CS168: Final Review

...

Intro to the Internet
Spring 2024

Agenda

- Topical Review [~1 hour]
- Final Exam Practice Questions [~50 minutes]

- Disclaimer: this review session is **not comprehensive**
  - Overview will only cover topics starting from Datacenters
    - All pre-Datacenters topics are covered in the Midterm Review
  - “Lectures 1-25, discussions 1-13, and all projects are in scope” - Ed

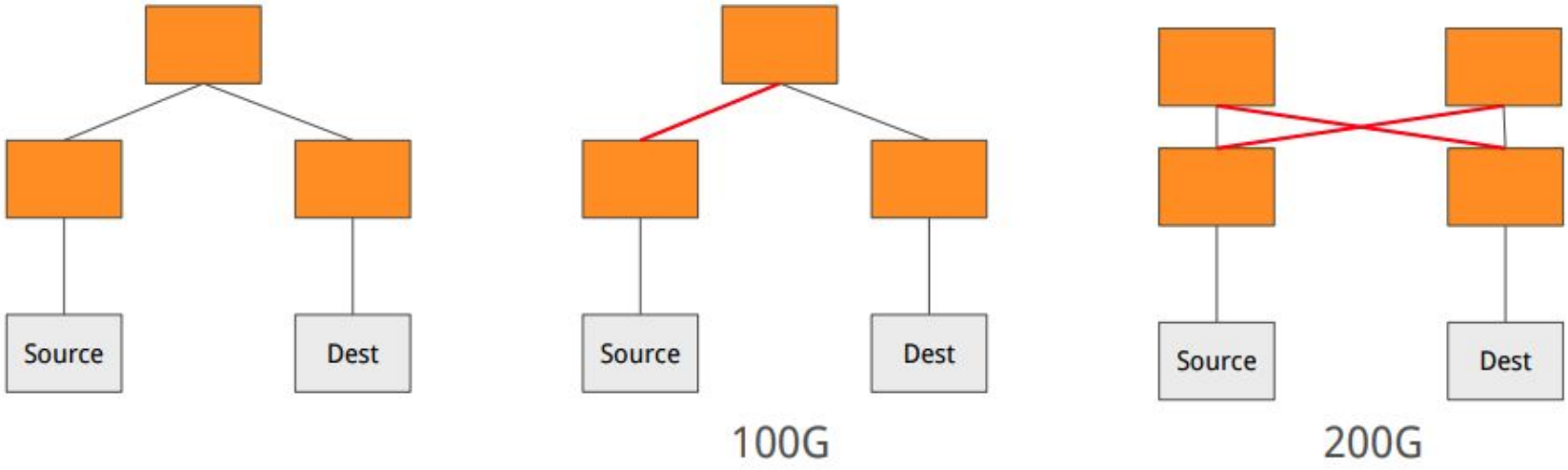
Datacenters

Datacenters

- How does one design datacenter networks?
- Consider new assumptions (i.e., different from what we learned in Internet).
  - **Single administrative control** over the network topology, traffic, and to some degree end hosts
  - Much more **homogenous**
  - Strong emphasis on **performance**
  - Few backwards compatibility requirements, **clean-slate solutions** welcome!

Bisection Bandwidth

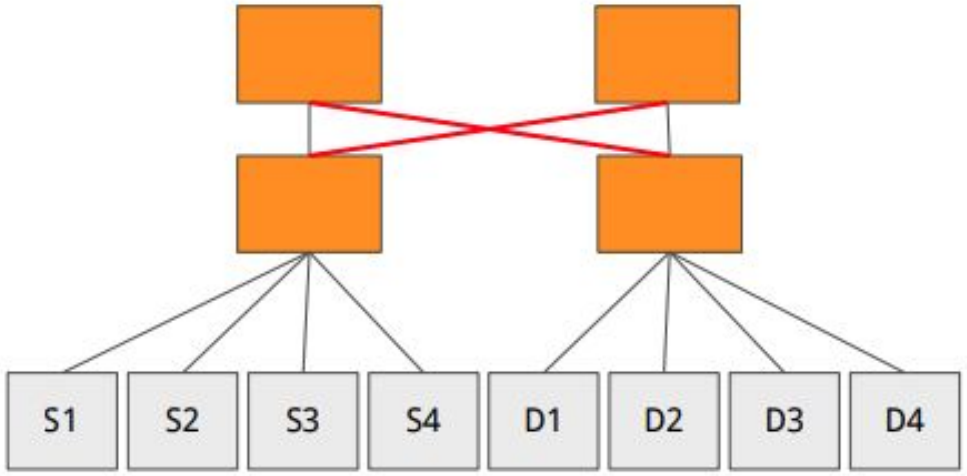
- We want a network with high bisection bandwidth:
  - Pick the number of links we must cut in order to partition a network into two halves
  - Bisection bandwidth is the sum of those bandwidths.



Bisection Bandwidth

- Full bisection bandwidth: Nodes in one partition can communicate simultaneously with nodes in the other partition at full rate.
  - Given N nodes, each with access link capacity R, bisection bandwidth = N/2 x R
- Oversubscription, informally, how far from the full bisection bandwidth we are
  - Formally: ratio of worst-case achievable bandwidth to full bisection bandwidth.

Bisection Bandwidth



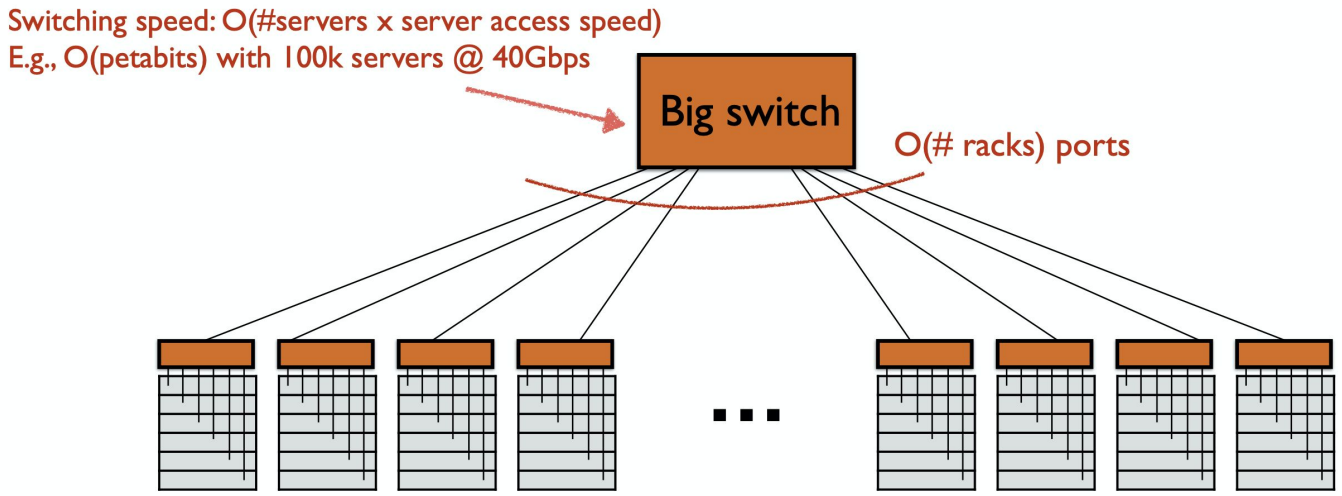
Bisection Bandwidth: 200G

Full Bisection Bandwidth: (8/2)*100G = 400G

Oversubscription: 200/400 = 2x

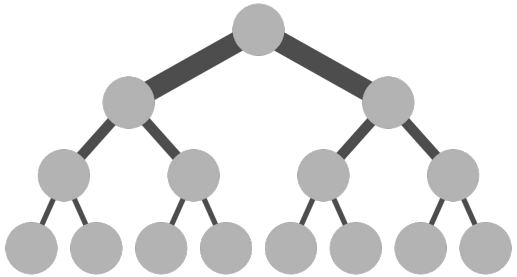
Big switch abstraction

- We want an abstraction of big switch;
- Naive solutions:
  - Server-to-server full-mesh;
  - A physical big switch?
- Either impractical or too costly ($$$); 10k hosts * 10k hosts = ~100M links
- Can we do better?

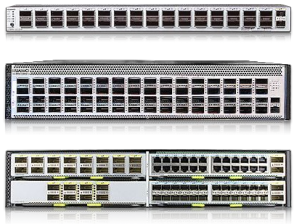


Design 1: Fat tree (scale-up)

- Borrowed straight from the High Performance Computing (i.e., supercomputers) community
- Use of big, non-commodity switches



- Problem? Scales badly in terms of cost
  - ..only a few switch vendors can make big switches
- Scales badly in terms of fault-tolerance

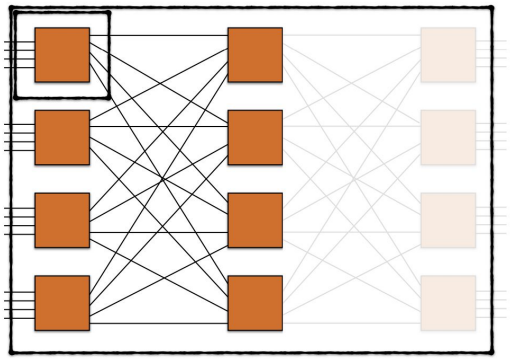


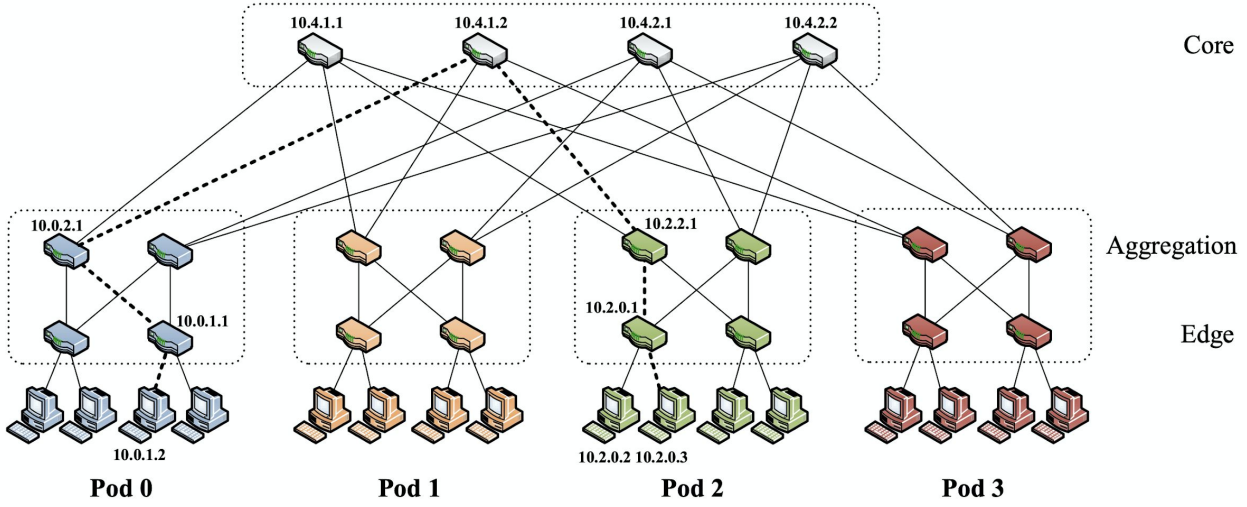
small switch

big switch

Design 2: Clos (scale-out)

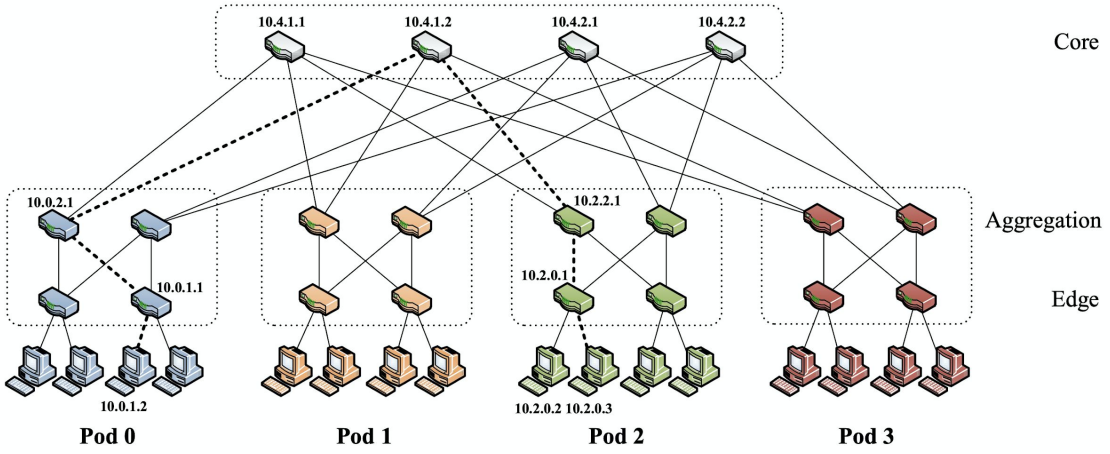
- Replace nodes in the fat tree with groups of cheap commodity switches
  - cheaper
  - high redundancy (bisection width)
- Allows *oversubscription* ratio of 1





A Closer look at Clos

- The concept of redundancy and the clos topology itself have many variations!
- We focus on the 3-tiered clos topology
- Homogenous switches (k ports) and links
- Each non-core switch has k/2 ports pointing north and k/2 pointing south

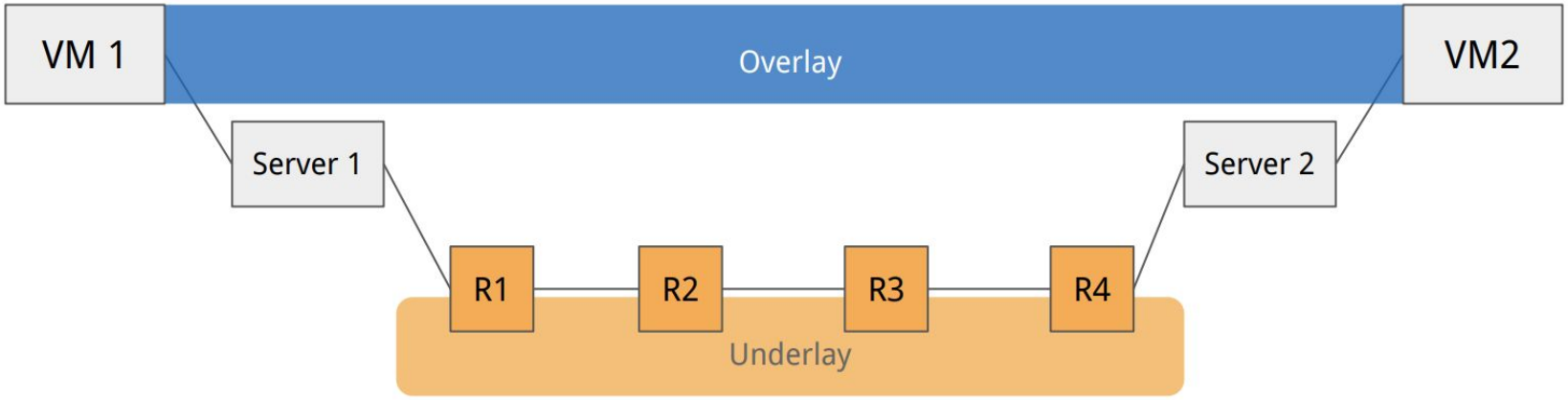


ECMP

- ECMP - Equal Cost Multi-Path
  - Goal: use multiple paths in network topology that are equal cost
  - Idea: Load-balance packets across different forwarding paths
- ECMP Hash Function:
  - f(src_ip, dst_ip, proto, src_port, dst_port)
  - “Per-flow” load balancing

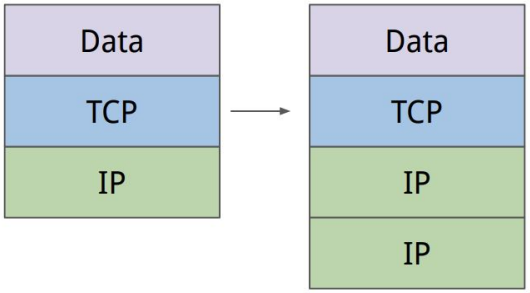
(Over/Under)lay

- With VMs, many hosts spin up frequently
- Addressing could become a mess!
  - And won't scale!
- Thus, we build **overlay** networks

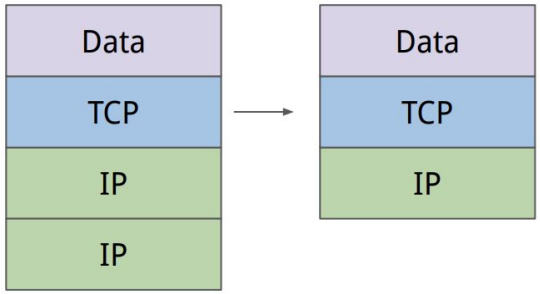


How? Encapsulation

- Encapsulation: put another header on the packet



- Decapsulation: remove extra headers that were added for encapsulation



SDN

What is SDN?

- A way to **manage** networks through a *logically* centralized controller

- What is management?
  - Generally, setting configuration for different protocols
    - So this can look like many different things!
    - Examples:
      - setting routing protocol configuration
      - setting routes directly
      - creating tunnels (encapsulation / decapsulation)

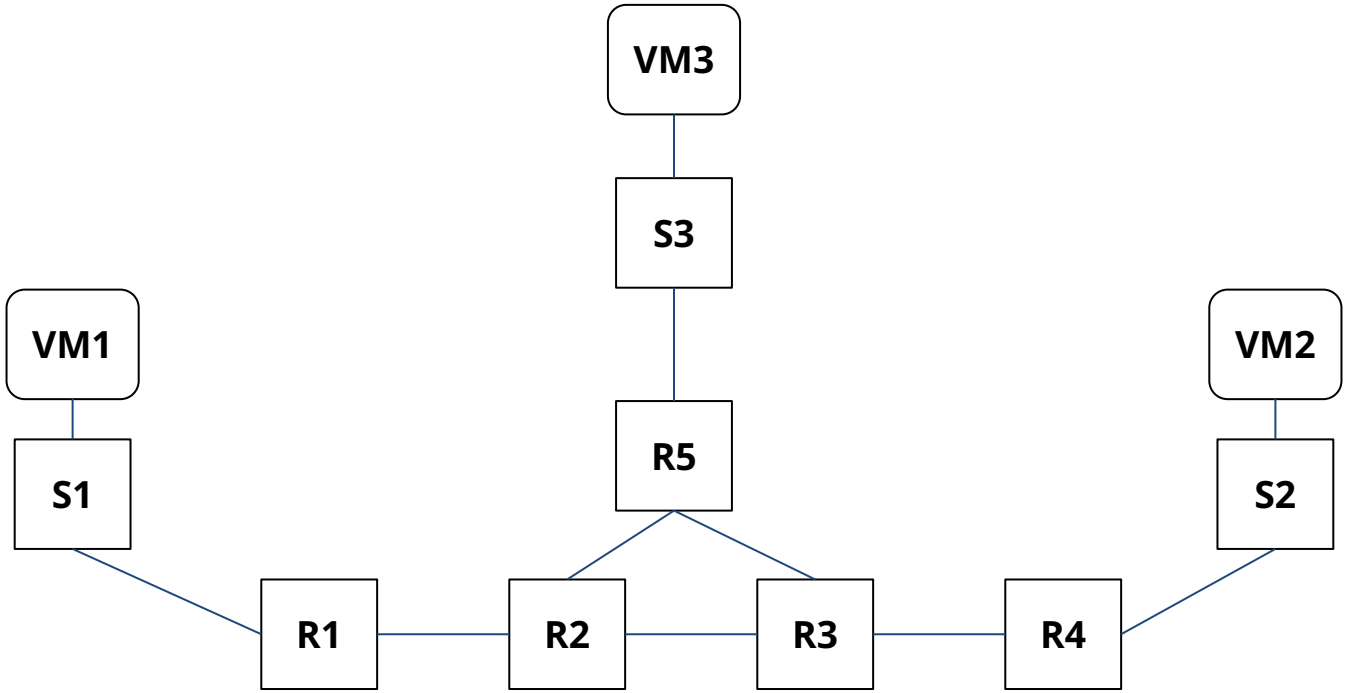
How SDN?

- In a lot of the cases, implement an **overlay** network on top of an **underlay** network
  - Specifics will vary → that's part of the appeal (customization)!

- What is an **overlay?**
  - A subset of the nodes in the underlay network connected together **using the destination-based connectivity achieved in the underlay network**

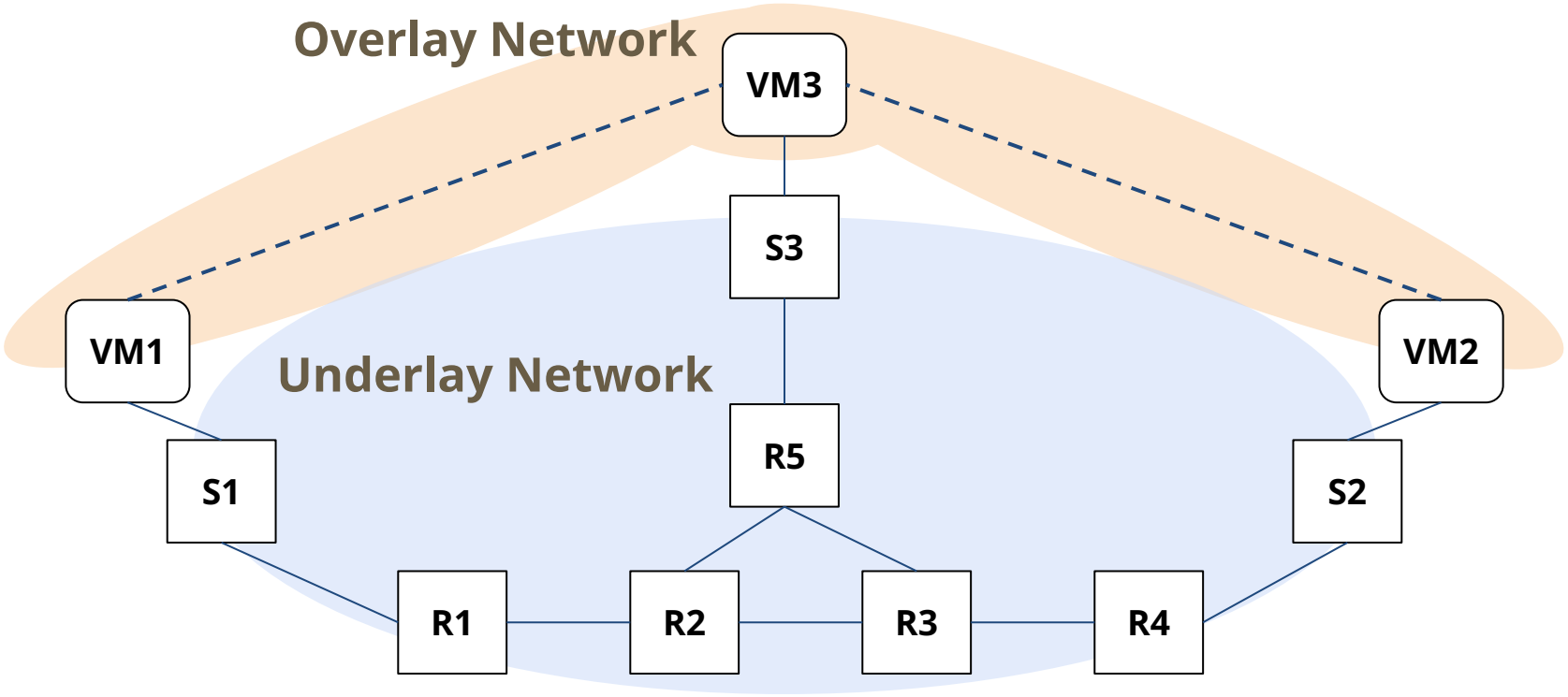
How SDN? - Network Virtualization

- Fix scalability problems of routing with multitenancy by creating virtual overlay networks



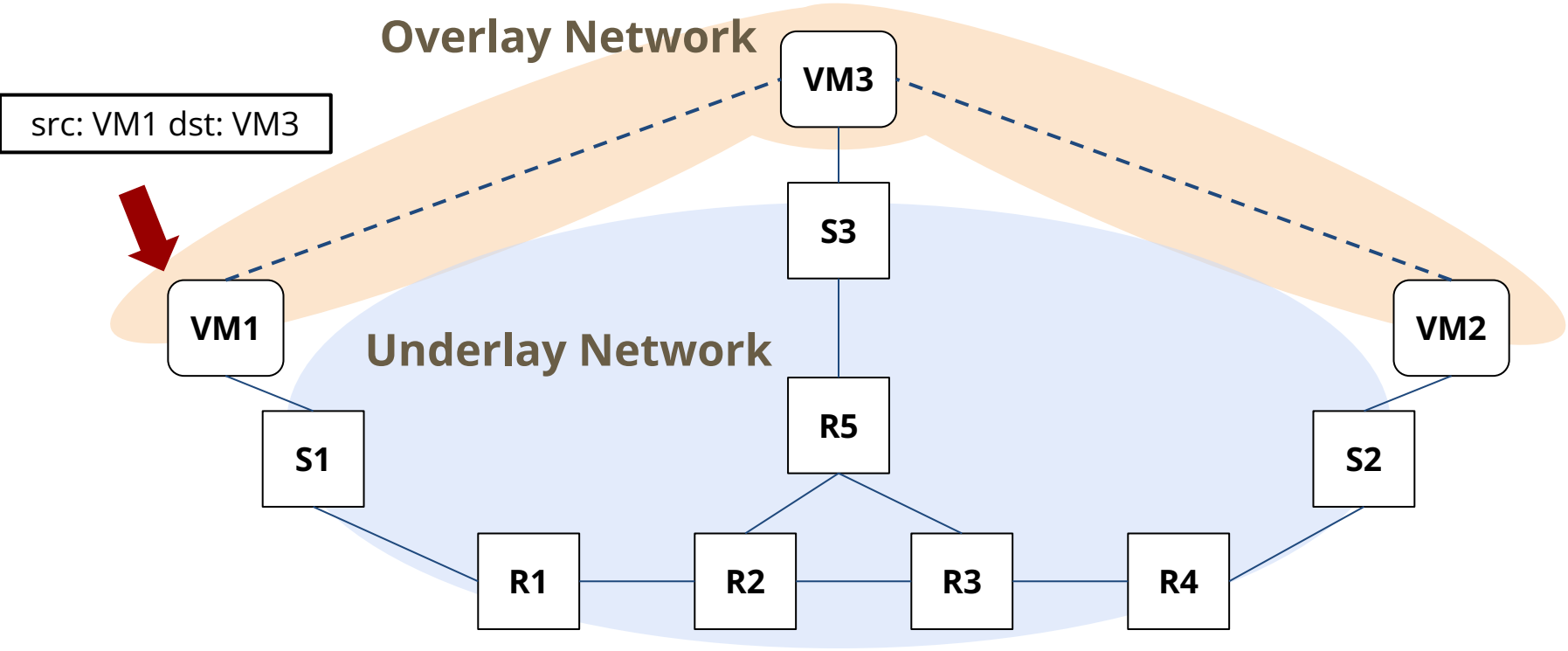
How SDN? - Network Virtualization

- Fix scalability problems of routing with multitenancy by creating virtual overlay networks



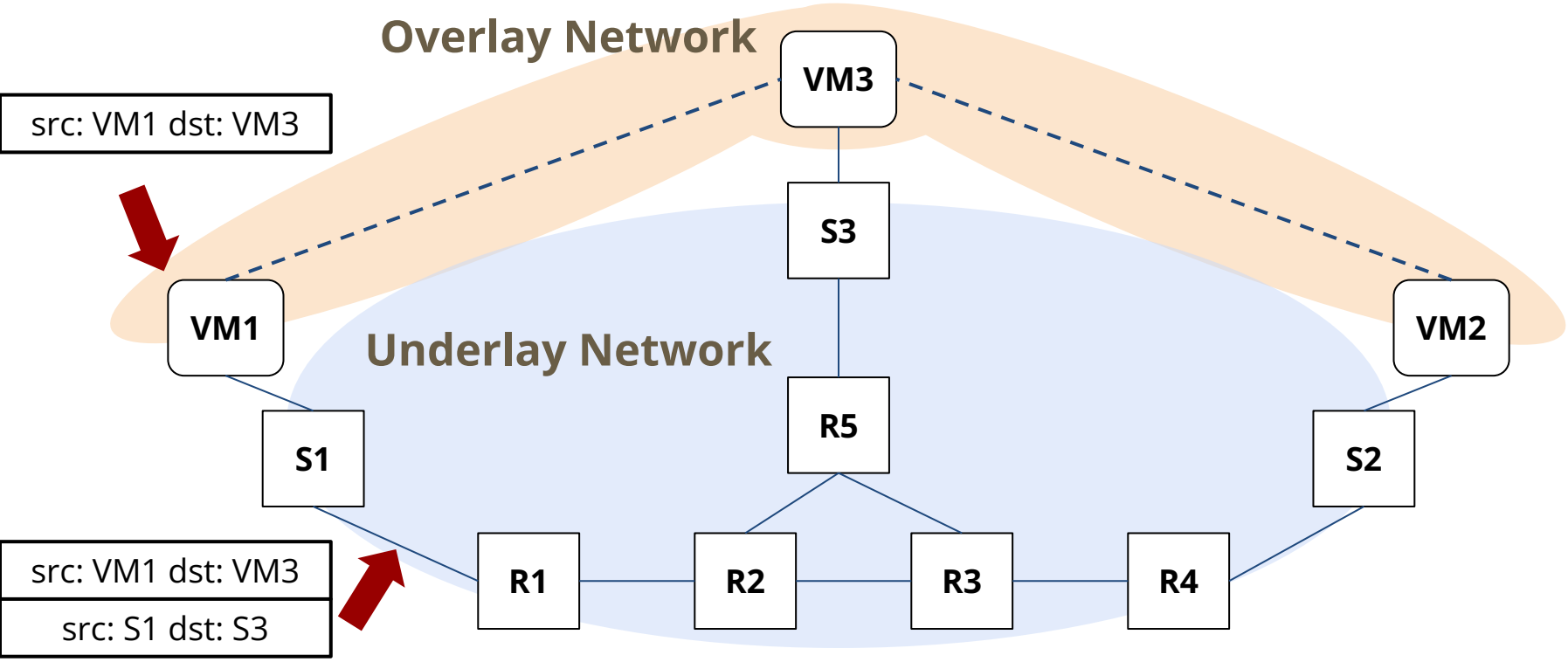
How SDN? - Network Virtualization

- Fix scalability problems of routing with multitenancy by creating virtual overlay networks



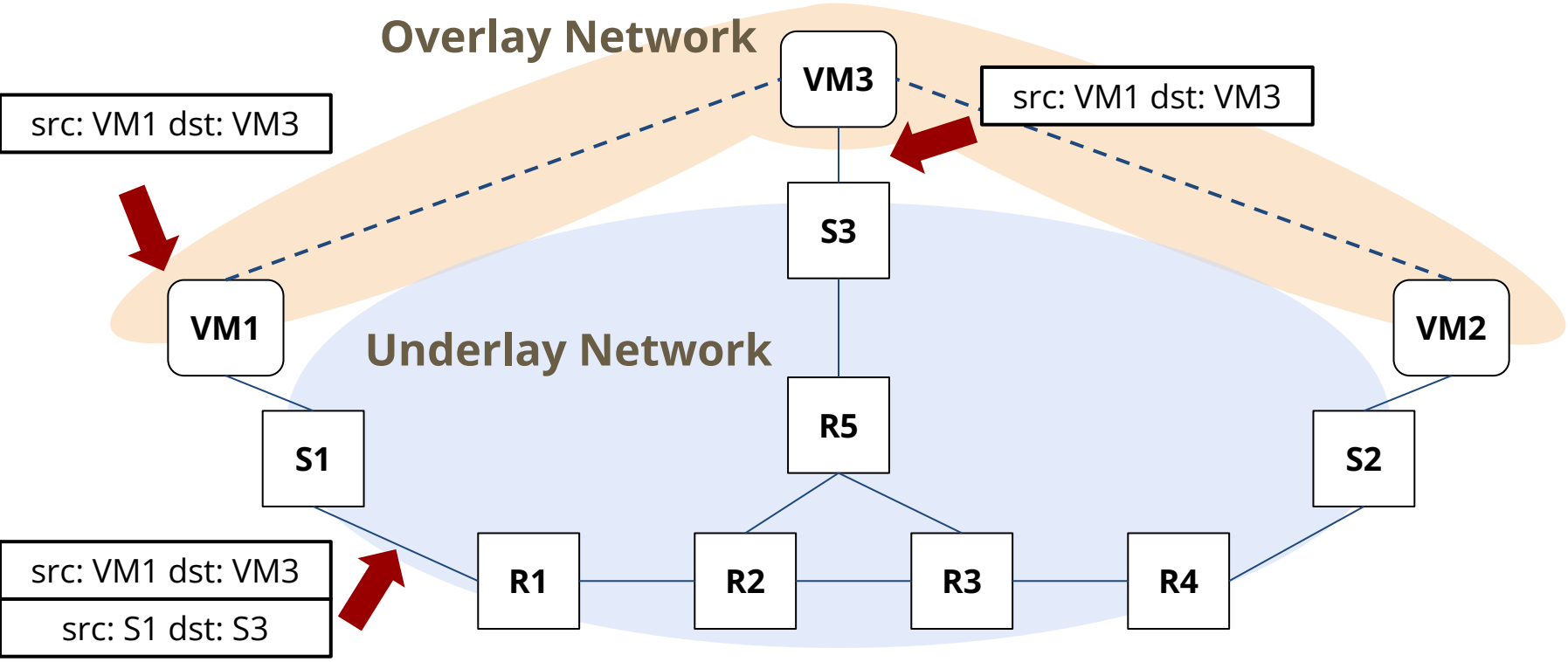
How SDN? - Network Virtualization

- Fix scalability problems of routing with multitenancy by creating virtual overlay networks



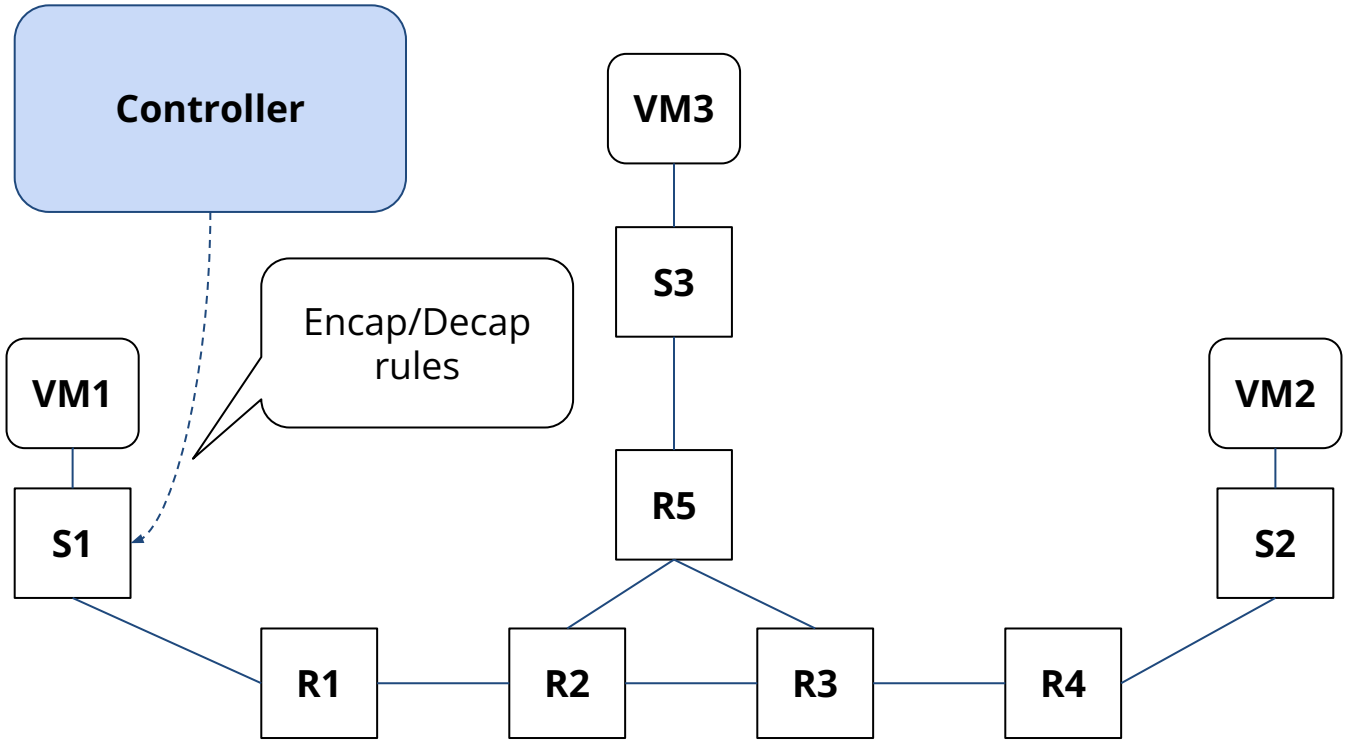
How SDN? - Network Virtualization

- Fix scalability problems of routing with multitenancy by creating virtual overlay networks



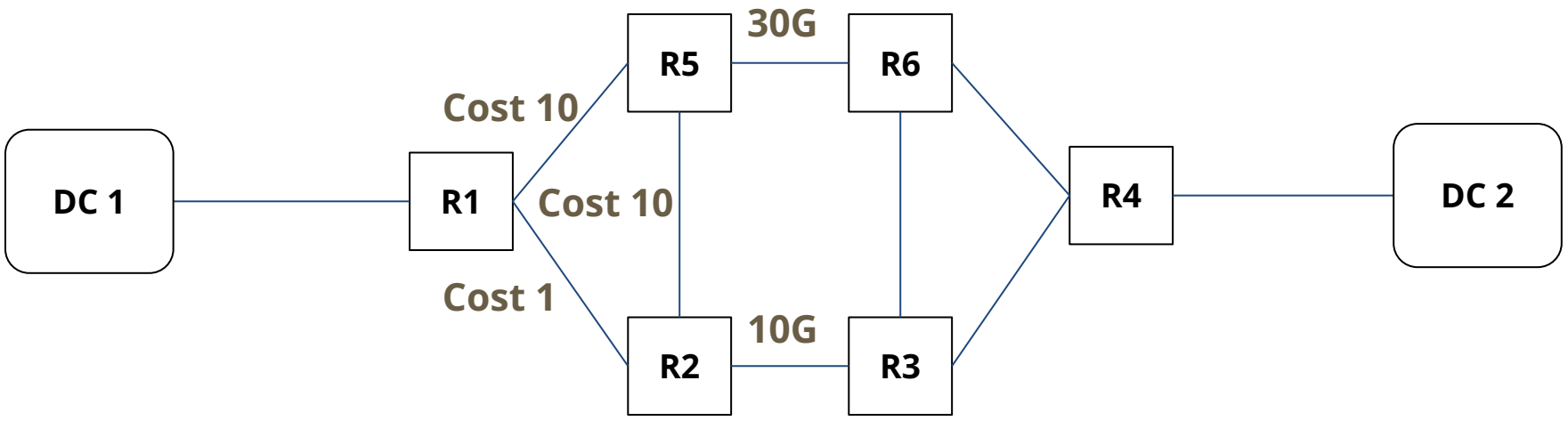
How SDN?

- Where did all this configuration to set up these overlays/underlays come from?
  - The **SDN controller**

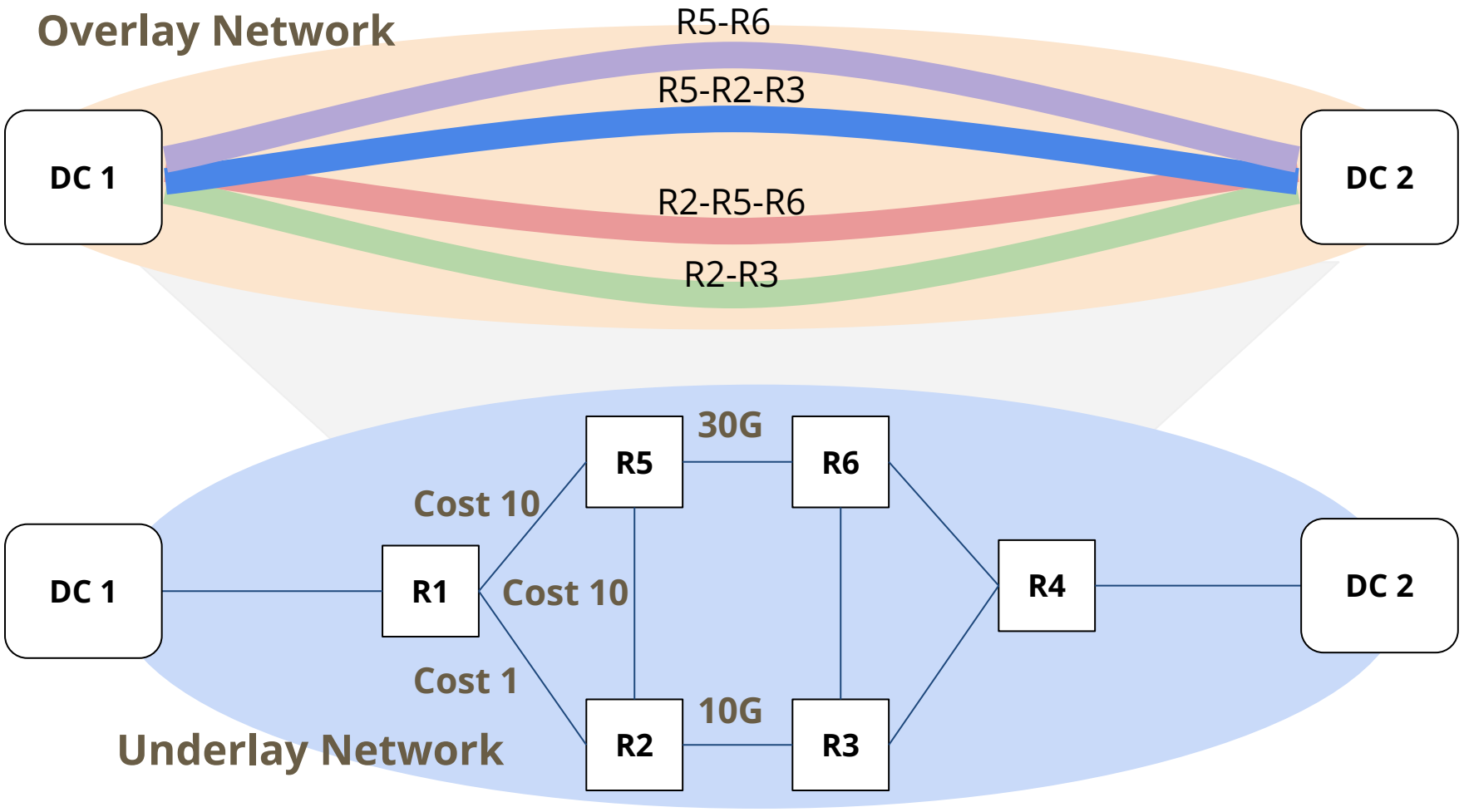


How SDN? - Traffic Engineering

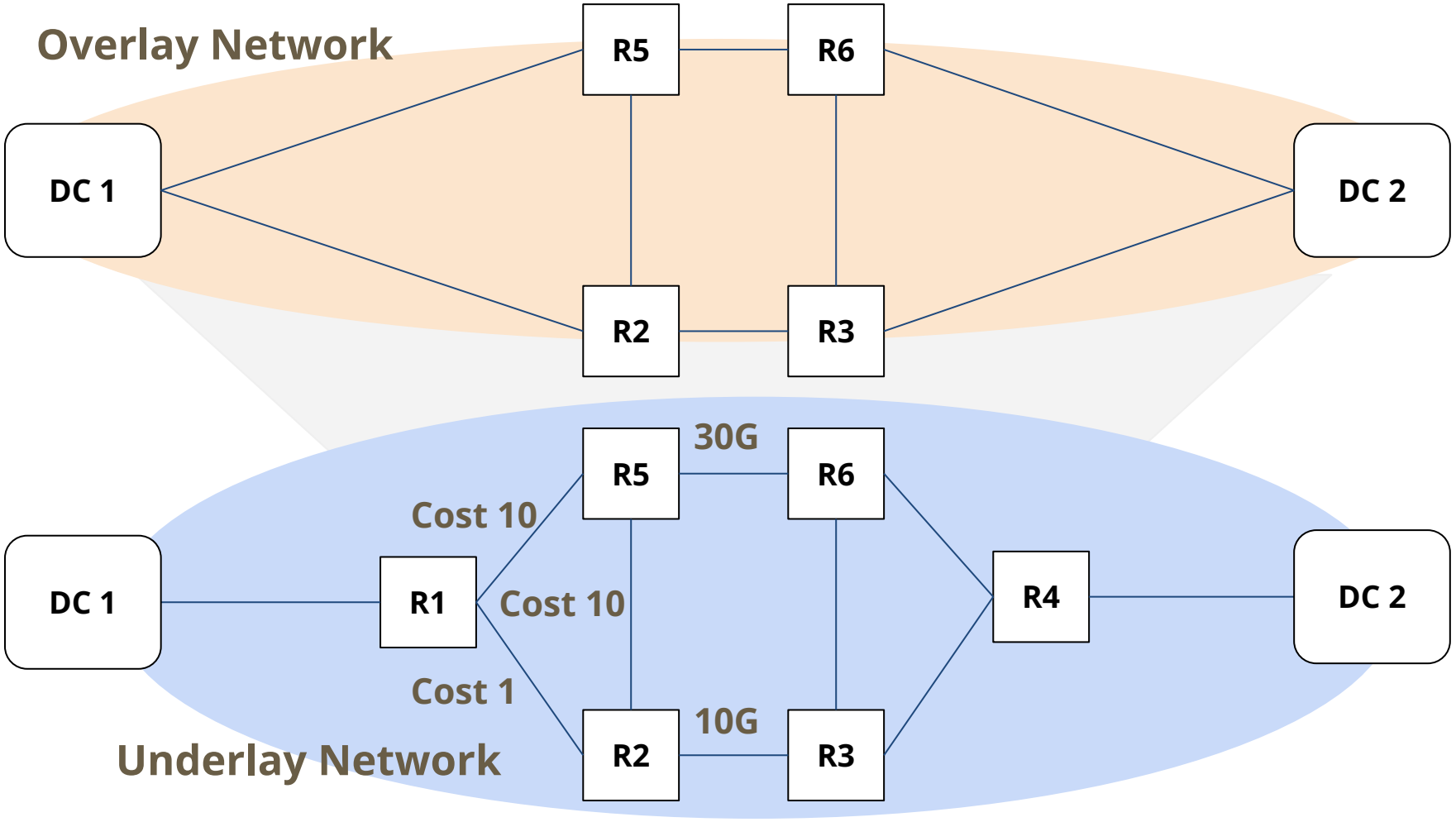
- Use labels in header to customize the path that traffic takes
  - But the existing network only does destination-based forwarding!
  - Exploit that to create new paths



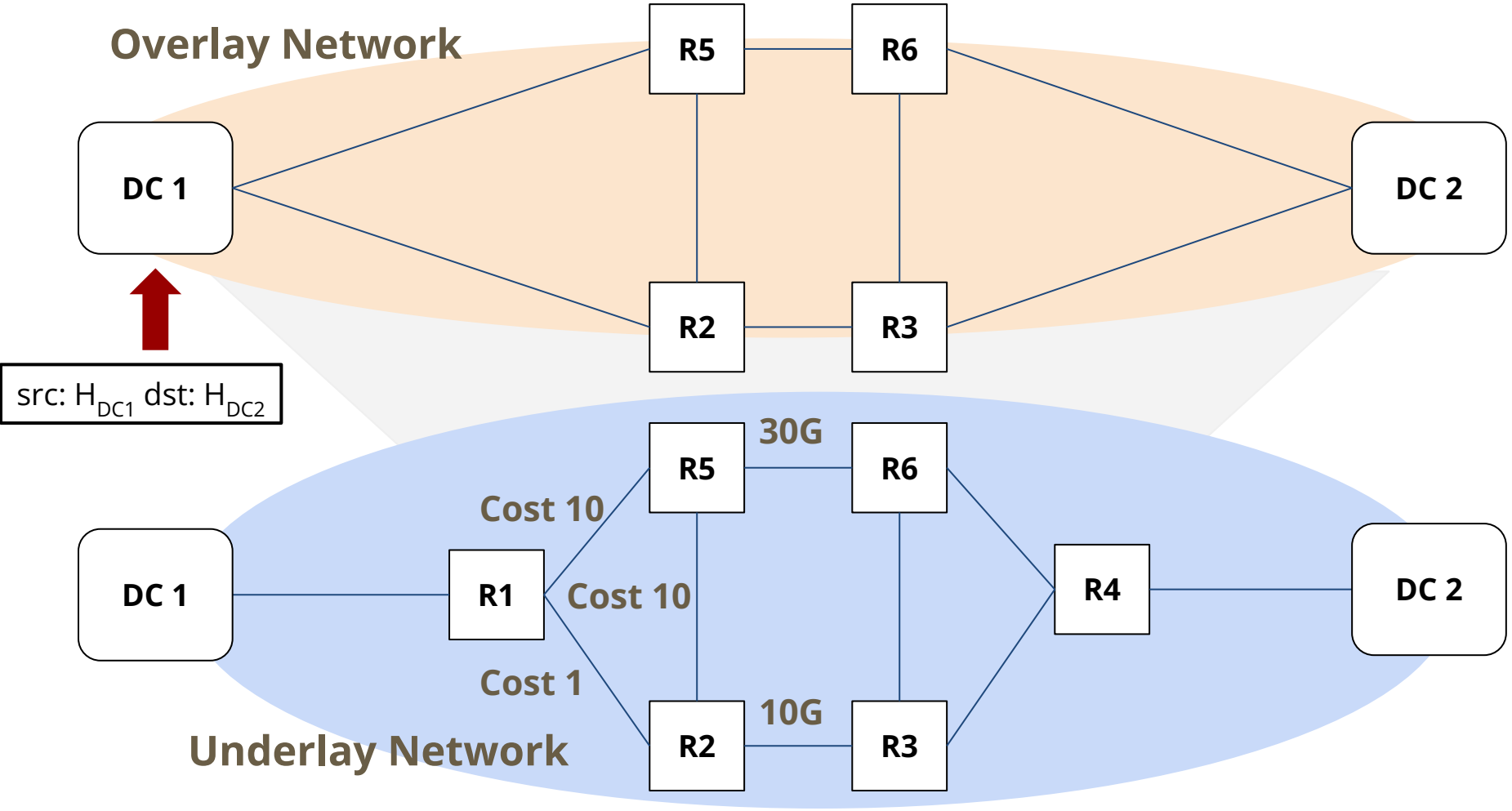
How SDN? - Traffic Engineering



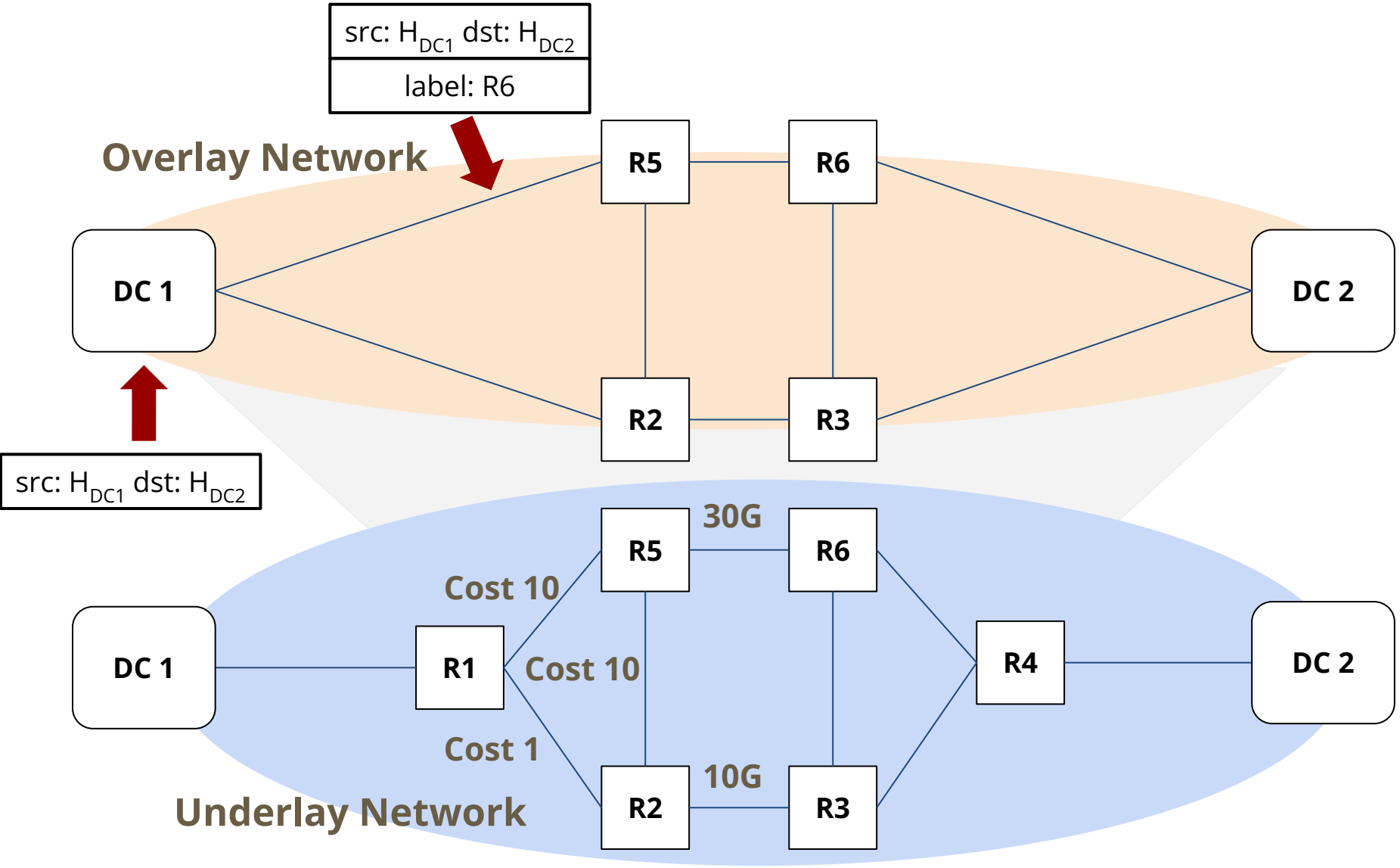
How SDN? - Traffic Engineering



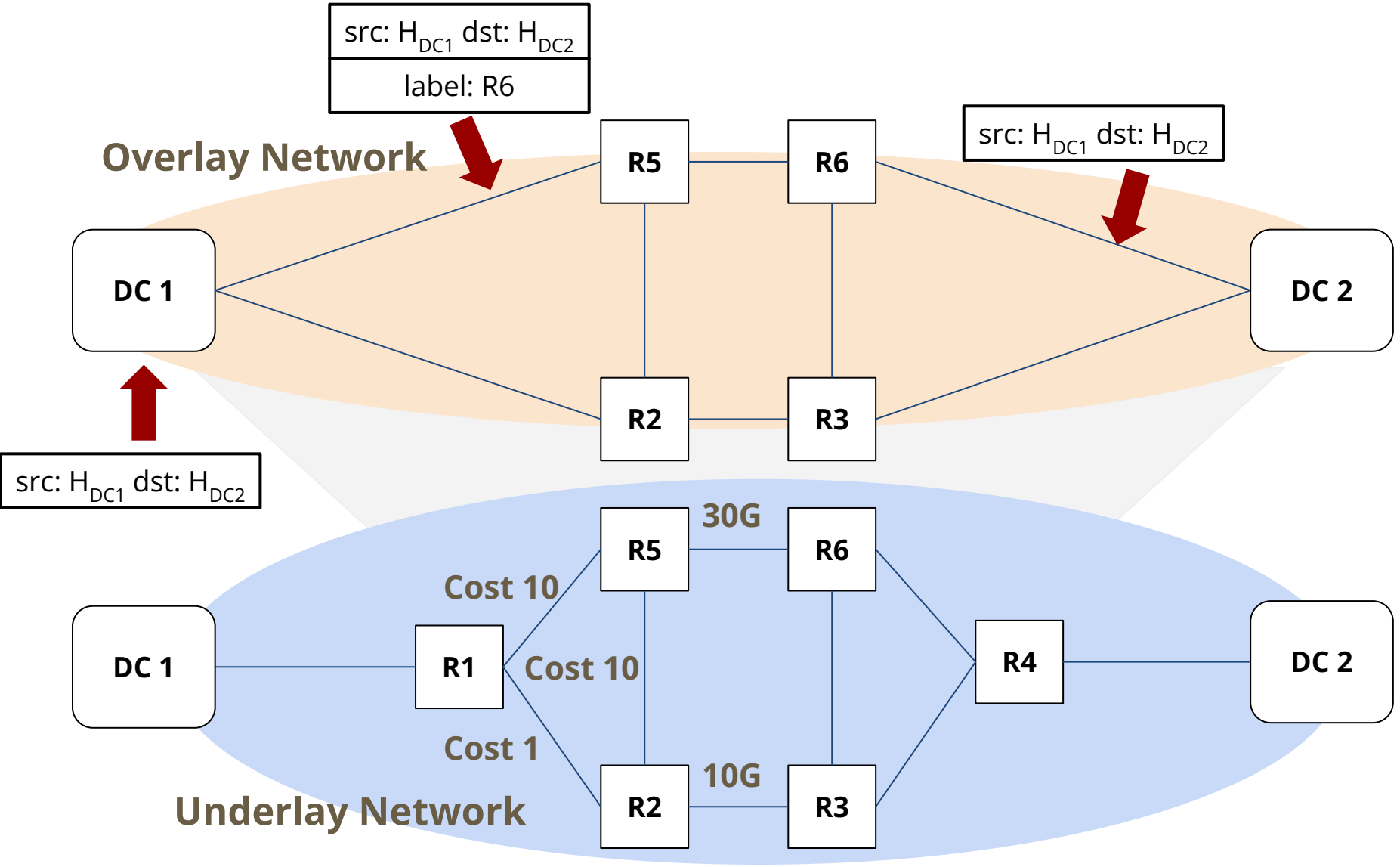
How SDN? - Traffic Engineering



How SDN? - Traffic Engineering



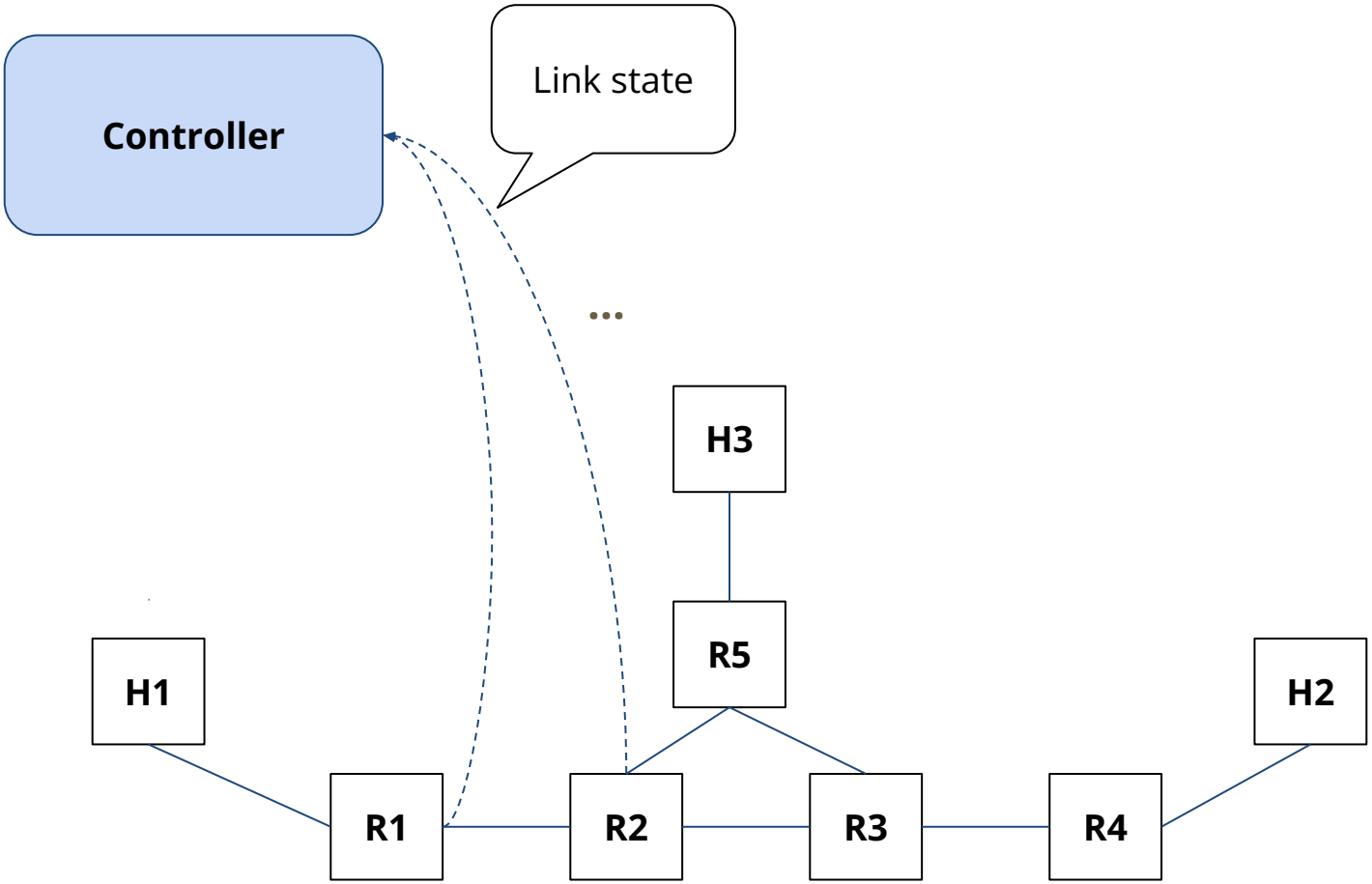
How SDN? - Traffic Engineering



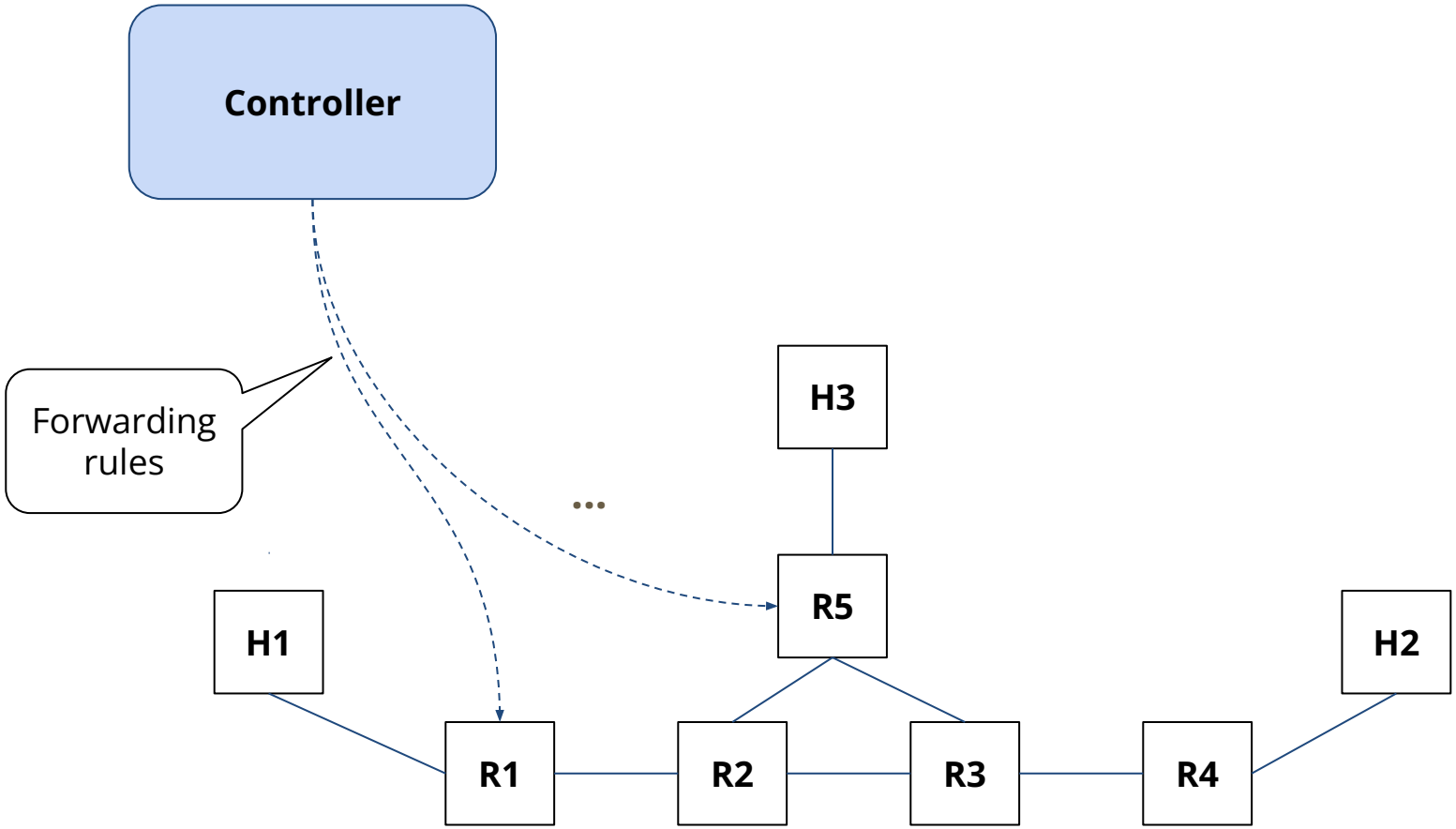
How SDN? - Forwarding

- Directly program forwarding rules into routers instead of using routing protocols
- Requires out-of-band connection with the controller

How SDN? - Forwarding

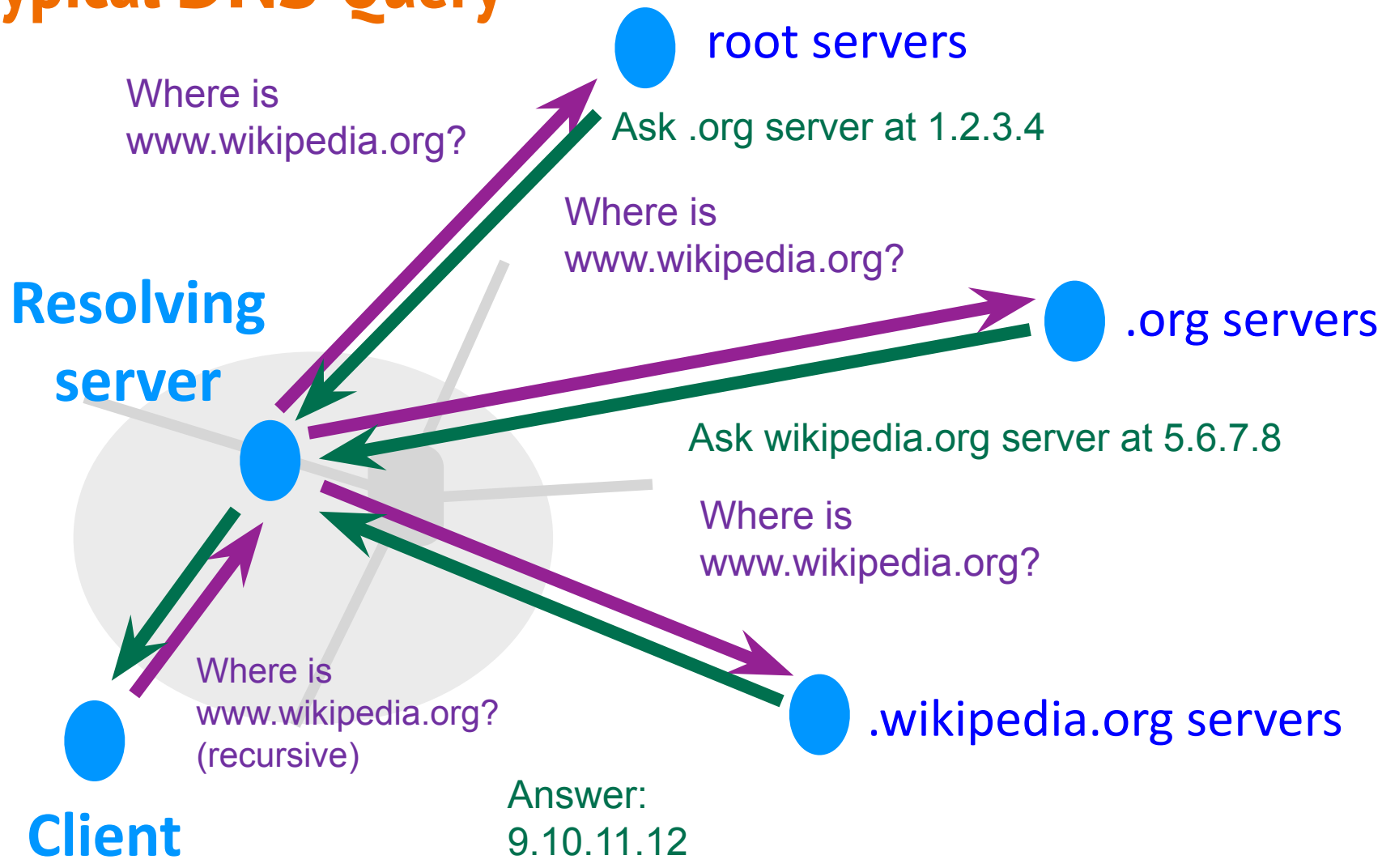


How SDN? - Forwarding



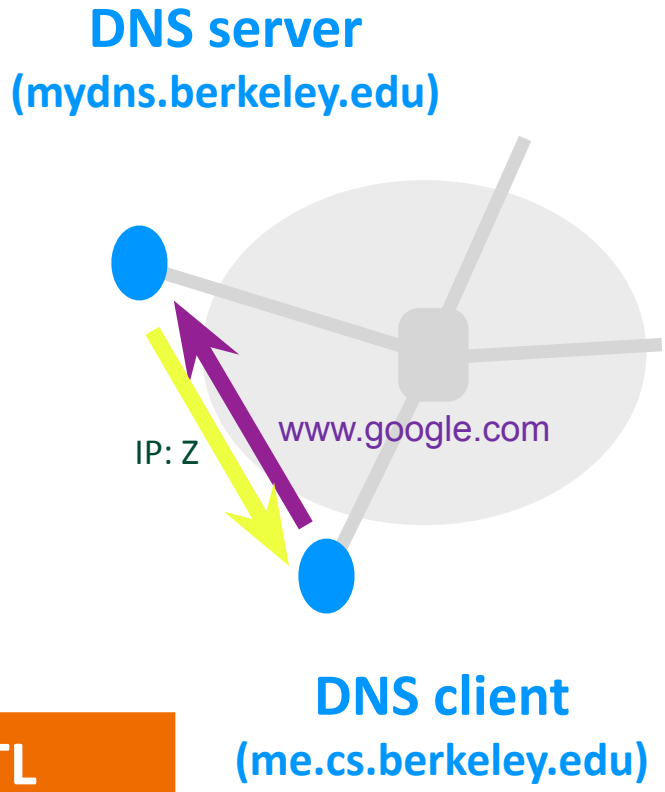
DNS, Web

Typical DNS Query



Caching DNS Responses

- DNS responses cached in DNS servers
  - Expire after TTL (time-to-live)
- Most popular sites visited often
  - Top sites frequently cached. Fast!



Hostname	IP	TTL
www.google.com	Z	60 min

Anatomy of a URL

`scheme://host[:port]/path/resource`

Scheme	Usually a protocol like (http, ftp, https, smtp, rtsp, *etc.*)
host	DNS hostname or an IP address
port	Defaults to protocol's standard port *e.g.* http: 80 https: 443
path	Traditionally reflects the file system
resource	Identifies the desired resource Can also extend to program executions: `http://us.f413.mail.yahoo.com/ym/ShowLetter?box=%40B%40Bulk&MsgId=2604_1744106_29699_1123_1261_0_28917_3552_1289957100&Search=&Nhead=f&YY=31454&order=down&sort=date&pos=0&view=a&head=b`

HTTP

- Client-server architecture
  - Server is “always on” and “well known”
  - Clients initiate contact to server

- Synchronous request/reply protocol
  - “Synchronous” means same HTTP session used for request and reply
  - Runs over TCP, Port 80

Requests and Responses

Request

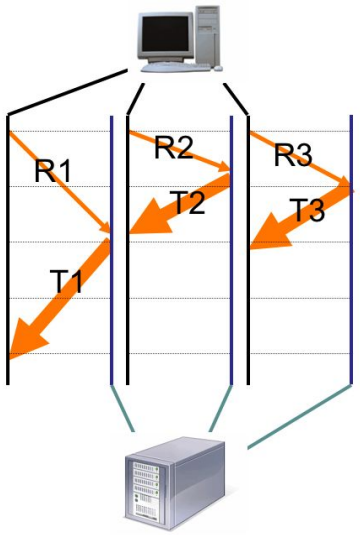
```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language: fr
(blank line)
```

Response

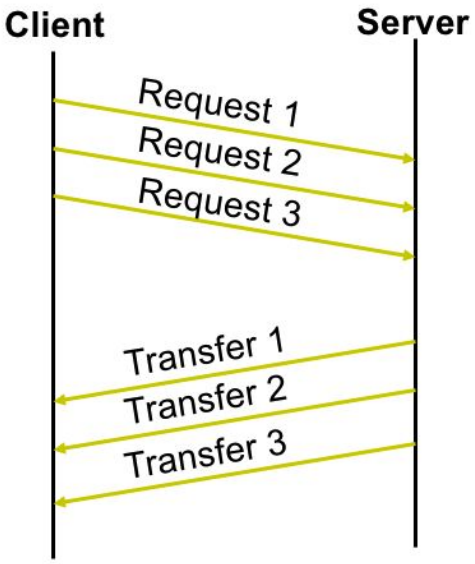
```
HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 2006 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 2006 ...
Content-Length: 6821
Content-Type: text/html
(blank line)
data data data data data ...
```

Request Patterns

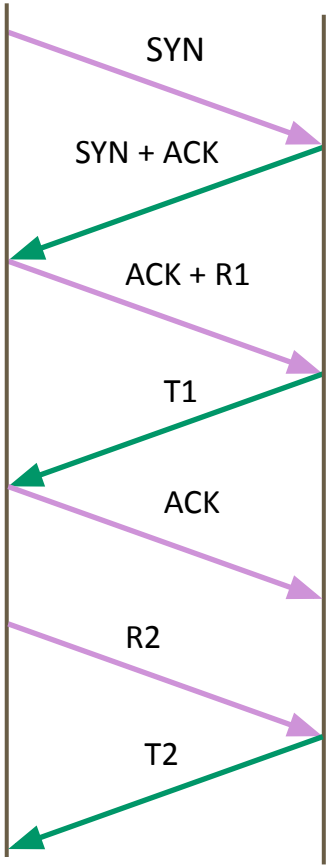
Concurrent



Pipelined



Persistent



Caching: How

- Idea: Replication
  - Replicate the content across multiple copies to reduce bottlenecks
- Implementation: content distribution networks (CDNs)
  - The client content provider modifies its content so that embedded URLs reference the new domains.
  - "Akamaize" content
  - e.g.: *http://www.netflix.com/tiger_king.jpg* might become *http://a1386.g.akamai.net/tiger_king.jpg*

Caching: CNAMEs

- Instead of using a weird URL, use CNAMEs!
  - List picture as *http://cdn.netflix.com/tiger_king.jpg*
  - The authoritative server for this is controlled by netflix
  - CNAME record aliases *cdn.netflix.com* to *a1386.g.akamai.net*
- Akamai handles sending to the closest server because *a1386.g.akamai.net* is under an Akamai nameserver

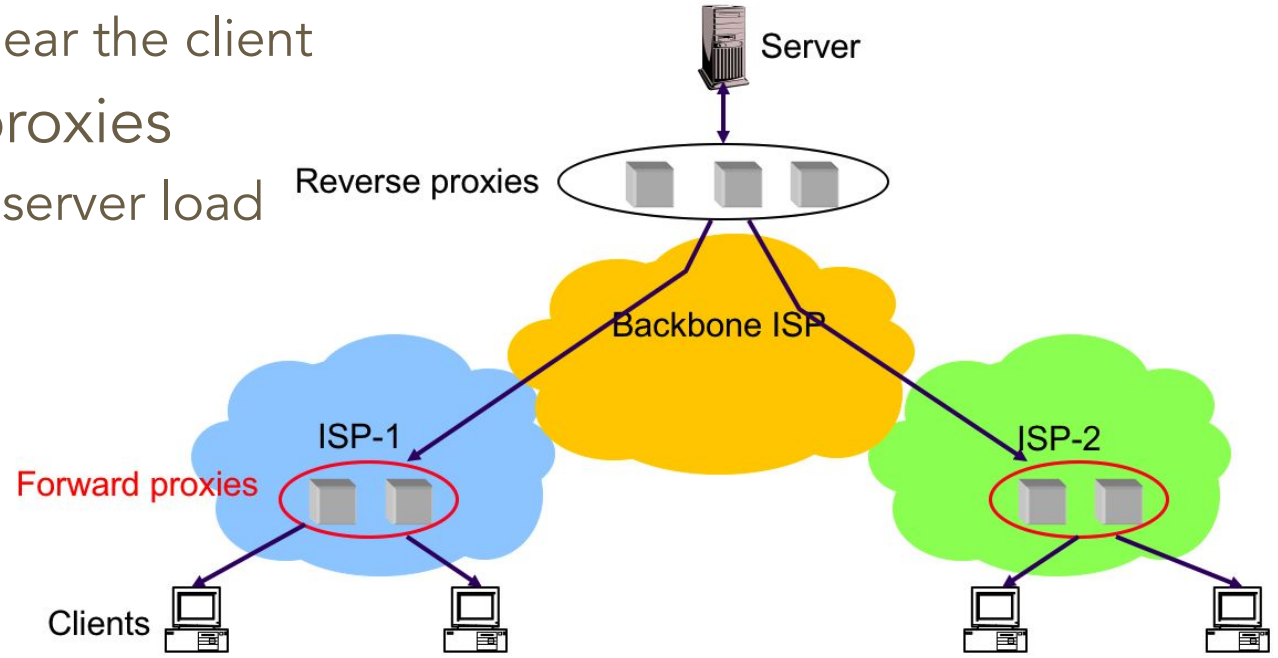
- Only change to the HTML is *http://www.netflix.com/tiger_king.jpg* became *http://cdn.netflix.com/tiger_king.jpg*

Caching: Specifics in HTTP

- GET Request header:
  - `If-Modified-Since` – returns "not modified" if resource not modified since specified time
  - `Cache-Control: no-cache` – ignore all caches; always get resource directly from server (think force refresh)
- Response header:
  - `Cache-Control: max-age=<ttl>` – TTL: how long to cache the resource
  - `Cache-Control: no-store` – Don't cache this
- When making request, if within the TTL, just load cached resource… otherwise, send with *if-modified.*
  - Server will either send a HTTP 304 ("Not Modified") or HTTP 200 (changed, and here's the new data)

Caching: Where

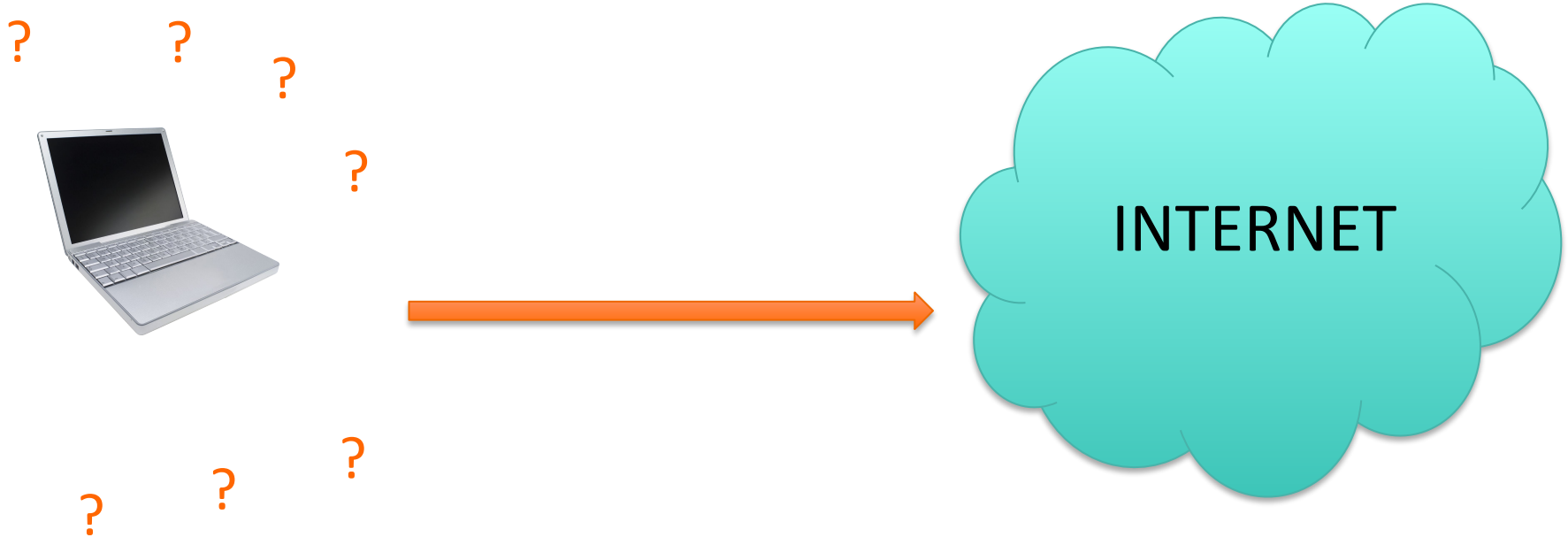
- Forward Proxies
  - Cache near the client
- Reverse proxies
  - Reduce server load



DHCP, ARP

DHCP

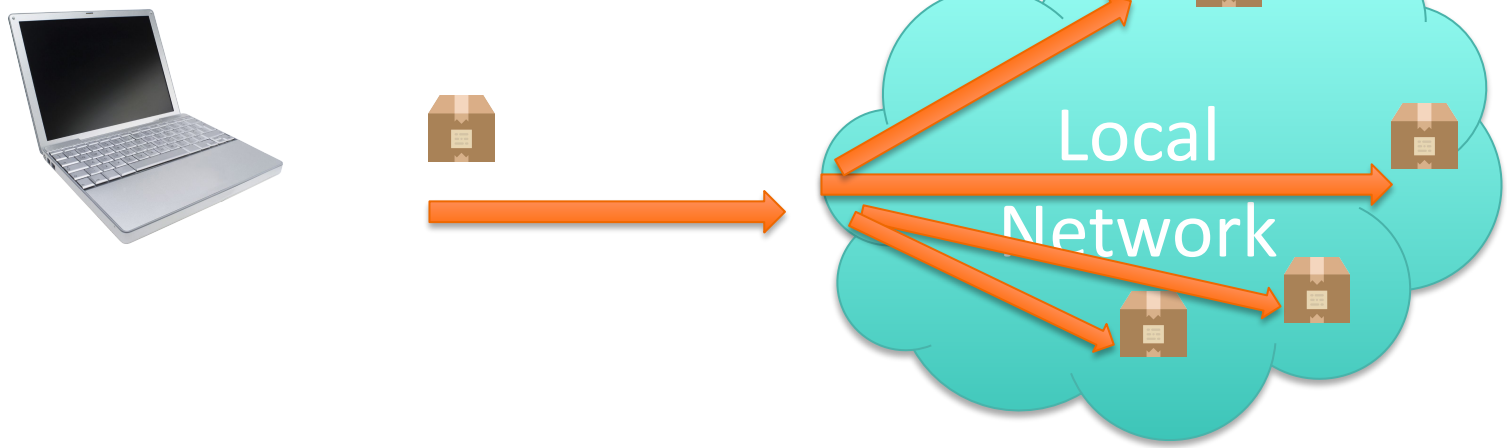
- Enables a host to learn about its….
  - Own IP Address
  - Network Mask
  - First-hop router's IP Address
  - DNS Server(s) IP Address(es)



DHCP Discovery

- DHCP server(s) located on same local network as host
- Host initially broadcasts **discover** message

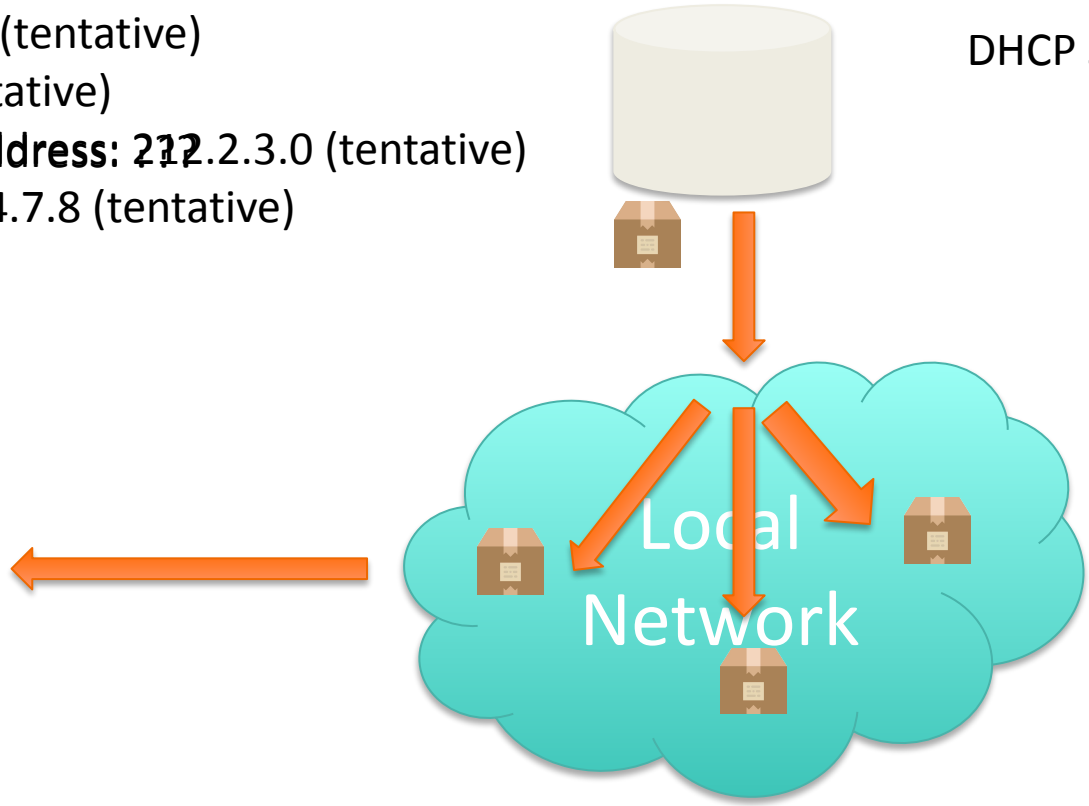
IP Address: ???
Subnet Mask: ???
First-hop Router IP Address: ???
DNS IP Address: ???



DHCP Offer

- DHCP server(s) responds by broadcasting **offer** message
- Message includes assigned IP address, network mask, first-hop router address, DNS server addresses

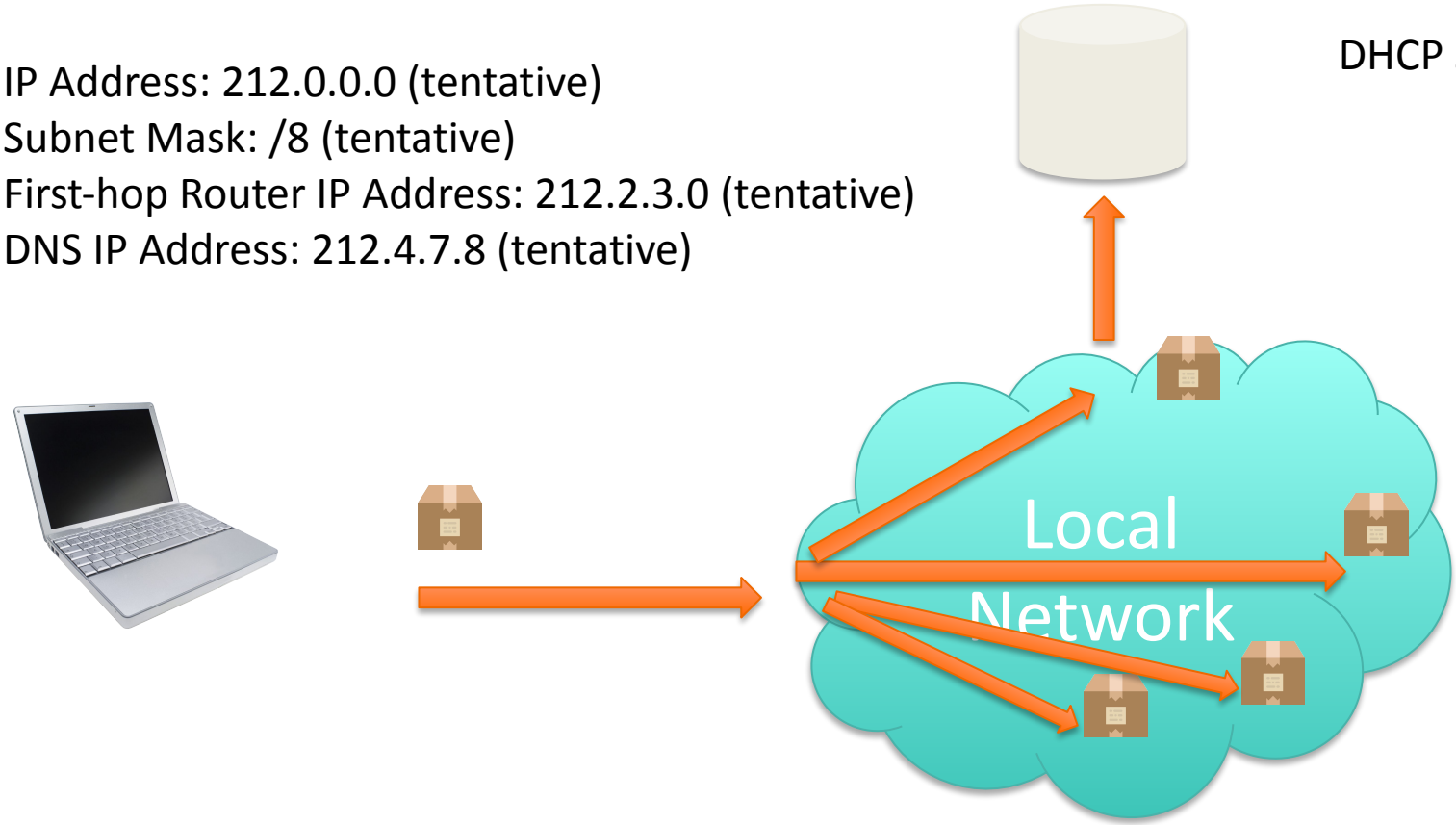
IP Address: 212.0.0.0 (tentative)
Subnet Mask: 255.255.255.0 (tentative)
First-hop Router IP Address: 212.2.3.0 (tentative)
DNS IP Address: 212.4.7.8 (tentative)



DHCP Request

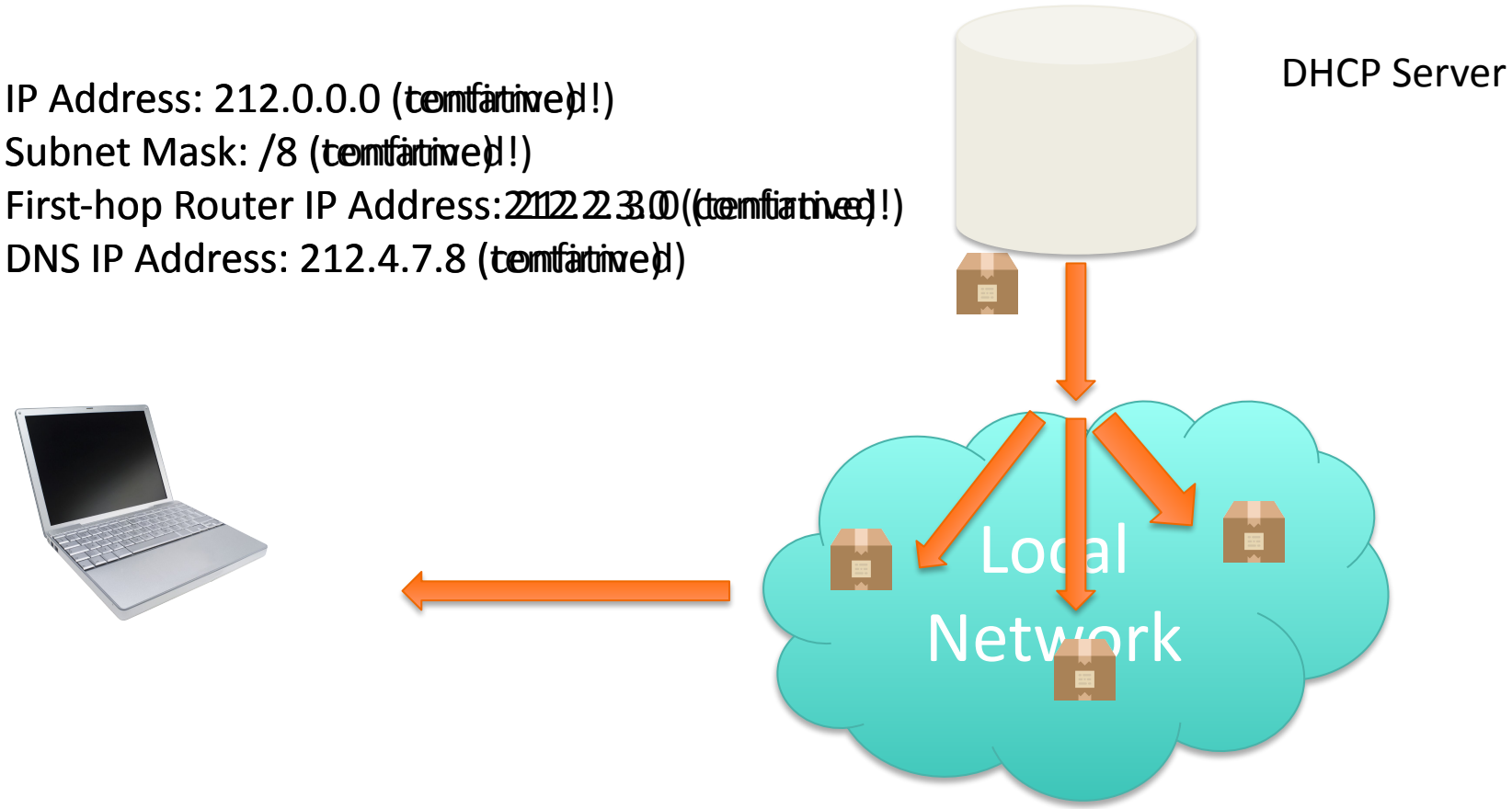
- Host responds by broadcasting **request** message
- This message identifies which offer was accepted (helps when there are multiple local DHCP servers)

IP Address: 212.0.0.0 (tentative)
Subnet Mask: /8 (tentative)
First-hop Router IP Address: 212.2.3.0 (tentative)
DNS IP Address: 212.4.7.8 (tentative)



DHCP Acknowledgement

- Chosen DHCP server responds by broadcasting **ACK**



ARP (Overview)

- When host sends packet, specify dest Ethernet address so packet can traverse local networks
- Each host has ARP table, which maps IP to Ethernet
- If mapping unknown, ask (solicit) local network by broadcasting "Who has IP address **x**?"
  - Host with IP **x** responds "My Ethernet address is **y**"

ARP Table

IP Addr.	Ethernet Addr.
a.b.c.d	40:11:11:11:11:11
a.b.c.a	50:37:11:11:11:11

Dest Host
IP: a.b.c.d
Ethernet address:
40:11:11:11:11:11

ARP (Within local network)

Destination is in same local network

- Use ARP table to lookup Ethernet address of dest
- Specify Ethernet address when sending packet

ARP Table

IP Addr.	Ethernet Addr.
a.b.c.d	40:11:11:11:11:11
a.b.c.a	50:37:11:11:11:11

Dest Host
IP: a.b.c.d
Ethernet address:
40:11:11:11:11:11

Src Host

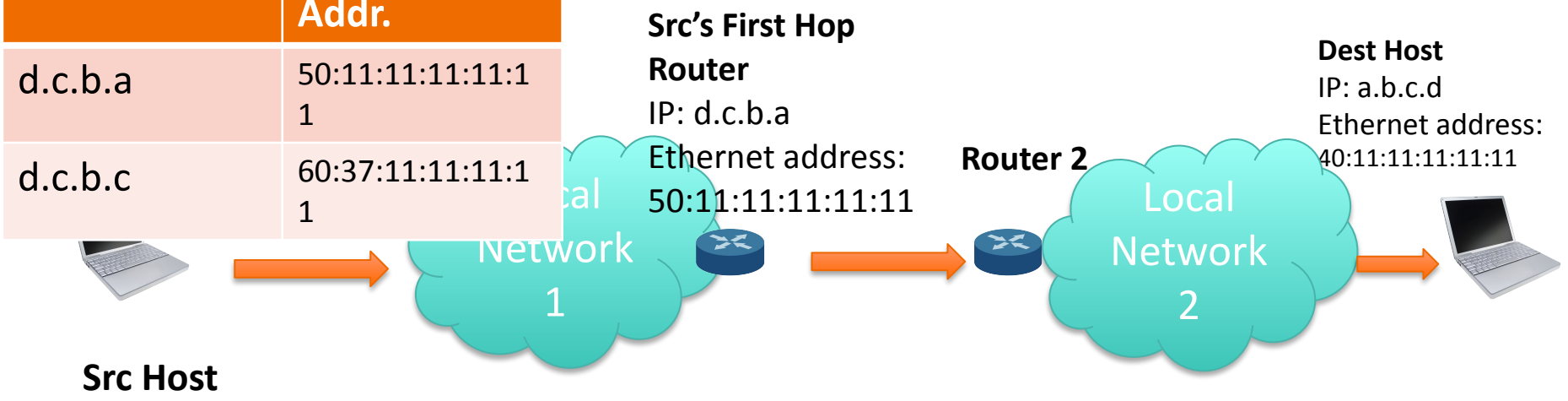
ARP (Across local networks)

Destination is NOT in same local network

- How can we tell?
  - Use subnet mask to check dest network address

ARP Table

IP Addr.	Ethernet Addr.
d.c.b.a	50:11:11:11:11:11
d.c.b.c	60:37:11:11:11:11

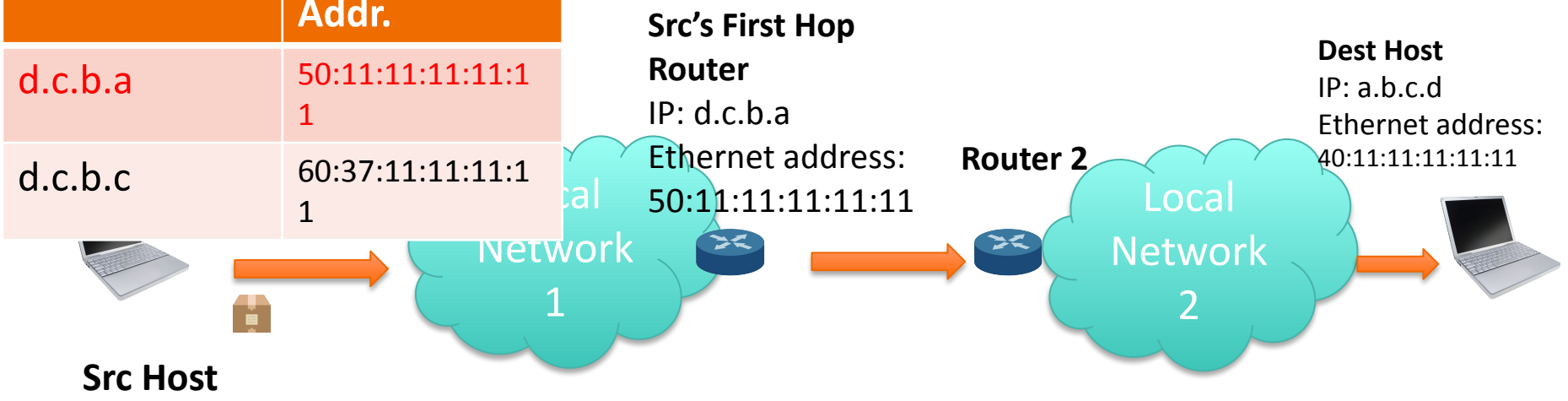


ARP (Across local networks)

- Use ARP table to lookup Ethernet address of first-hop-router (which is in same local network)
  - We know router's IP address through DHCP!
- Specify **first-hop router**'s Ethernet address in packet and send packet

ARP Table

IP Addr.	Ethernet Addr.
d.c.b.a	50:11:11:11:11:11
d.c.b.c	60:37:11:11:11:11

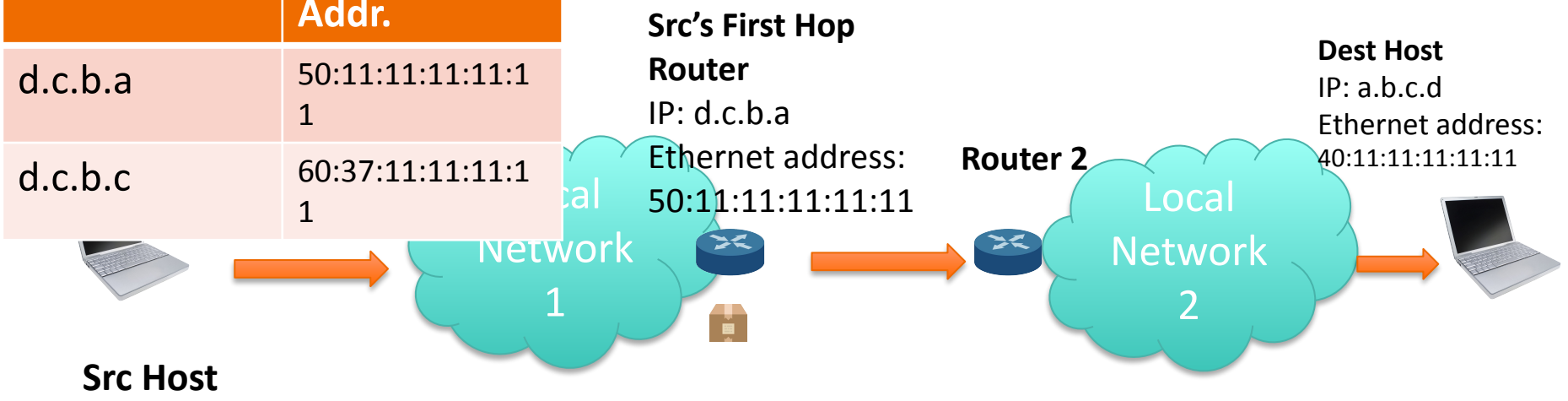


ARP (Across local networks)

- First-hop router will route packet to router 2 using dest IP address
  - Dest IP address is a.b.c.d. in this example

ARP Table

IP Addr.	Ethernet Addr.
d.c.b.a	50:11:11:11:11:11
d.c.b.c	60:37:11:11:11:11



ARP (Across local networks)

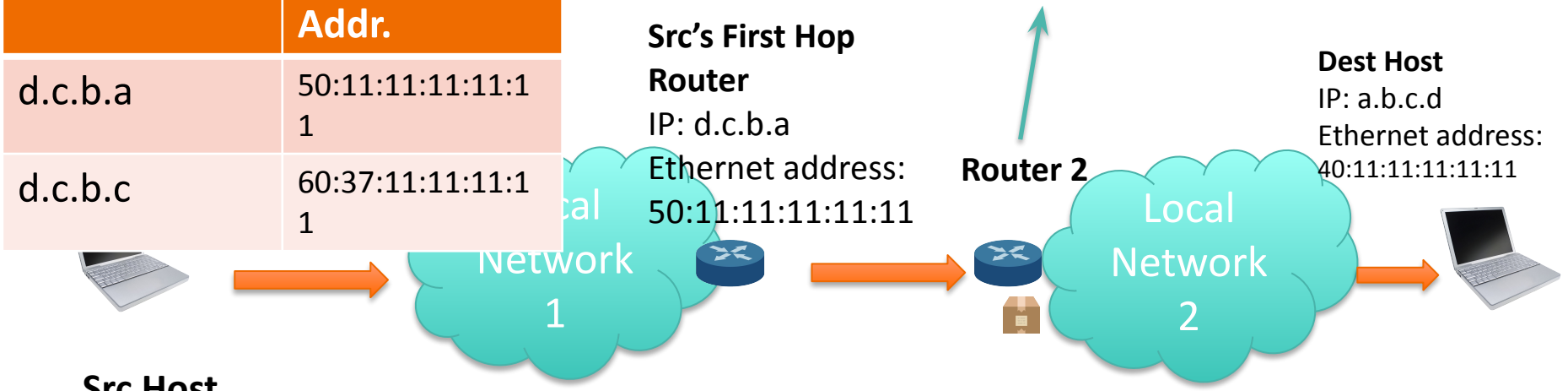
- Router 2 will use its ARP table to **set** packet's dest Ethernet address to actual dest host's Ethernet address:
  - Router 2 then sends packet to dest host

ARP Table

IP Addr.	Ethernet Addr.
a.b.c.d	40:11:11:11:11:11
a.b.c.c	70:33:33:33:11:11

ARP Table

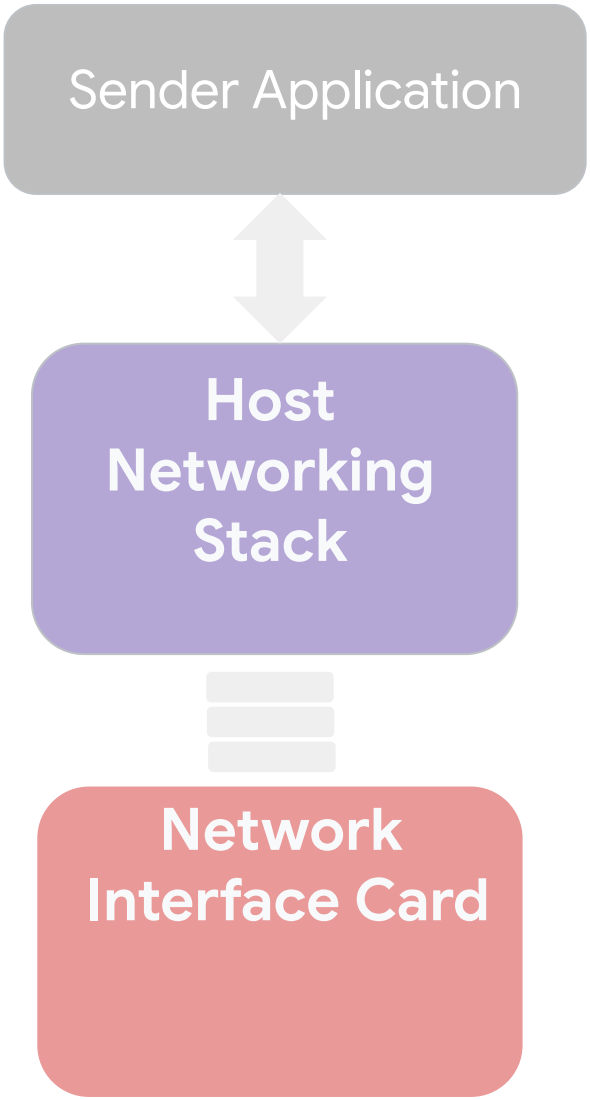
IP Addr.	Ethernet Addr.
d.c.b.a	50:11:11:11:11:11
d.c.b.c	60:37:11:11:11:11



Host Networking

What is Host Networking?

- Everything at the host that enables it to use the network
  - Linux networking stack, network driver, NIC, etc...



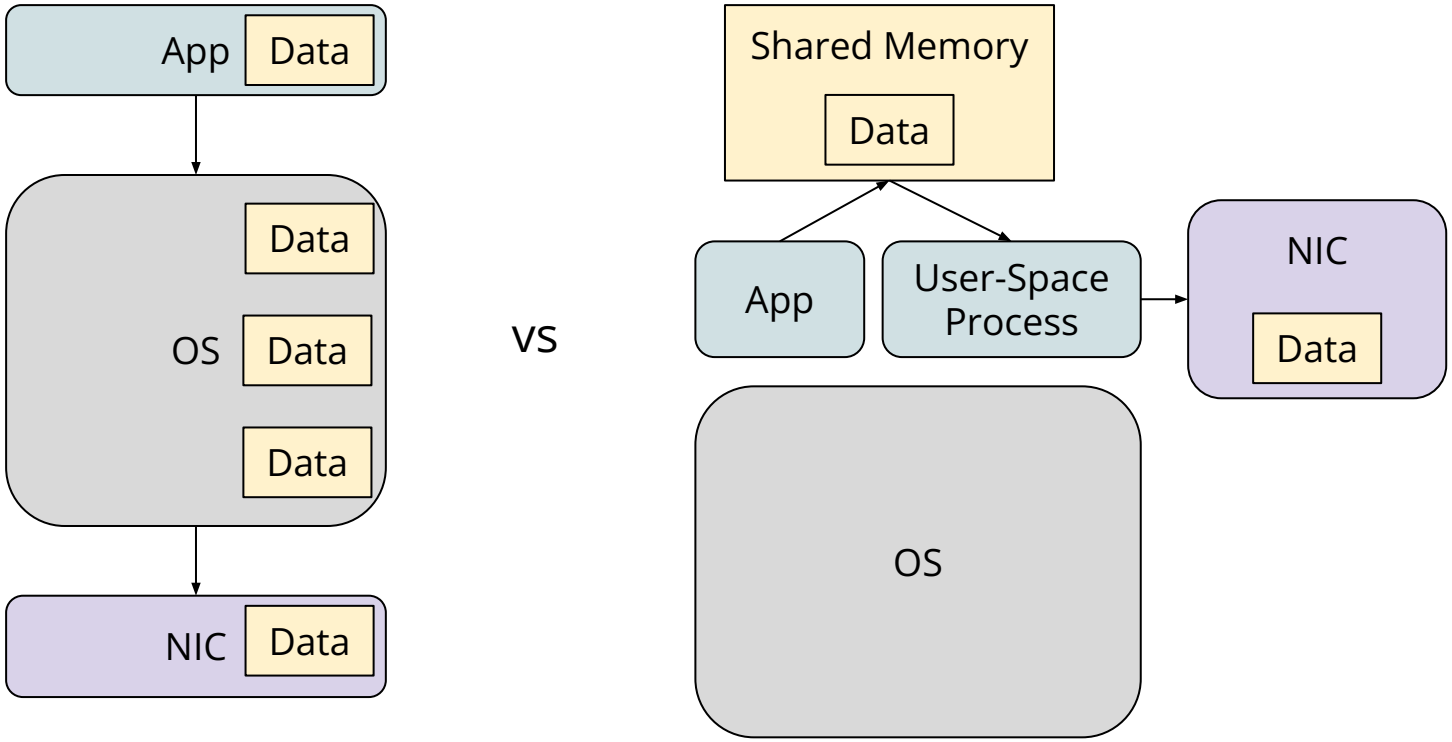
Advanced Host Networking

- Datacenter operators have the incentive to put a lot of effort into optimization
  - And have the means: total administrative control

- Some opportunities for optimization
  - Kernel bypass (avoid kernel development)
  - NIC offloads (save CPU cycles)
  - RDMA (save CPU cycles, and better performance)
  - Other features (better performance)
    - Congestion control
    - Load balancing
    - Traffic shaping
    - QoS

Kernel Bypass

- Run networking stack in user space rather than kernel space

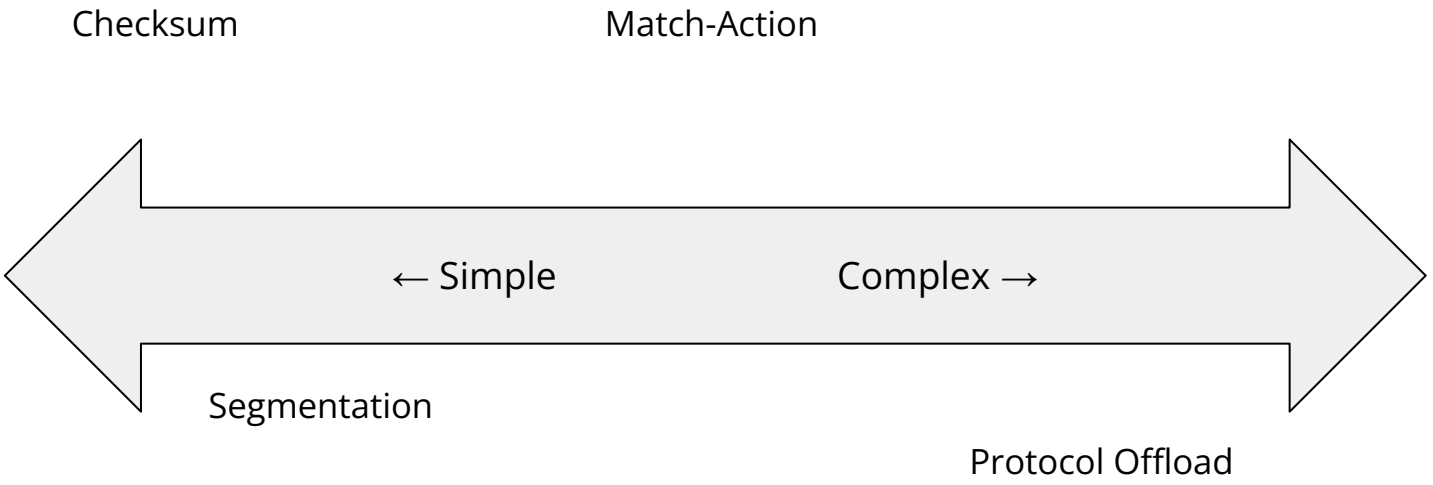


NIC Offloads

- What is offloading?
  - Implementing some tasks in hardware to free up CPU cycles
  - In our case, putting some networking tasks in the NIC instead of the kernel
- Why offload?
  - Save CPU cycles
  - Performance gains

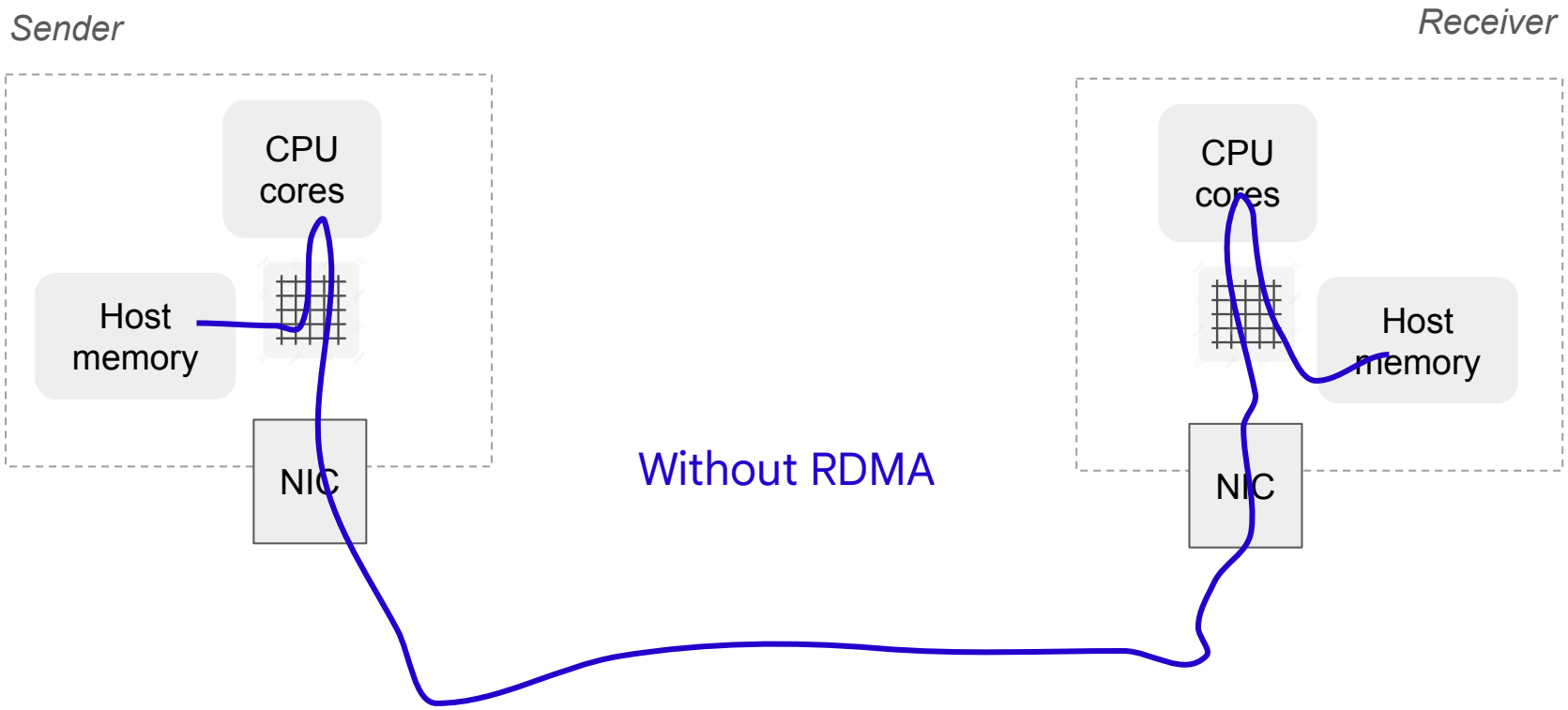
NIC Offloads

- Range from simple (checksum computation) to advanced (protocol offload)



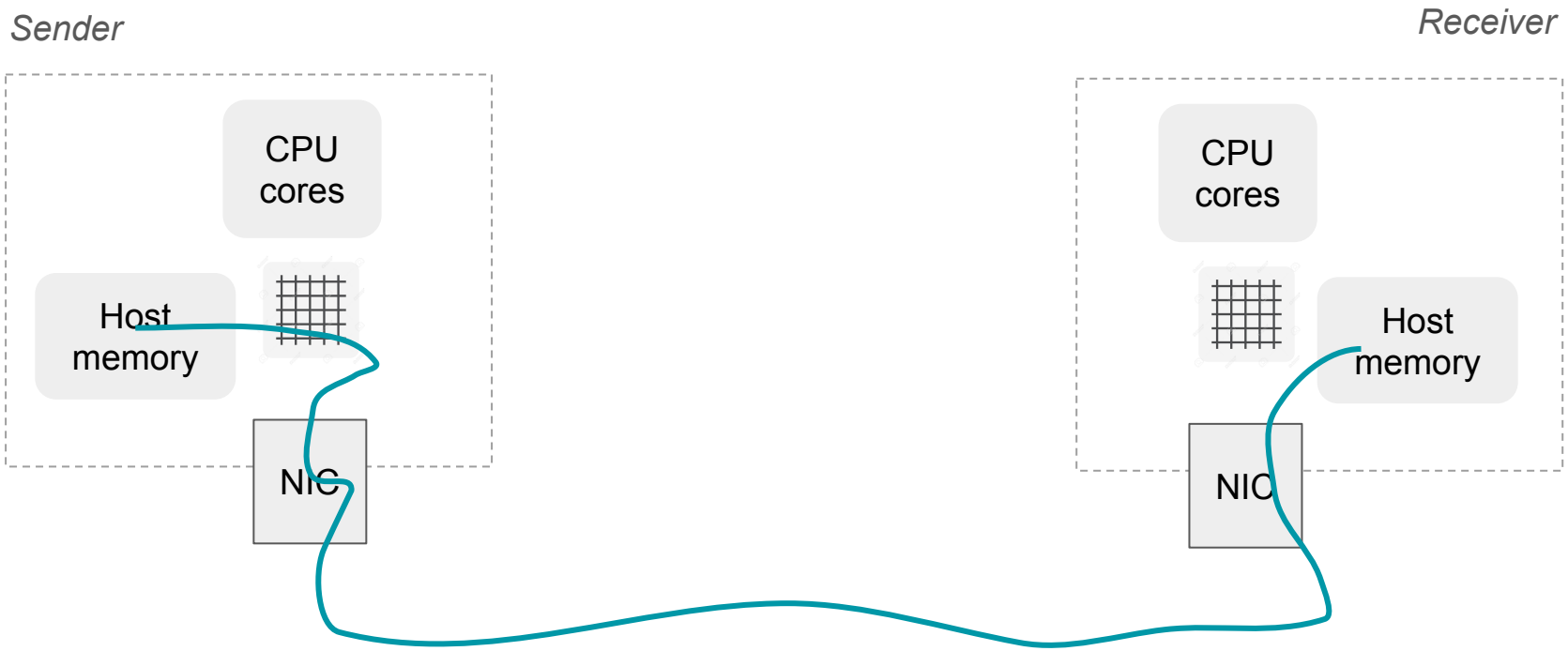
RDMA

- Remote Direct Memory Access
- Removes CPU from transfers (almost entirely)



RDMA

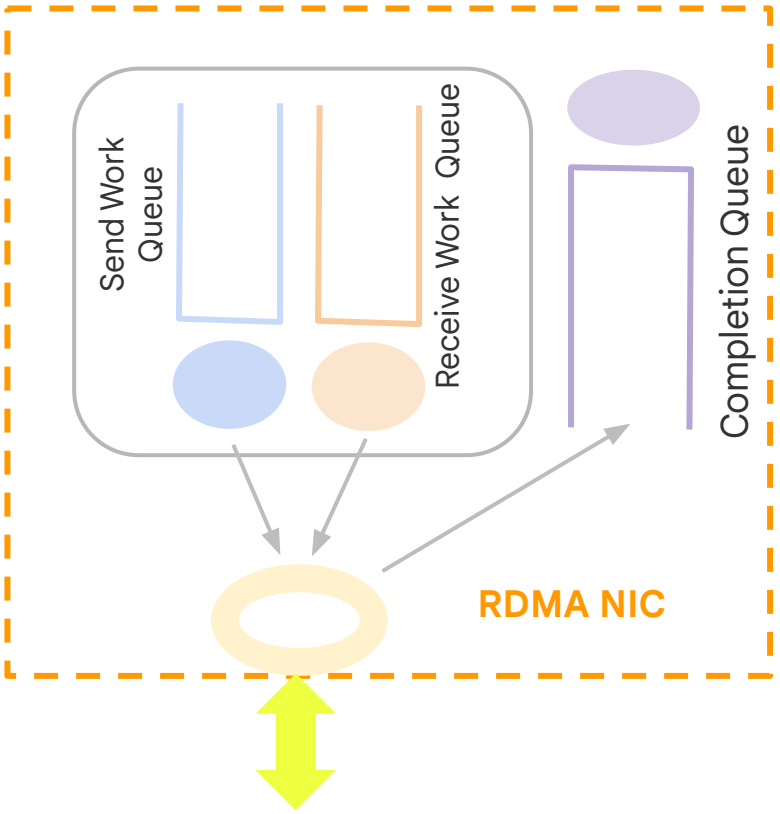
- Remote Direct Memory Access
- Removes CPU from transfers (almost entirely)



With RDMA

RDMA - How?

- Queue pairs
  - Send and receive queues with Work Queue Elements (WQEs) in them
- WQEs
  - Pointer to memory to transfer / receive data in
- Completion Queue and Completion Queue Elements (CQEs)
  - Signals about whether transfers have completed or not



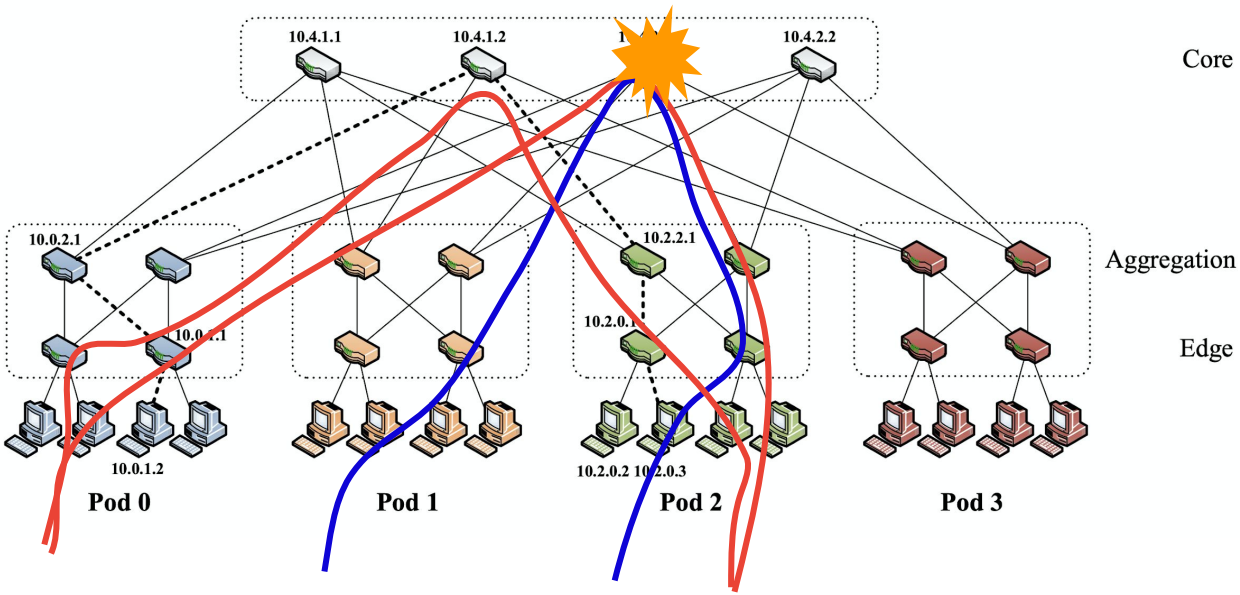
Advanced Features - Congestion Control

- Problem:
  - TCP is **buffer-filling**
  - Loss is a coarse signal
- Solution: use delay as a signal
  - Challenging to get a correct measurement
  - Why is this approach better in a DC?
- Note: this is not the only way to do advanced CC
  - Very active area of research!

```
if RTT < Target
    increase cwnd
   (Additively)
else
    decrease cwnd
   (Multiplicatively)
```

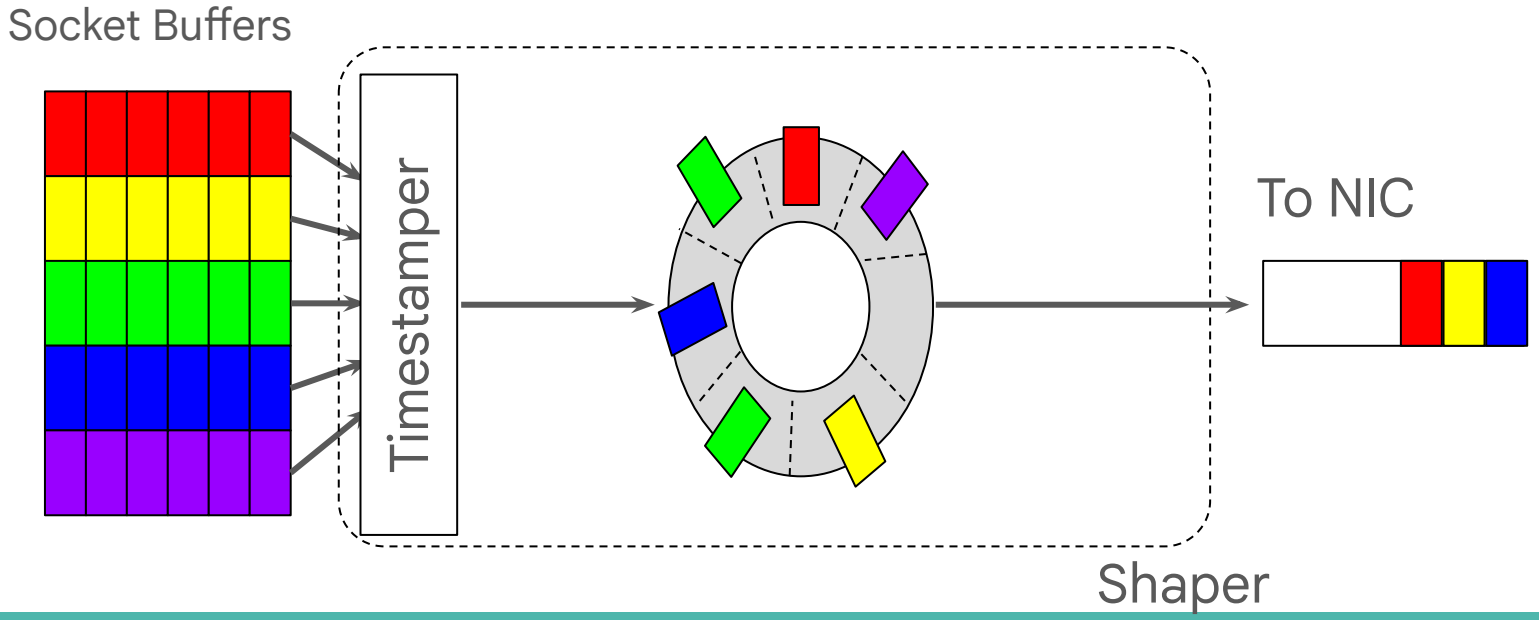
Advanced Features - Load Balancing

- Problem: ECMP can have collisions that cause hotspots
- Solution: repath based on congestion signals
  - N



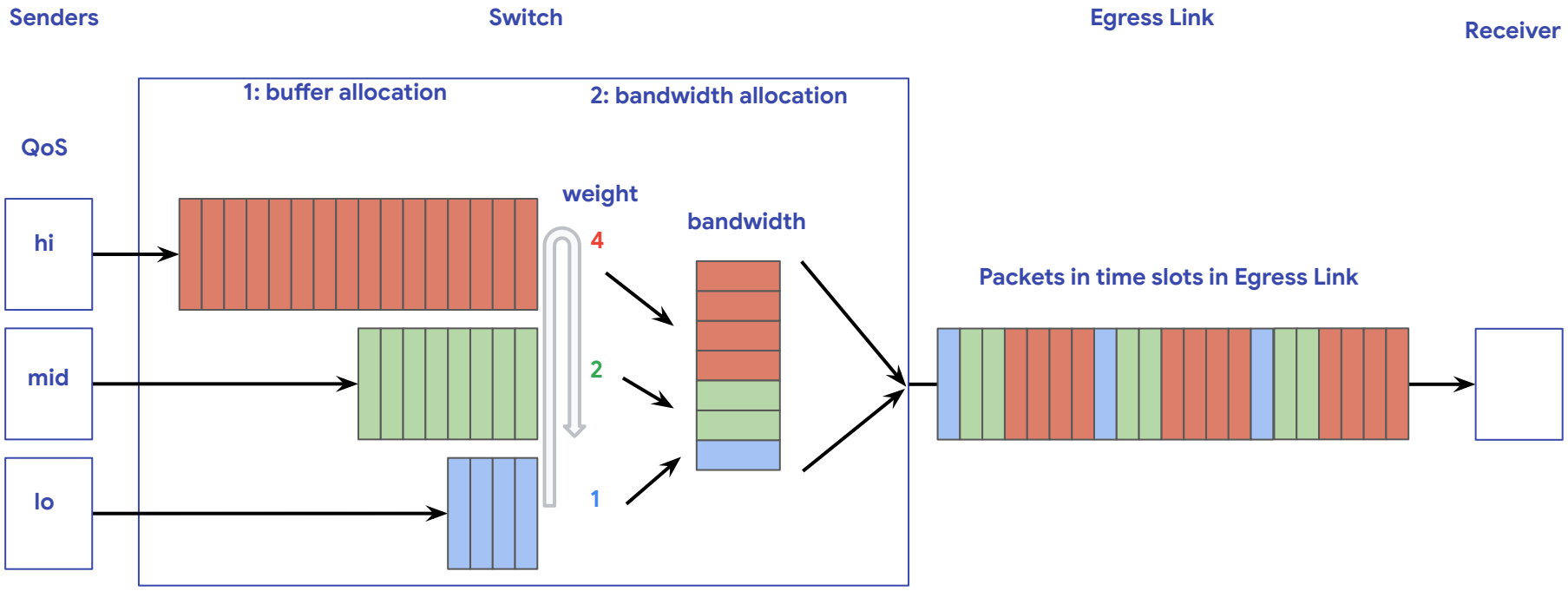
Advanced Features - Traffic Shaping

- Problem: Need to share bandwidth according to policy
  - Note: congestion control alone cannot do this
- Solution:
  - Classify traffic and rate limit it
  - Use a time wheel because managing multiple queues is difficult



Advanced Features - QoS

- Problem: Need a way to express how much of a resource a flow should get
- Solution: priority classes and enforcement in the queue



Traffic Shaping vs QoS

- Traffic Shaping:
  - Occurs at the host
  - Specifies the amount of bandwidth each type of traffic receives

- QoS (Quality of Service):
  - **Only comes into play when link is at max capacity**
  - When there is contention on resources, assigns priority to offload packets from the queue based on the class it belongs to

Wireless + Cellular

What is wireless communication?

- Transmit information without contact (e.g. with EM waves)

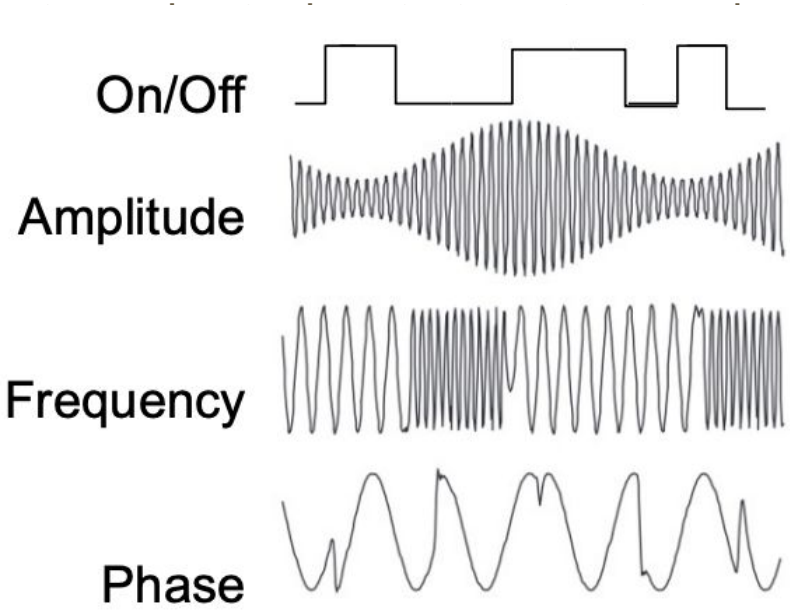
How is wireless different from wired?

Wireless ...

- is a fundamentally shared medium
- signals attenuate significantly with distance
- environments can change rapidly
- packet collisions are hard to detect

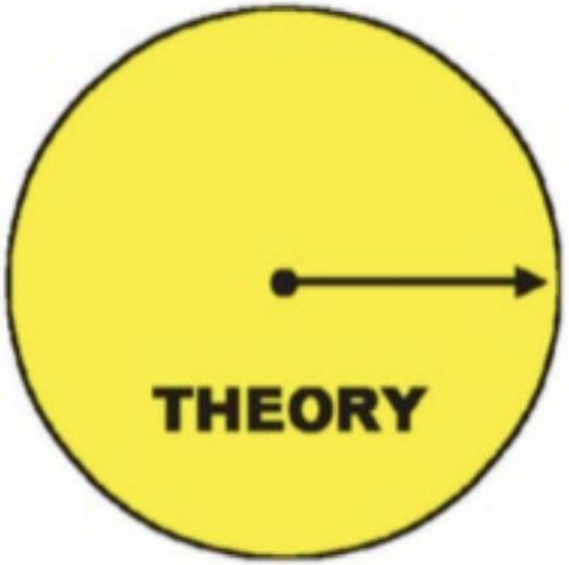
How is data encoded?

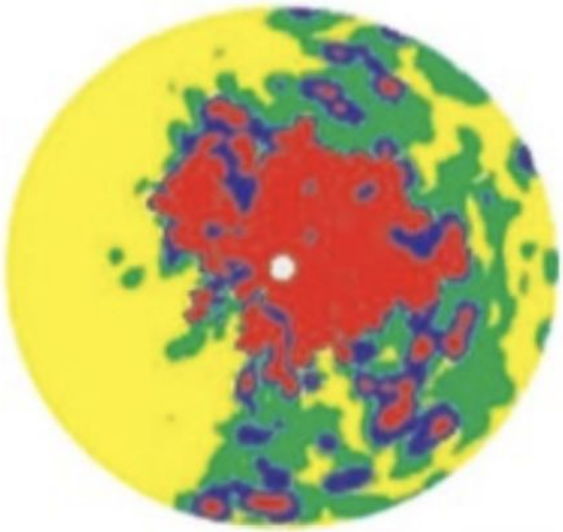
Physical Layer Modu ap to digital values (i.e. 1 or 0)



Details out of scope!! Check out EE117, EE 121, EE122

The wireless medium is messy





Medium Access Control

How to share the same transmission technology?

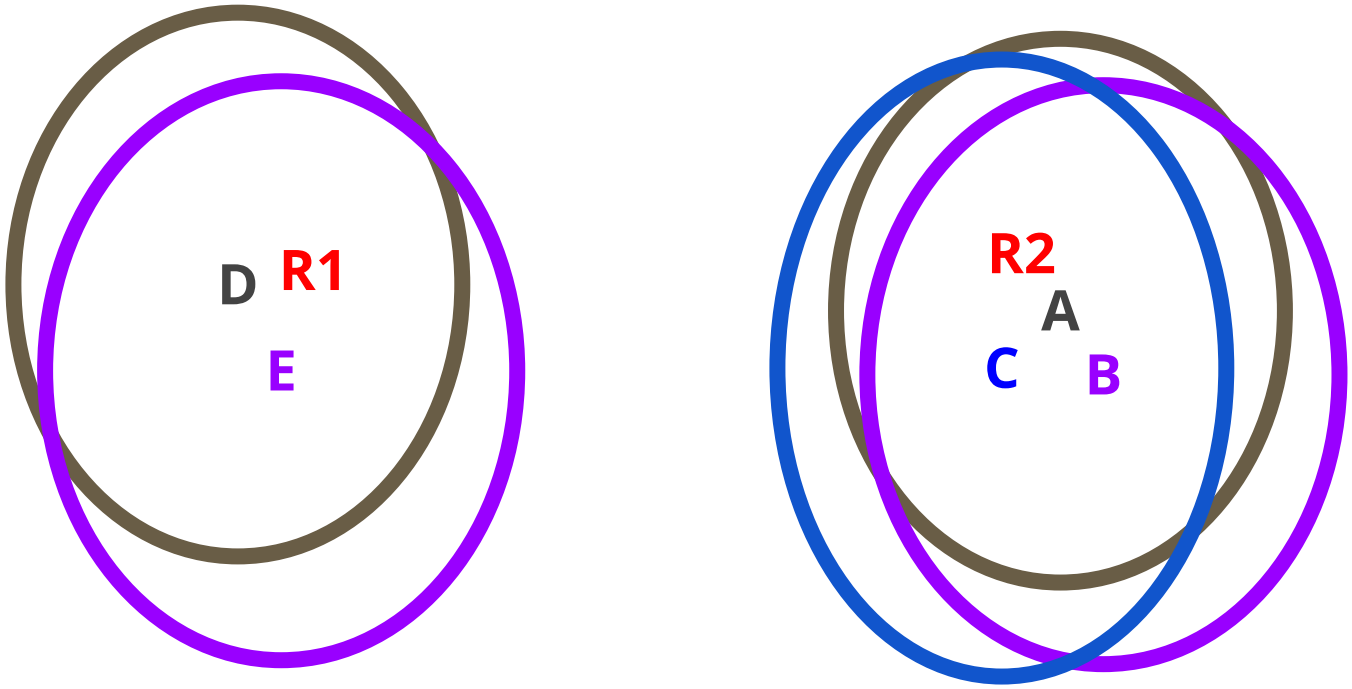
Key problem: wireless collisions are often unexpected (highly mobile, dynamic environment) and hard to detect (failure is the only indication

Solution: CSMA

Carrier Sense Multiple Access (CSMA)

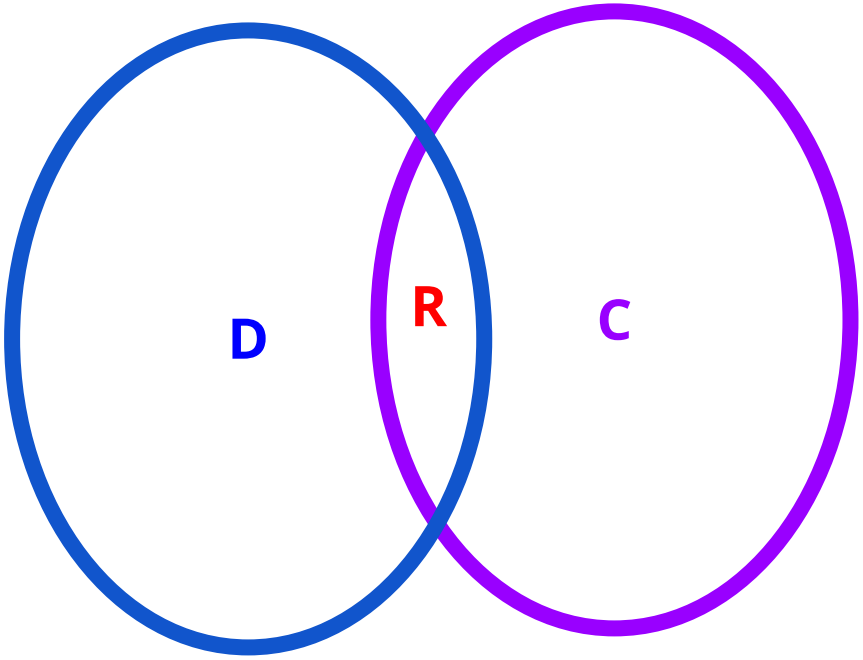
Key idea: listen for others on the medium and don't transmit if busy

- Works great when ranges of involved transmitters/receivers are all overlapping or completely disconnected



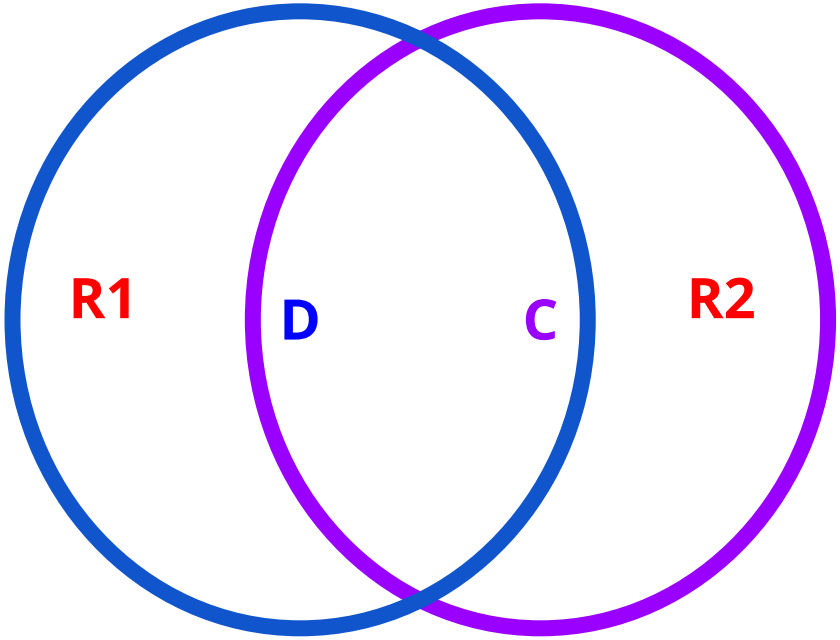
Hidden Terminal Problem

- A problem: two transmitters can’t hear each other



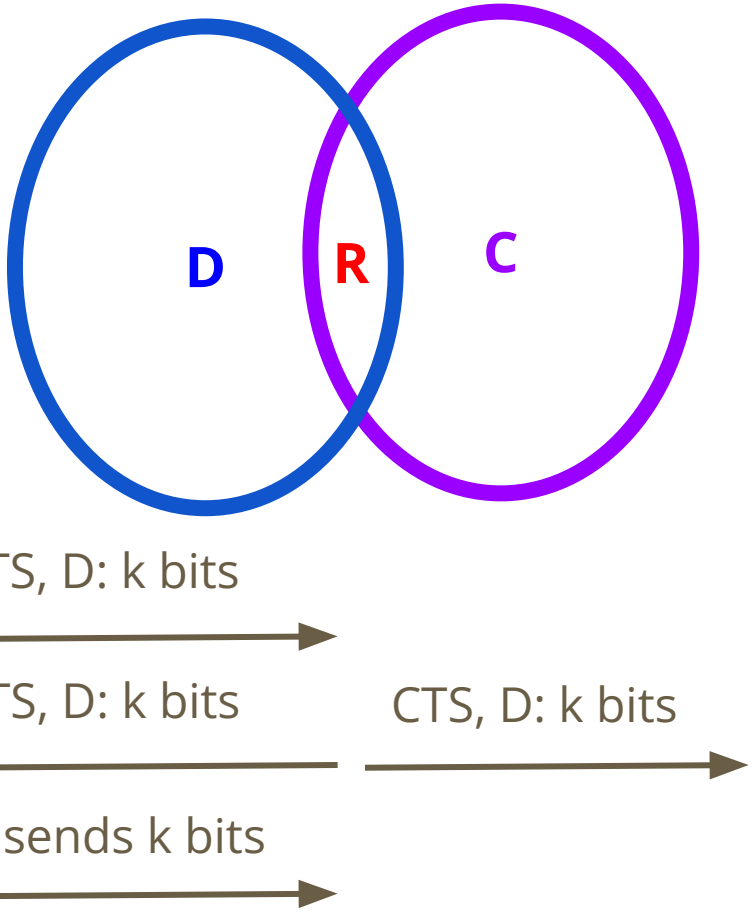
Exposed Terminal Problem

- Another problem: ‘safe’ transmissions could be blocked since D and C hear each other transmitting



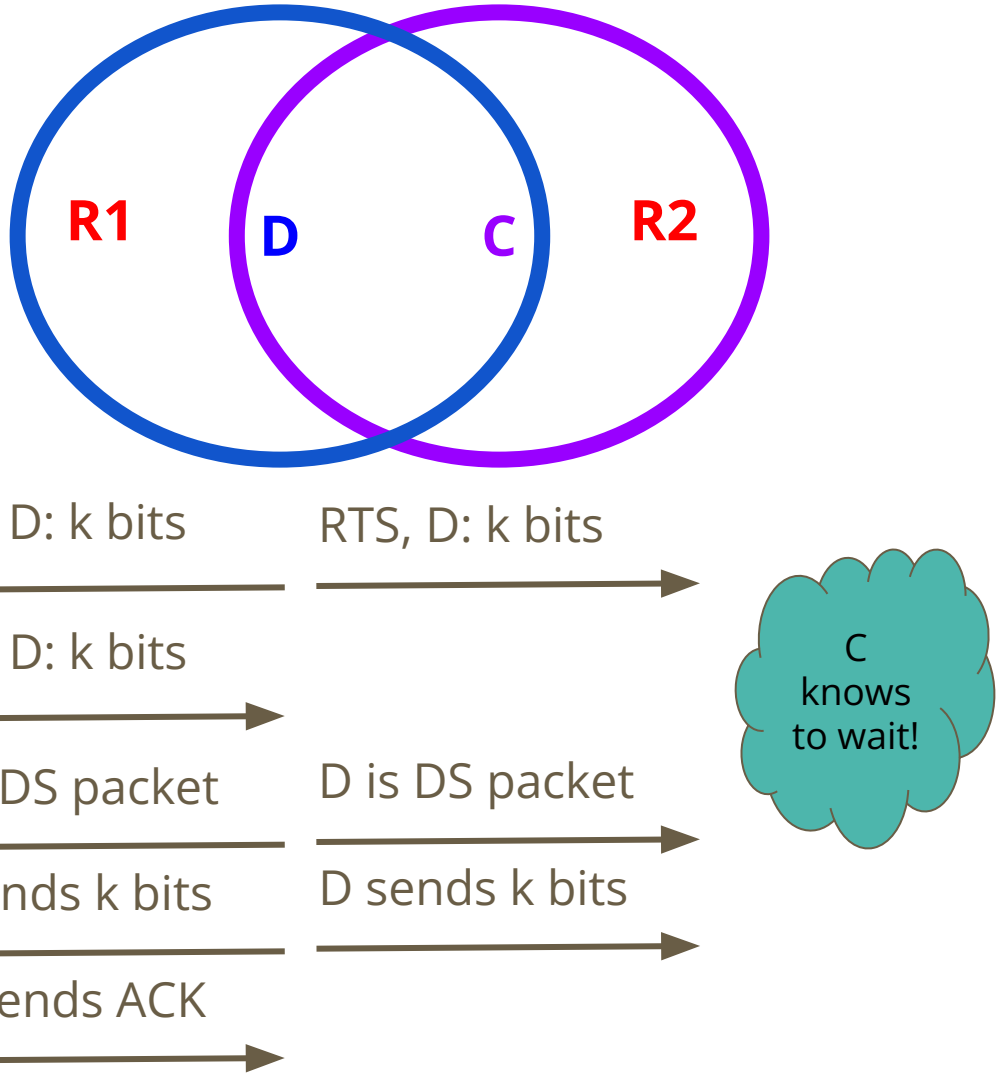
Request to Send / Clear to Send (RTS/CTS)

- Key idea: send a request-to-send and wait for a clear-to-send before transmitting



Multiple Access with Collision Avoidance for Wireless (MACAW)

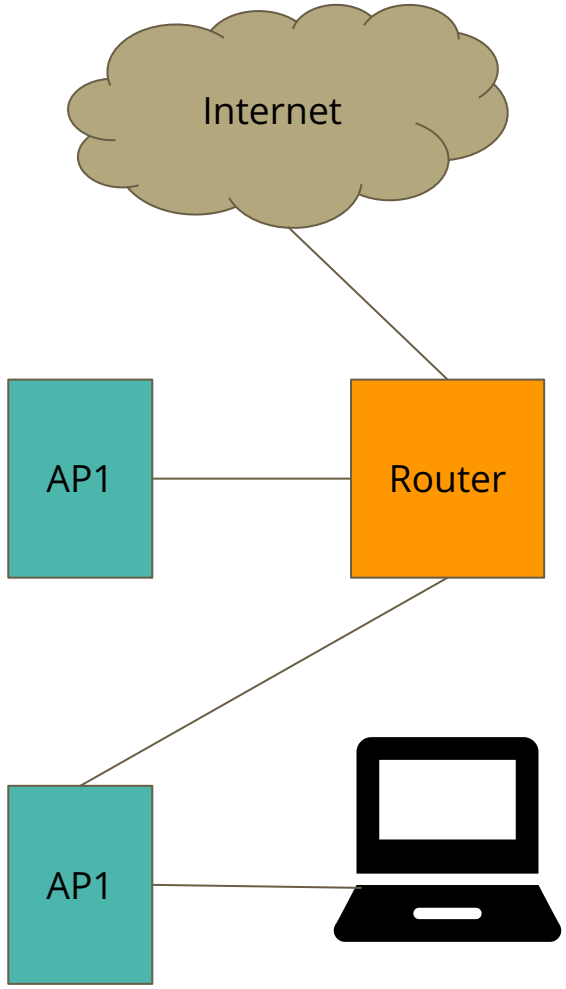
Key idea: Use RTS/CTS, but add backoff to deal with RTS collisions and reliability by using acks and a data sending indicator



WiFi

How does WiFi (802.11) work?

- Designed for a limited area (i.e. home)
- Access points (APs) set to a channel
- APs broadcast beacon messages with SSID (Service Set Identifier) and MAC Address periodically
- Hosts scan all channels to discover APs
- Hosts associate with AP



How does WiFi (802.11) work?

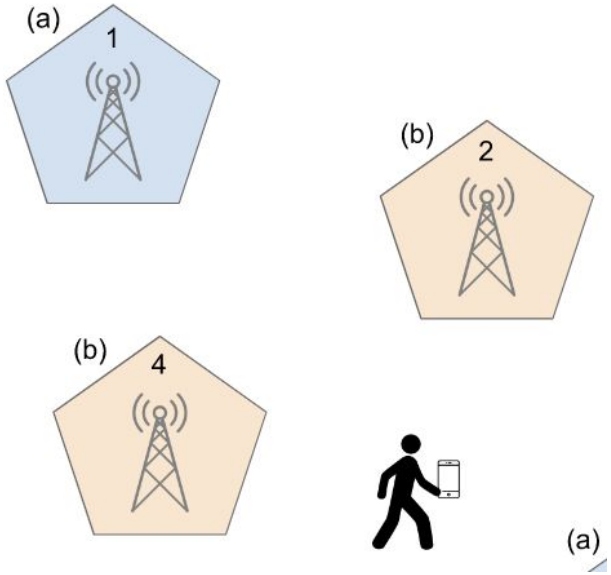
- Essentially uses CSMA (this is over simplified but details are out of scope)
- RTS/CTS is an optional feature (mainly used for large data transfers)

- Why would WiFi not use a full MACAW protocol all the time?

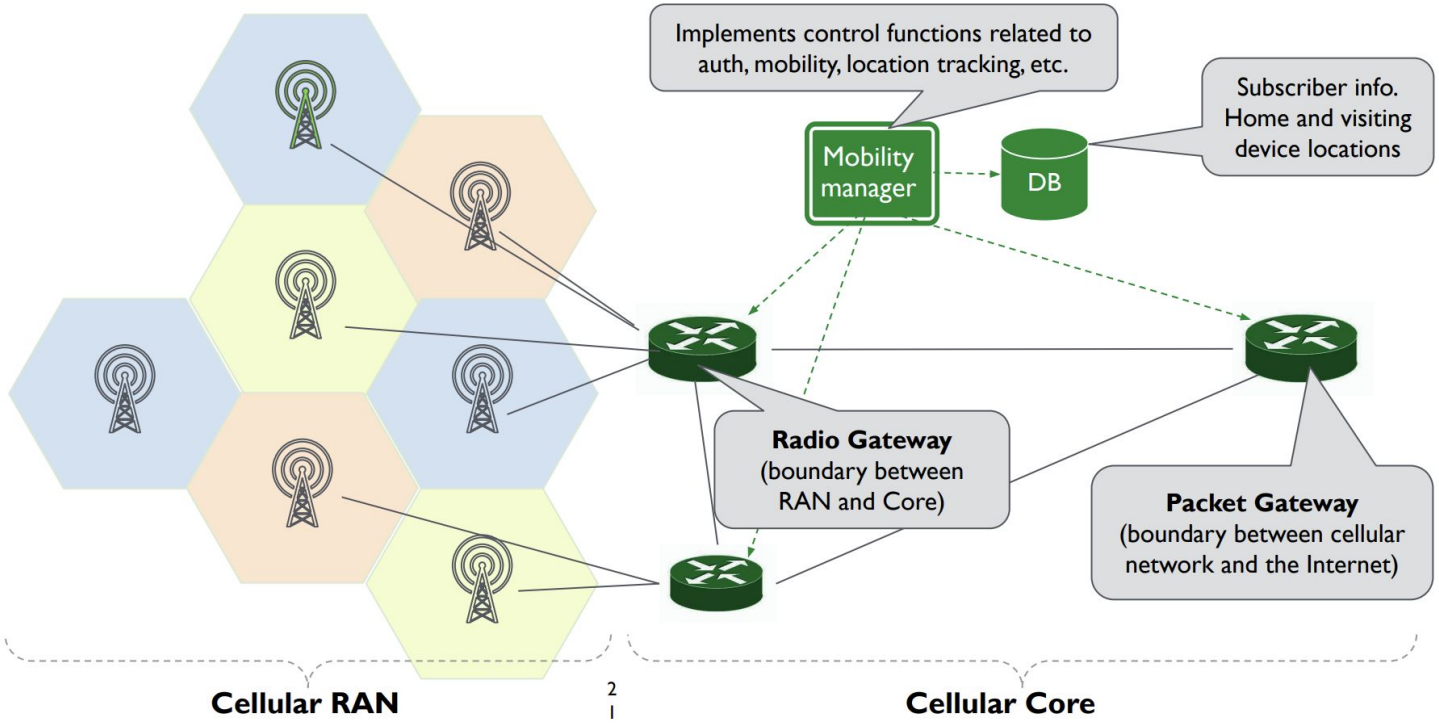
Cellular

How does cellular work?

- Key goal: Mobility
- Authentication and accountability are also first order goals
- Two Major Components:
  - Cellular RAN: between devices and cell towers
  - Cellular Core: between gateways and internet

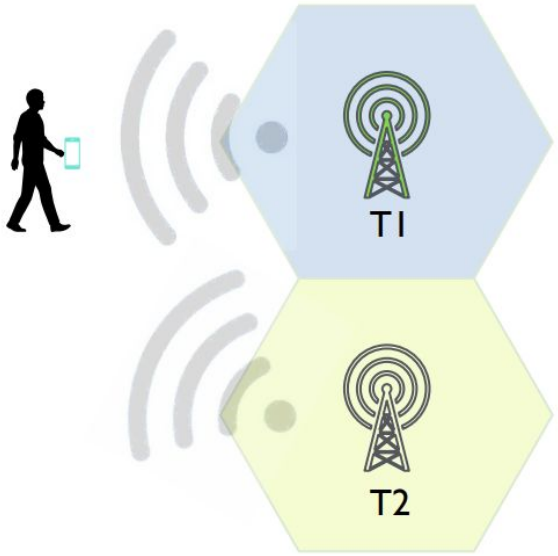


How does cellular work?



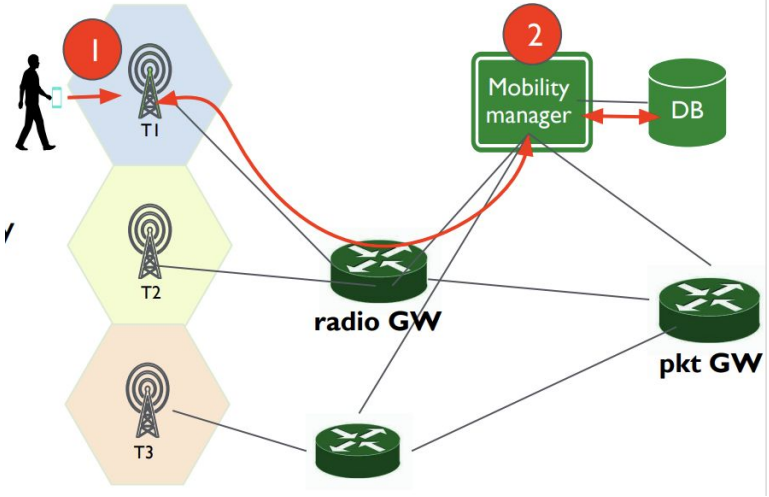
Discovery

- Towers transmit periodic “beacons” which ID the operator
- Device picks the best [illegible] ngth



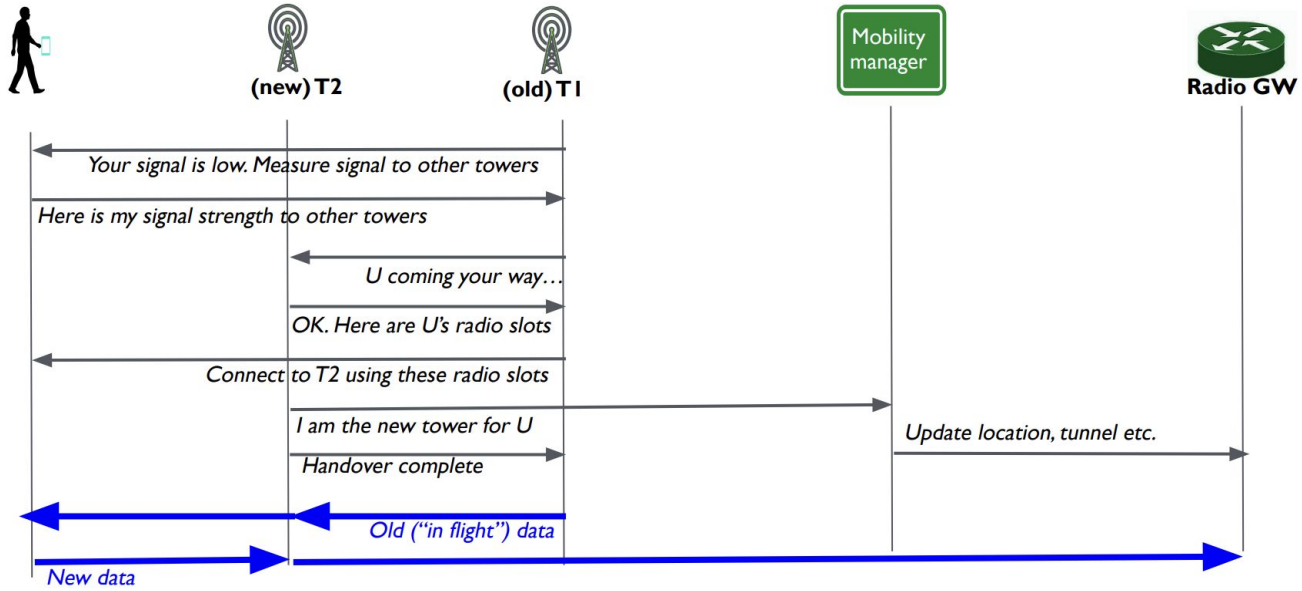
Attachment

- Device sends “attach request” to tower with best signal strength owned by operator.
- Request is processed/authenticated by the mobility manager



Handover

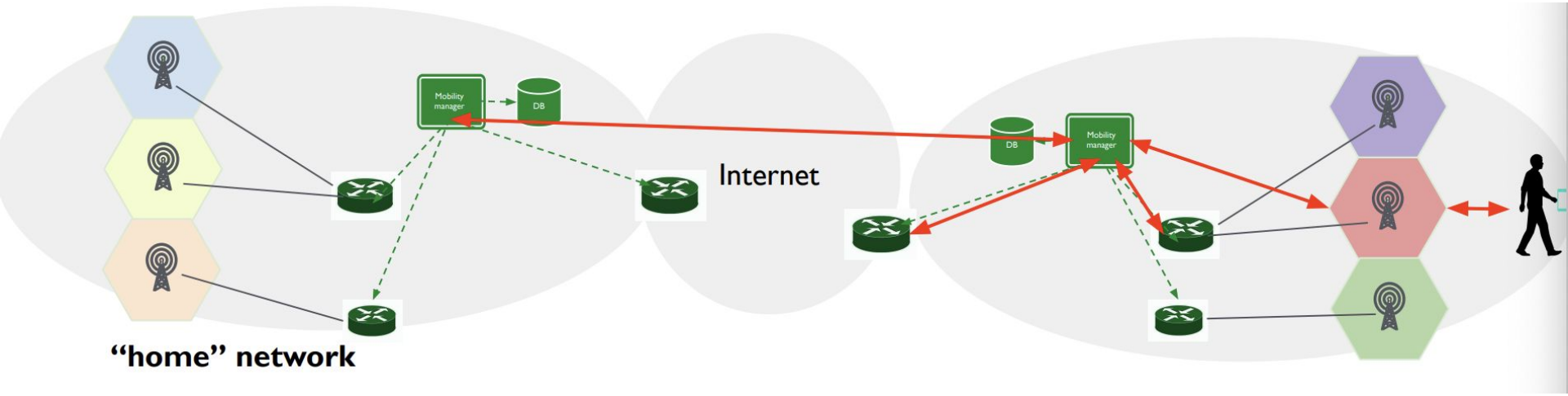
- Coop eway



Acknowledgements: adapted from Raj Jain's lectures

Roaming

- Mobility manager in the “visited” network has to check with the “home” network mobility manager for authentication.
- Can either route data through home network packet gateway or visited network packet gateway



Practice Exam Questions