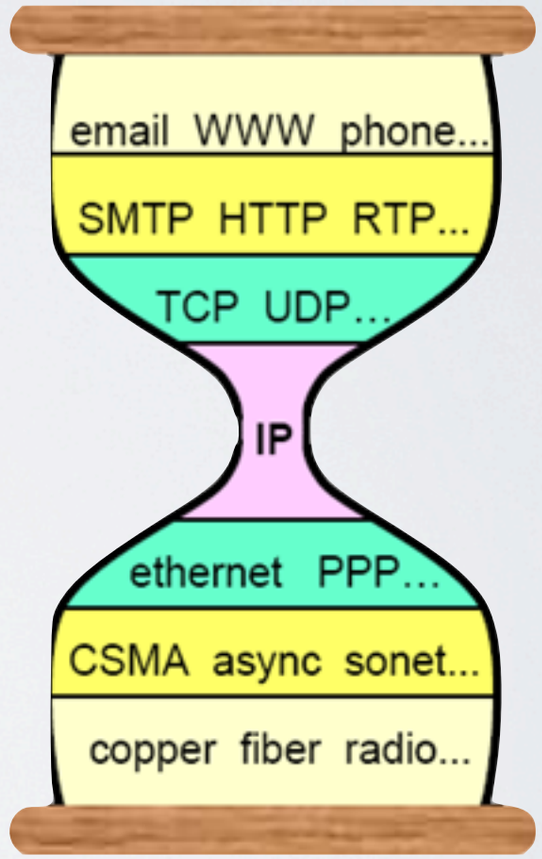
ROUTING IN PURE OPENFLOW ENVIRONMENT

by can.

some
backgrounds

WHY EXISTS IP?

- the *Internet* Protocol
- supposed to be globally unique
- "everything over IP"
- designed with routing in mind



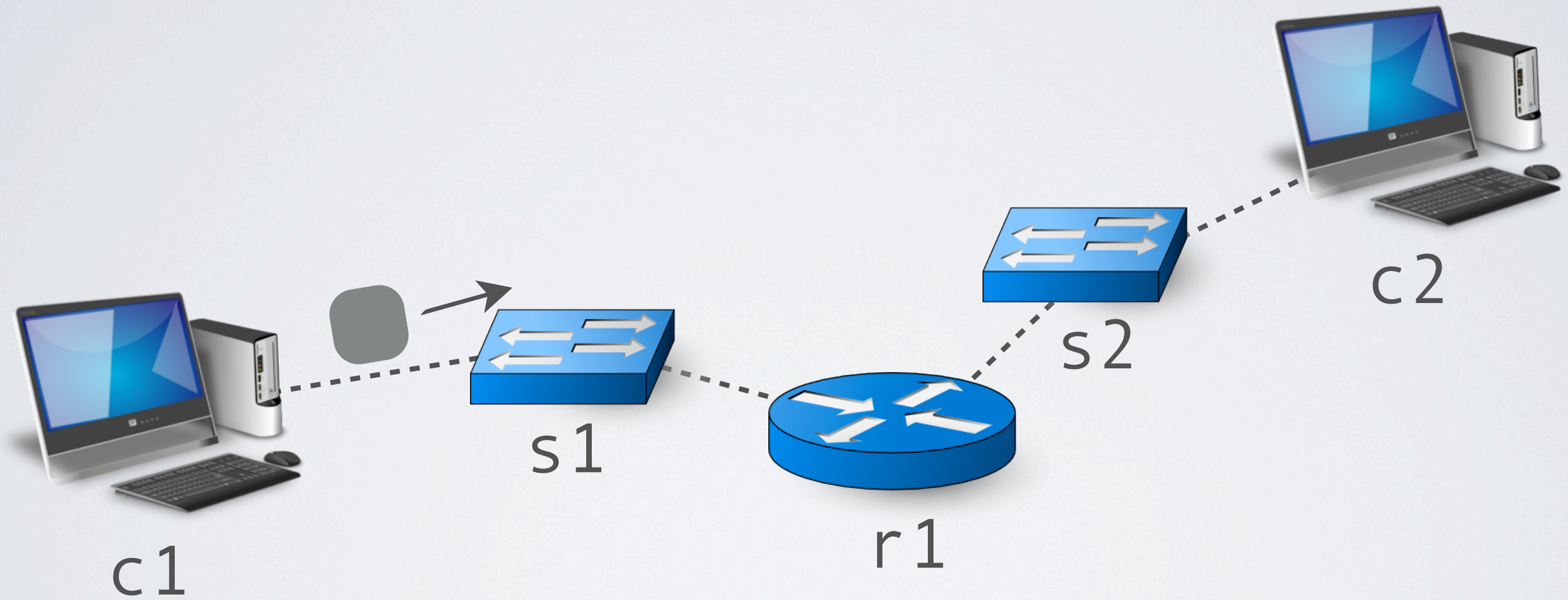
WHY EXISTS MAC?

- by MAC I mean *Media Access Control* layer

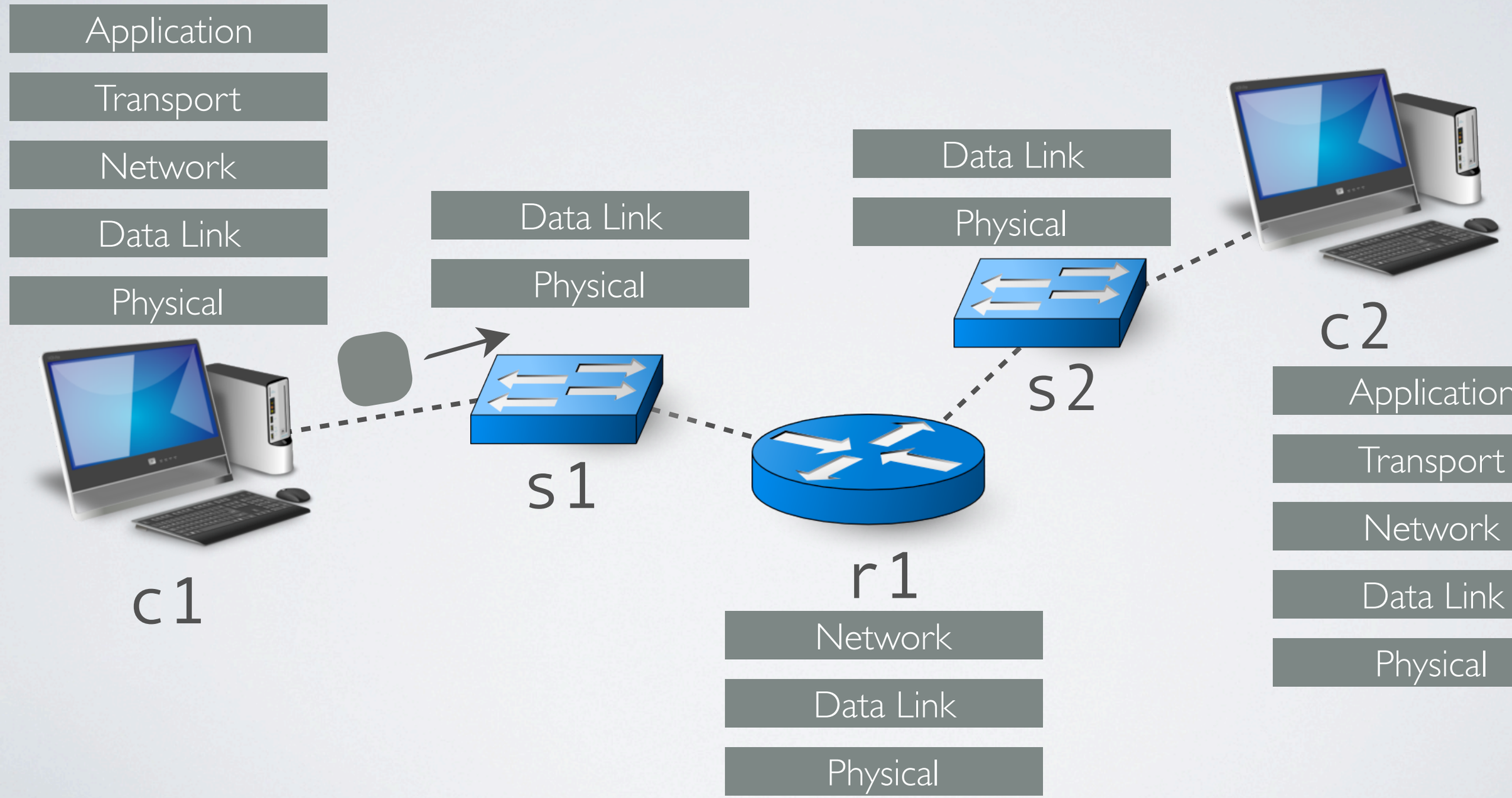
- in a wired switching ethernet, the MAC layer is really doing nothing

- but in other types of networks, especially wireless ones, it means a lot

HOW A PACKET TRAVELS?

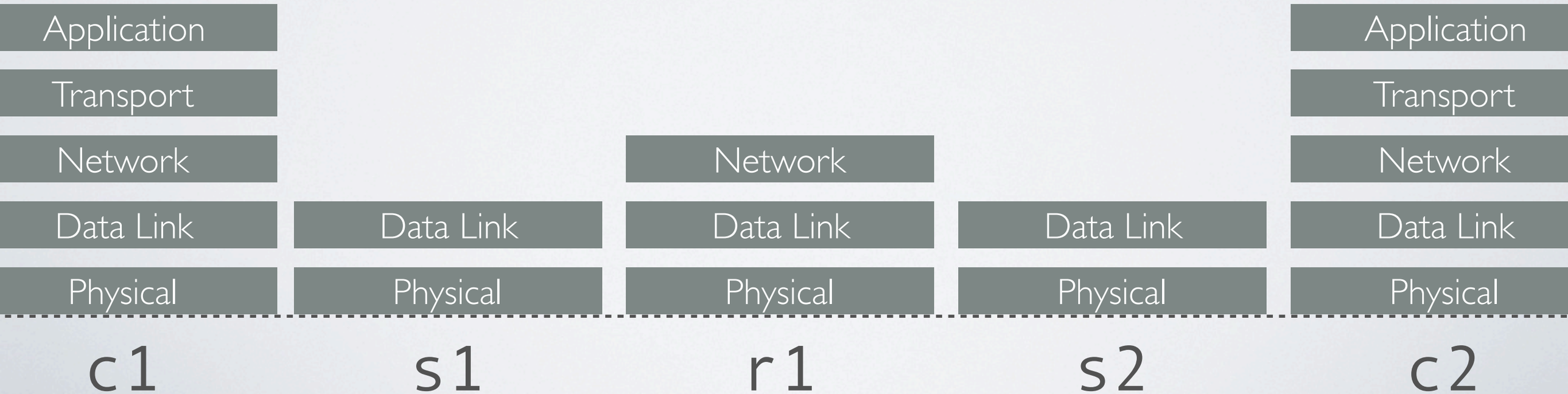


HOW A PACKET TRAVELS?

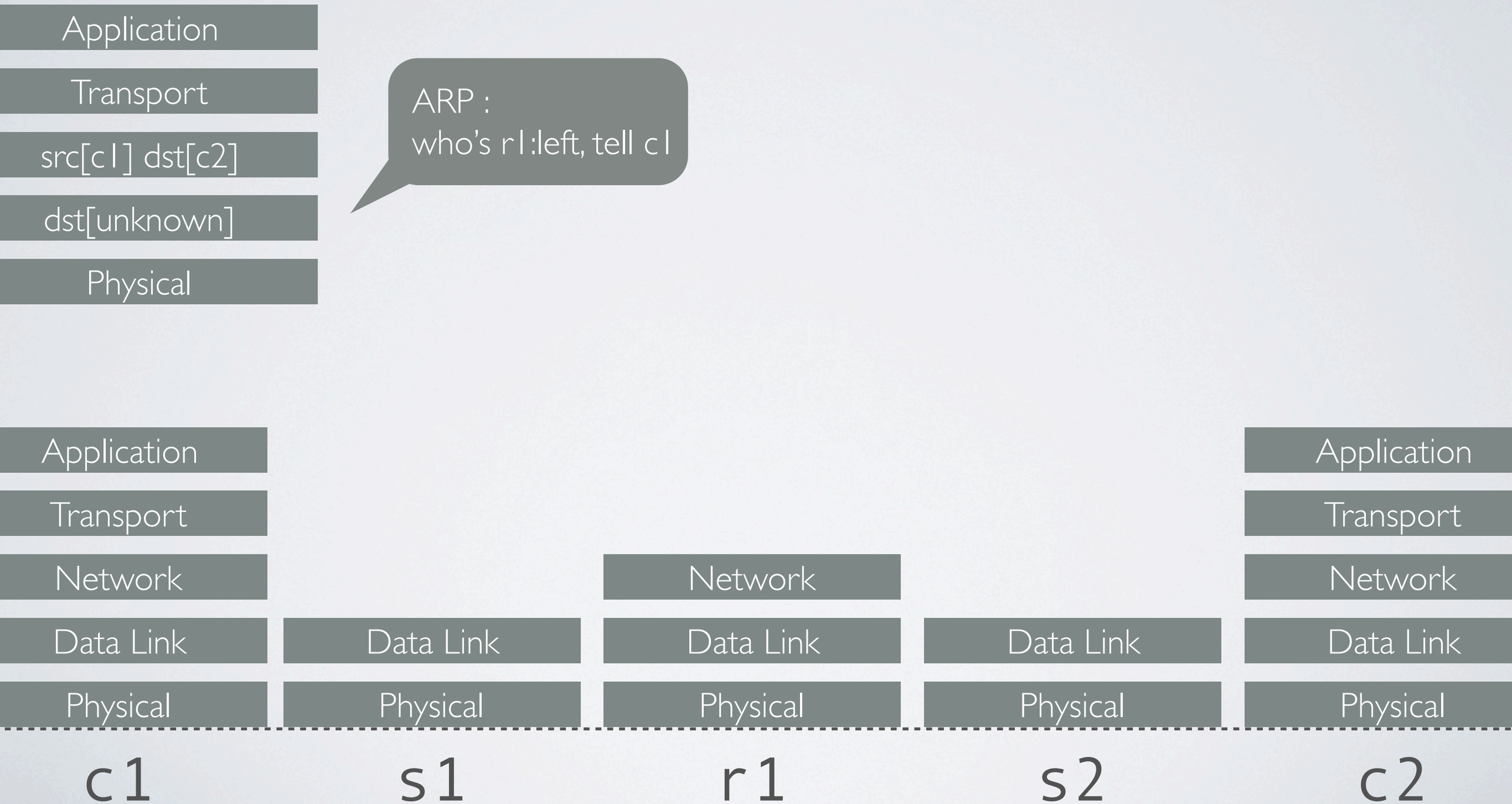


HOW A PACKET TRAVELS?

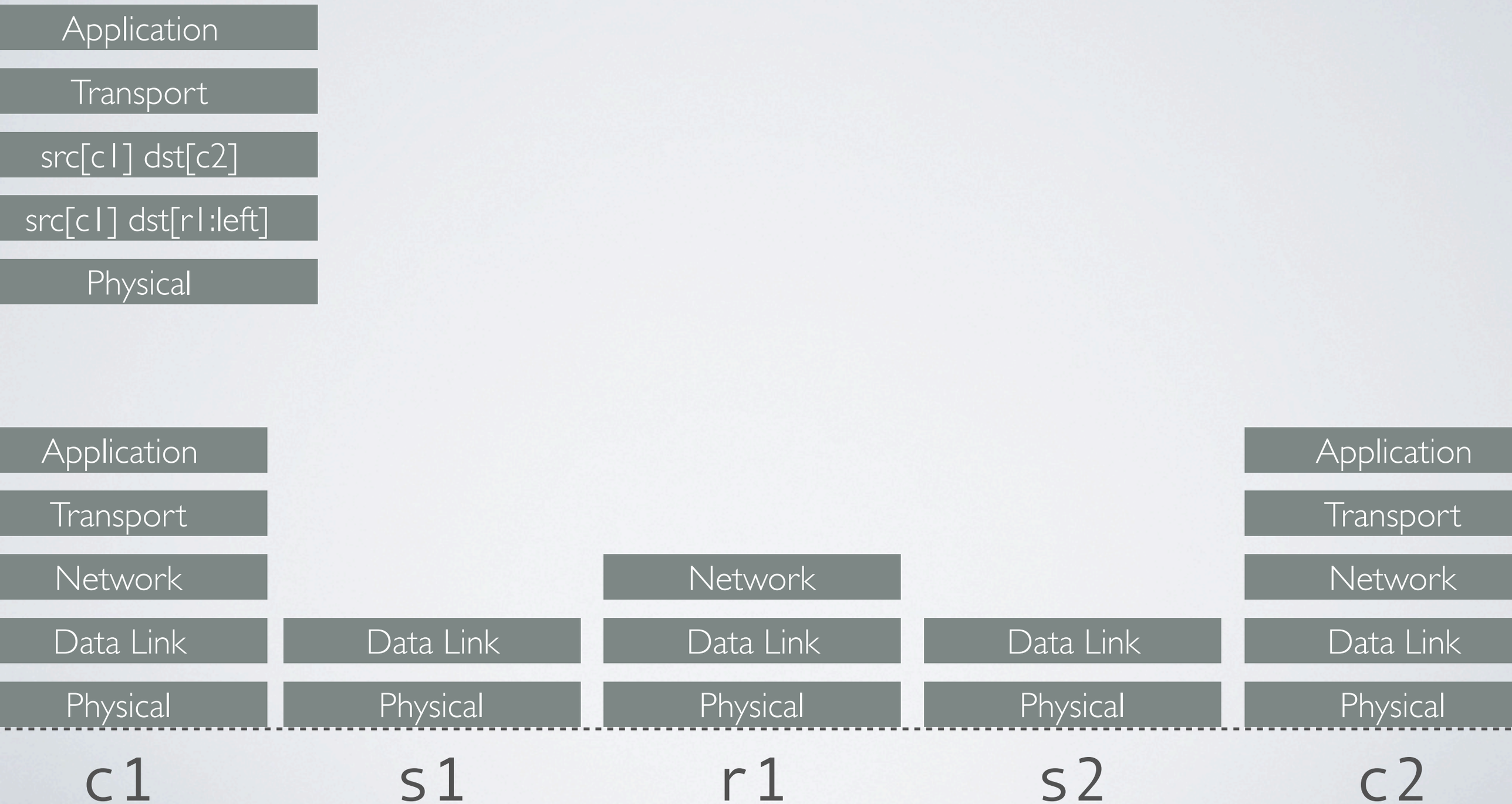
packet p1: c1 => c2
gateway[c1] = r1.left
gateway[c2] = r1.right
every table of any devices are empty



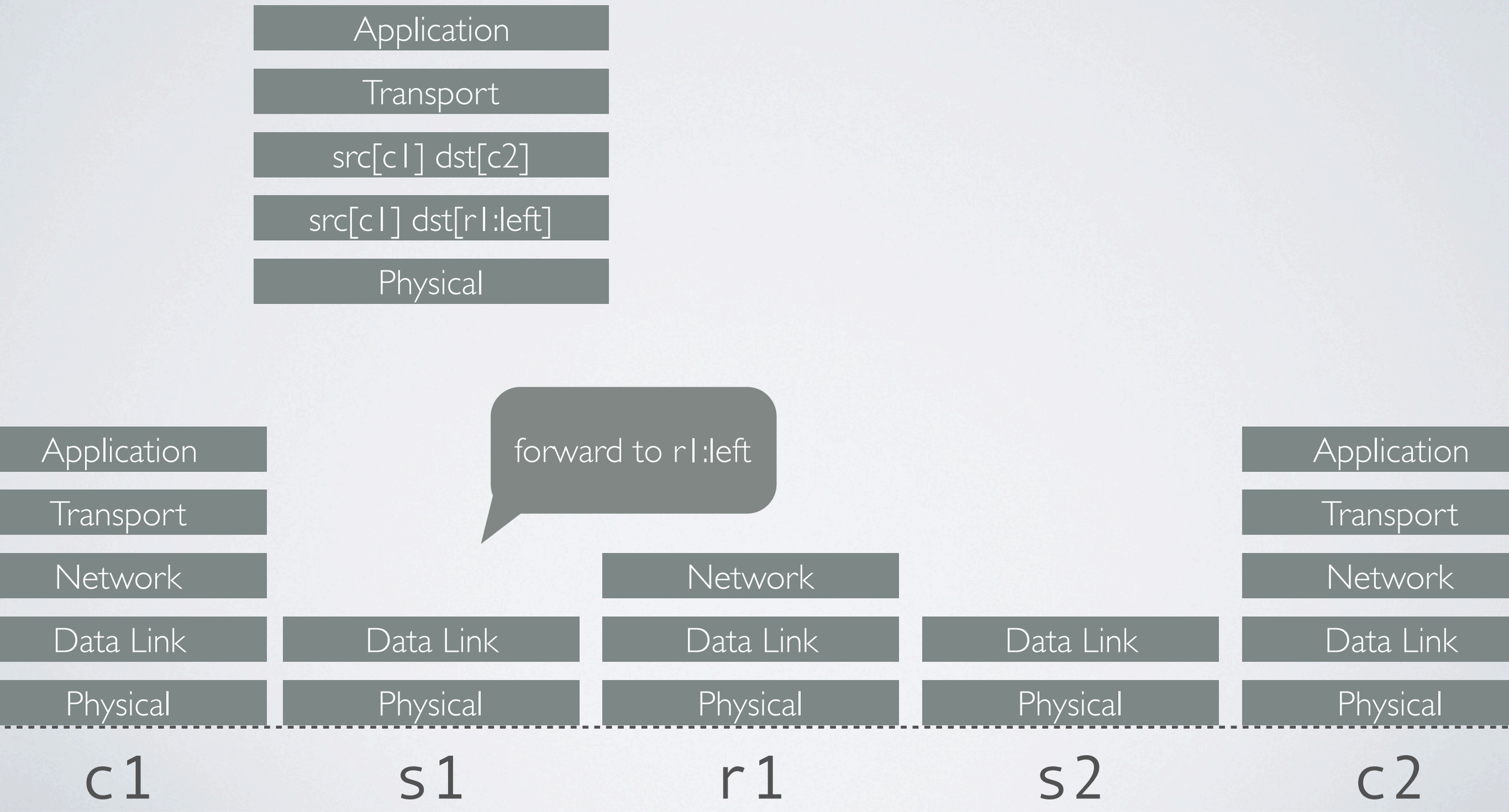
HOW A PACKET TRAVELS?



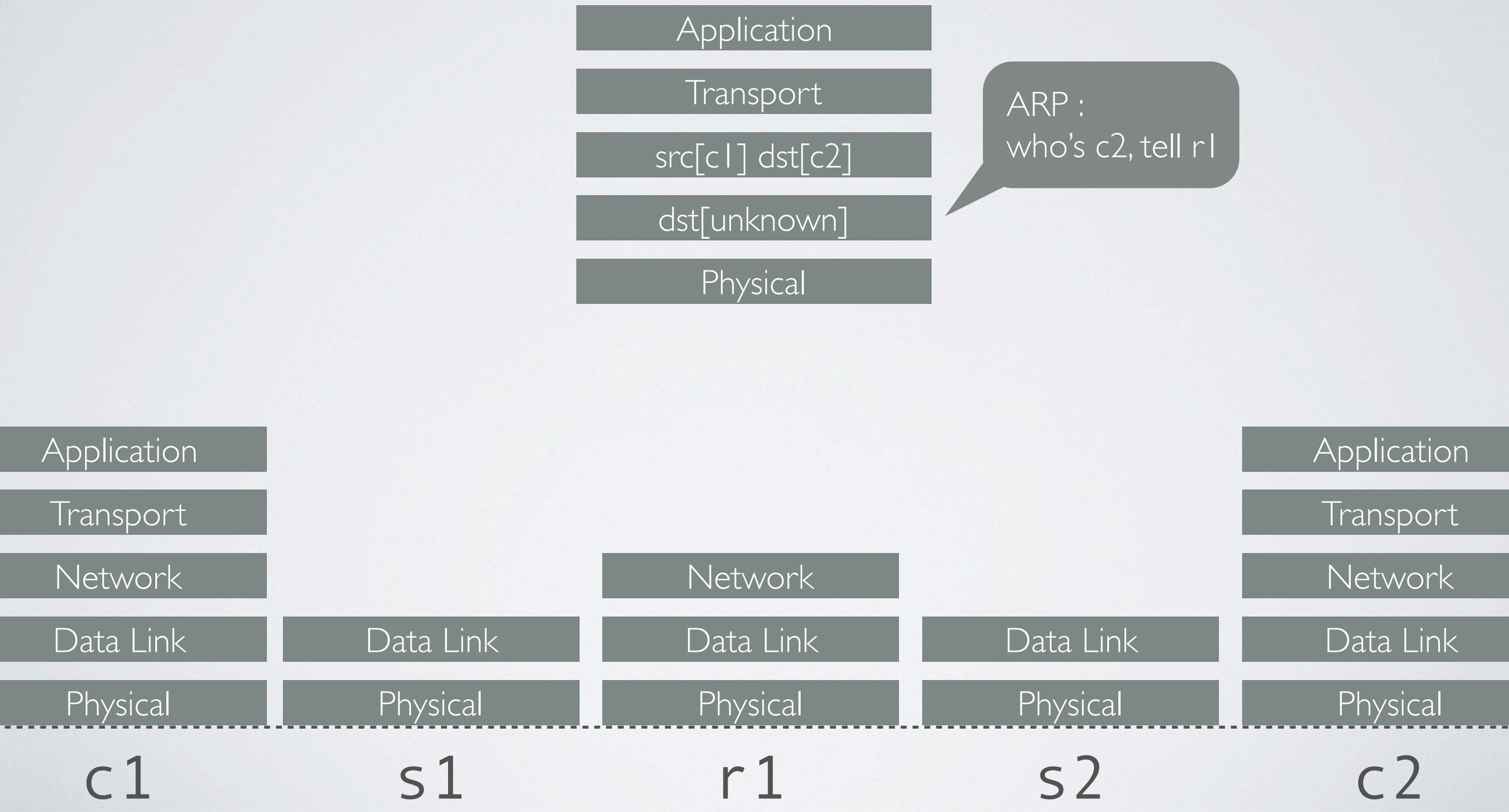
HOW A PACKET TRAVELS?



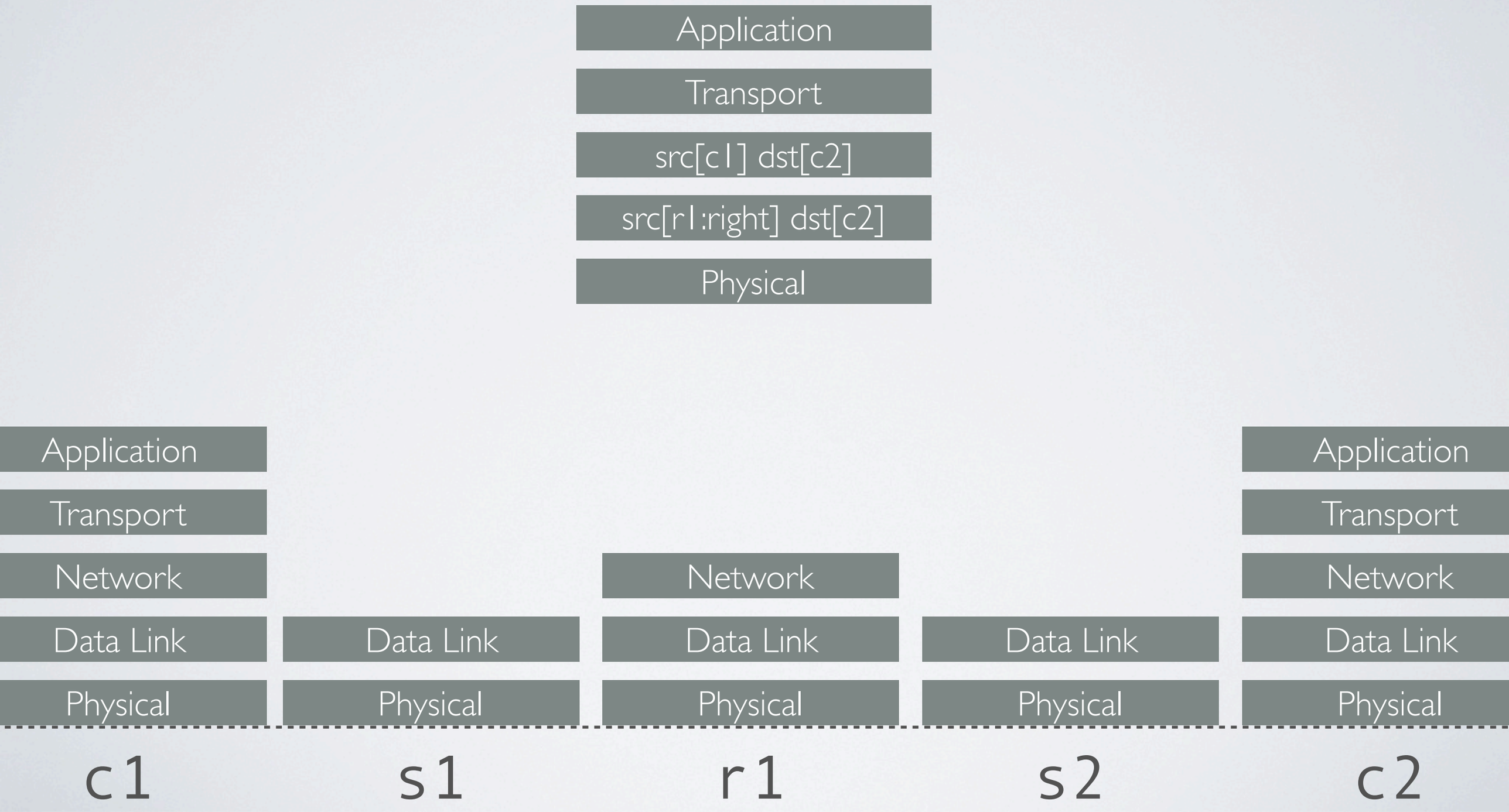
HOW A PACKET TRAVELS?



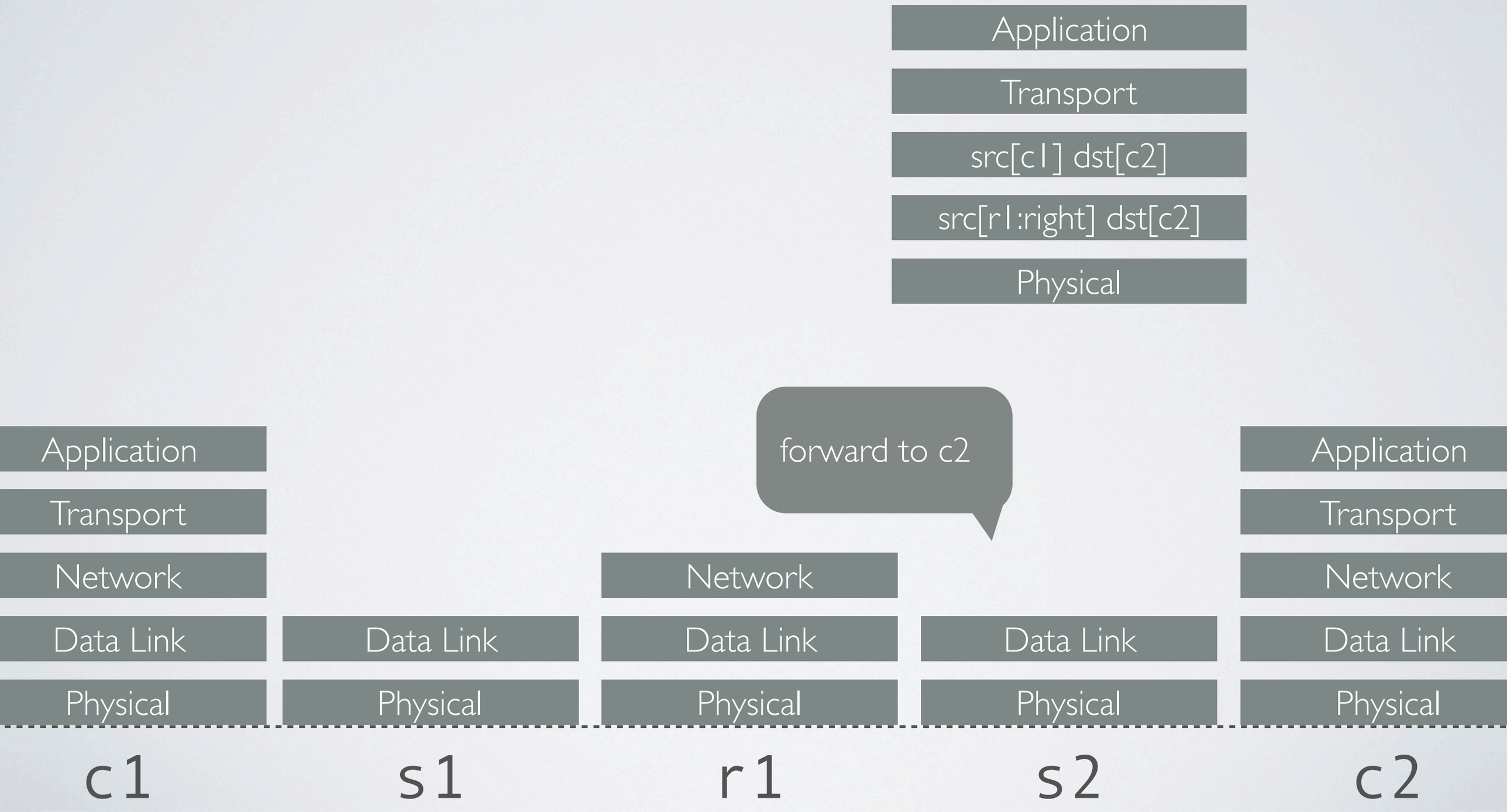
HOW A PACKET TRAVELS?



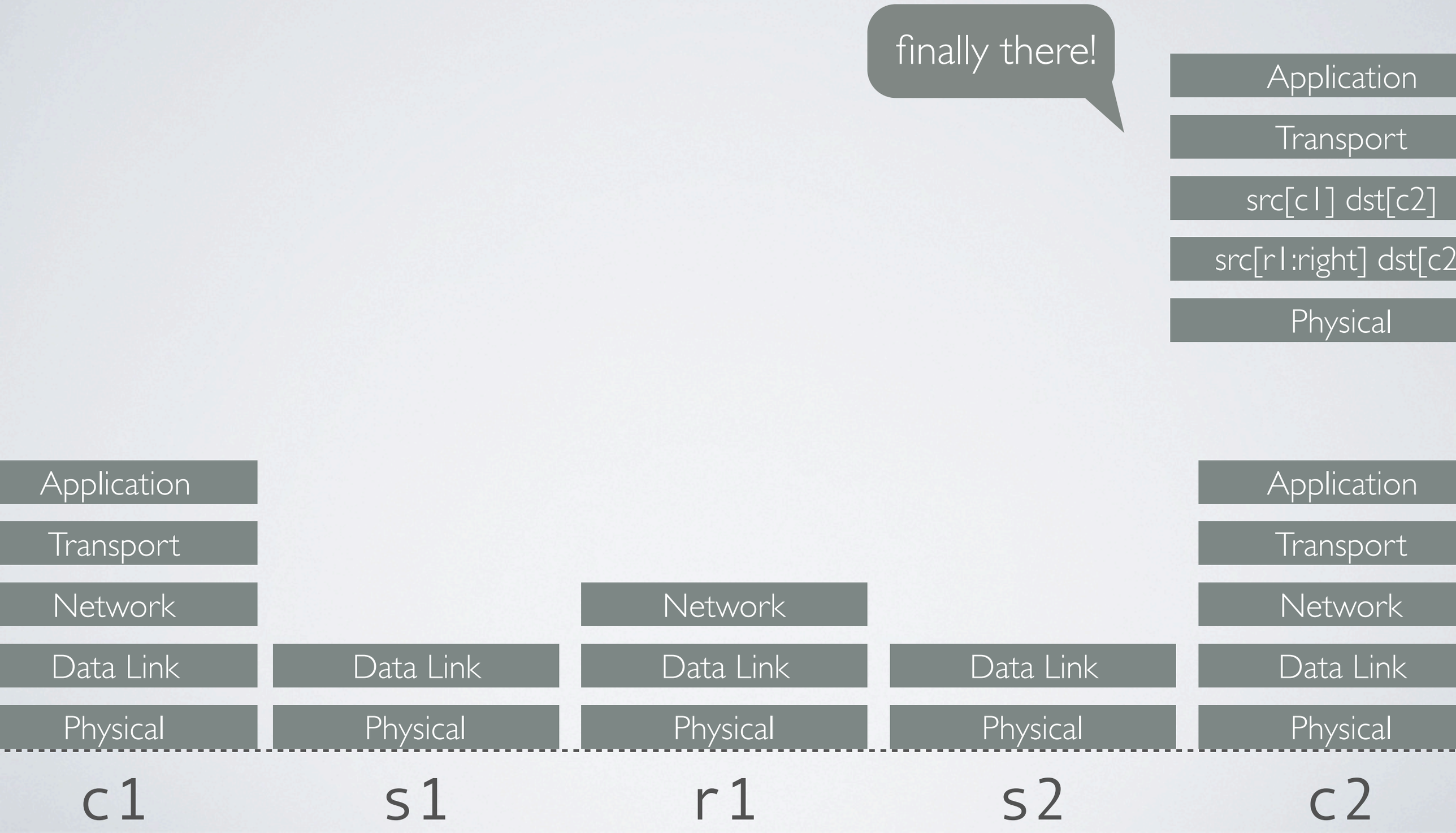
HOW A PACKET TRAVELS?



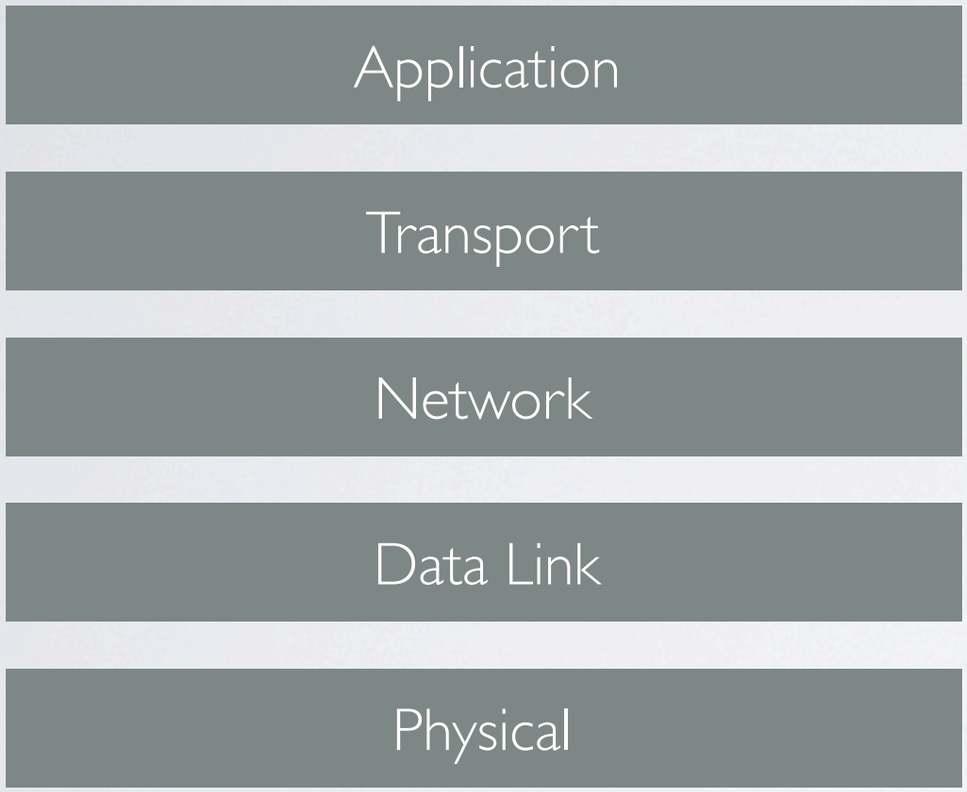
HOW A PACKET TRAVELS?



HOW A PACKET TRAVELS?

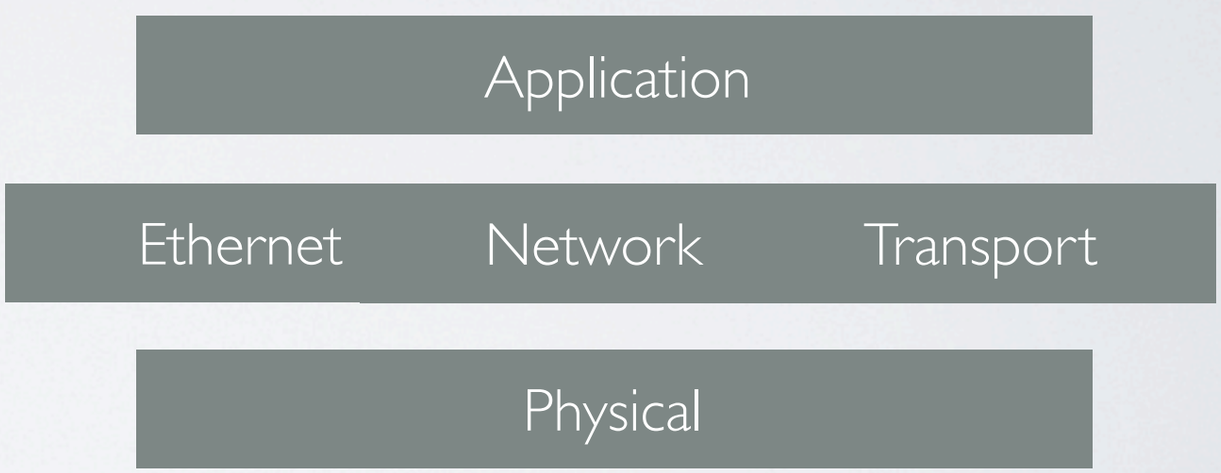


NETWORK IN OPENFLOW'S EYE



standard TCP/IP

powerful, yet less clean



OpenFlow

DEFINE "ROUTE"

- Find a way from host A to host B
- More precisely, from network A to network B
- So,
  - there should be different subnets
  - hosts should be configured with gateways
  - network is well planned

design
& reason

A LEARNING ROUTER?

- sketch:

```
for every switch:
  learn IP <--> mac of host
  learn IP <--> switch:port(position)

when a packet comes in:
  if we learned the IP:
    forward to the switch:port
    install flow entry
  else:
    flood
```

- plug-and-play, no configuration needed

A LEARNING ROUTER?

- But there's a problem:
  - We cannot handle loops
- That's why switches have *Spanning Tree Protocol(STP)*
- But STP doesn't fit for routing for lack of efficiency

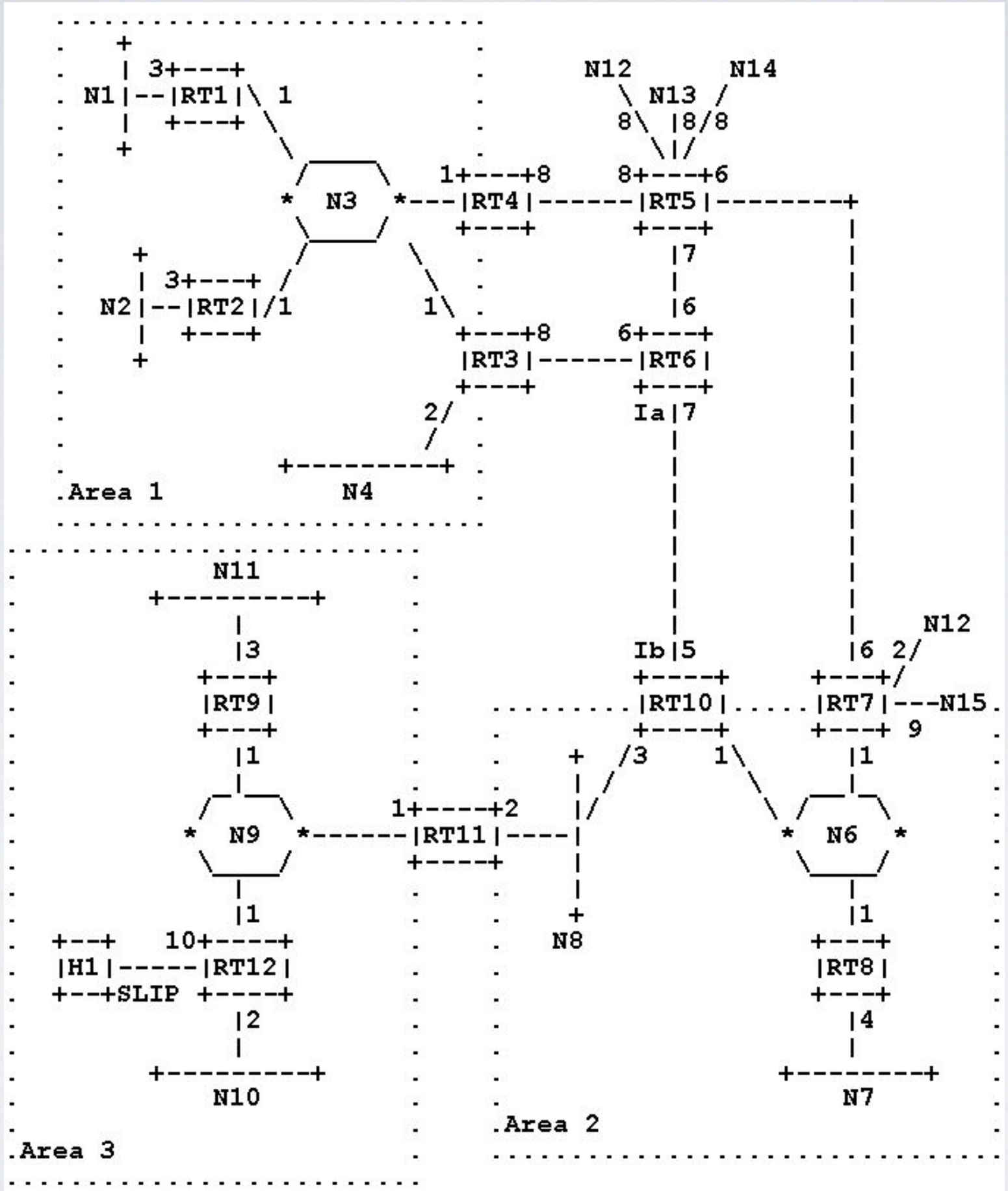
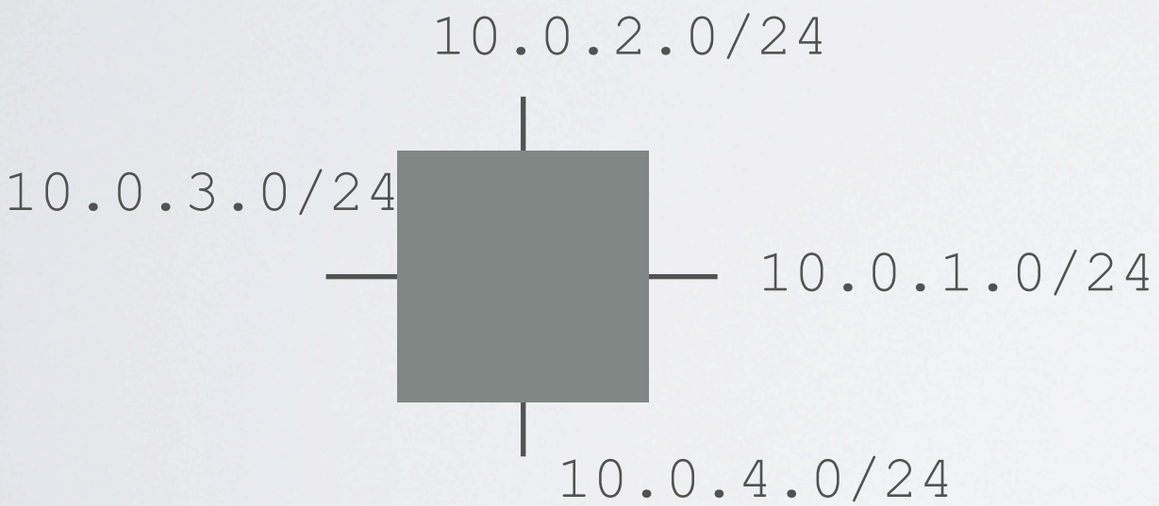


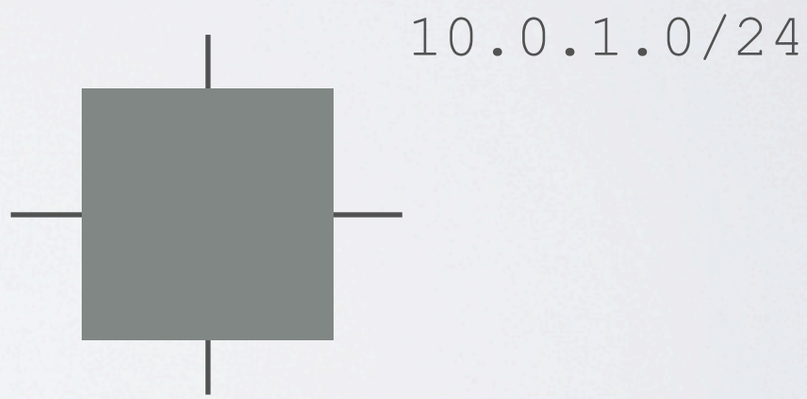
Figure 6: A sample OSPF area configuration

SWITCH OR ROUTER?

- an OpenFlow switch could act like an L2 switch, or an L3 router



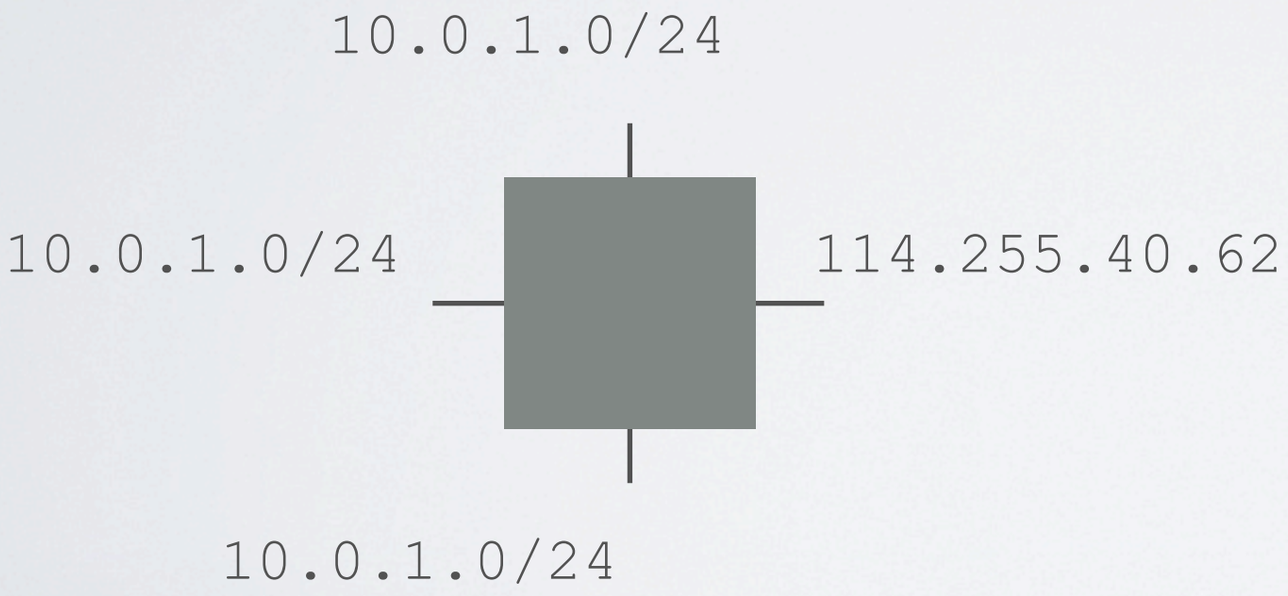
Router

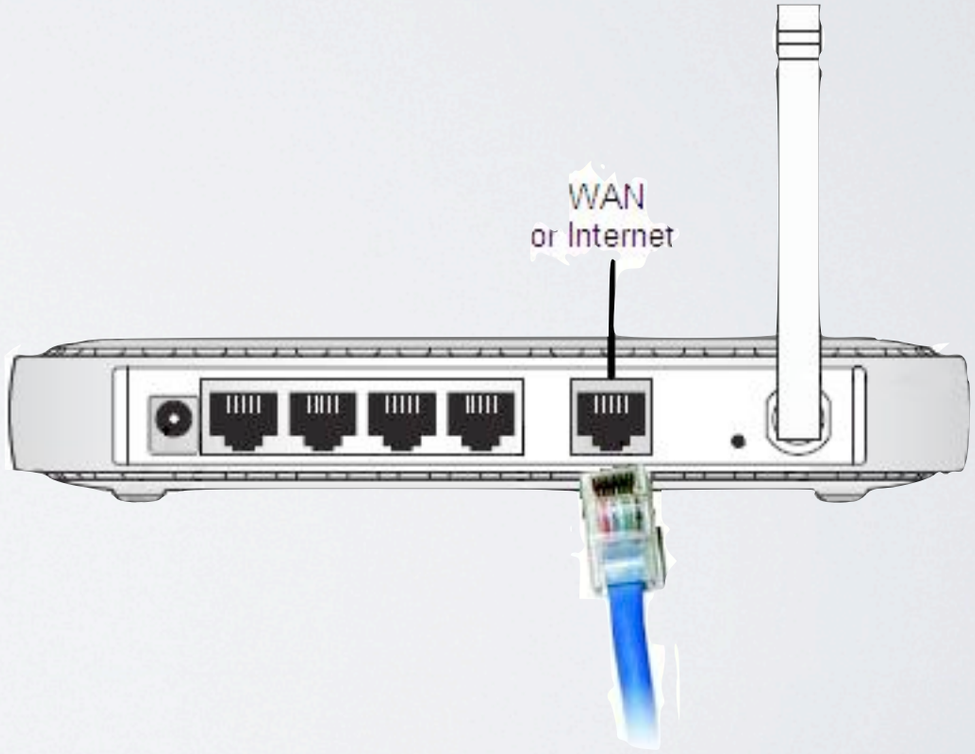


Switch

SWITCH OR ROUTER?

- what if some ports are connected to same subnet?
- need *Network Address Translation(NAT)*





DESIGN

- port-based mode configuration:
  - if the port is configured with an IP address, then the port is in ROUTER_MODE
  - if the port has no IP address, then the port is in SWITCH_MODE
- then the OpenFlow switch has 3 modes:
  - ROUTER_MODE if all ports are in ROUTER_MODE
  - SWITCH_MODE if all ports are in SWITCH_MODE
  - else HYBRID_MODE

DESIGN

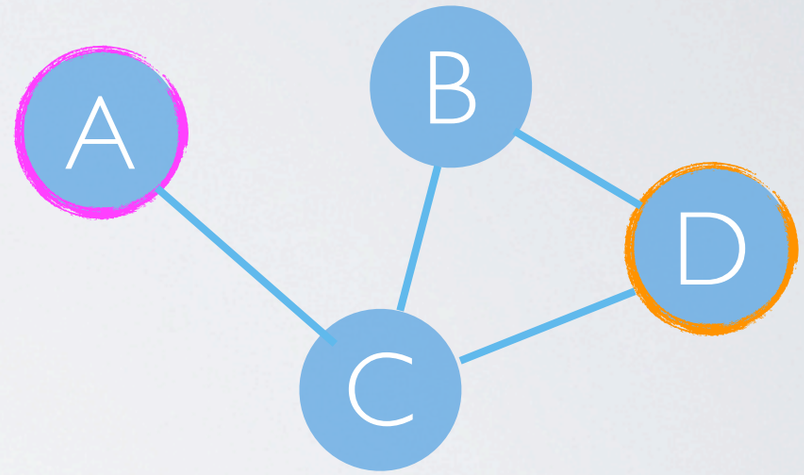
- switch could be configured through configuration file, or a telnet command line interface

- then if nothing is configured, all ports are in SWITCH_MODE and all OpenFlow switches act like normal L2 learning switches

- NAT is not supported, so the SWITCH_MODE ports in a hybrid switch have no access to the outer world

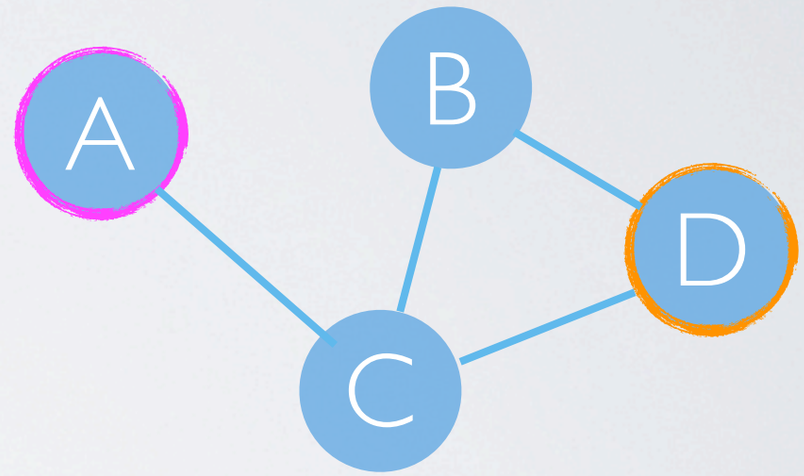
HOW & WHEN TO INSTALL FLOW ENTRY?

- when a packet_in, calculate the path to destination
- for every switch in the path, install flow entry in *reverse* order



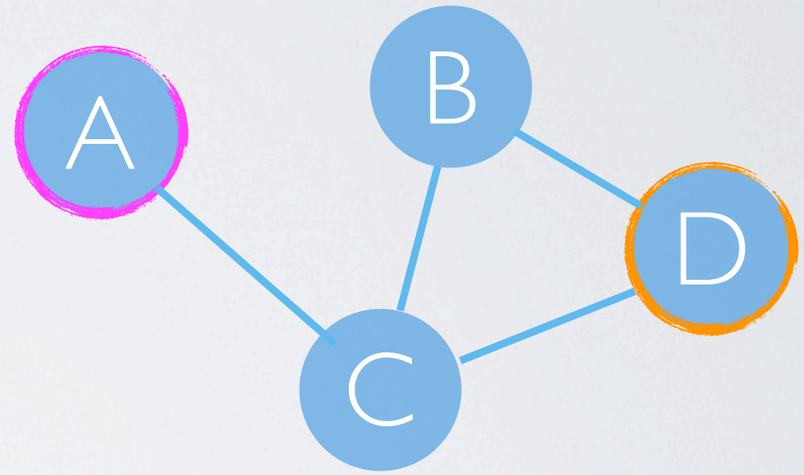
HOW & WHEN TO INSTALL FLOW ENTRY?

- but the flow entries also expire in reverse order
- this trick only puts off packet_in messages
- so no real benefits



DESIGN

- maintain a routing information data structure:
  - if routing information is up-to-date, ROUTE_CLEAN
  - else ROUTE_DIRTY
- when packet_in:
  - if ROUTE_CLEAN, install the flow entry on this switch, for next hop
  - else, re-calculate routing information, and install flow entry



ACTING LIKE A GATEWAY

- response to ARP request
  - response to ICMP Echo request(ping)
  - using ARP to get MAC address of a host
- it turns out to implement a *protocol suite*, not just "routing"

“改不动一个字”

- as you can see I tried many different ways to improve/simplify/blahblah the routing mechanism, but at last I just copied the current implementation

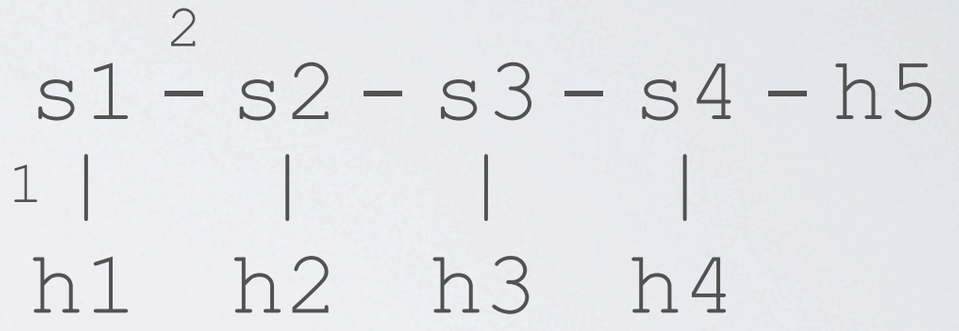
- that’s why we call it “classic”

3

demo

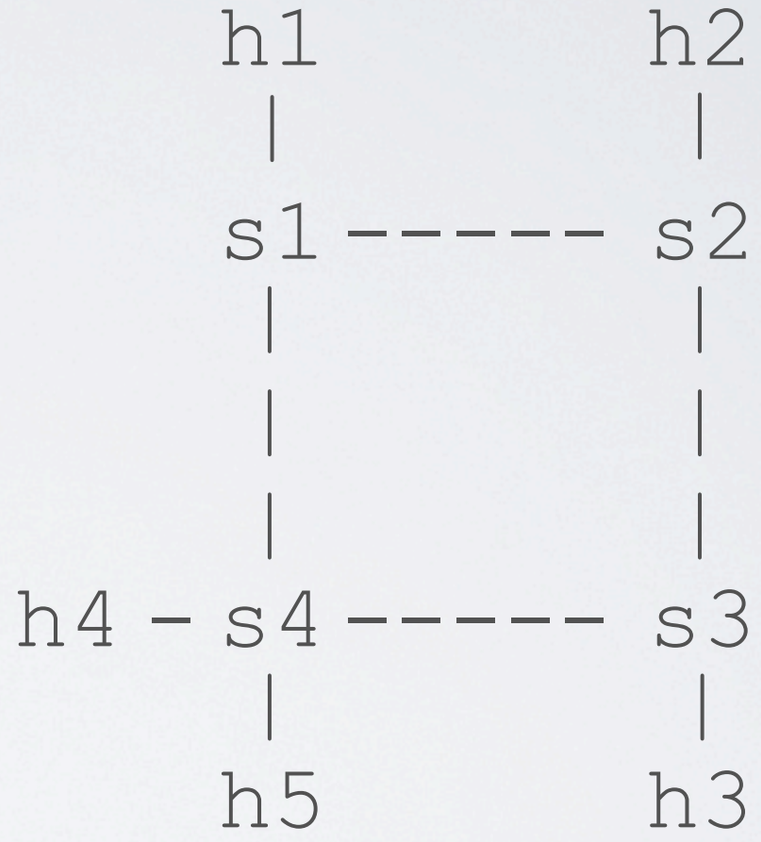
A LINEAR TOPOLOGY

- s1: router
  - 1: 10.0.1.254, 10.0.1.0/24
  - 2: 10.0.0.254, 10.0.0.0/24
- h1: 10.0.1.1
- s2-s4: switch
- h2-h5: 10.0.0.2-10.0.0.5



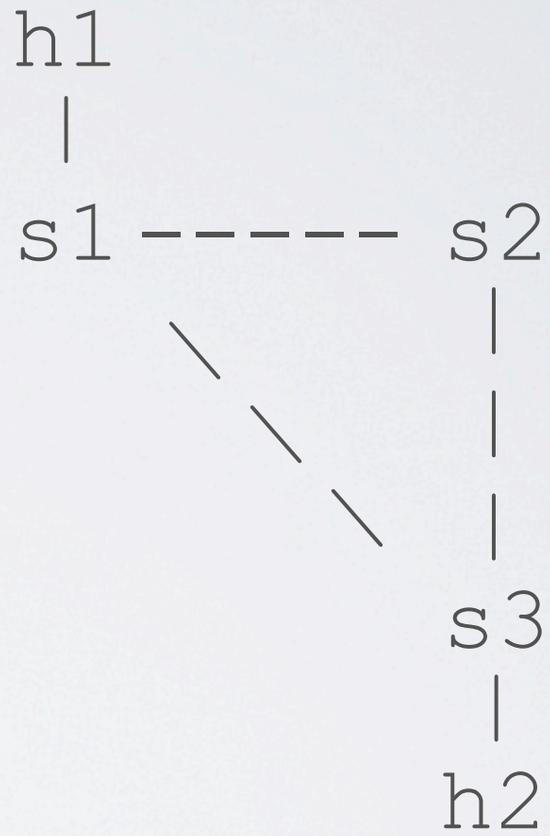
A MUCH COMPLICATED TOPOLOGY

- h1: 10.0.1.1
- h2: 10.0.2.2
- h3: 10.0.3.3
- h4: 10.0.4.4
- h5: 10.0.4.5



MANUALLY SET SWITCHES

- h1: 10.0.1.1
- h2: 10.0.2.2



4

open questions
& future plan

IS "PARTLY DIRTY" POSSIBLE?

- in the current implementation, a topology update causes the recalculation of whole network information

```
if topology_update:
  network.info_condition = ROUTE_DIRTY
... ... ...
if network.info_condition == ROUTE_DIRTY:
  network.calculate()
```

IPV6

- as I have stated, it's actually implementing a protocol suite
- including NDP, ICMPv6, and of course IPv6

> someone who did. I think the major parts of the problem are: 1) Adding the OpenFlow extension(s), 2) Proposing how to handle IPv6 addresses in pox.lib.addresses and implementing that, 3) Adding an ipv6 class to pox.lib.packet.

CODE

- https://github.com/cannium/openflow_routing
- running with *POX*, test environment is *mininet*
- definitely need more tests !! so play with it !!

Q&A