Lesson 06-06: SDN and ICMP

CS 326E Elements of Networking

Mikyung Han

mhan@cs.utexas.edu

Internet Reference Model

Example Protocols	Layer	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

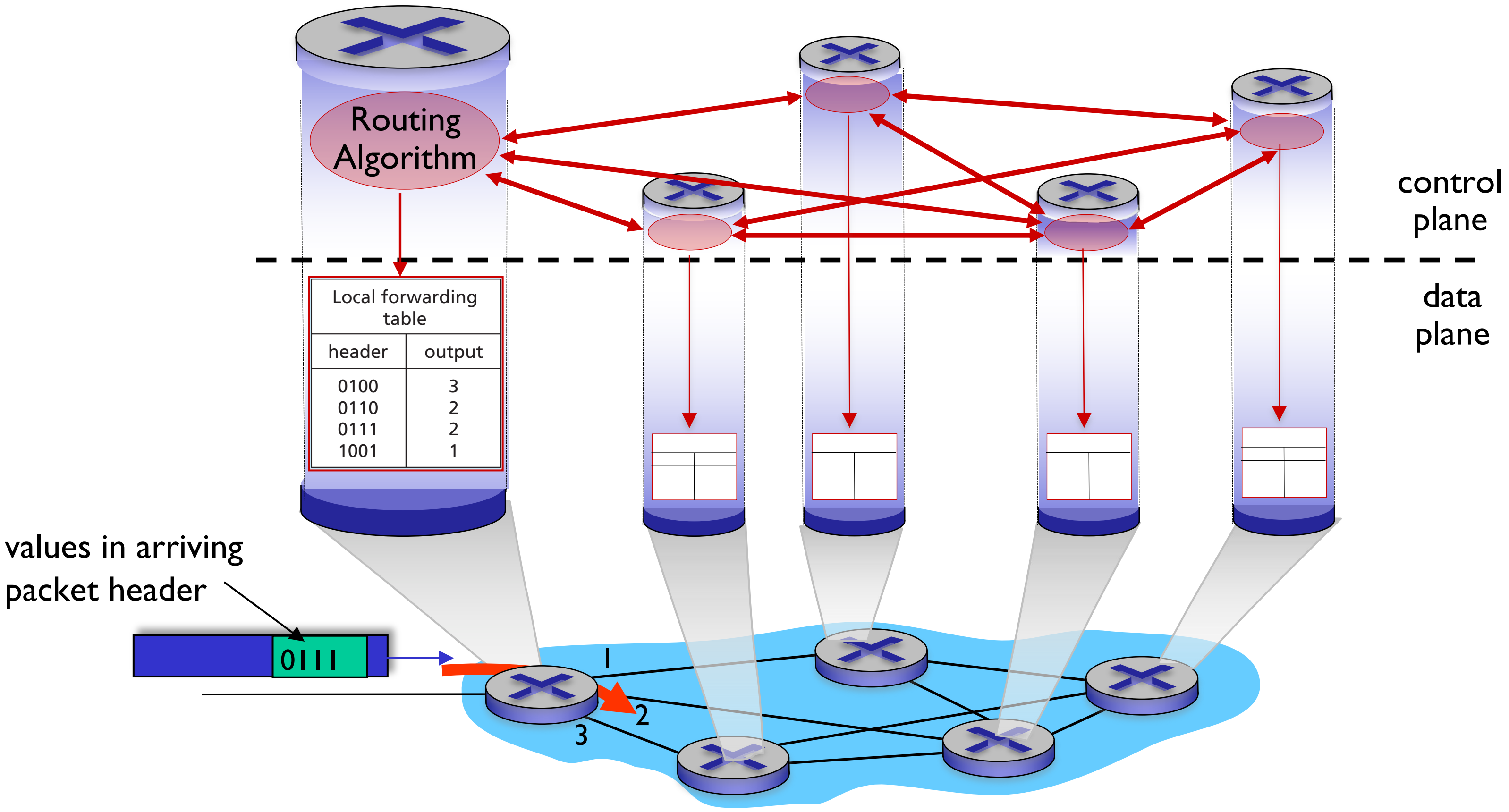
Outline

1. Why SDN?

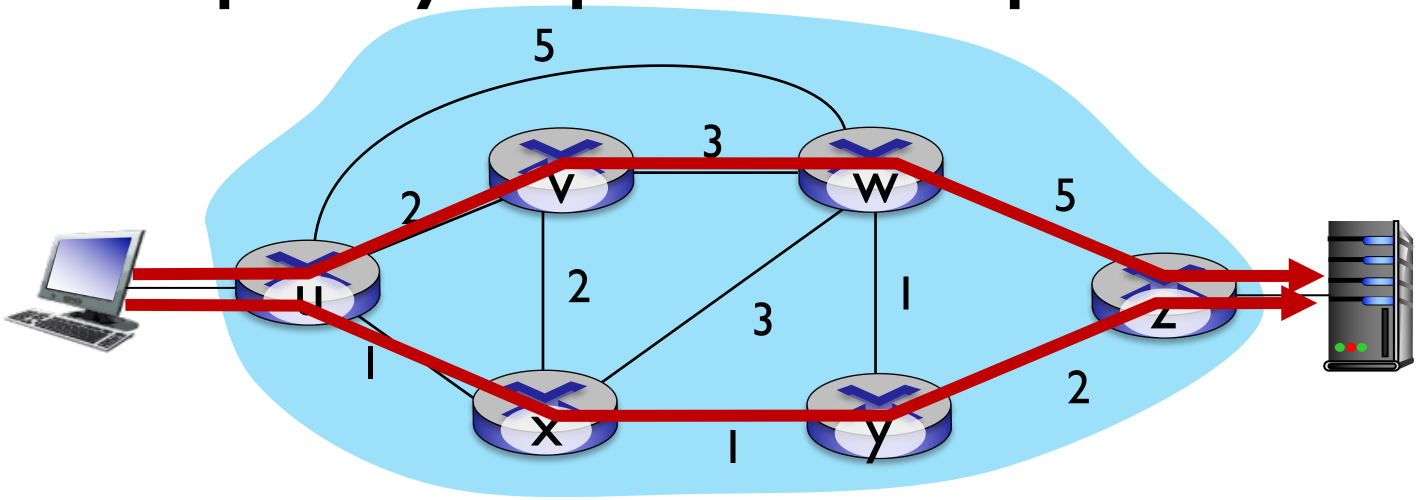
Motivation:
What is difficult/impossible in traditional routing?

Traditional per-router control plane

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



Traditional routing: Not easy to specify a preferred path

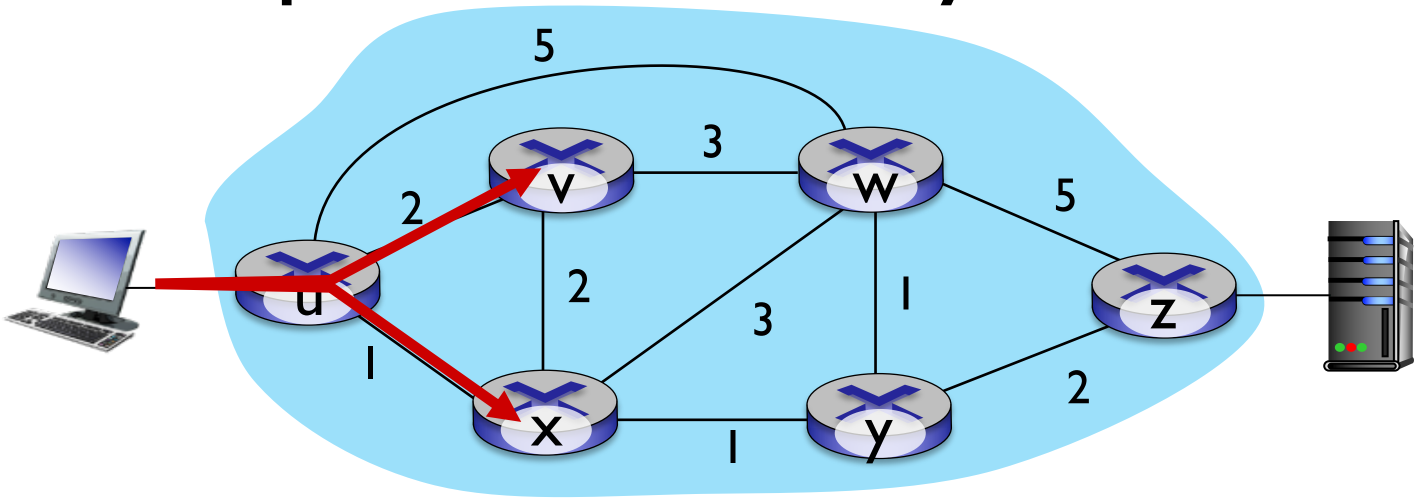


Q: what if network operator wants u-to-z traffic to flow along uvwz, rather than uxyz?

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

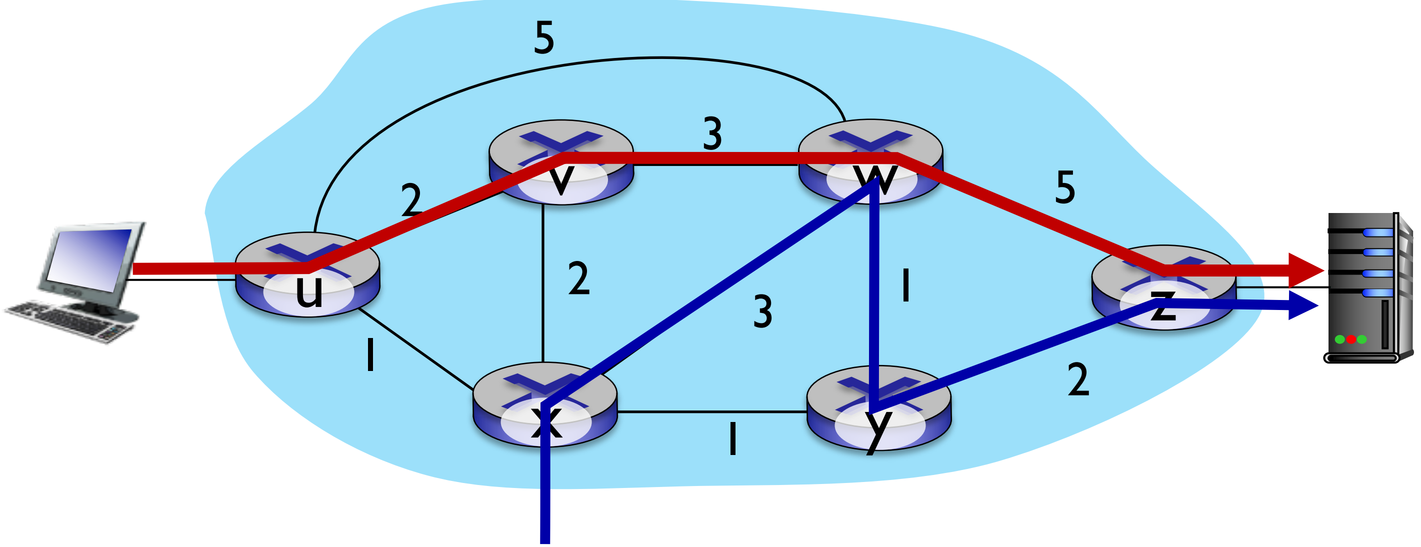
link weights are only control "knobs": not much control!

Traditional routing: Impossible to split traffic evenly



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)

Traditional routing: Impossible to use different routes for different flows

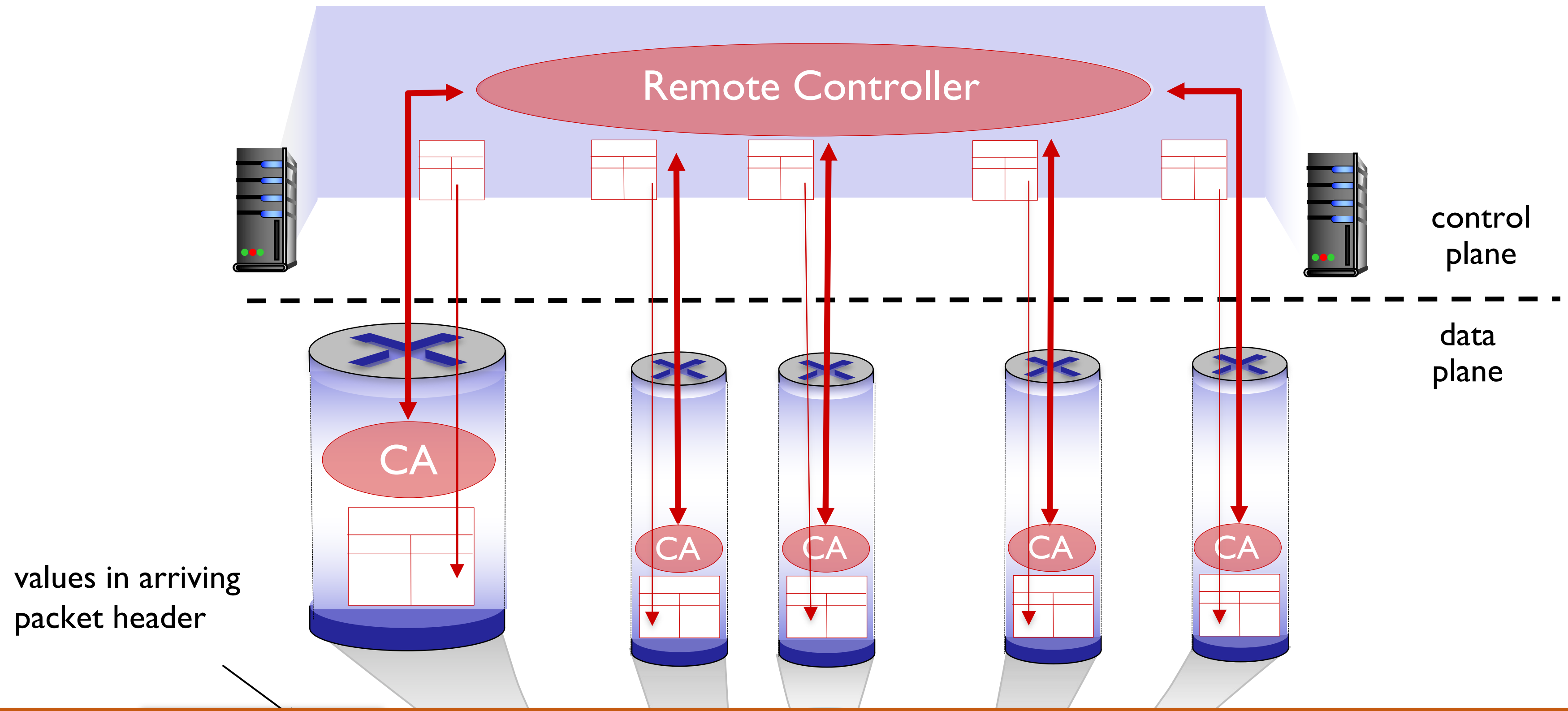


Q: what if w wants to route blue and red traffic differently from w to z?

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

GF can be used to achieve any routing desired!

SDN uses a logically centralized control plane



Remote controller computes and installs forwarding tables to each router

Why logically centralized control plane?

- **Easier management**
  - Less router misconfigurations
  - Greater flexibility of traffic flows
- **Allows "programmable" network**
- **Unbundling allowed rich innovation**
  - Functionality/implementations divided into 3 entities
    - SDN-controlled switches
    - SDN controller
    - Network-control applications
  - No longer "monolithic" or "vertically integrated" into a single router/switch

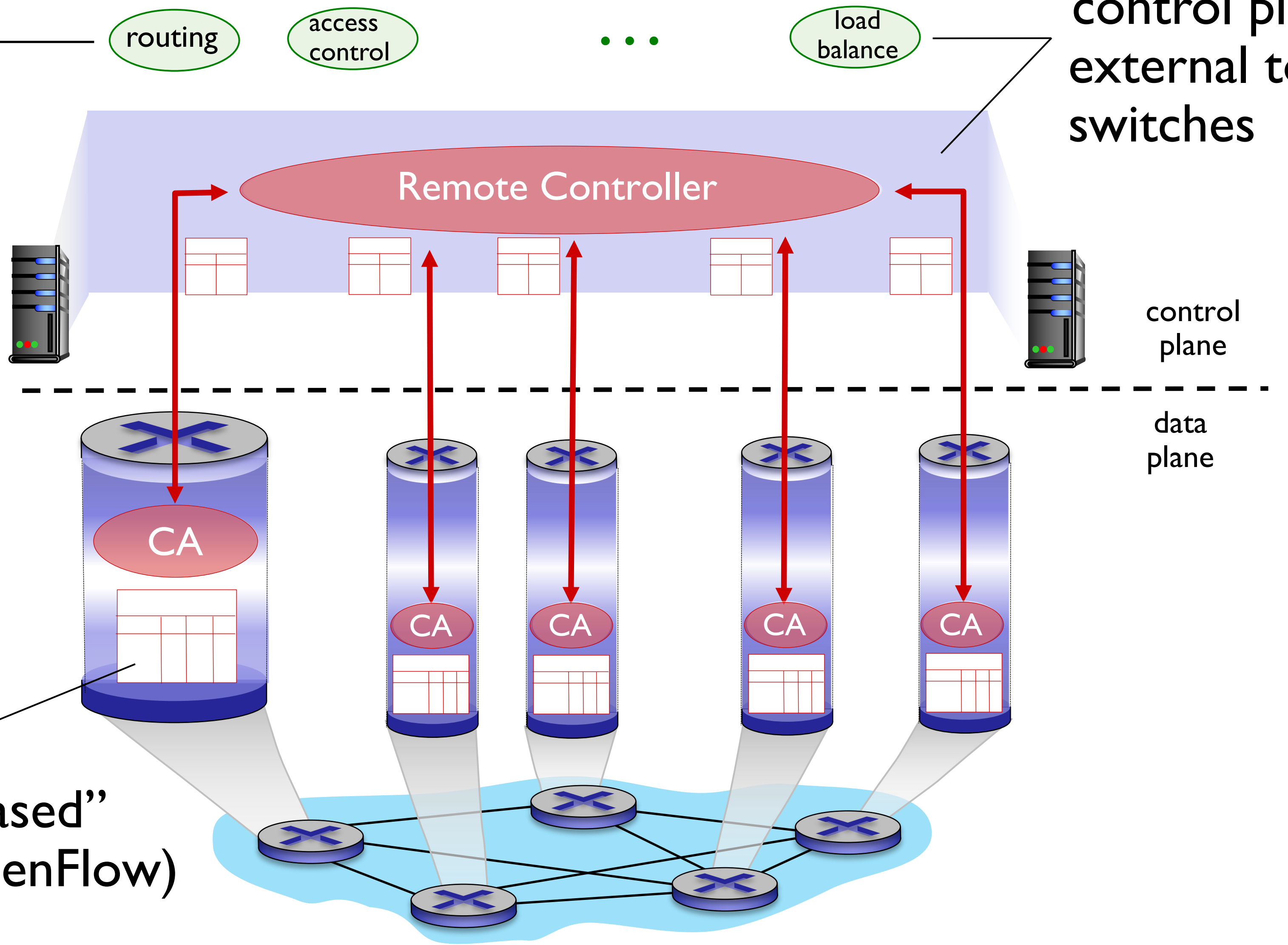
Outline

1. Why SDN?
2. SDN architecture

SDN architecture

programmable network-control applications

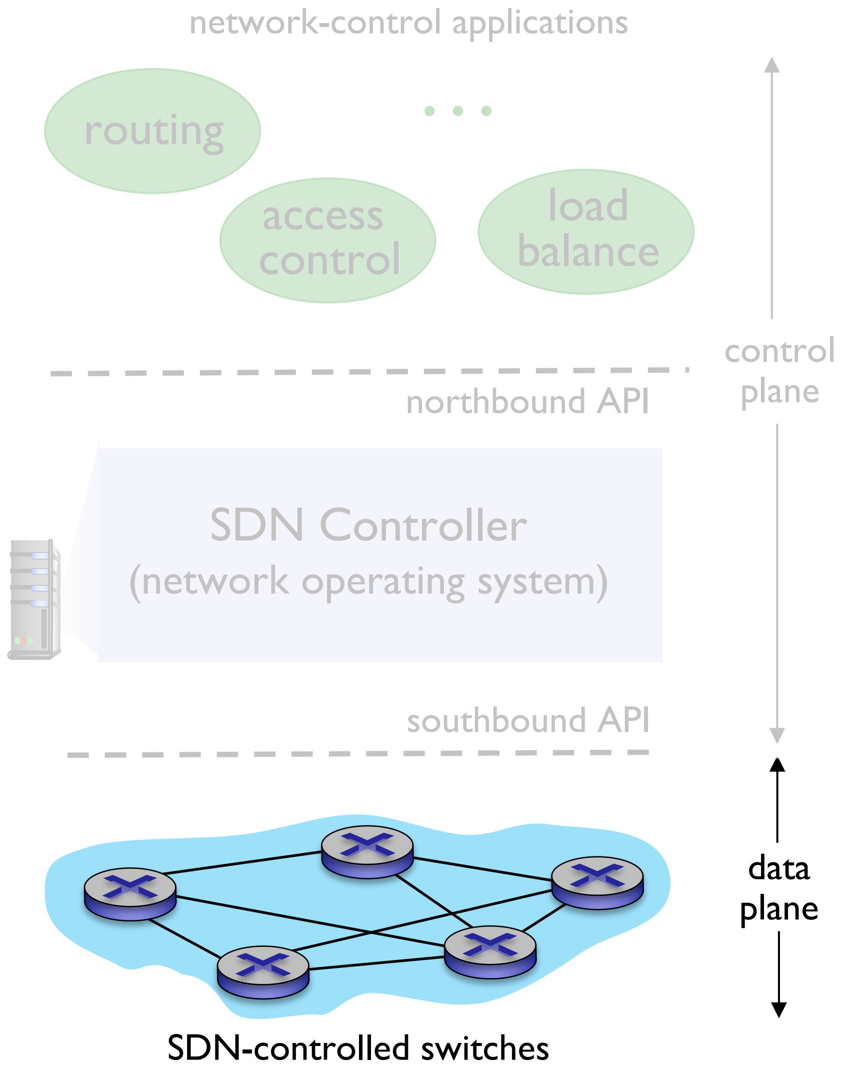
control plane functions external to data-plane switches



generalized "flow-based" forwarding (e.g., OpenFlow)

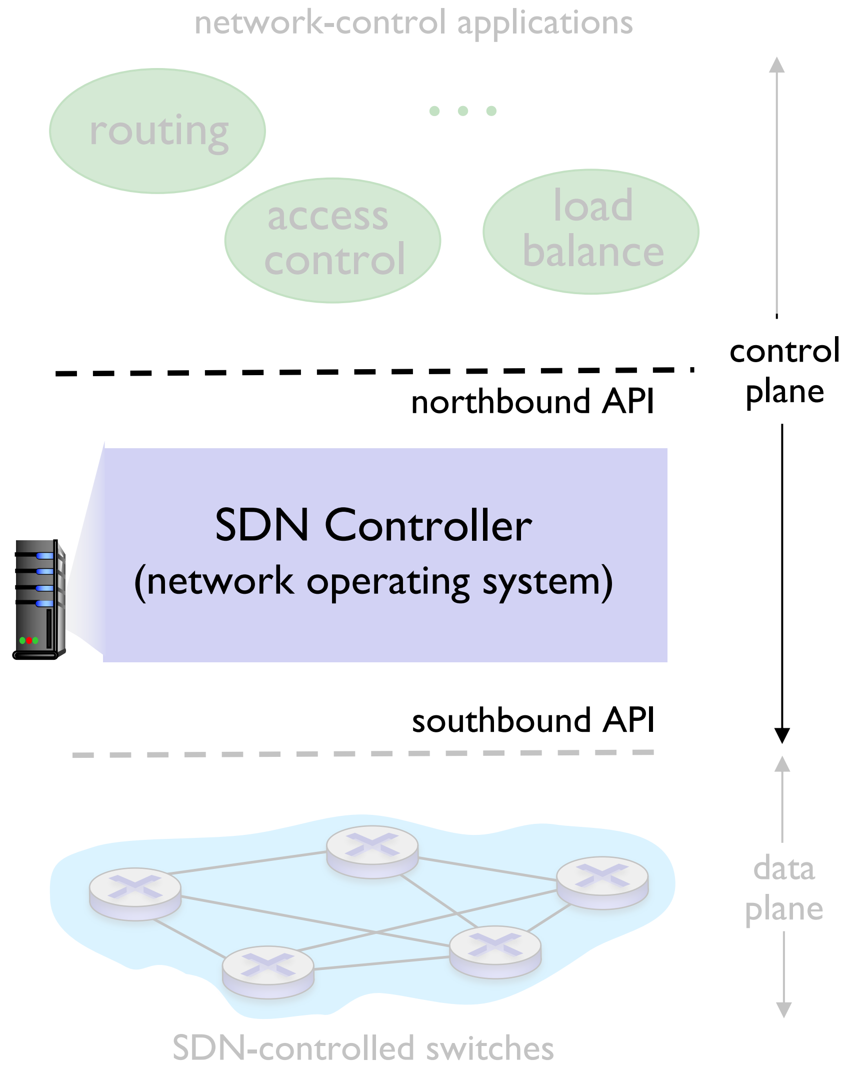
#1 Data-plane switches

- fast, simple, commodity switches implementing GF in hardware
- flow table computed, installed under controller supervision
- Communicates with SDN controller via protocol like OpenFlow



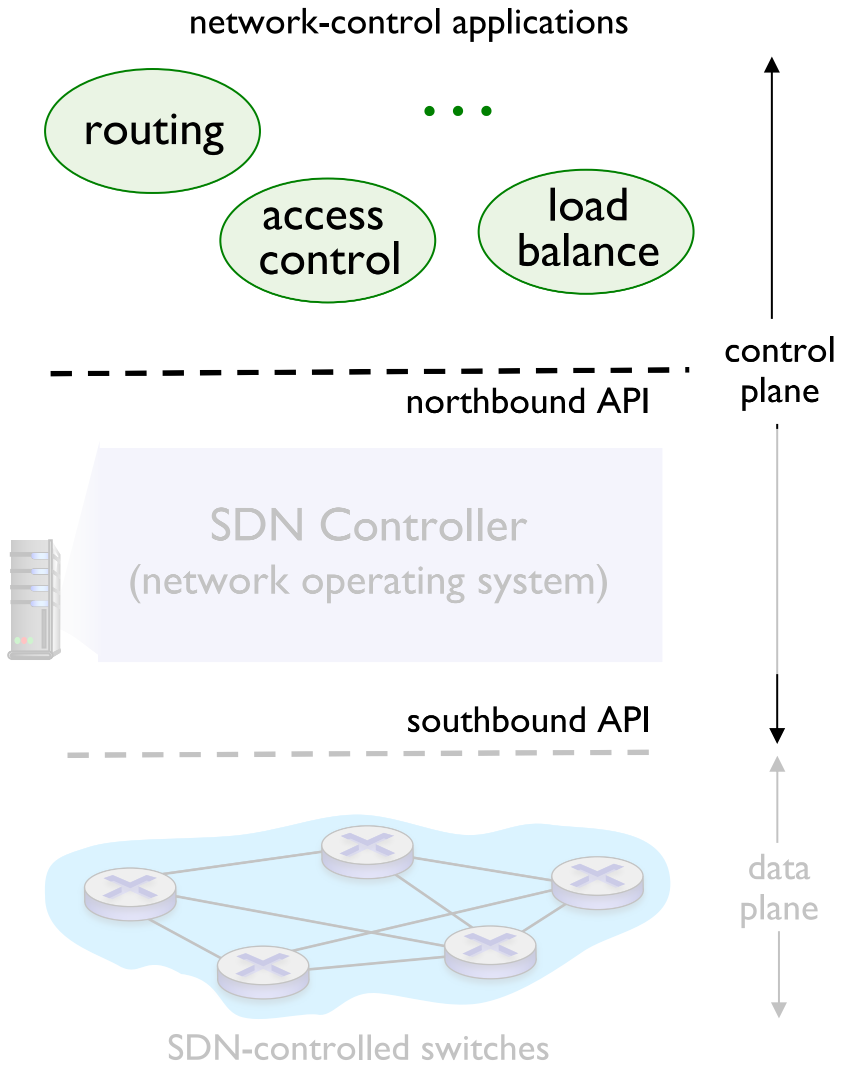
#2 SDN Controller (aka Network OS)

- maintains network state information
- interacts with both network switches "below" and network control applications "above"
- implemented as distributed system for performance, scalability, fault-tolerance, robustness

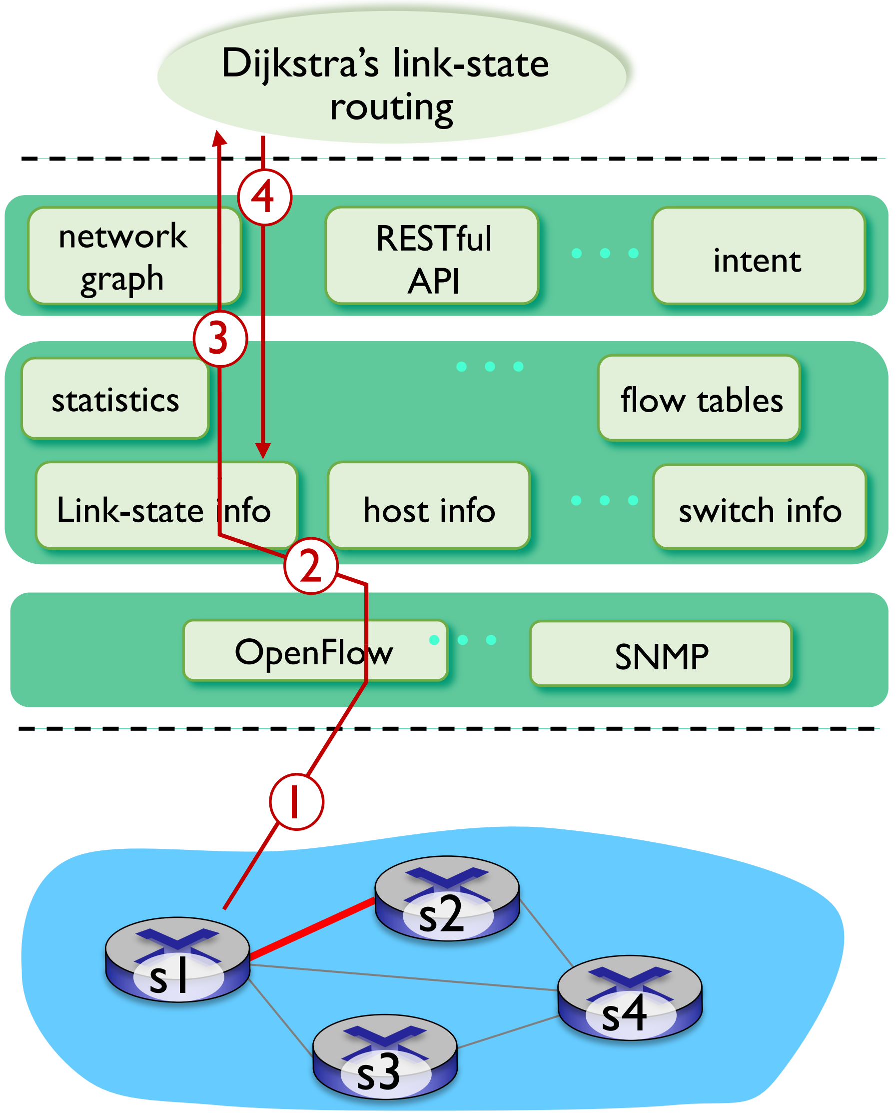


#3 Network-control applications

- "brains" of control: implement control functions depending on current network status
  - New routing algo, policy, etc…
- unbundled: can be provided by 3rd party: distinct from routing vendor, or SDN controller



SDN: control/data plane interaction example



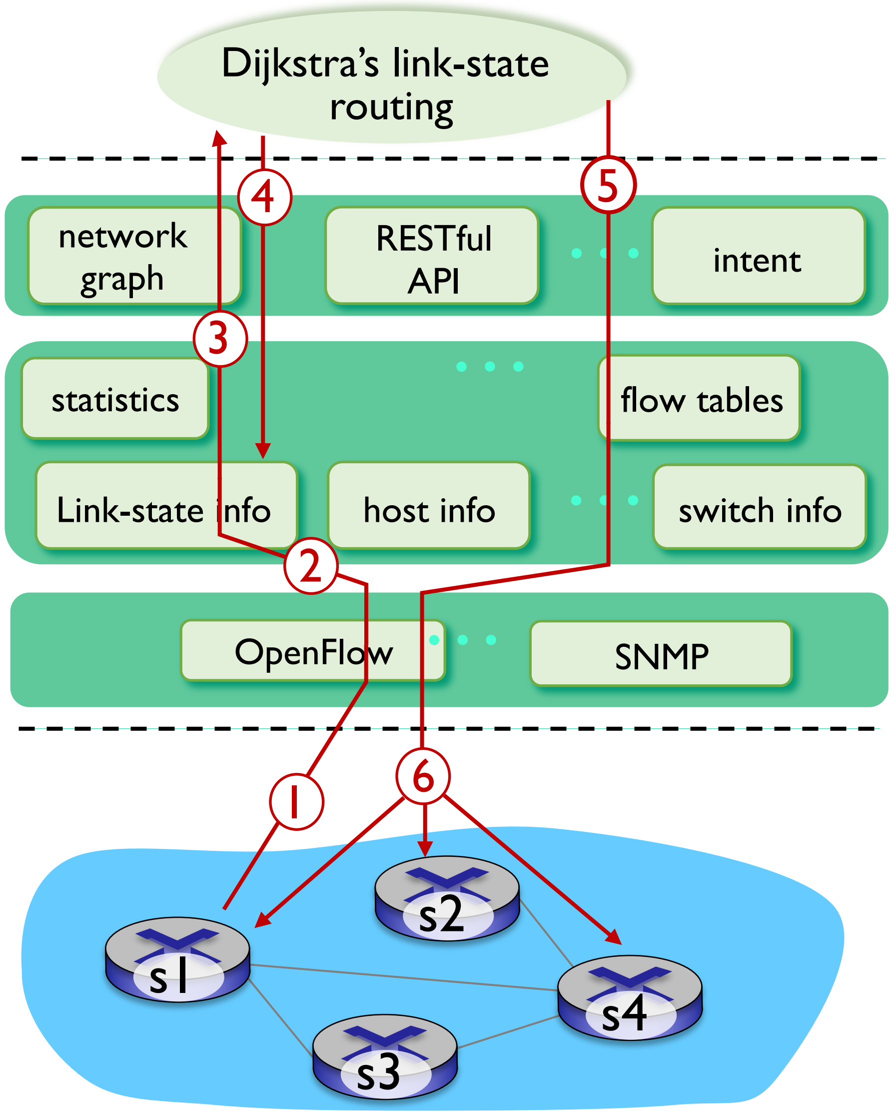
① S1, experiencing link failure uses OpenFlow port status message to notify controller

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
➜  ~
```

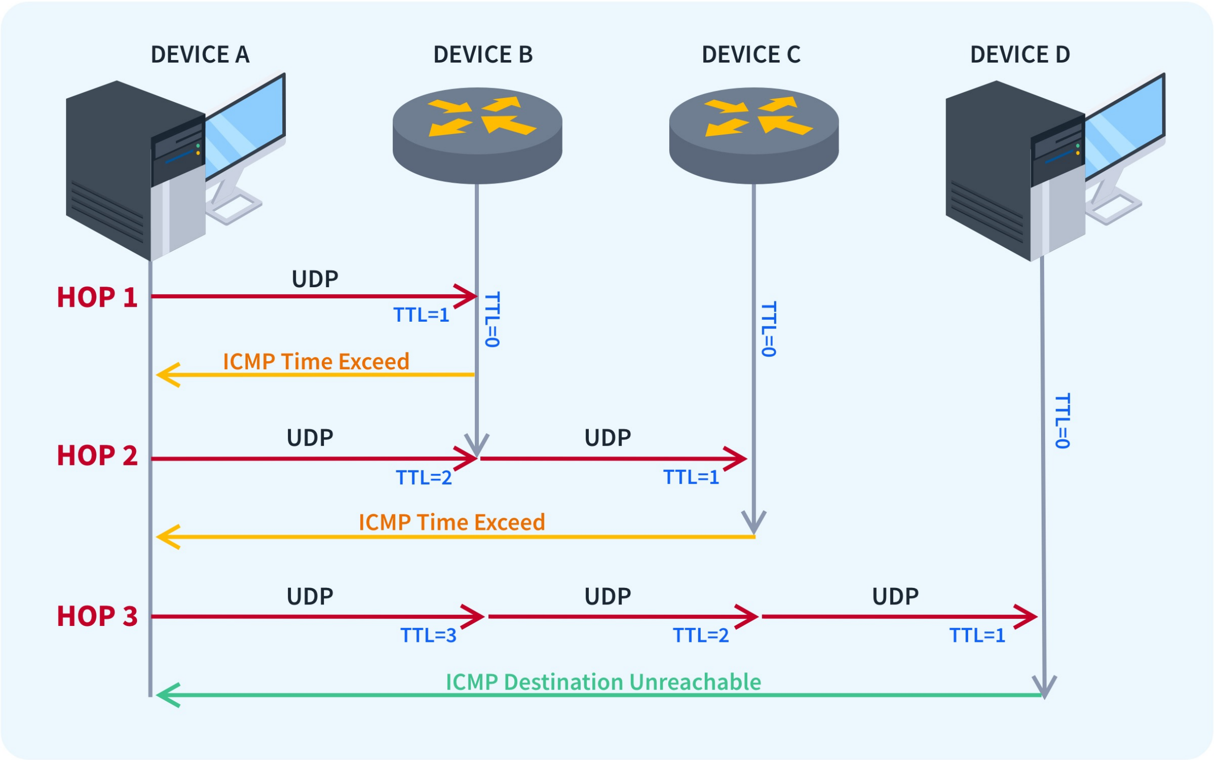
Traceroute: "Show me routes from here to X"

```
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
```

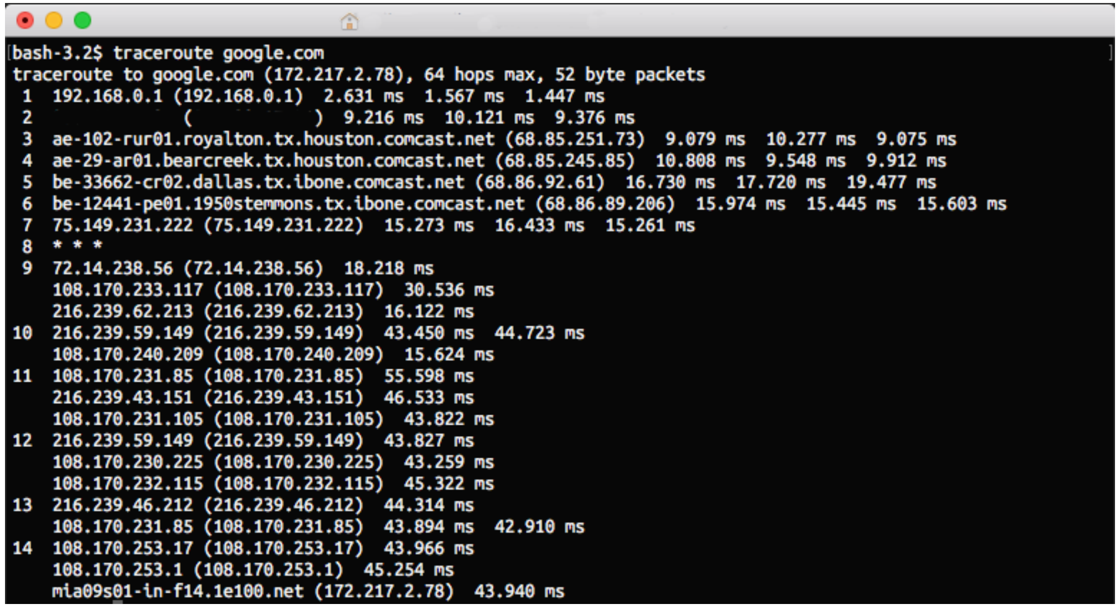
```
mhan in key in ~ via 🐍 v2.7.18 took 25s
❯ traceroute dns.google
traceroute to dns.google (8.8.4.4), 30 hops max, 60 byte packets
 1  cs-gw.cs.utexas.edu (128.83.139.1)  0.521 ms  0.465 ms  0.459 ms
 2  cs45k-cs65k-po1-p2p.gw.utexas.edu (128.83.37.65)  7.845 ms  7.806 ms  7.819 ms
 3  nocb-p2p-gdc.gw.utexas.edu (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
  - …
- router at which TTL expires sends back TTL expired (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