Reviewing End-to-End Operation

CS168

Sarah McClure and Sylvia Ratnasamy

Fall 2022

Plan

- Combination of review and hands-on illustration
- Start with end-to-end operation
  - Focus on how L2 and L3 co-exist
- If time permits, will review HTTP and DNS
  - Steps involved in downloading a webpage

L2 and L3 together

- Simplified Internet for today's review:
  - IPv4 (L3) network as a collection of Ethernet (L2) networks
  - Ethernet networks may be switched or shared

- L2 and L3 have separate addressing, routing, forwarding

- We'll review how, when, and why L2 and L3 work together

L2 and L3 have separate addressing

- MAC (L2) addresses
  - Hard-coded ("burned in") by device manufacturer
  - Not aggregation-friendly
  - Portable, and can stay the same as the host moves (*topology independent*)
  - Used to get packet between interfaces on the same L2 link/network

- IP (L3) addresses
  - Assigned by network operators; configured or learned dynamically (DHCP)
  - Hierarchical structure and allocation allows aggregation
  - Not portable and depends on where the host is attached (*topology dependent*)
  - Used to get the packet to the destination IP "subnet"

Questions (1)

- Why do we need both L3 and L2 addresses?
  - Consider the following questions
- Why not use MAC addresses at L3?
  - Can’t aggregate, so can’t possibly scale
- Why not use IP addresses at L2?
  - Bootstrapping problem!
  - Need to assign addresses, but can’t reach host without it having a local address
- Why do we need both L2 and L3?
  - L2 and L3 have different tasks, and different constraints
  - One needs to scale, other requires no configuration

Bootstrap and discovery

- A host A is “born” knowing only its MAC address
- Must discover some information before it can communicate with a remote host B
- What is my (A’s) IP address?
  - DHCP
- What is B’s IP address?
  - DNS
- What is B’s MAC address? (if B is local)
  - ARP
- What is my first-hop router’s IP address (needed if B is remote)
  - DHCP
- What is my first-hop router’s MAC address?
  - ARP

ARP and DHCP

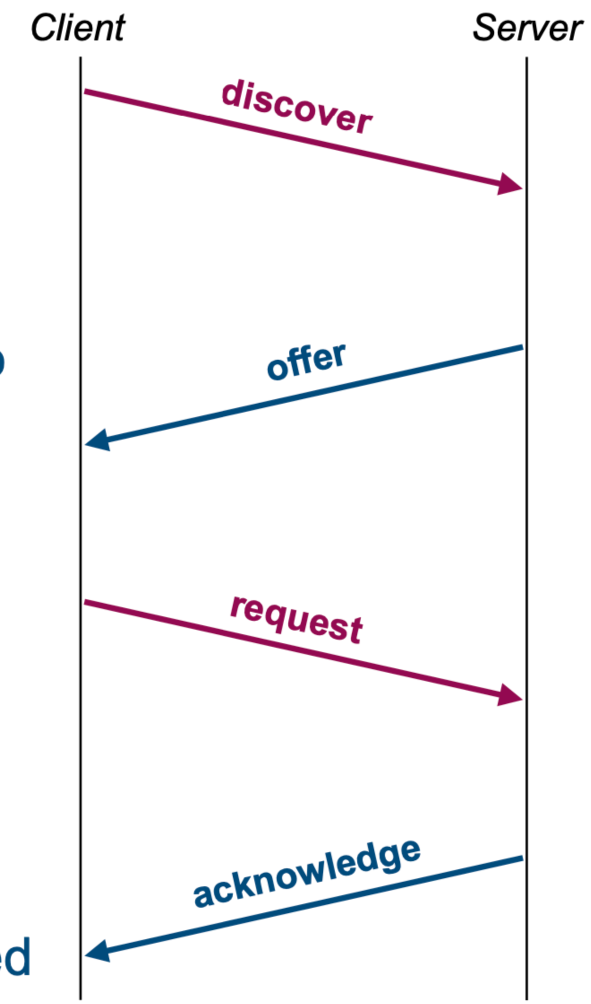
- Discovery protocols
  - ARP → Address Resolution Protocol
  - DHCP → Dynamic Host Configuration Protocol
  - Confined to the host's local L2 network
  - Rely on broadcast capability (as most discovery protocols)

- ARP
  - Initiating host broadcasts query: "*Who has IP address* w.x.y.z"?
  - Host with w.x.y.z responds (unicast): "*I am* w.x.y.z *and my MAC address is* a1:b2:c3:d4:e5:f6"

- DHCP
  - Used by a host to learn (bootstrap itself) about its L3 context
  - Discovers its IP address, netmask, IP address of first-hop router, IP address of its DNS resolver

DHCP

- Client sends a **discover message** — asks for config info
- Server(s) send(s) **offer message** with config info (e.g., particular IP)
- Client sends **request message** to accept a particular offer
- Server sends **acknowledge message** to confirm request granted



- DHCP uses UDP (built on IP)
- Client doesn't know DHCP server address!
- Client doesn't have its own address yet
- So, must broadcast
- That is, uses broadcast IP address as destination
  - 255.255.255.255
- And broadcast Ethernet address as destination
  - FF:FF:FF:FF:FF:FF

Key ideas in both ARP and DHCP

- **Broadcasting**: can use broadcast to make contact
  - Scalable because of limited size

- **Caching**: remember results for a while
  - Store the information you learn to reduce overhead
  - Associate a time-to-live field with the information
  - … and either refresh or discard the information
  - Key for robustness in the face of unpredictable change

Questions (2)

- Difference between broadcast and flood?
  - Broadcast is a <u>communication abstraction</u> for delivery to everyone (within some scope)
  - Flooding is a <u>mechanism</u>: e.g., may be used to implement broadcast; but also unicast for unknown dst
- What communication abstractions have we studied?
  - Unicast, multicast, anycast, broadcast
- What packets does a host see? (i.e., what packets will its NIC pass up to the host OS)
  - Packets that match the host's MAC address OR match the broadcast address
  - Note: host OS will discard packets that don't match IP address of host (or broadcast)
- Is ARP an L2 or L3 protocol?
  - Controversial but general consensus is L2 (though it takes L3 address as parameter)

L2 and L3 have separate forwarding and routing

- **Ethernet (L2):** Learning switches, STP, flooding, *etc.* in switched Ethernet
  - Routing may be “trivial” in the case of shared media Ethernet

- **IP (L3):** Configured (static, default) or learned via routing protocols (OSPF, BGP, etc)

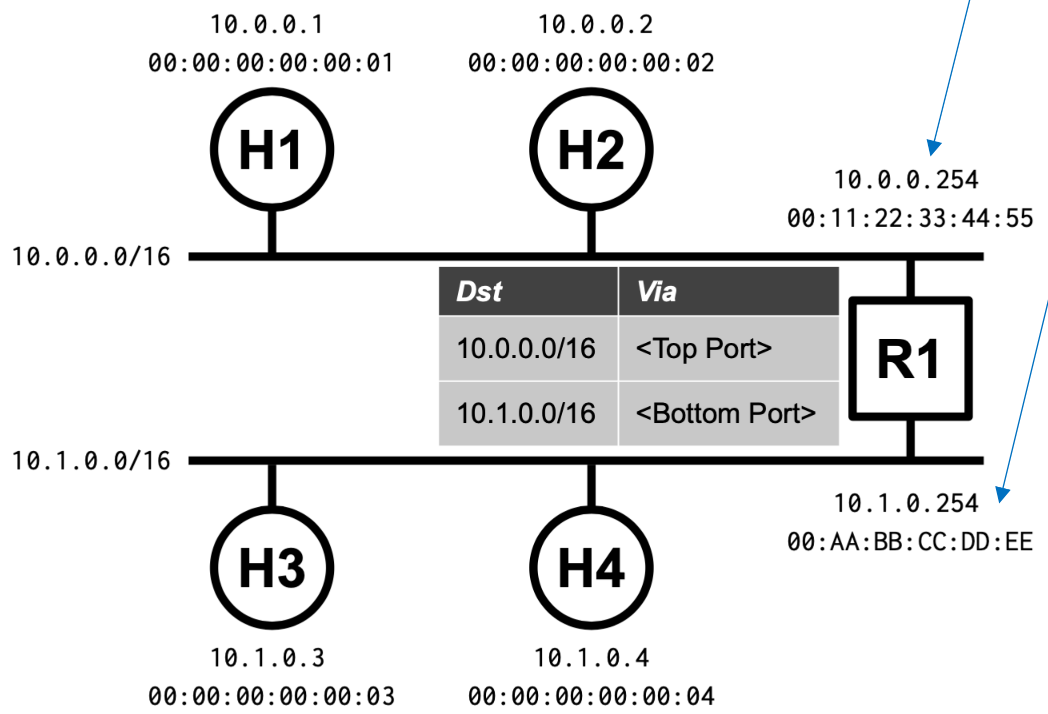
- A packet contains both an L2 and L3 header, with L2 and L3 addresses respectively
  - L3 addresses generally stay the same throughout (recall: L3 is global)
  - L2 addresses change as packet moves from one L2 network to another (recall: L2 is local)

- A packet generally forwarded based on L2 and L3
  - Each relevant at different points in the packet’s journey
  - Examples coming up

What about routers?

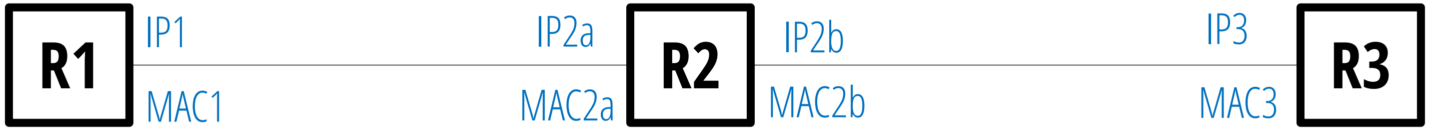
- Generally, have an L2 and L3 address *per interface*
- A router might interconnect two Ethernet networks (Murphy's example)

Recall, Murphy's lecture:



What about routers?

- Generally, have an L2 and L3 address *per interface*
- A router might interconnect two Ethernet networks (Murphy's example)
- Two routers might be interconnected by an Ethernet "network"
  - E.g., MAC1 and MAC2a on one Ethernet network; MAC2b and MAC3 on another



What about routers?

- Generally, have an L2 and L3 address *per interface*
- A router might interconnect two Ethernet networks (Murphy’s example)
- Two routers might be interconnected by an Ethernet “network”
  - E.g., MAC1 and MAC2a on one Ethernet network; MAC2b and MAC3 on another

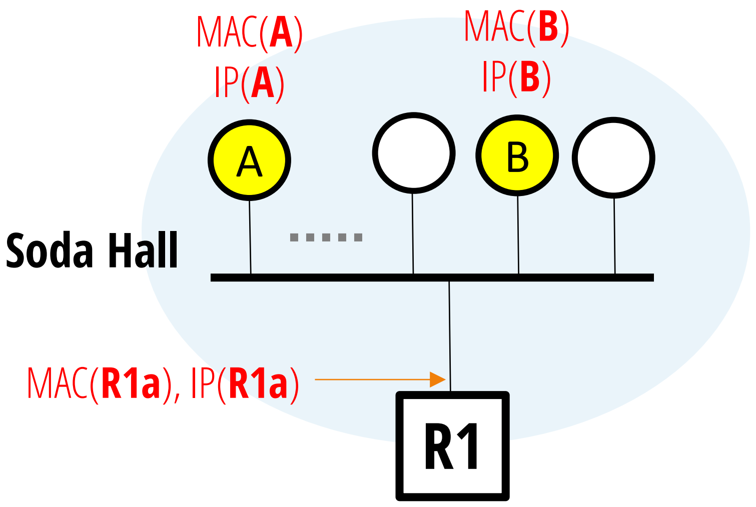
- More questions
  - Where do routers get their IP address? (configured or DHCP)
  - How do routers discover their neighboring router’s L2/L3 address? (config., ARP, other specialized protocols)

Which Layer Does What?

- App hands data to L4, specifies destination address
- L4 packetizes data, hands packet(s) to L3
- L3 decides if destination is on same network
  - Identifies whether next-hop IP is destination or router
- L3 hands packet to L2
  - With next-hop IP address as parameter
- L2 ARP resolves this IP address into MAC address

Let’s work through an example …

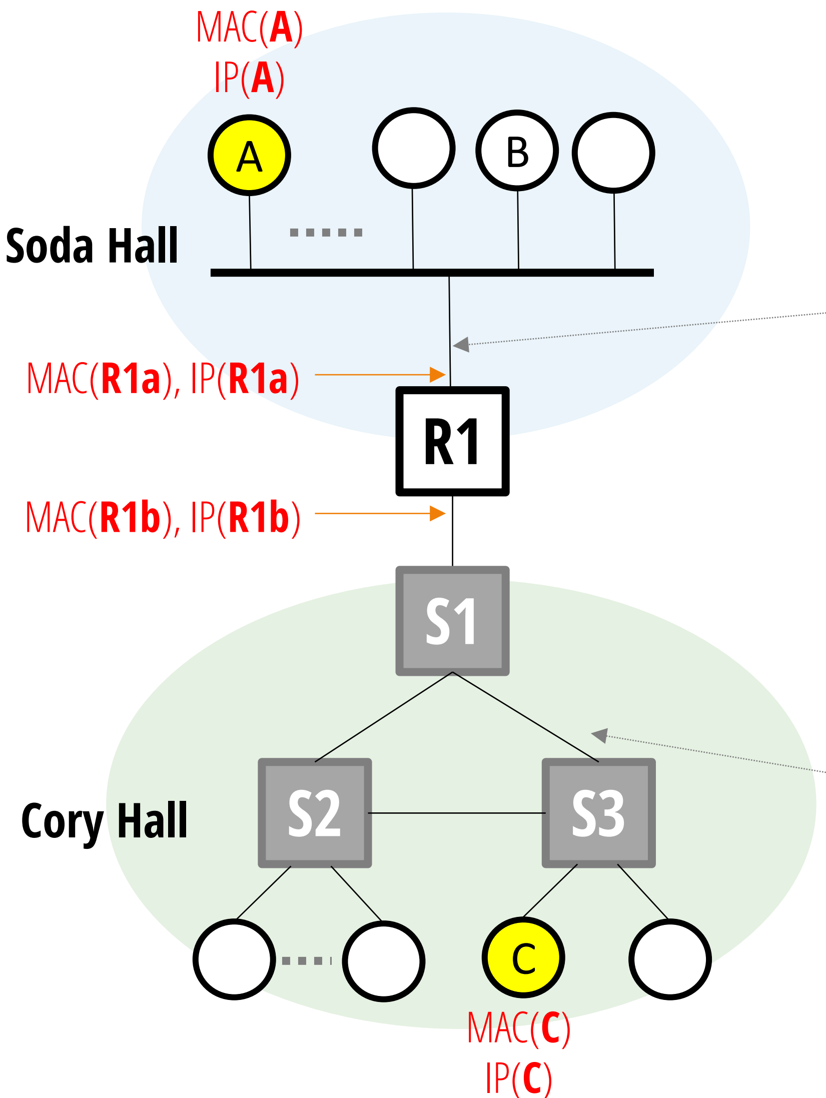
Putting it all together



Scenario#1: Host A sends a packet to host B

- What path does the packet take?
  - A to B directly over the shared Ethernet
- L2 and L3 addresses in the packet?
  - Destination: L3 = IP(B), L2 = MAC(B)
  - Source: L3 = IP(A), L2 = MAC(A)
- Who made the "routing" decision?

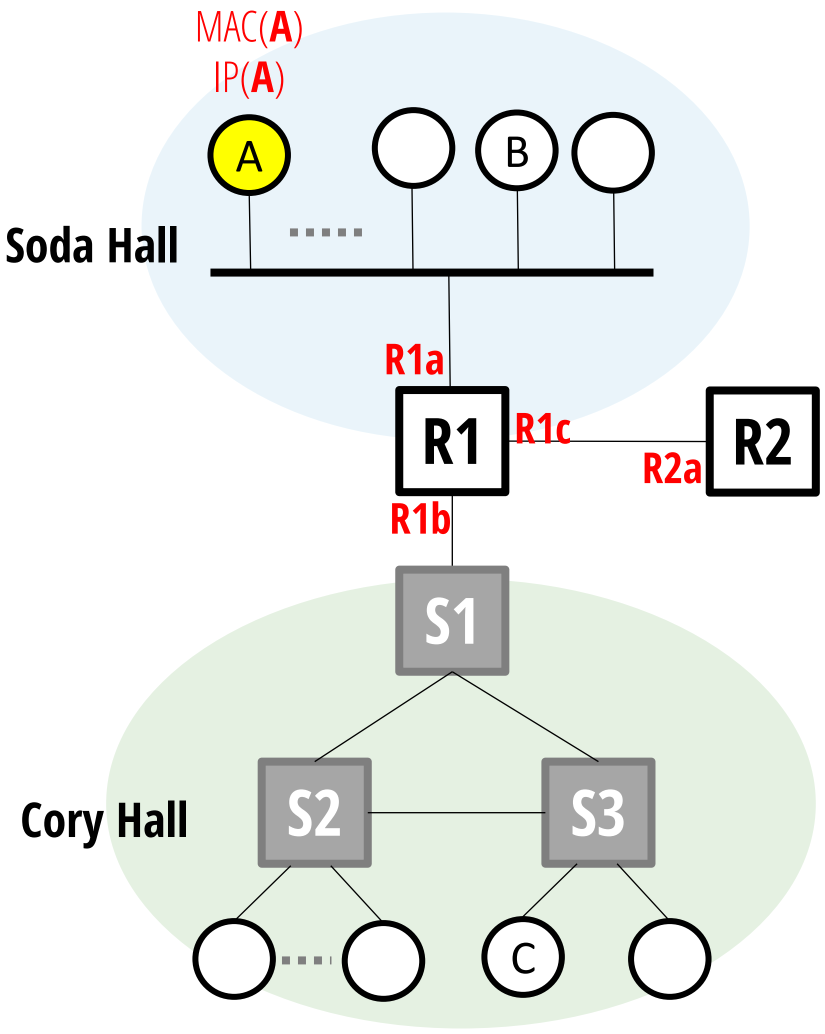
Putting it all together



Scenario#2: Host A sends a packet to host C

- What path does the packet take?
  - A → R1 → S1 → S3 (say) → C
- L2 and L3 addresses in the packet when it arrives at R1?
  - Destination: L3 = IP(C), L2 = **MAC(R1a)**
  - Source: L3 = IP(A), L2 = MAC(A)
- Based on what address does R1 make its forwarding decision?
  - IP(C)
  - What routing entries does R1 have and how did they get there?
    - Static routes for prefixes corresponding to IP(R1a) and IP(R1b) [Soda & Cory subnets]
- L2 and L3 addresses in the packet when it arrives at S3?
  - Destination: L3 = IP(C), L2 = **MAC(C)**
  - Source: L3 = IP(A), L2 = **MAC(R1b)**

Putting it all together

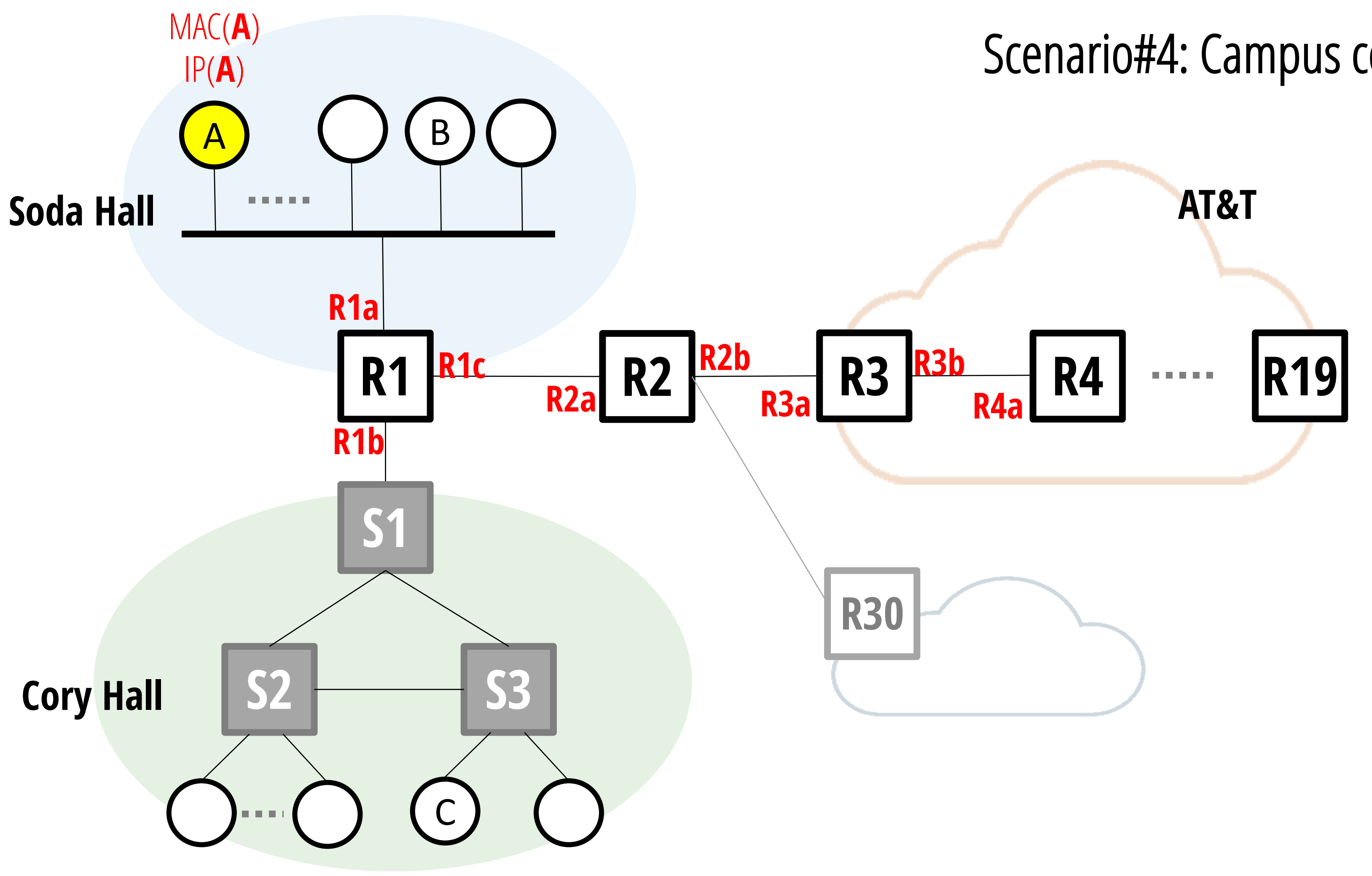


Scenario#3: EECS connects R1 to campus router R2

- (assume R2 is connected to other routers but R1 isn't)
- What static route(s) might we want to add to R1's routing table?
- What static route(s) might we want to add to R2's routing table?
- How might we avoid configuring static routes?

Putting it all together

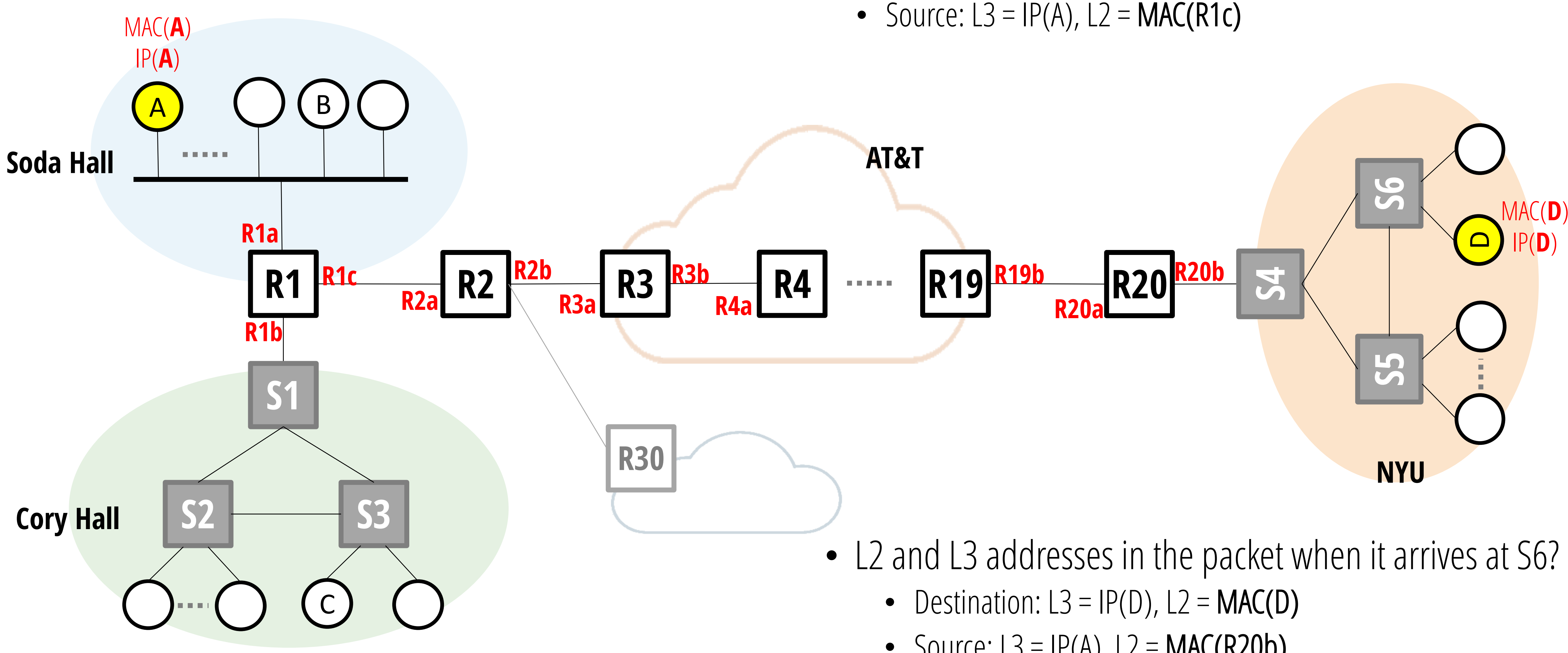
Scenario#4: Campus connects R2 to an AT&T router R3



Putting it all together

Scenario#5: Host A sends a packet to Host D

- L2 and L3 addresses in the packet when it arrives at R2?
  - Destination: L3 = IP(D), L2 = **MAC(R2a)**
  - Source: L3 = IP(A), L2 = **MAC(R1c)**



- L2 and L3 addresses in the packet when it arrives at S6?
  - Destination: L3 = IP(D), L2 = **MAC(D)**
  - Source: L3 = IP(A), L2 = **MAC(R20b)**

Questions?

Sarah’s demo

DNS: Quick review

- Why we need it? Convert names to IP addresses

- Design based on three intertwined hierarchies
  - **Naming structure**: names are hiearchical (cs.berkeley.edu)
  - **Management**: hierarchy of authority over names
  - **Infrastructure**: hierarchy of DNS servers
  - What are some pros and cons of this?

- Names are “resolved” by starting at the root and querying down the hierarchy

- Availability / scalability / performance: via partitioning, replication, caching

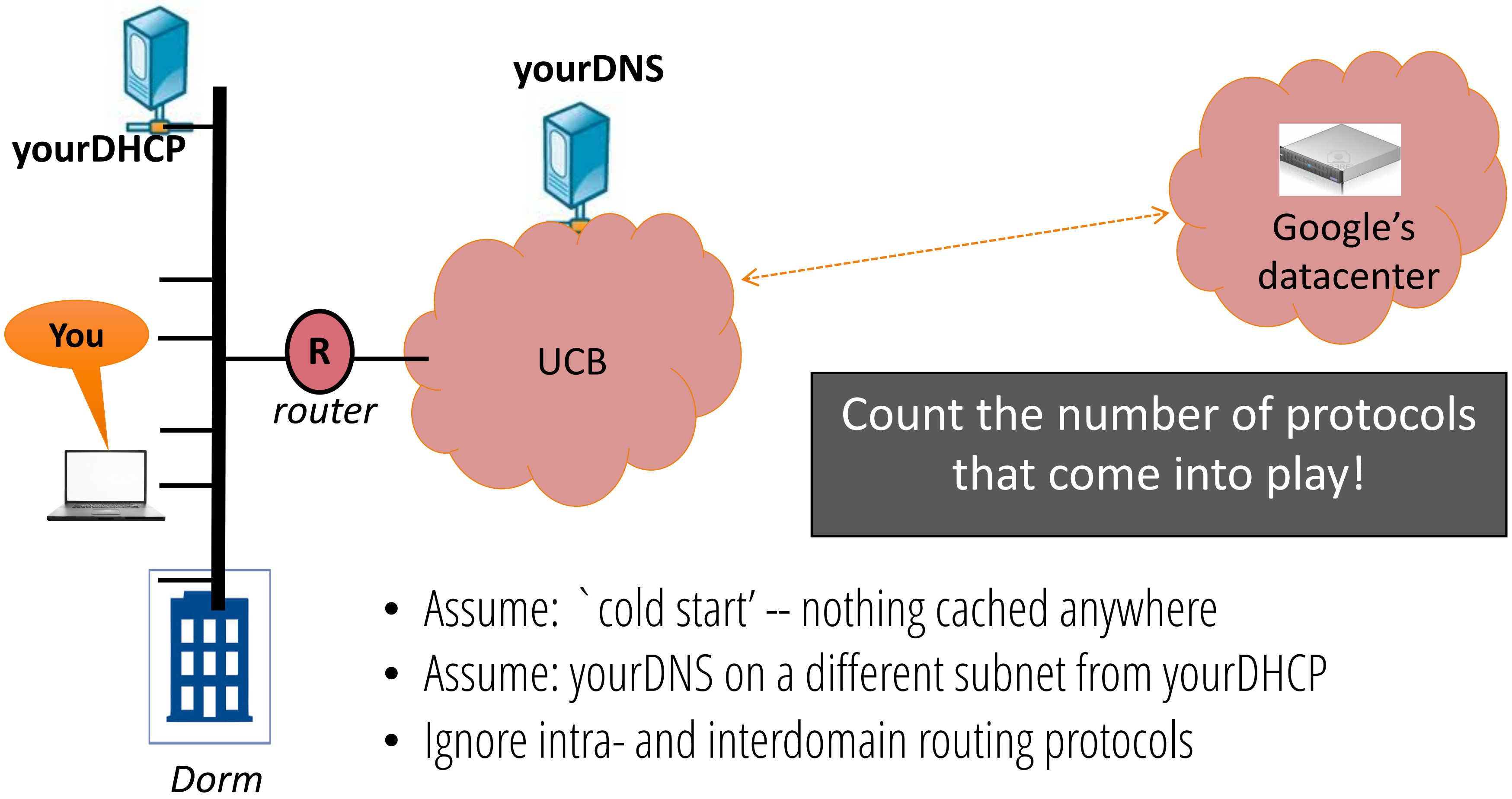
HTTP / Web: Quick Review

- Essential components:
  - **HTML**: content with links
  - **URL**: reference to content (lot going on in a URL! – protocol, name, location, resource, parameters...)
  - **Infrastructure**: Client browsers and Web servers
  - **HTTP**: protocol used to fetch content from servers

- Availability, scalability, performance
  - Caching: at browser and forward/reverse proxy servers controlled by HTTP caching directives
  - CDNs: 3rd-party entity that replicates/caches/serves your content using their infrastructure
  - TCP optimizations: concurrent, persistent, pipelined connections amortize TCP setup overhead

Putting the pieces together (again)

Walk through the steps required to download www.google.com/index.html from your laptop



Count the number of protocols that come into play!

- Assume: `cold start' -- nothing cached anywhere
- Assume: yourDNS on a different subnet from yourDHCP
- Ignore intra- and interdomain routing protocols

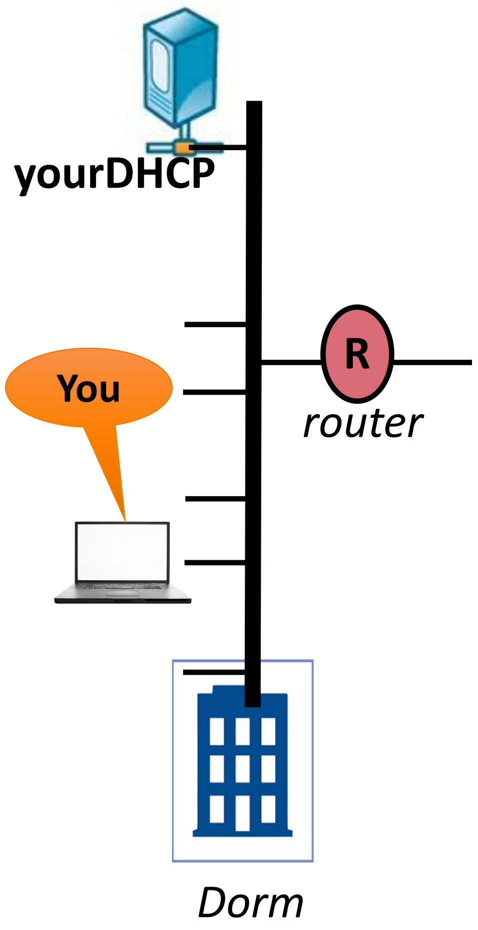
Step 1: Self discovery

- You use DHCP to discover bootstrap parameters
  - your IP addr (u.u.u.u)
  - your DNS server's IP (u.dns.ip.addr)
  - R's IP address (r.r.r.r)
  - ..

- Exchange between you and yourDHCP

Ethernet	IP	UDP	DHCP

- Protocol count = 4



Next...

- You are ready to contact www.google.com

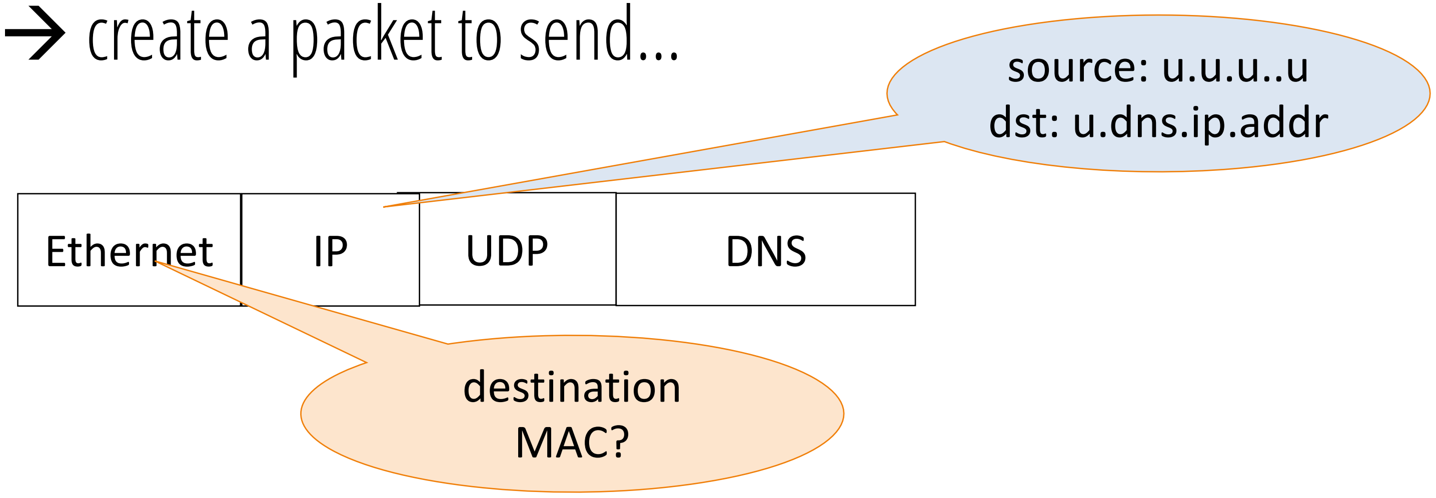
→ need an IP address for www.google.com

→ need to ask google's DNS server

→ need to ask my DNS server to ask google's DNS...

→ I know my DNS server's IP addr is u.dns.ip.addr

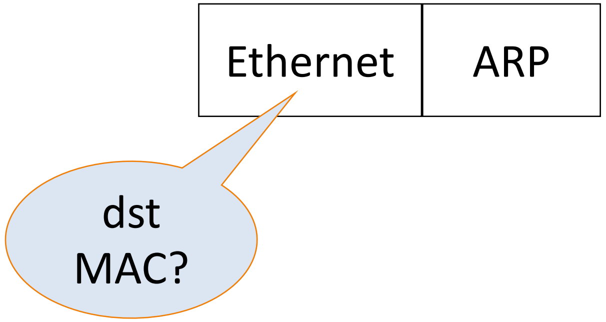
→ create a packet to send...



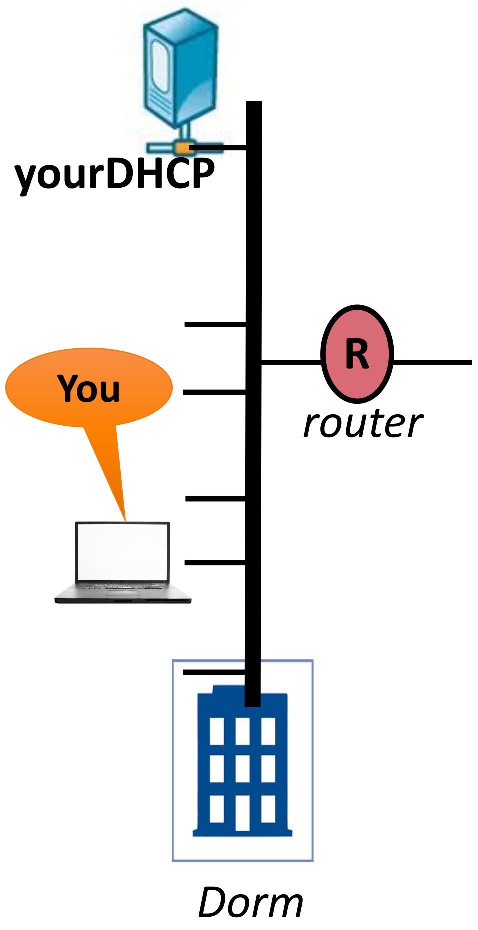
Step 2: Getting out the door

- You use ARP to discover the MAC address of R

- Exchange between you and R

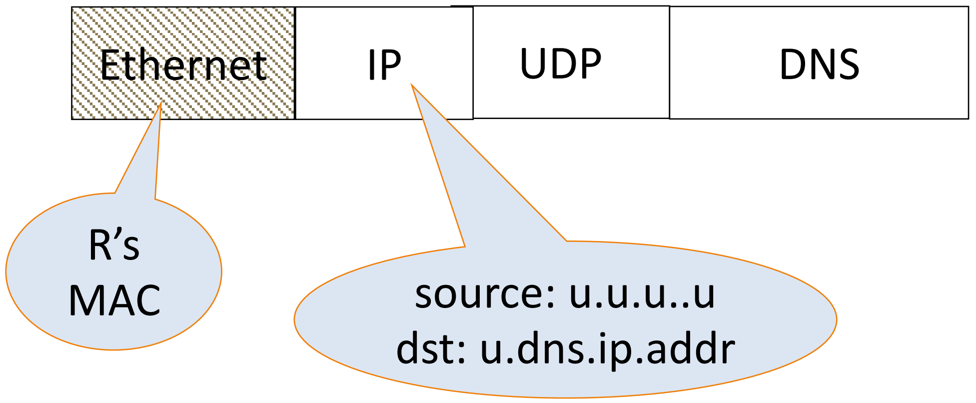


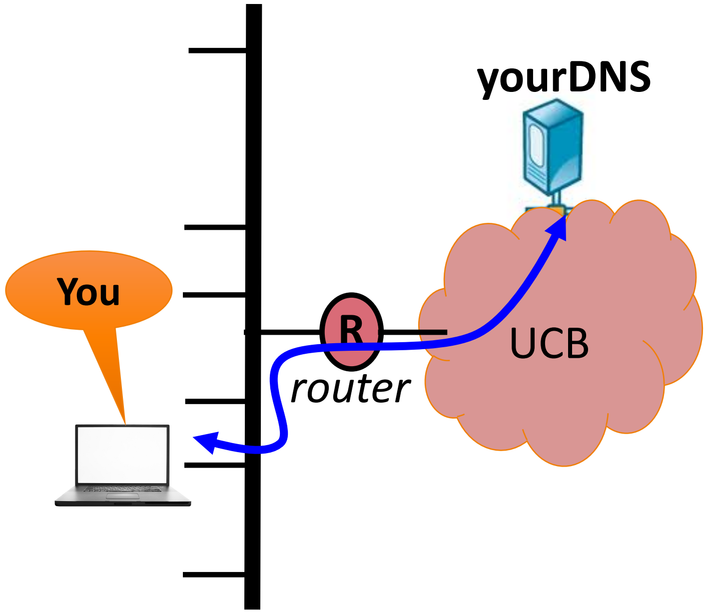
- Protocol count = 5



Step 3: Send a DNS request

- Exchange between you and yourDNS
- Now ready to send that packet

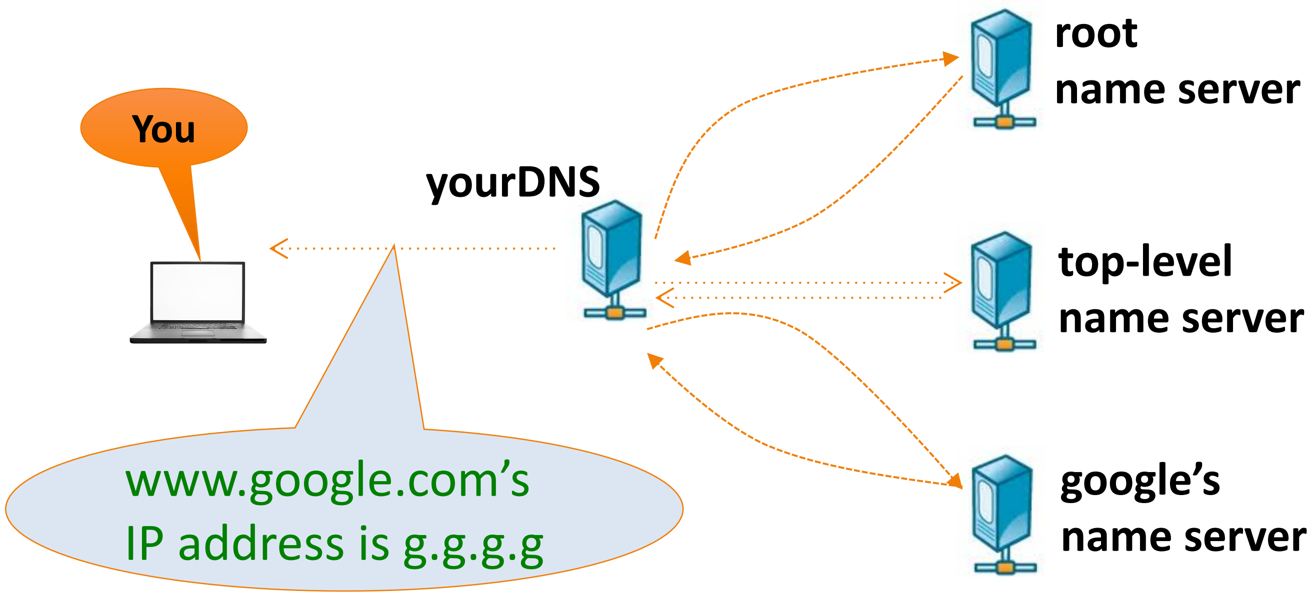




- Protocol count = 6

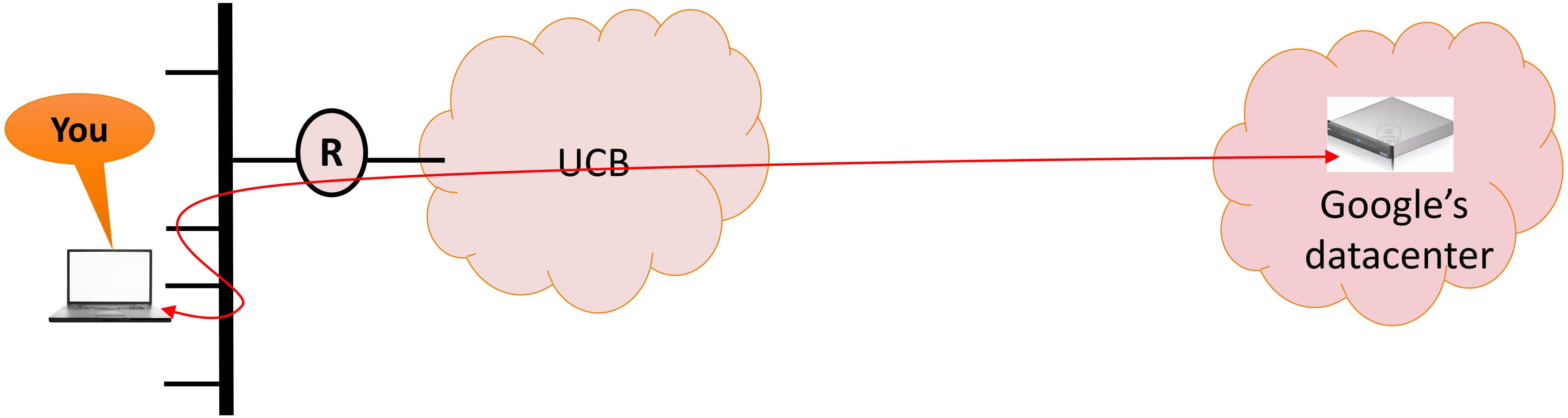
Step 4: yourDNS does its thing

- yourDNS resolves www.google.com

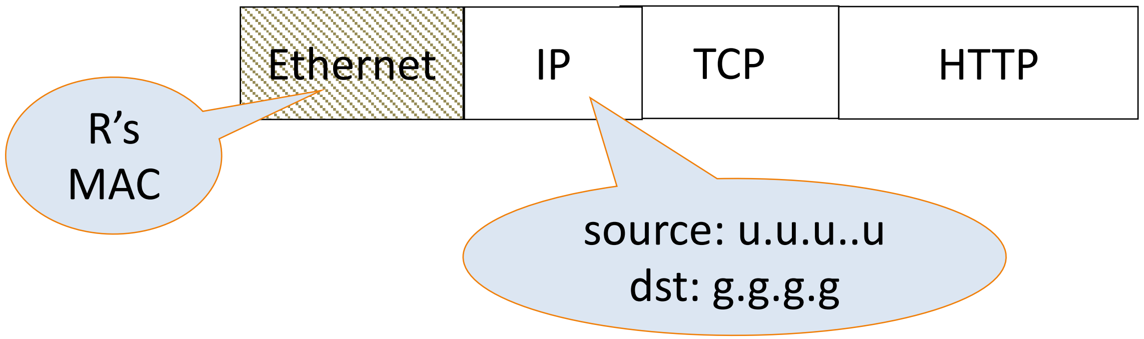


- Protocol count = 6

Step 5: Getting the content (at last)



- Exchange between you and google's server at g.g.g.g



- Protocol count = 8

Recap: Name discovery/resolution

- MAC addresses?
  - my own: hardcoded
  - others: ARP (given IP address)

- IP addresses?
  - my own: DHCP
  - others: DNS (given domain name)
    - how do I bootstrap DNS communication? (DHCP)

- Domain names?
  - search engines

Sarah’s demo#2

Questions?

Backup

Putting it all together