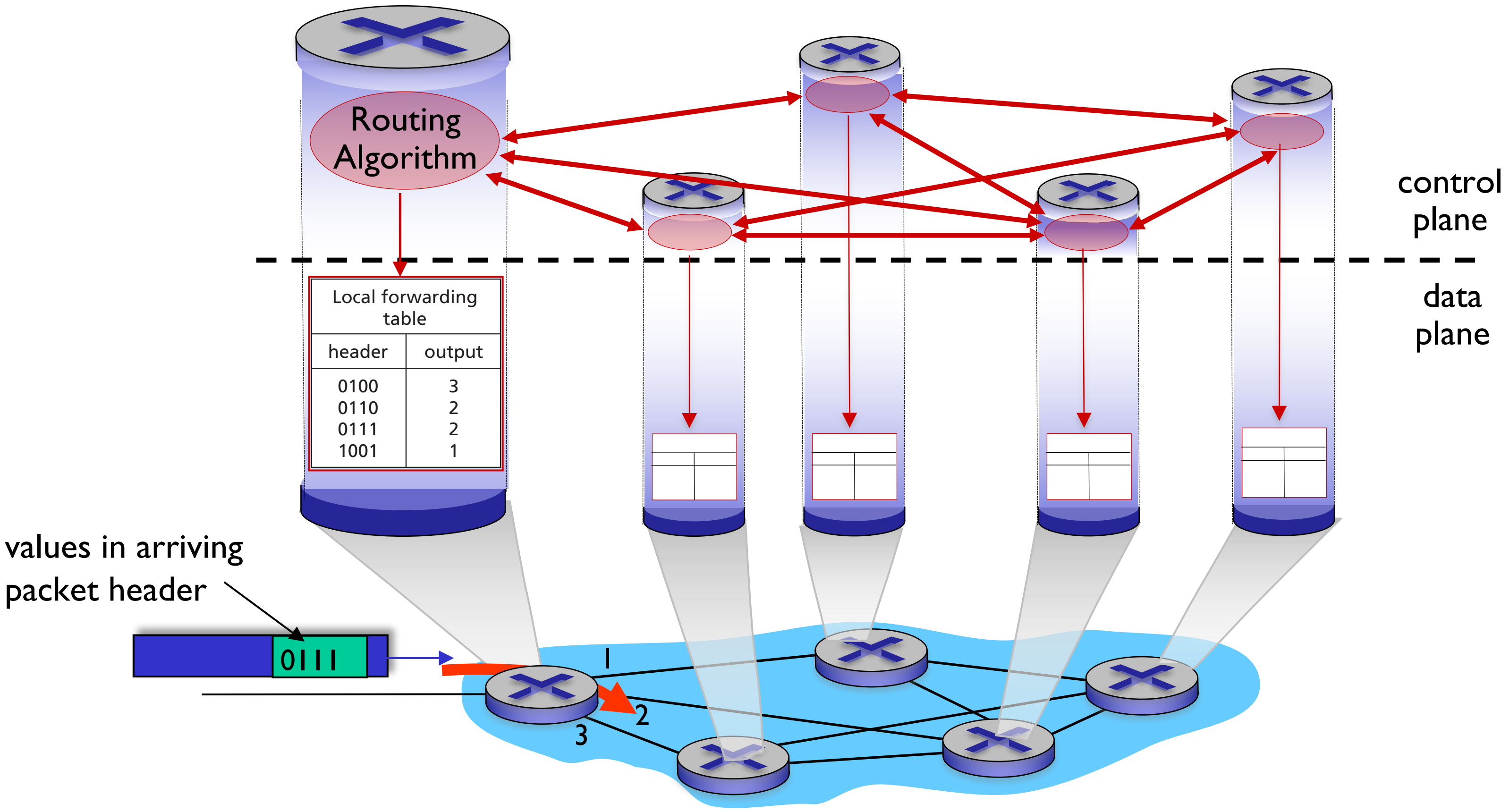
2

Outline

1. Why Software Defined Network?

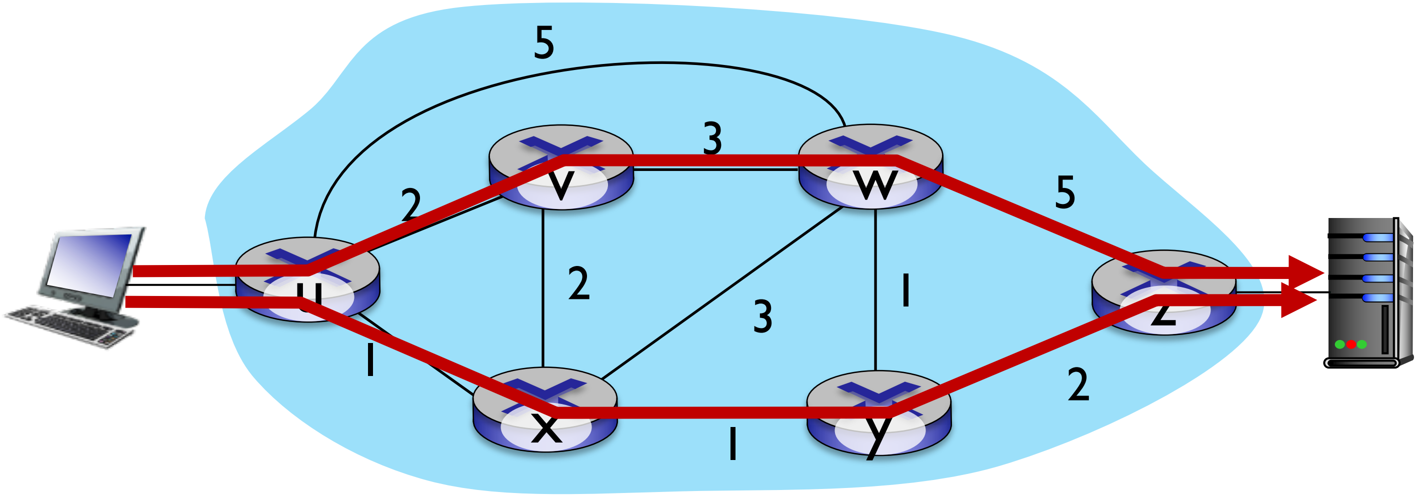
Traditional per-router control plane

each router computes its own forwarding table after exchanging control plane info with other routers



Motivation: What is difficult/impossible in traditional routing?

Difficult to specify a preferred path with traditional routing

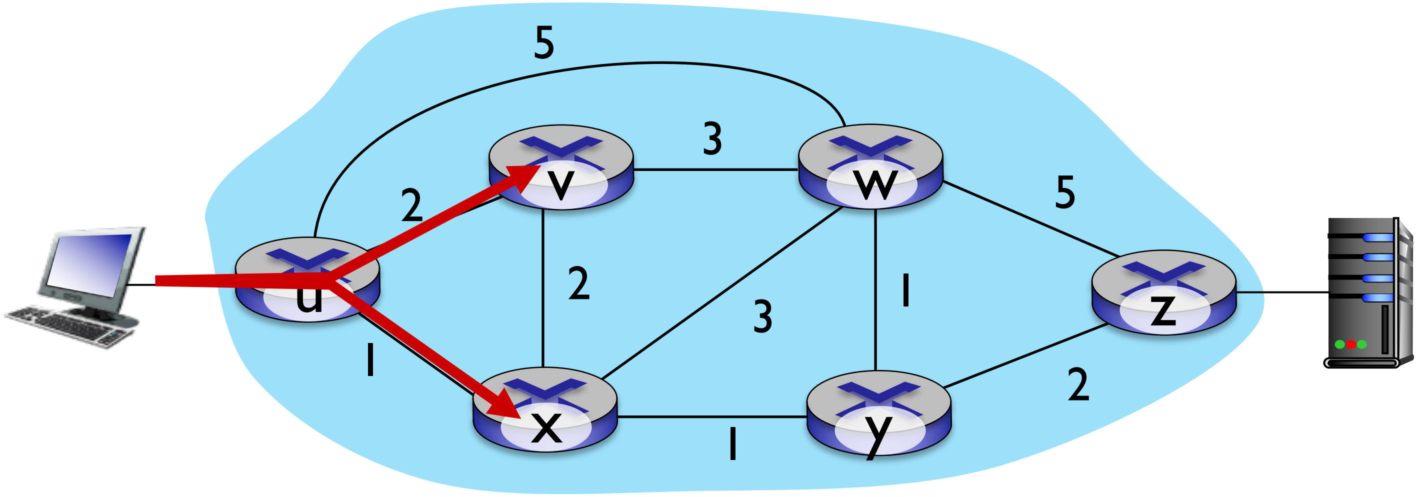


Q: what if network operator wants u-to-z traffic to flow along uvwz, rather than uxyz?

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!

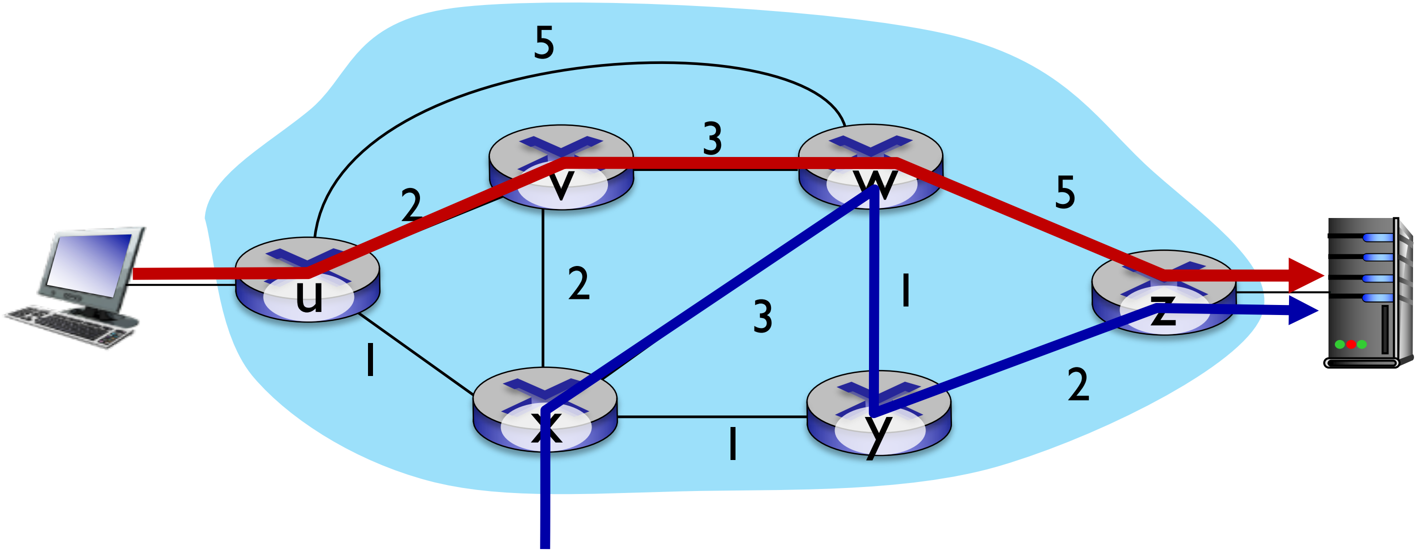
Traditional routing cannot split traffic



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

A: can't do it

With traditional routing, Impossible to use different routes for different flows



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)

GF and SDN can be used to achieve any routing desired!

Imagine your life as a network administrator

You just bought bunch of routers…
The routers come pre-baked with a set of protocols
You cannot directly do what you want!

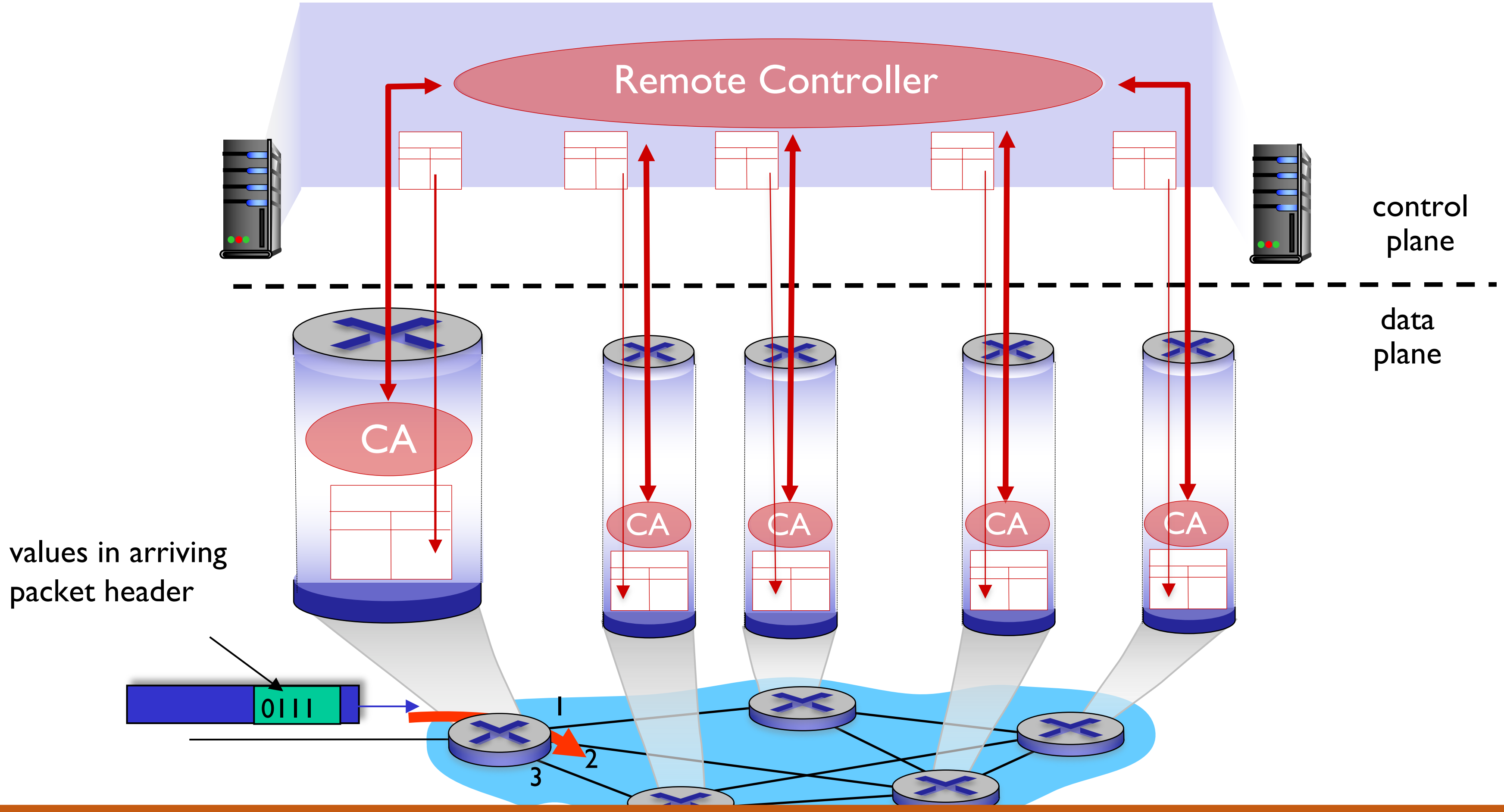
Need more knobs for better traffic engineering and
more dials for better understanding of whole network

SDN moved control plane out of individual routers

Traditionally both control plane and data plane lived in one place (router)

In SDN a controller software defines what network (routers) should do

SDN uses a logically centralized control plane



Remote controller computes, installs forwarding tables in routers

Why logically centralized control plane?

- Easier management
  - Less router misconfigurations
  - Greater flexibility of traffic flows
- Allows “programmable” network
- Unbundling allowed rich innovation
  - No longer ”monolithic” or “vertically integrated” into a single router/switch

SDN analogy: mainframe to PC revolution

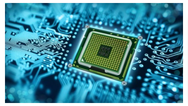
Vertically integrated
Closed, proprietary
Slow innovation
Small industry

App

Open Interface

Windows or Linux or MAC OS

Open Interface



Microprocessor

Horizontal
Open interfaces
Rapid innovation
Huge industry

* Slide courtesy: N. McKeown

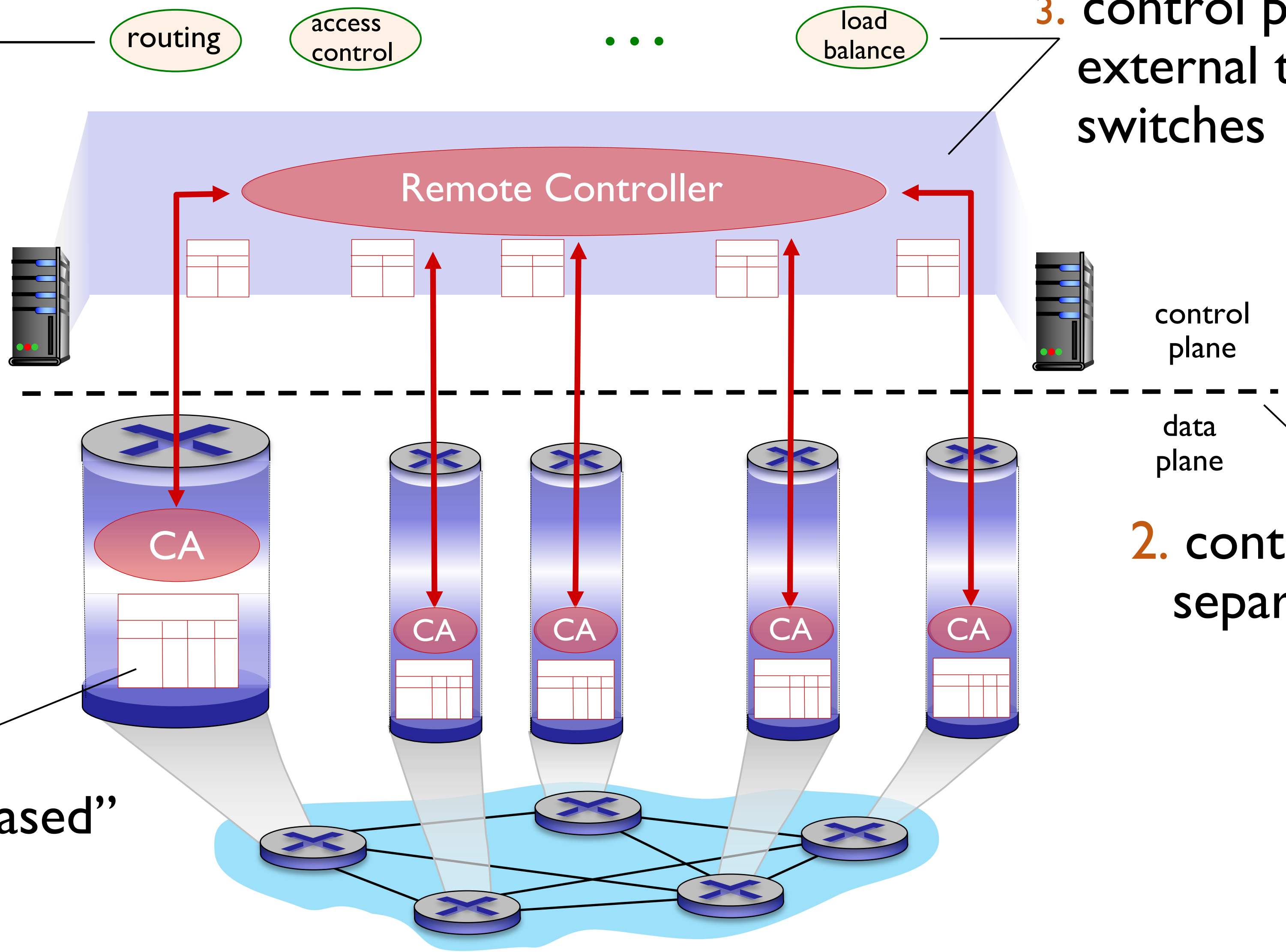
Outline

1. Why SDN?
2. SDN architecture

Software Defined Networking Highlights

4. programmable control applications

3. control plane functions external to data-plane switches



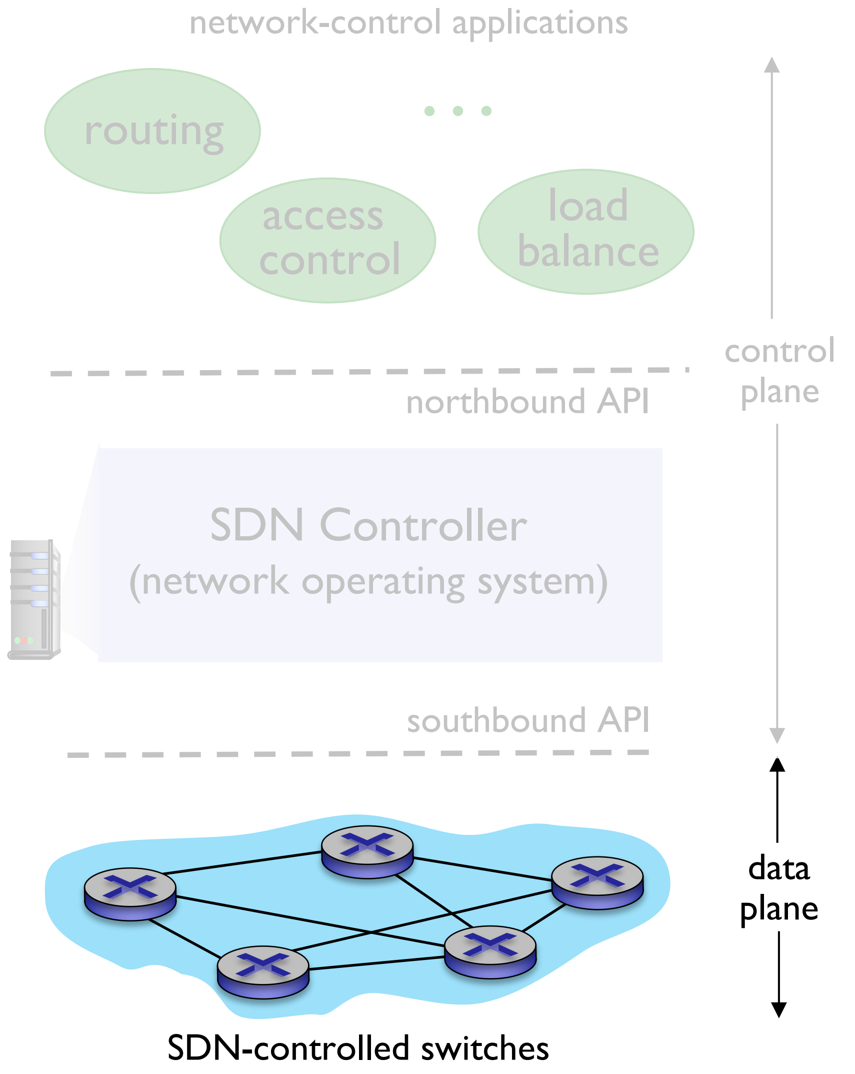
2. control, data plane separation

1: generalized "flow-based" forwarding

What consists of SDN?

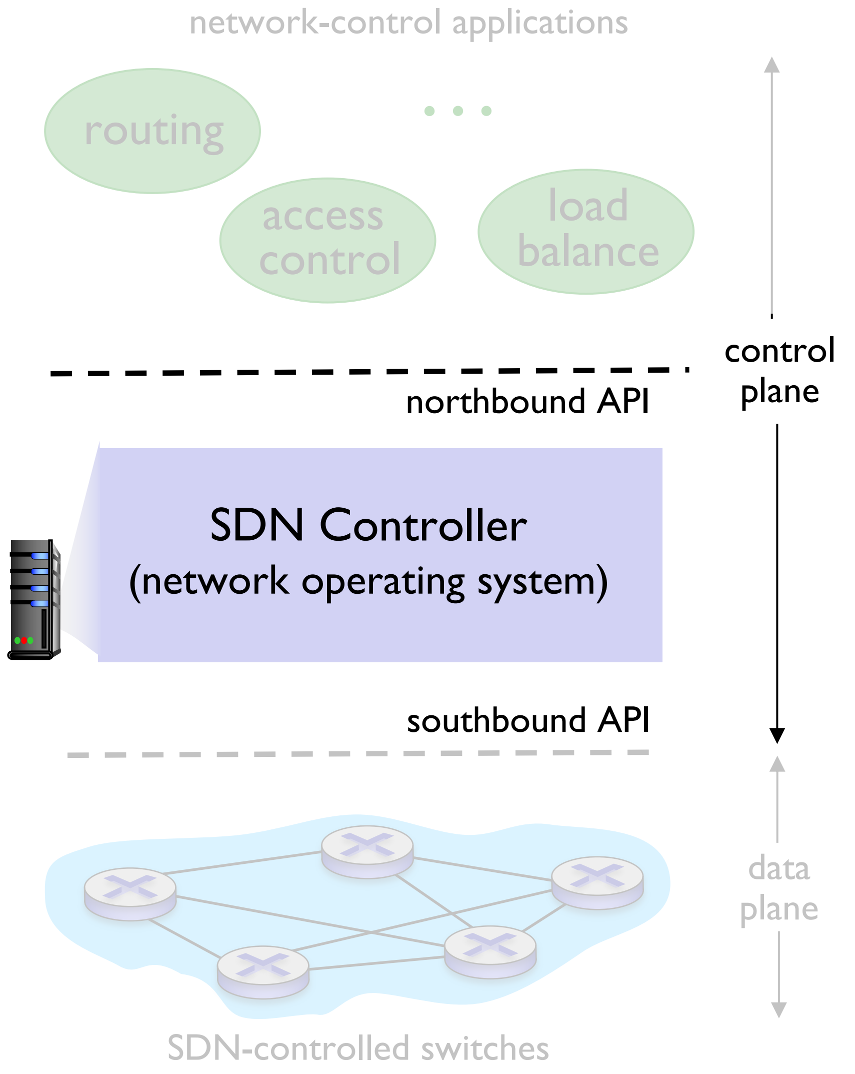
#1 Data-plane switches

- fast, simple, commodity switches implementing generalized data-plane forwarding in hardware
- flow (forwarding) table computed, installed under controller supervision
- Use OpenFlow protocol to communicate with the controller



#2 SDN Controller (Network OS)

- maintain network state information
- implemented as distributed system for performance, scalability, fault-tolerance, robustness
- interacts with network applications “above” via northbound API
- interacts with network switches “below” via southbound API

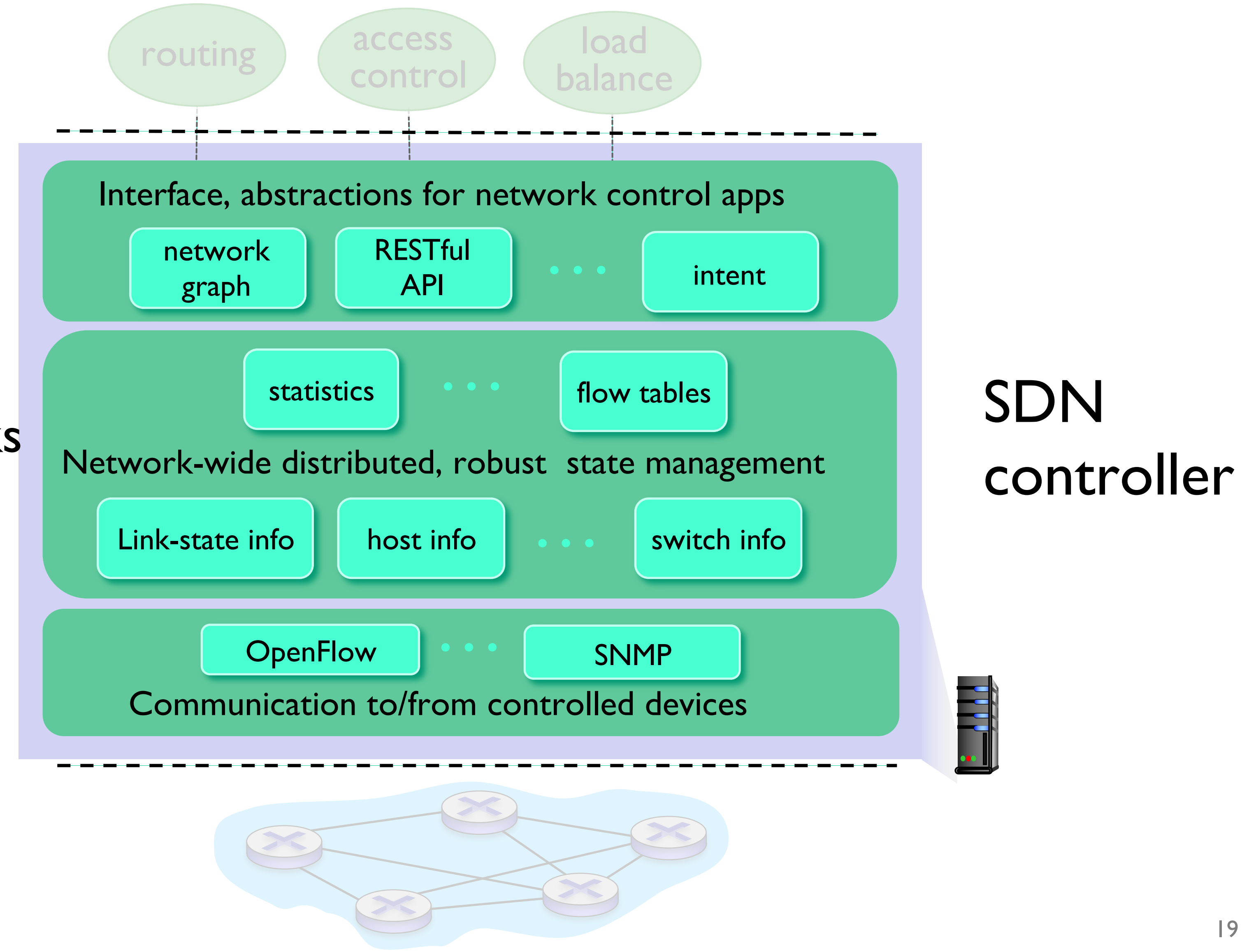


#2 SDN controller details: What is inside?

interface layer to network control apps: abstractions API

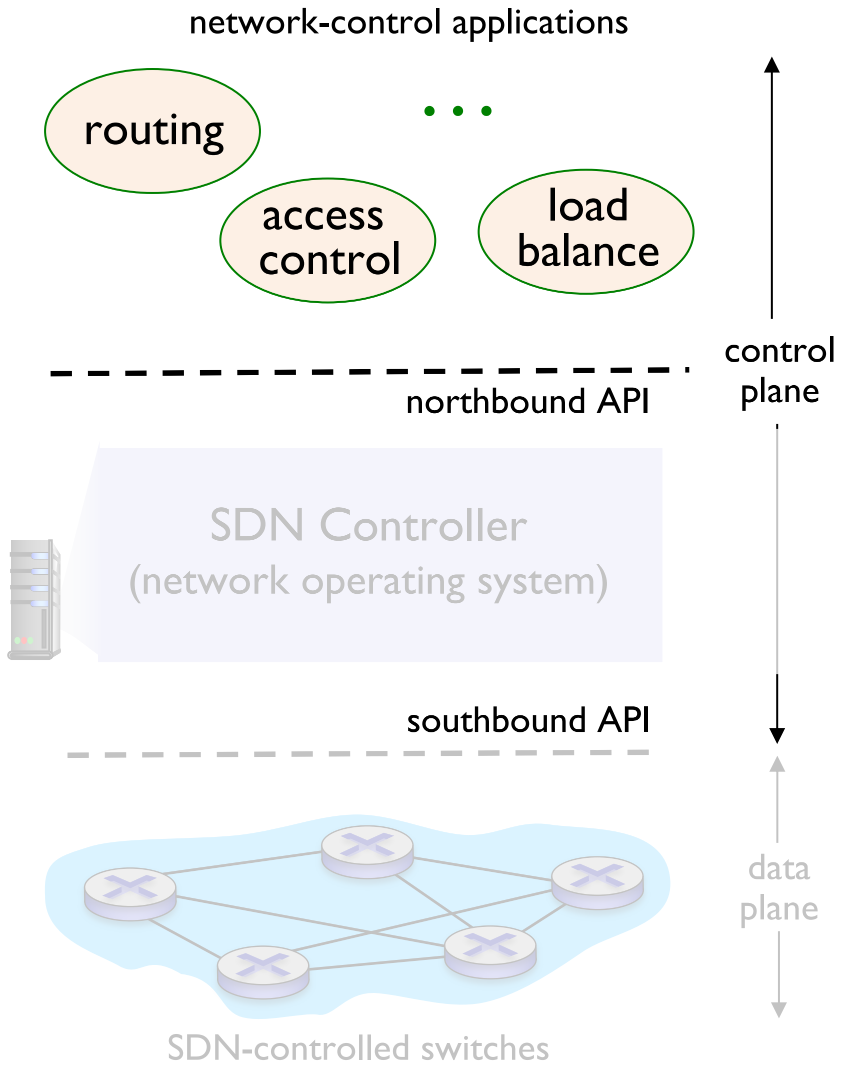
network-wide state management : state of networks links, switches, services: a distributed database

communication: communicate between SDN controller and controlled switches

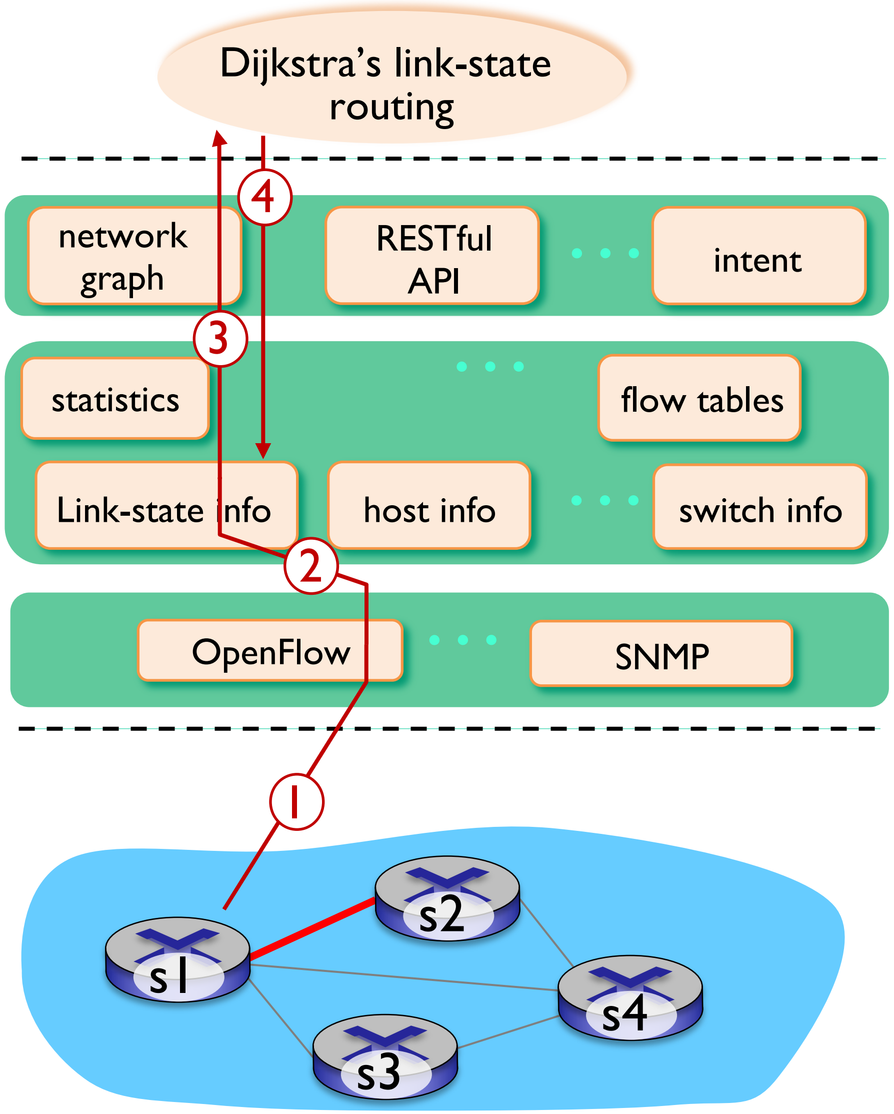


#3 Network-control Applications

- "brains" of control: implement control functions using lower-level services, API provided by SDN controller
- unbundled: can be provided by 3rd party: distinct from routing vendor, or SDN controller

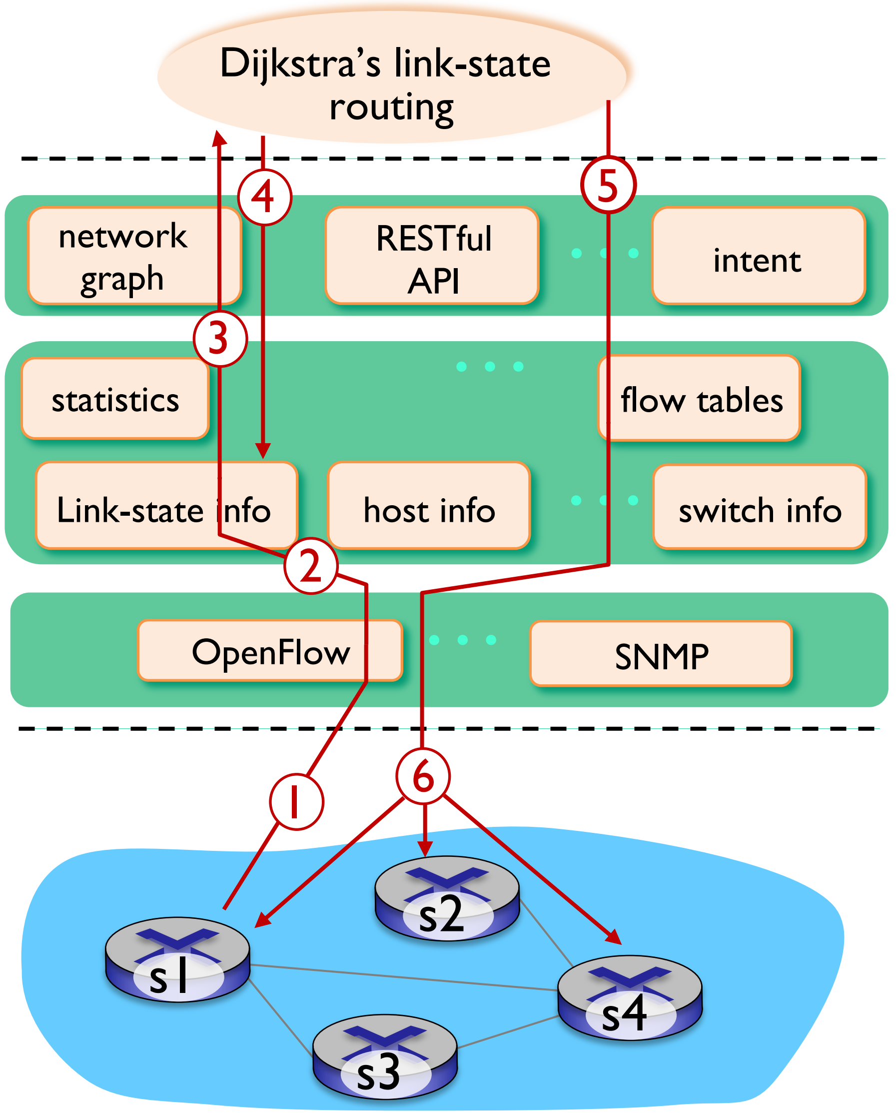


SDN: control/data plane interaction example



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 when ever link status changes. It is called.
4. Dijkstra's routing algorithm access network graph info, link state info in controller, computes new routes

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

Outline

1. Why SDN?
2. SDN architecture
3. ICMP

ICMP is used by network devices to diagnose network communication issues

Mainly to figure out

- Is destination network reachable?
- Is destination host reachable?
- Is destination port reachable?

ICMP is considered network layer protocol

Because

- ICMP helps diagnosing network layer

But

- ICMP is implemented in one layer above network layer
- ICMP messages are carried by IP datagram as part of IP payload

ICMP Packet Format

ICMP header comes after IPv4 and IPv6 packet header.

Type(8 bit)	Code(8 bit)	CheckSum(16 bit)
Extended Header(32 bit)		
Data/Payload(Variable Length)		

ICMPv4 Packet Format

Ping: "are you there?"

Src sends ICMP echo request every n seconds

Dst replies with ICMP echo reply

```
[➜  ~ ping cnn.com
PING cnn.com (151.101.65.67): 56 data bytes
64 bytes from 151.101.65.67: icmp_seq=0 ttl=57 time=4.958 ms
64 bytes from 151.101.65.67: icmp_seq=1 ttl=57 time=4.875 ms
64 bytes from 151.101.65.67: icmp_seq=2 ttl=57 time=4.956 ms
64 bytes from 151.101.65.67: icmp_seq=3 ttl=57 time=11.490 ms
64 bytes from 151.101.65.67: icmp_seq=4 ttl=57 time=11.315 ms
64 bytes from 151.101.65.67: icmp_seq=5 ttl=57 time=5.640 ms
64 bytes from 151.101.65.67: icmp_seq=6 ttl=57 time=11.444 ms
64 bytes from 151.101.65.67: icmp_seq=7 ttl=57 time=12.050 ms
64 bytes from 151.101.65.67: icmp_seq=8 ttl=57 time=14.593 ms
64 bytes from 151.101.65.67: icmp_seq=9 ttl=57 time=11.237 ms
64 bytes from 151.101.65.67: icmp_seq=10 ttl=57 time=5.606 ms
64 bytes from 151.101.65.67: icmp_seq=11 ttl=57 time=5.181 ms
64 bytes from 151.101.65.67: icmp_seq=12 ttl=57 time=11.252 ms
64 bytes from 151.101.65.67: icmp_seq=13 ttl=57 time=11.359 ms
64 bytes from 151.101.65.67: icmp_seq=14 ttl=57 time=11.343 ms
^C
--- cnn.com ping statistics ---
15 packets transmitted, 15 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 4.875/9.153/14.593/3.327 ms
➜  ~
```

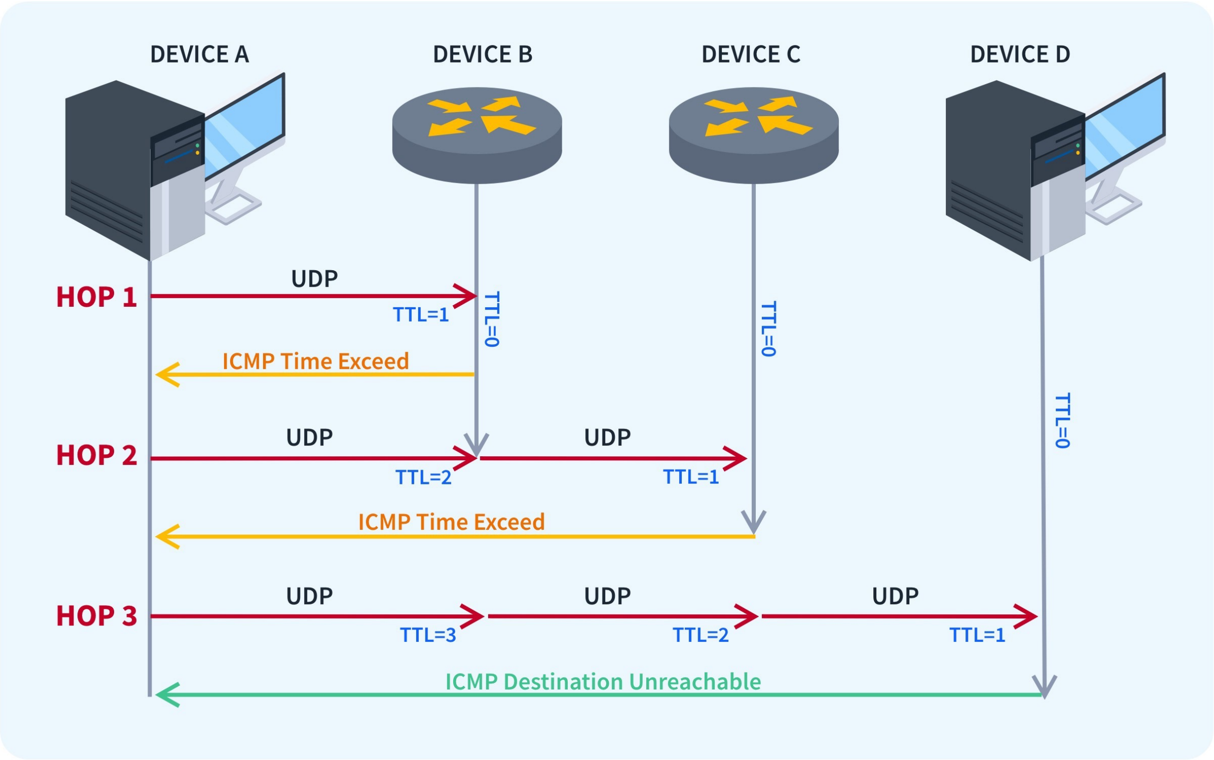
Traceroute: “Show me routes from here to X”

```
mhan in key in ~ via 🐍 v2.7.18
❯ traceroute sorry.cs.utexas.edu
traceroute to sorry.cs.utexas.edu (128.83.130.135), 30 hops max, 60 byte packets
 1  sorry.cs.utexas.edu (128.83.130.135)  0.468 ms  0.407 ms  0.362 ms
```

```
mhan in key in ~ via 🐍 v2.7.18 took 25s
❯ traceroute dns.google
traceroute to dns.google (8.8.4.4), 30 hops max, 60 byte packets
 1  cs-gw.cs.utexas.edu (128.83.139.1)  0.521 ms  0.465 ms  0.459 ms
 2  cs45k-cs65k-po1-p2p.gw.utexas.edu (128.83.37.65)  7.845 ms  7.806 ms  7.819 ms
 3  nocb-p2p-gdc.gw.utexas.edu (10.83.8.49)  0.460 ms napm-p2p-gdc.gw.utexas.edu (10.83.7.49)  0.455 ms  0.674 ms
 4  10.83.10.50 (10.83.10.50)  0.561 ms 10.83.9.50 (10.83.9.50)  0.405 ms  0.437 ms
 5  nocb1-nocb10-p2p10.gw.utexas.edu (10.83.10.34)  2.382 ms napm1-napm9-p2p9.gw.utexas.edu (10.83.9.30)  2.972 ms nocb1-
.gw.utexas.edu (10.83.10.34)  2.330 ms
 6  dlls-lvl3-isp-ae0-630.tx-bb.net (192.12.10.17)  5.061 ms  5.119 ms  5.072 ms
 7  74.125.50.226 (74.125.50.226)  7.219 ms * *
 8  74.125.50.226 (74.125.50.226)  6.315 ms * *
 9  dns.google (8.8.4.4)  5.240 ms *  5.330 ms
```

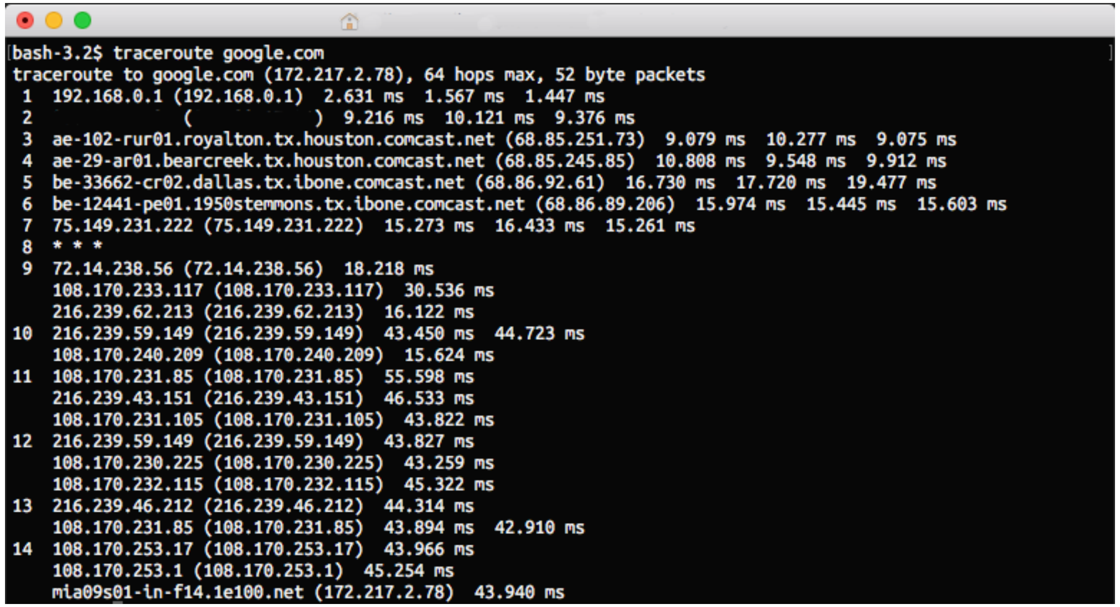
How to implement traceroute? Use TTL!

- ICMP dictates…
  - Any router/host must respond upon receiving a packet with TTL = 0 to original src
  - Any host must respond upon receiving a packet with non-existing port to original src



Traceroute shows each hop from src to dst

- src sends out UDP segments with unlikely port number
  - Segment 1 has TTL of 1: expires at 1st hop
  - Segment 2 has TTL of 2: expires at 2nd hop
  - Segment 3 has TTL of 3: expires at 3rd hop
  - …

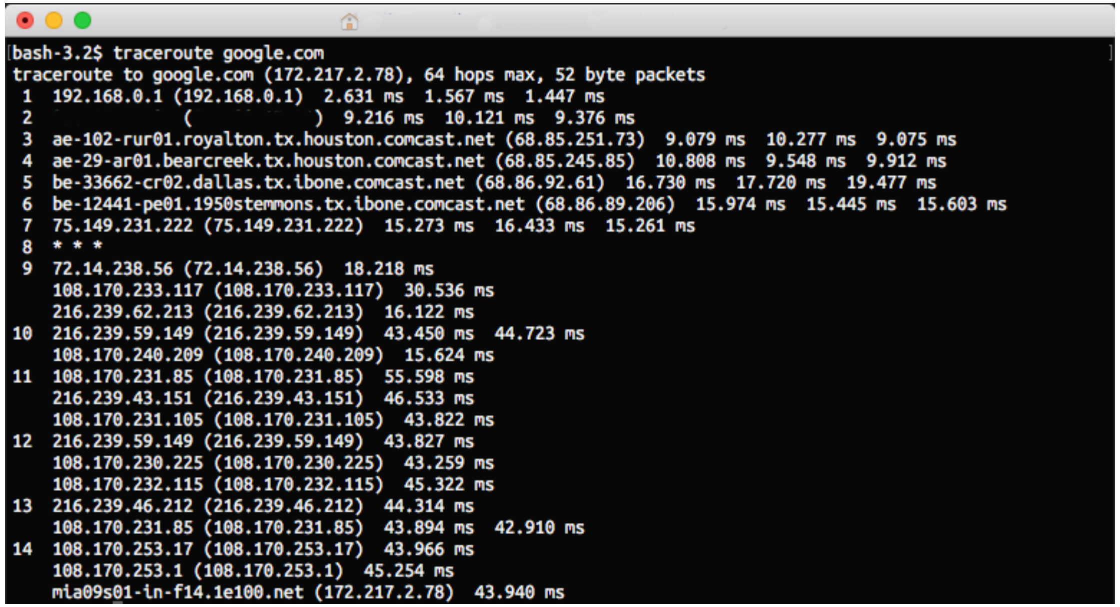


```
bash-3.2$ traceroute google.com
traceroute to google.com (172.217.2.78), 64 hops max, 52 byte packets
 1  192.168.0.1 (192.168.0.1)  2.631 ms  1.567 ms  1.447 ms
 2              (            )  9.216 ms  10.121 ms  9.376 ms
 3  ae-102-rur01.royalton.tx.houston.comcast.net (68.85.251.73)  9.079 ms  10.277 ms  9.075 ms
 4  ae-29-ar01.bearcreek.tx.houston.comcast.net (68.85.245.85)  10.808 ms  9.548 ms  9.912 ms
 5  be-33662-cr02.dallas.tx.ibone.comcast.net (68.86.92.61)  16.730 ms  17.720 ms  19.477 ms
 6  be-12441-pe01.1950stemmons.tx.ibone.comcast.net (68.86.89.206)  15.974 ms  15.445 ms  15.603 ms
 7  75.149.231.222 (75.149.231.222)  15.273 ms  16.433 ms  15.261 ms
 8  * * *
 9  72.14.238.56 (72.14.238.56)  18.218 ms
    108.170.233.117 (108.170.233.117)  30.536 ms
    216.239.62.213 (216.239.62.213)  16.122 ms
10  216.239.59.149 (216.239.59.149)  43.450 ms  44.723 ms
    108.170.240.209 (108.170.240.209)  15.624 ms
11  108.170.231.85 (108.170.231.85)  55.598 ms
    216.239.43.151 (216.239.43.151)  46.533 ms
    108.170.231.105 (108.170.231.105)  43.822 ms
12  216.239.59.149 (216.239.59.149)  43.827 ms
    108.170.230.225 (108.170.230.225)  43.259 ms
    108.170.232.115 (108.170.232.115)  45.322 ms
13  216.239.46.212 (216.239.46.212)  44.314 ms
    108.170.231.85 (108.170.231.85)  43.894 ms  42.910 ms
14  108.170.253.17 (108.170.253.17)  43.966 ms
    108.170.253.1 (108.170.253.1)  45.254 ms
    mia09s01-in-f14.1e100.net (172.217.2.78)  43.940 ms
```

- router at which TTL expires sends back Time Exceeded (ICMP warning) to back to src
- dst with no such UDP port open sends dst port unreachable (ICMP warning) back to src

Traceroute shows each hop from src to dst

- src sends out UDP segments with unlikely port number
  - Segment 1 has TTL of 1: expires at 1st hop
  - Segment 2 has TTL of 2: expires at 2nd hop
  - Segment 3 has TTL of 3: expires at 3rd hop
  - …



- router at which TTL expires sends back Time Exceeded (ICMP warning) to back to src
- dst with no such UDP port open sends dst port unreachable (ICMP warning) back to src

How does src know when to stop sending segment?

Acknowledgements

Slides are adopted from Kurose’ Computer Networking Slides