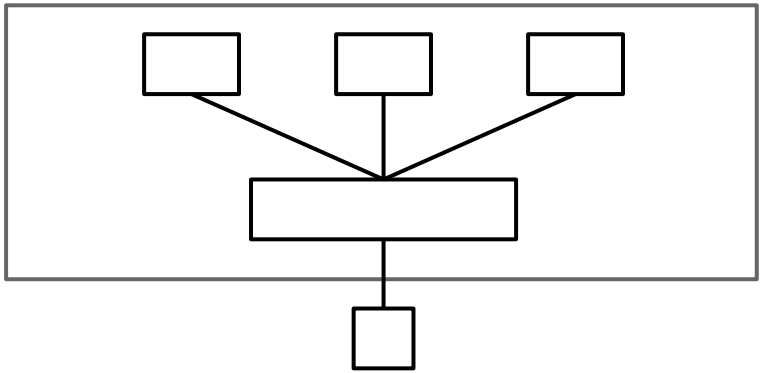


Lecture 20 (Datacenters 2)

Datacenter Routing

CS 168, Fall 2026 @ UC Berkeley

Slides credit: Sylvia Ratnasamy, Rob Shakir, Peyrin Kao, Ankit Singla, Murphy McCauley

Datacenter Routing

Lecture 19, CS 168, Spring 2026

Datacenter Routing

Datacenter Addressing

Virtualization and Encapsulation

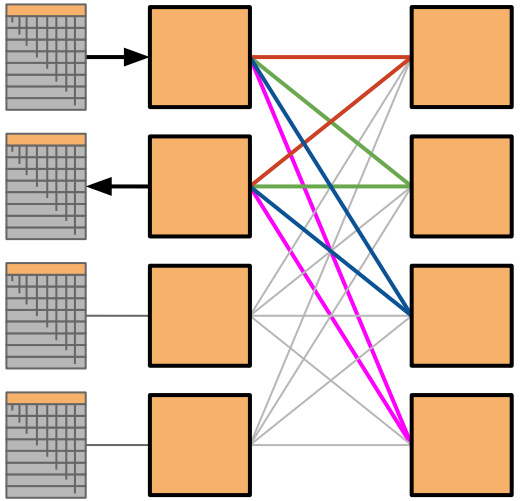
- Virtualization
- Overlay and Underlay Networks
- Multi-Tenancy and Private Networks

Why are Datacenters Different? – Multiple Paths

Recall: In a Clos network, there are many paths between two servers.

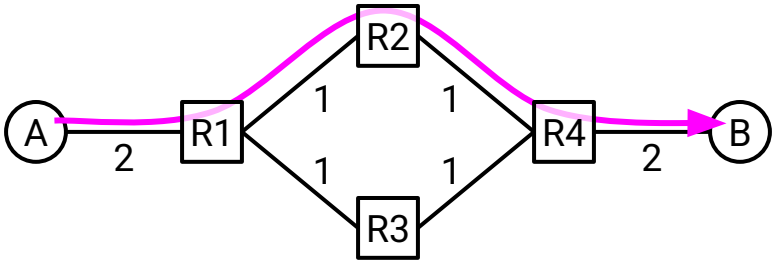
Our routing algorithms so far pick a *single* path from source to destination.

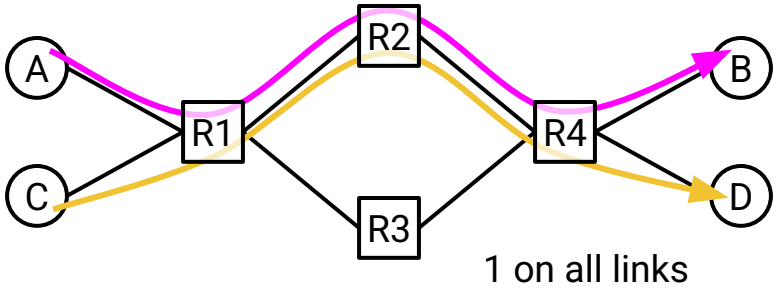
How do we modify our routing protocols to find multiple paths?



Why are Datacenters Different? – Multiple Paths

We want routing protocols to find multiple paths between two hosts.





Bandwidth:

- If A→B only used a single path, it can only send at 1 Gbps.
- If A→B split traffic across both paths, it can send at 2 Gbps.
- Our routing protocols can only send traffic along one path.

Coordination:

- A→B and C→D might send along the same path, overloading it.
- C→D should switch to using the bottom path instead.
- Our routing protocols don't assume coordination between flows.

Why are Datacenters Different? – Multiple Paths

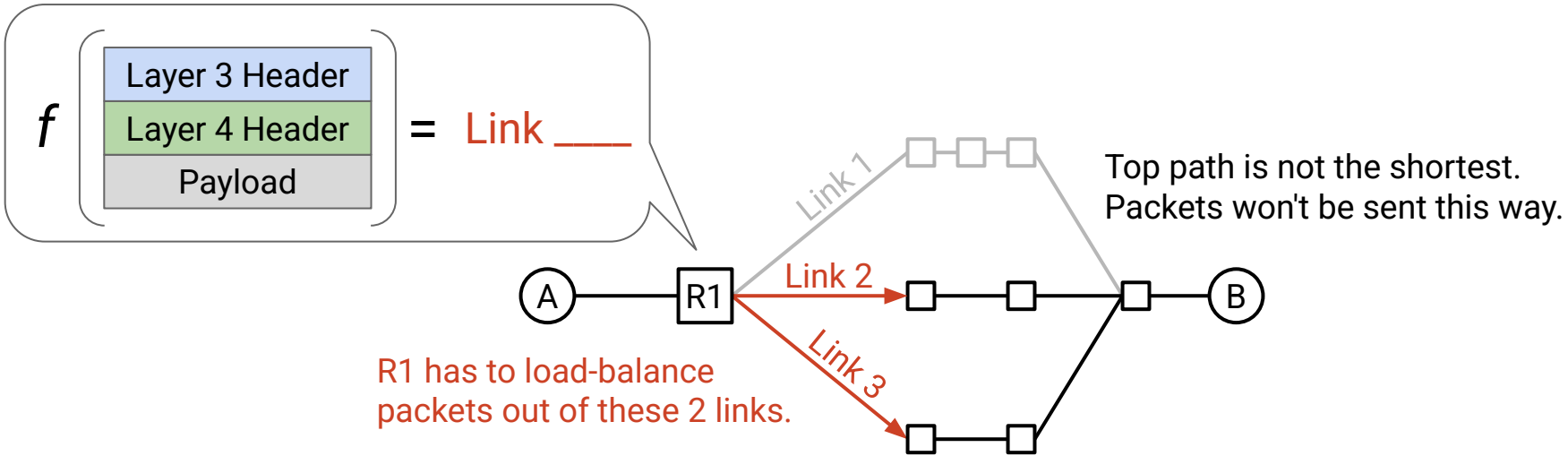
Equal Cost Multi-Path (ECMP) finds all of the shortest paths (with equal cost).

Then, we load-balance packets across those paths.

ECMP Load-Balancing

If there are multiple shortest paths, how does the router load-balance packets between those paths?

- We need a function to compute a link for each packet: *f*(packet) → link.
- Routers receive a packet and run this function to pick a link.
- What's a good *f*?



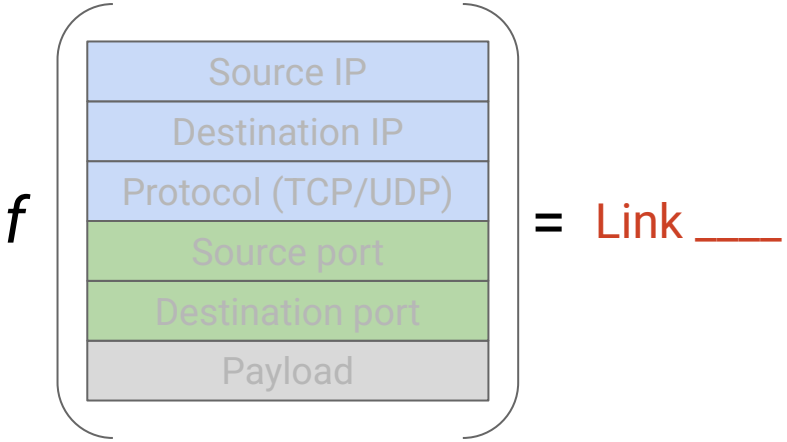
ECMP Load-Balancing Strategy #1 – Round-Robin

Round-robin: Ignore packet contents, and alternate sending between links.

- Link 1, 2, 1, 2, 1, 2, ...

Problem: TCP packet reordering.

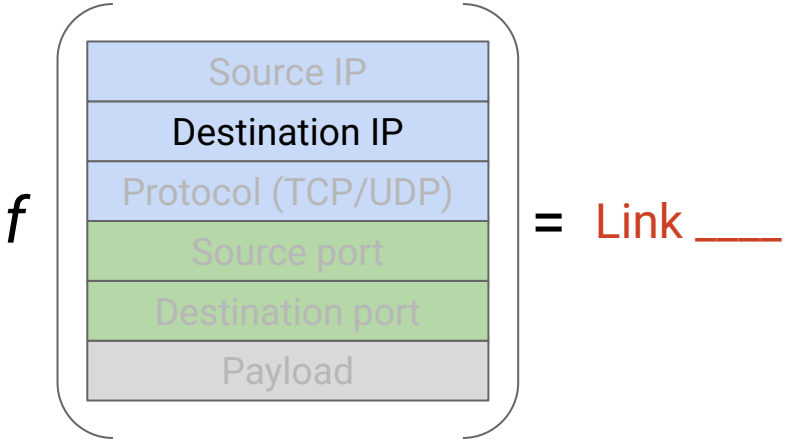
- Equal-cost paths might have different latency. (*Cost could be based on something else.*)
- If odd packets on slow path, and even packets on fast path:
  The recipient gets all the even packets before the odd packets.
- Recipient has to buffer packets, resulting in poor performance.



ECMP Load-Balancing Strategy #2 – Destination-Based

Use destination IP to choose link.

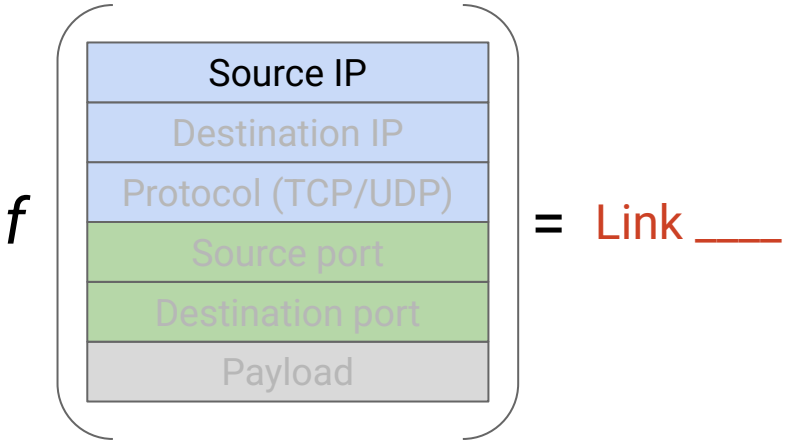
Problem: If lots of sources sending to the same destination, one link is overloaded.



ECMP Load-Balancing Strategy #3 – Source-Based

Use source IP to choose link.

Problem: If the same source is sending to lots of destinations, one link is overloaded.

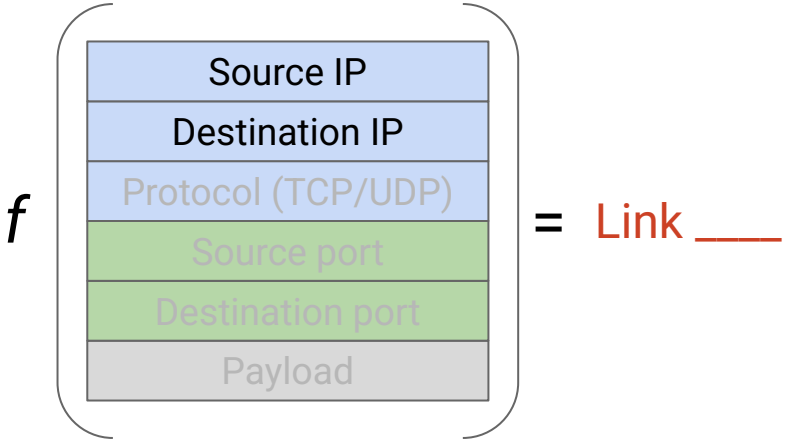


ECMP Load-Balancing Strategy #4 – IP-Based

Use source and destination IP to choose link.

Using both values helps spread out packets across links.

Problem: What if there are multiple large flows between the same two servers?



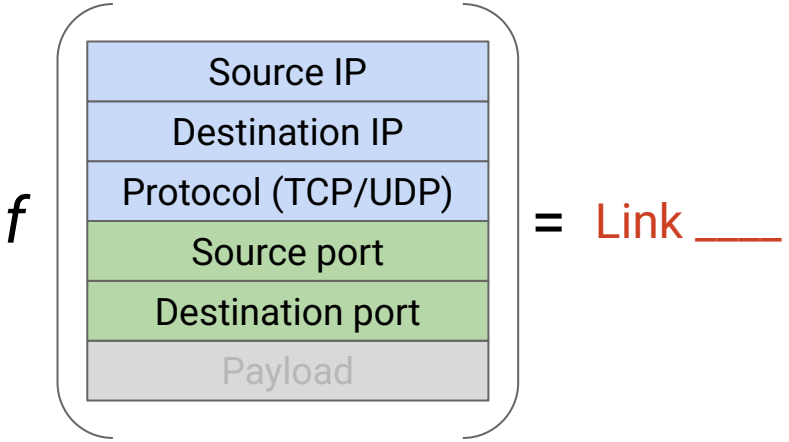
ECMP Load-Balancing Strategy #5 – Flow-Based

Use 5 values to choose link:

- Source and destination IP.
- Source and destination port.
- Protocol (TCP or UDP).

This is called **per-flow** load-balancing.

- Each flow has a unique **5-tuple**.
- All packets in the same flow use the same link (no reordering problems).
- Modern commodity routers have support for reading these 5 values.



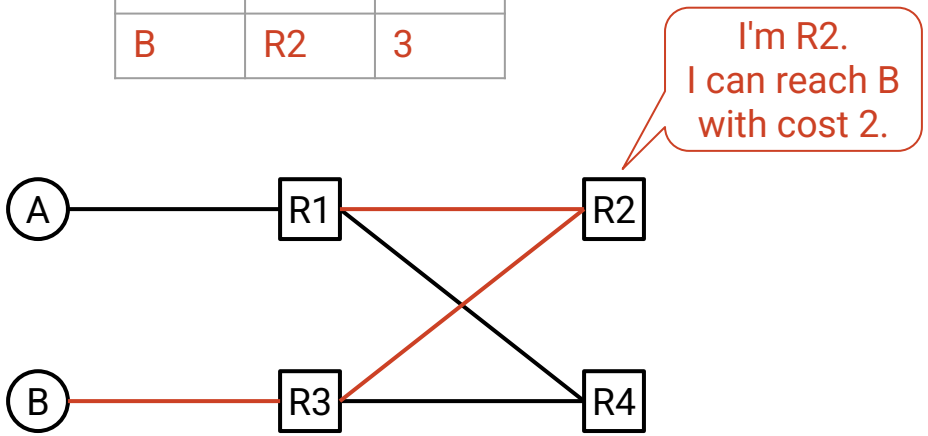
Note: This does not account for flows being different sizes.
Tracking flow size is more complex, for not a lot of benefit.

Multi-Path Distance-Vector Protocols

How do we adjust distance-vector protocols to support multiple paths?

- Recall distance-vector: Routers advertise paths and costs.
- If the advertised path is better than the current best path, accept.

R1's forwarding table		
To:	Via:	Cost:
B	R2	3



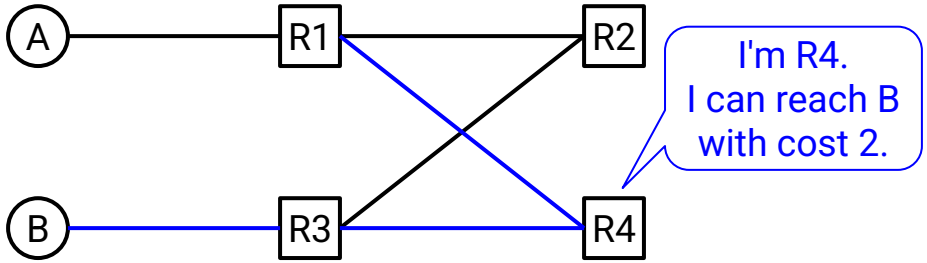
Multi-Path Distance-Vector Protocols

Normal distance-vector:

- If the advertised path has the *same* cost as the current best-known cost, reject.

R1's forwarding table		
To:	Via:	Cost:
B	R2	3

I already have a cost-3 path to B.
Your path is not better, so I'll ignore it.



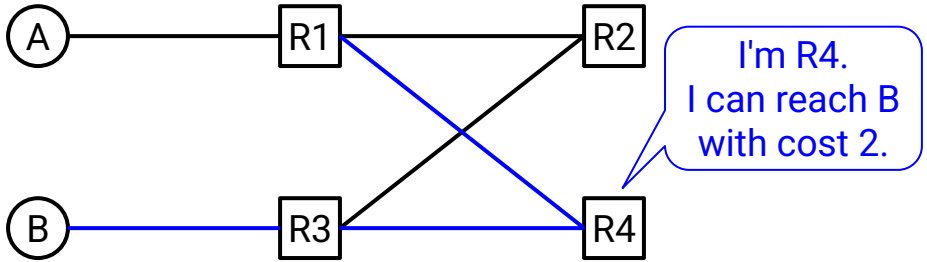
Multi-Path Distance-Vector Protocols

Multi-path distance-vector:

- If the advertised path has the *same* cost as the current best-known cost, *accept*.
- The forwarding table can now store multiple next-hops per destination.
- Use *f* (load-balancing) to choose a next-hop for each packet.

R1's forwarding table		
To:	Via:	Cost:
B	R2	3
B	R4	3

Your path is equally good.
I'll remember both paths.



Multi-Path Link-State Protocols

Recall link-state: Each router stores the full network graph.

- Each router uses the graph to calculate shortest path to each destination.
- Must extend protocol to calculate *all* shortest paths to each destination.
- The forwarding table can now store multiple next-hops per destination.
  - Just like multi-path distance-vector.

Datacenter Addressing

Lecture 19, CS 168, Spring 2026

Datacenter Routing

Datacenter Addressing

Virtualization and Encapsulation

- Virtualization
- Overlay and Underlay Networks
- Multi-Tenancy and Private Networks

Why are Datacenters Different? – Scaling Routing

Scaling routing protocols in datacenters:

- Distance-vector: Separate advertisements per destination.
  - In a datacenter: 100,000+ destinations being advertised.
- Link-state: Advertisements get flooded along every link.
  - In a datacenter: Clos networks can have 10,000+ links.

Recall: Clos networks scale by using commodity switches.

- Memory, CPU, and forwarding table resources are limited.
- We can't store table entries for every destination.

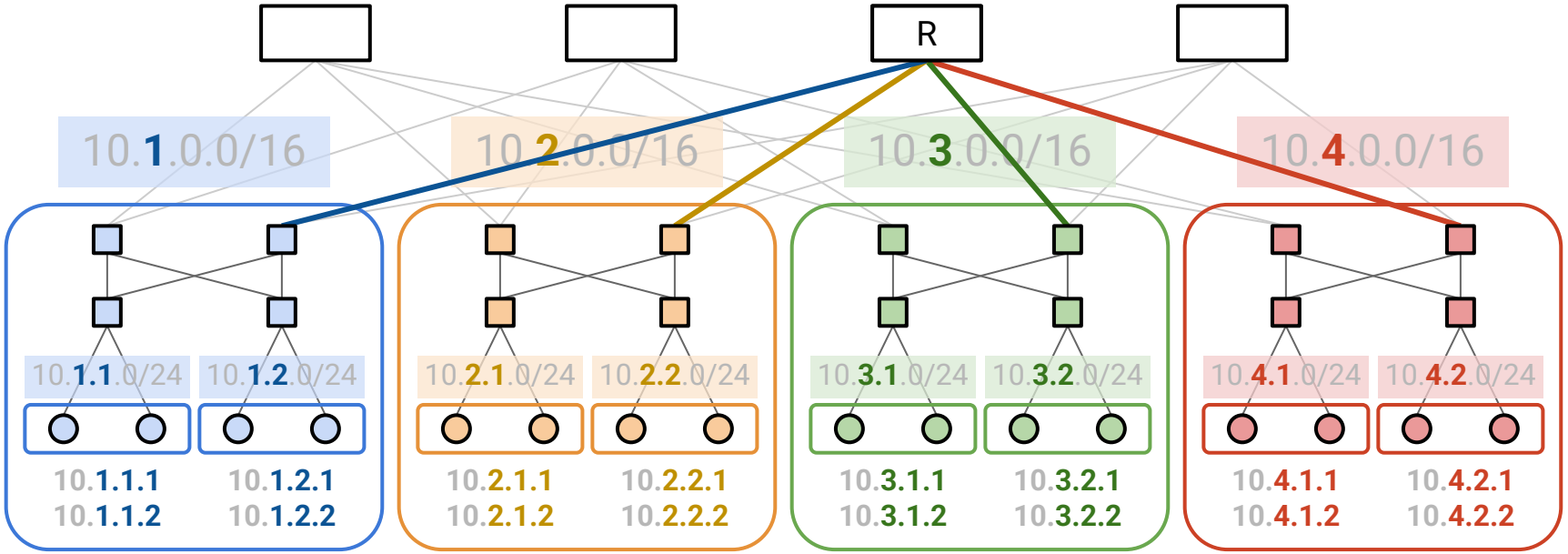
Topology-Aware Addressing

We can scale routing using hierarchical addressing.

- On the Internet: Hierarchy is based on geography and organizations.
- In datacenters: Hierarchy is based on physical organization in the datacenter.
  - The address tells us *where* in the building the host is.
  - Exploits the idea that the topology is *regular* (e.g. racks organized in rows).

Topology-Aware Addressing

R's forwarding table	
To:	Use:
10.1.0.0/16	Link 1
10.2.0.0/16	Link 2
10.3.0.0/16	Link 3
10.4.0.0/16	Link 4



Topology-Aware Addressing

Routing aggregation makes our forwarding tables:

- Smaller: One entry represents a range of hosts.
- More stable: If a host inside a subnet goes away, R's table doesn't change.

R's forwarding table	
To:	Use:
10.**1**.0.0/16	**Link 1**
10.**2**.0.0/16	**Link 2**
10.**3**.0.0/16	**Link 3**
10.**4**.0.0/16	**Link 4**

Nice example of what can be achieved in a controlled network.

- Requires the operator to assign all the addresses.

Virtualization

Lecture 19, CS 168, Spring 2026

Datacenter Routing

Datacenter Addressing

Virtualization and Encapsulation

- **Virtualization**
- Overlay and Underlay Networks
- Multi-Tenancy and Private Networks

Physical Datacenter Limitations

If we hosted applications directly on servers, we'd have some problems.

- Google introduces a new application: Somebody has to install a new server.
- The application grows: Somebody has to install more servers.
- A link goes down: Somebody has to move the server somewhere else.

Fundamental issue:

- Physical datacenters are rigid and structured.
- But, applications are constantly changing.
  - Hosts are rapidly added, removed, and moved.
- Changing physical infrastructure is hard.

Physical Datacenter Limitations

Scaling is another problem.

- A lightweight application might not need all the server resources.
- But the application still wants its own server (e.g. for security reasons).
- Inefficient use of resources.

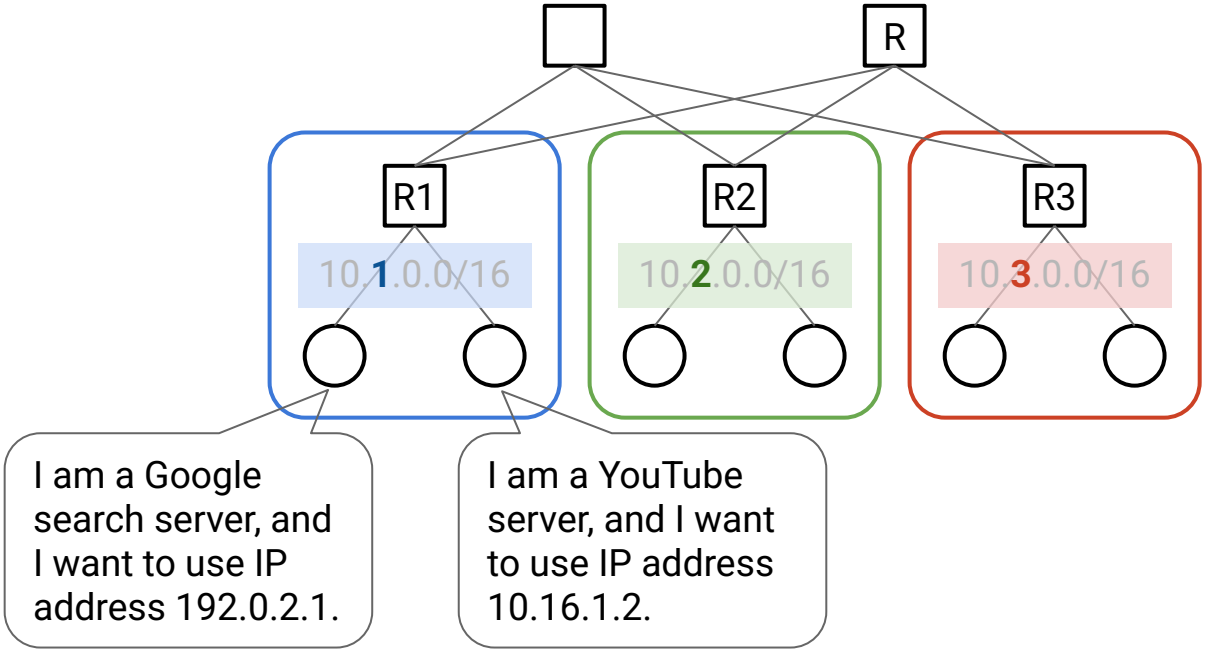
Routing is another problem.

- Applications might want to choose their own IP addresses.
  - Example: It wants to follow organizational hierarchy.
  - Its chosen address might not be consistent with datacenter addresses.
- If the application moves elsewhere in the datacenter, it might want to keep its original address.
  - But the datacenter addressing requires us to change addresses.

Physical Datacenter Limitations

Applications choosing IP addresses is incompatible with our datacenter addressing.

Applications moving and keeping the same IP address is also incompatible.



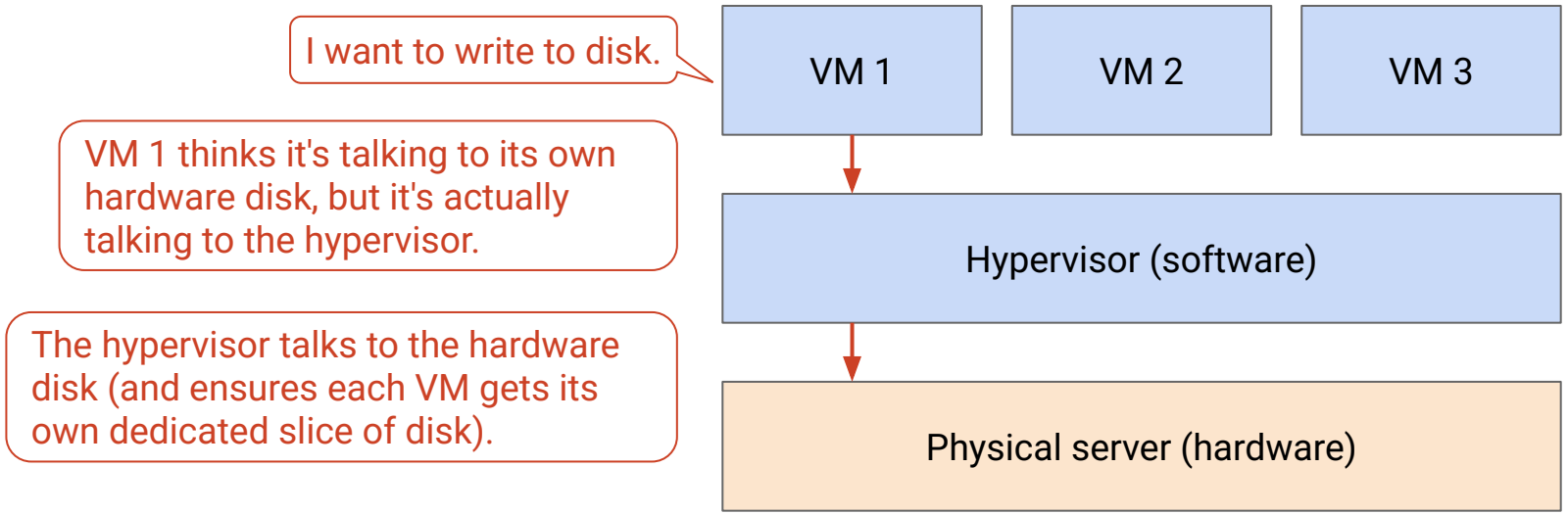
R's forwarding table	
To:	Next hop:
10.2.0.0/16	R2
10.3.0.0/16	R3
192.0.2.1	R1
10.16.1.2	R1

We can't aggregate these rows!

Virtual Machines

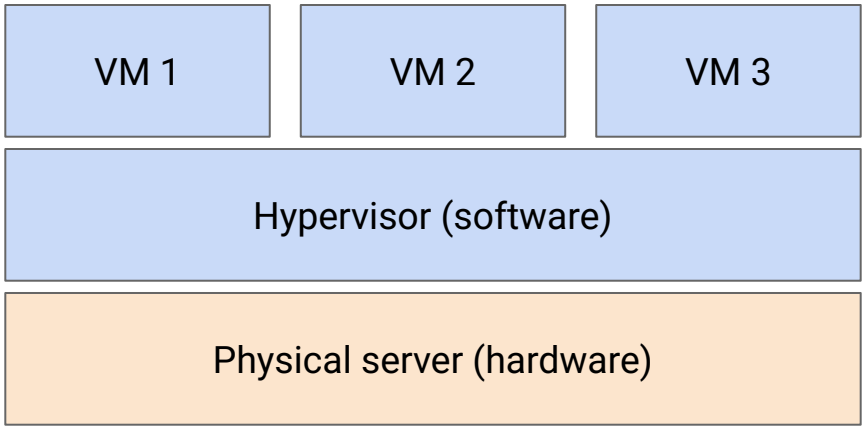
Virtualization lets us run one or more virtual machines on a single physical machine.

- Each virtual machine (VM) provides the illusion of a dedicated physical machine.
- The physical machine runs a **hypervisor** that gives each VM its own operating system, and manages resources between VMs.



Why is virtualization useful?

- We can rapidly add, remove, and move VMs without touching physical servers.
- Multiple applications can run on a single server.
  - Hypervisor can enforce separation of servers (e.g. for security).
  - More efficient use of resources.



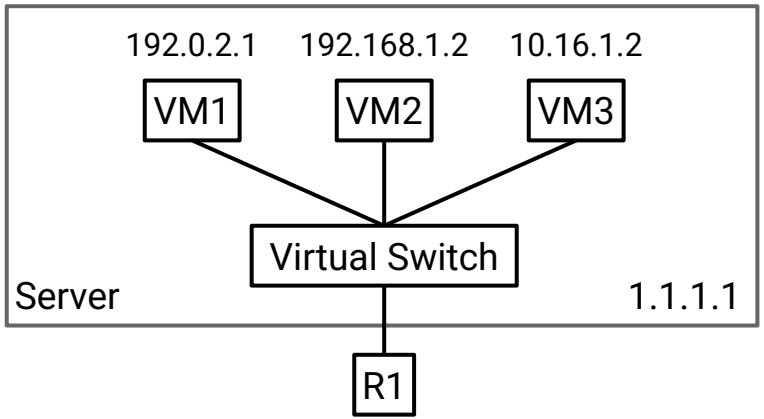
Virtual Switches

The physical server has a single network card (NIC) and a single IP address.

- We need to give each VM the illusion of its own dedicated NIC and IP address.
- The physical NIC receives packets meant for several VMs.

Solution: Run a **virtual switch** in software, on the server.

- Does the same things as a real switch (e.g. forwards packets).
- Each VM is connected to the virtual switch (in software).
- The virtual switch uses the physical NIC to send and receive packets.



Note: Virtual switches can run in software on a general-purpose CPU because they serve less traffic than a real-world hardware switch. (Just the traffic from the VMs on that server.)

Overlay and Underlay Networks

Lecture 19, CS 168, Spring 2026

Datacenter Routing

Datacenter Addressing

Virtualization and Encapsulation

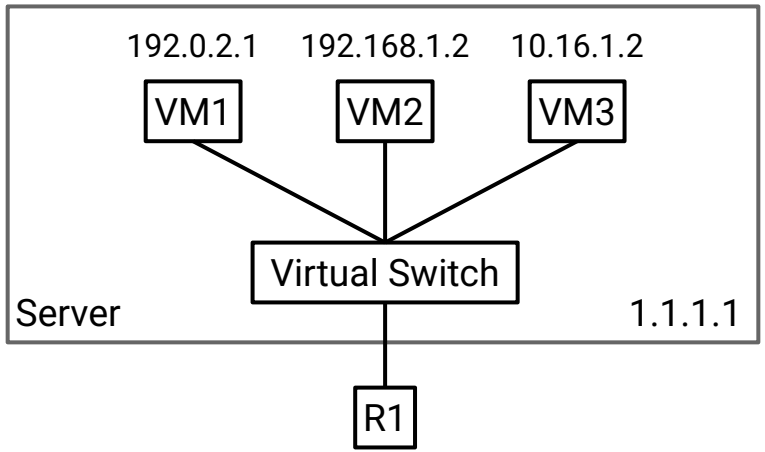
- Virtualization
- **Overlay and Underlay Networks**
- Multi-Tenancy and Private Networks

Routing with Virtualization

VMs let us easily add/remove servers, and use server resources efficiently.

But we still haven't solved our routing problem.

- VM addresses don't fit in the datacenter addressing scheme.
- Datacenter routers can't use aggregation to scale.



Problem: Routing with Virtualization

Key problem: We have 2 different addressing systems to think about.

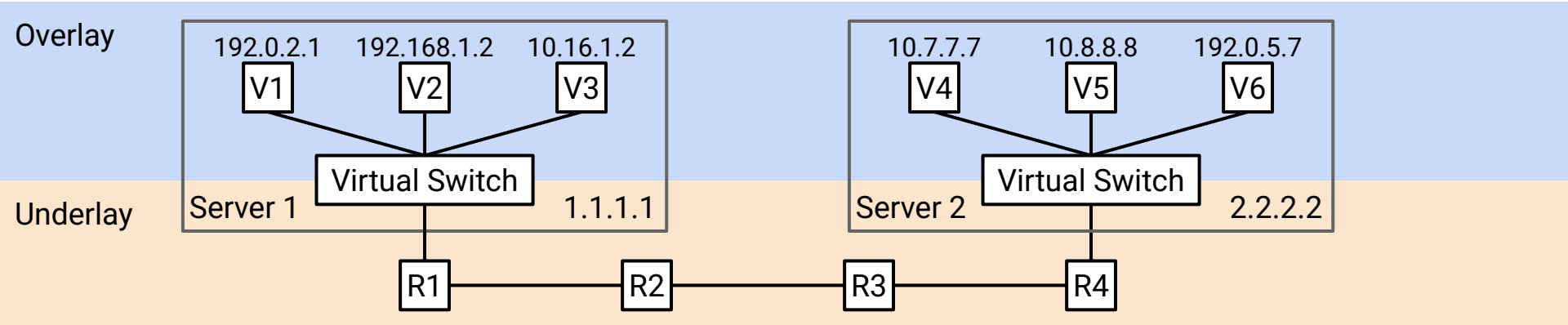
- Think of them as two "sub-layers" inside Layer 3 (IP).

The **underlay network** thinks in terms of physical addresses.

- Physical servers and routers in the datacenter.
- Addresses based on datacenter topology.

The **overlay network** (VMs) thinks in terms of virtual addresses.

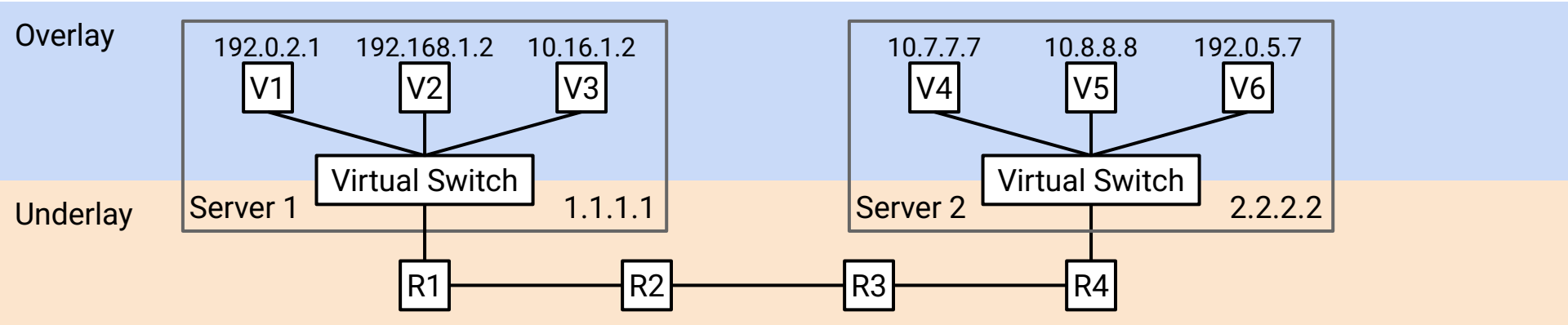
- Addresses based on organizational hierarchy.



Problem: Routing with Virtualization

Ideally, we want to think about each layer separately.

- Underlay scales by aggregating physical addresses (ignoring virtual addresses).
- Overlay scales because any-to-any routing isn't needed.
  - Example: A YouTube VM only needs to know about other YouTube VMs.
- Each layer should only need to think about its own addressing system.



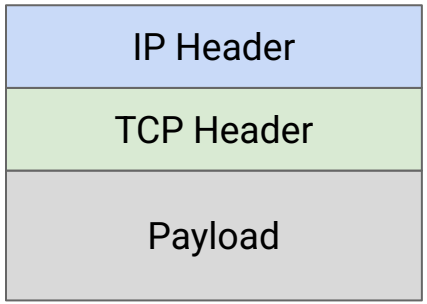
Solution: A New Layer

How do we bridge the gap between the overlay and underlay?

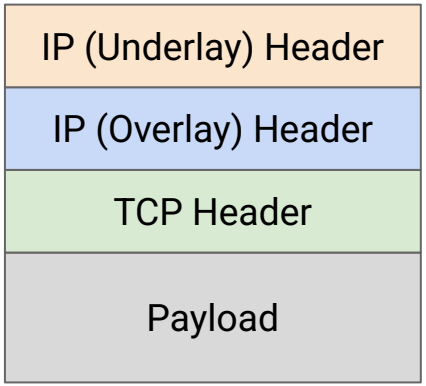
Use the same strategies with layering and header from the Internet design!

- We could introduce an extra layer (with an extra header).
- One layer thinks about the underlay – physical addresses.
- The other layer thinks about the overlay – virtual addresses.

The new layer could be a second IP header, or a new kind of header (not IP).



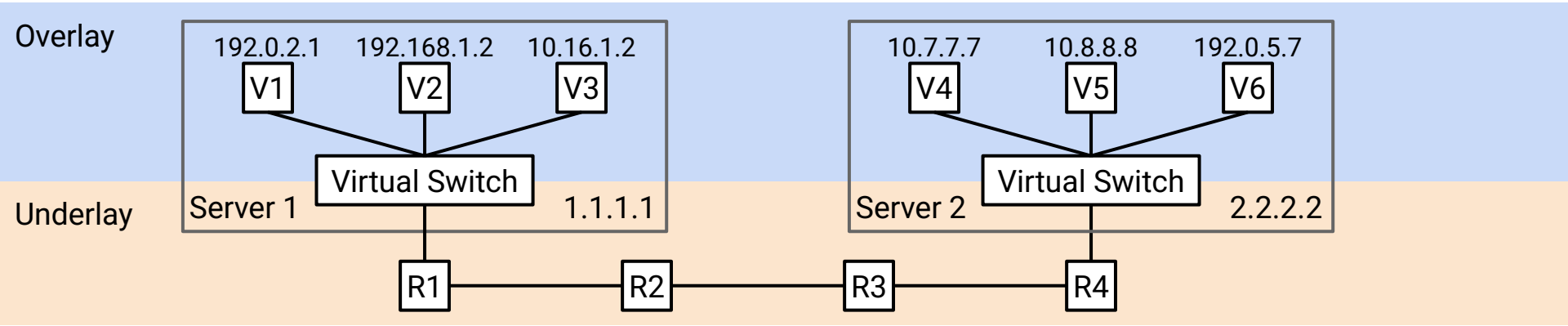
Original design.



The new design.

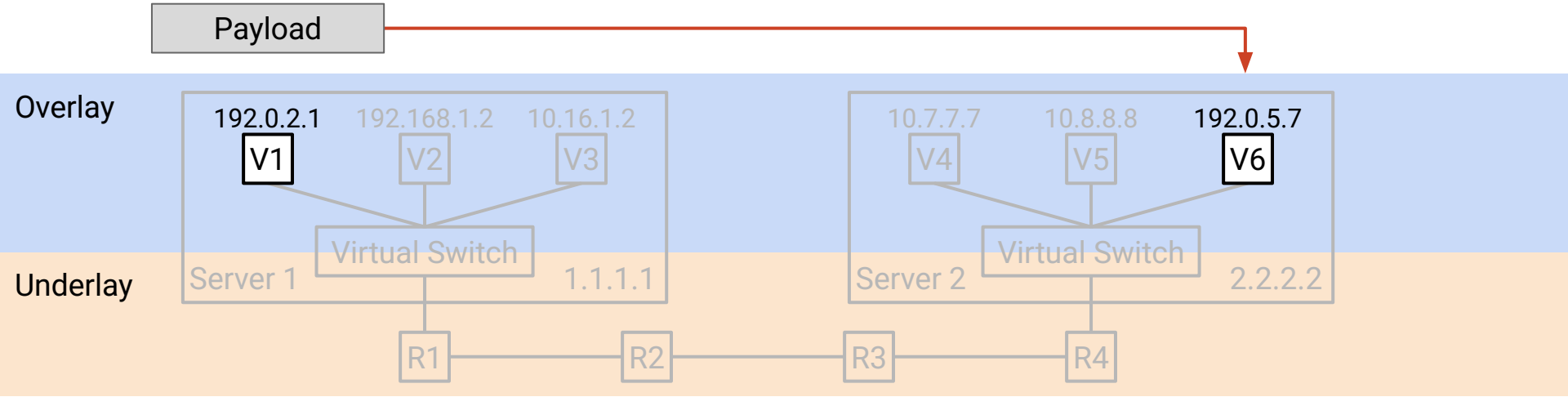
Encapsulation and Decapsulation

Let's see how to use the new layer to connect the overlay and underlay networks.



Encapsulation and Decapsulation

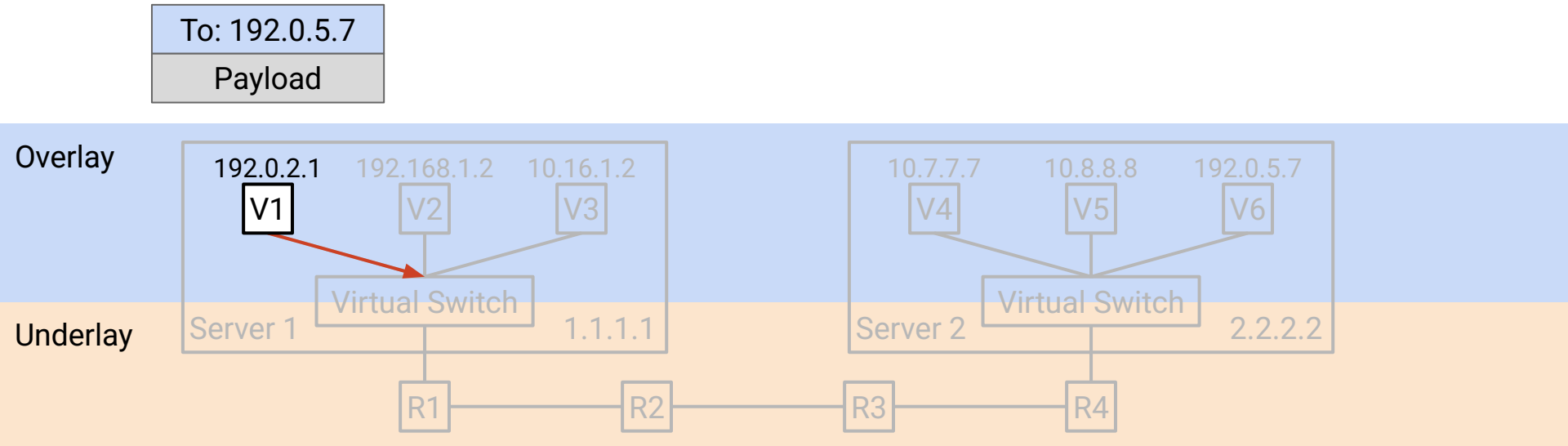
Our goal: VM1 wants to talk to VM6.



Encapsulation and Decapsulation (Step 1/5)

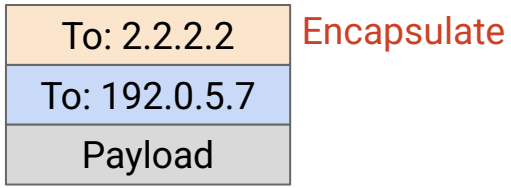
VM1 adds an overlay header with the destination's *virtual* address.

Then, VM1 passes the packet to the virtual switch.



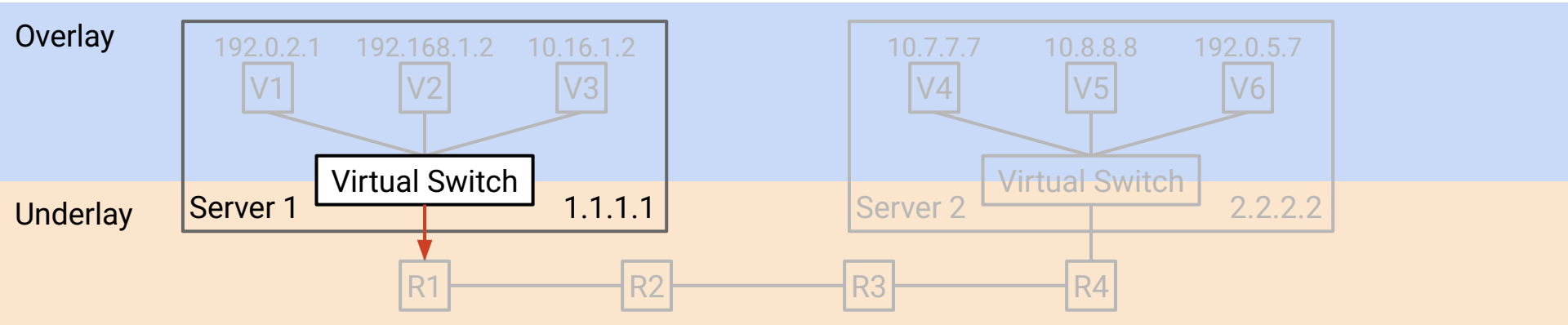
Encapsulation and Decapsulation (Step 2/5)

The virtual switch reads the virtual address and looks up the matching physical address. Then, it adds (*encapsulates*) a new header with the physical address.



Encapsulate

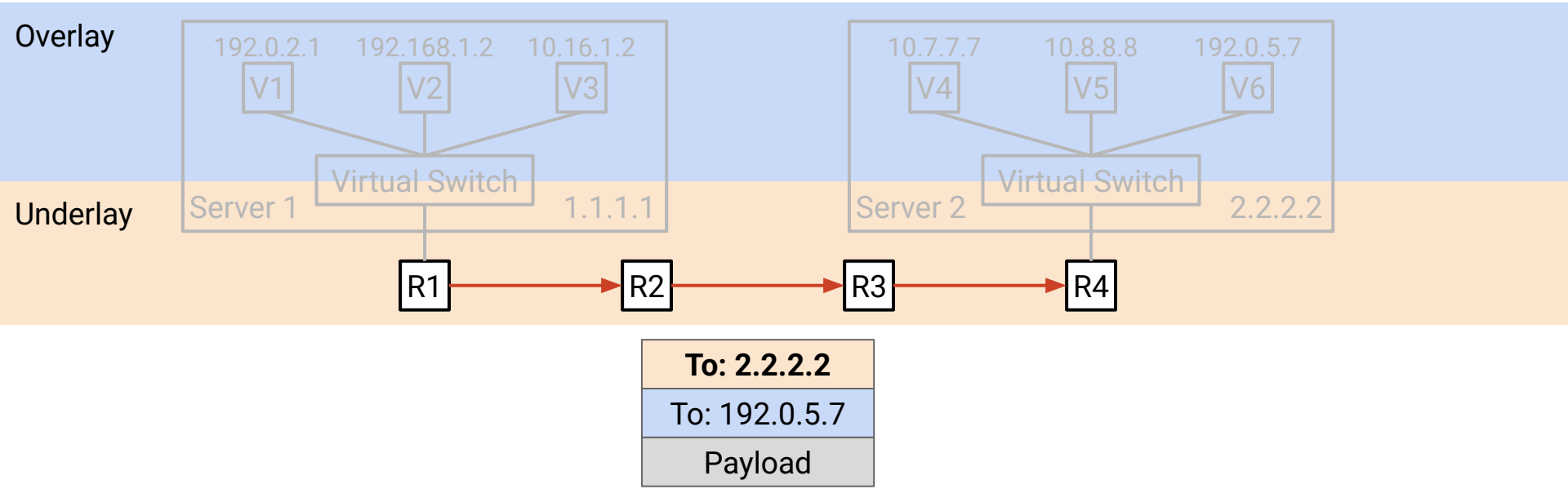
Haven't discussed how to look up yet. For now, it's magic.



Then, the virtual switch forwards the packet to routers in the datacenter.

Encapsulation and Decapsulation (Step 3/5)

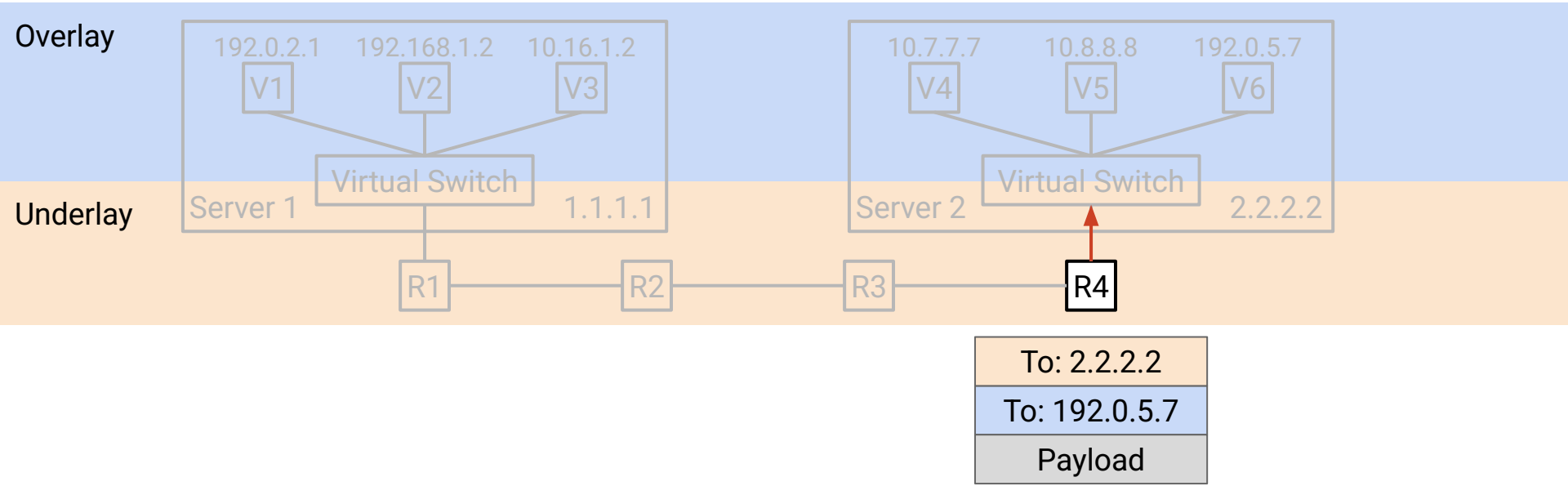
The routers in the datacenter forward the packet according to its *physical* (underlay) address. No need to think about virtual addresses!



Encapsulation and Decapsulation (Step 4/5)

Eventually, R4 receives the packet and reads its *physical* (underlay) destination address, 2.2.2.2.

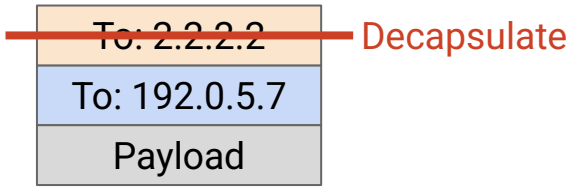
R4 is connected to physical server 2.2.2.2, so it forwards the packet to the server.

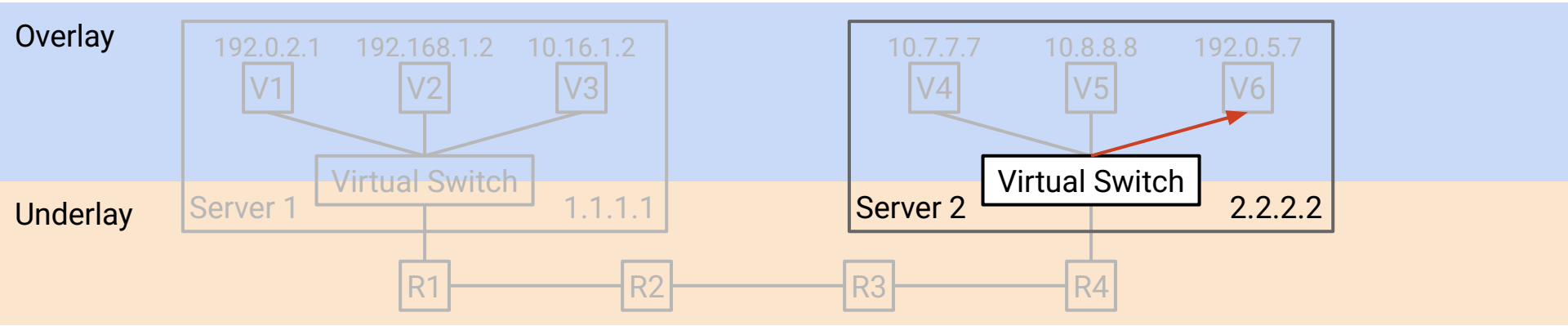


Encapsulation and Decapsulation (Step 5/5)

The virtual switch at 2.2.2.2 sees a packet destined for itself.

The virtual switch removes (*decapsulates*) the underlay header, revealing the virtual address of the destination.

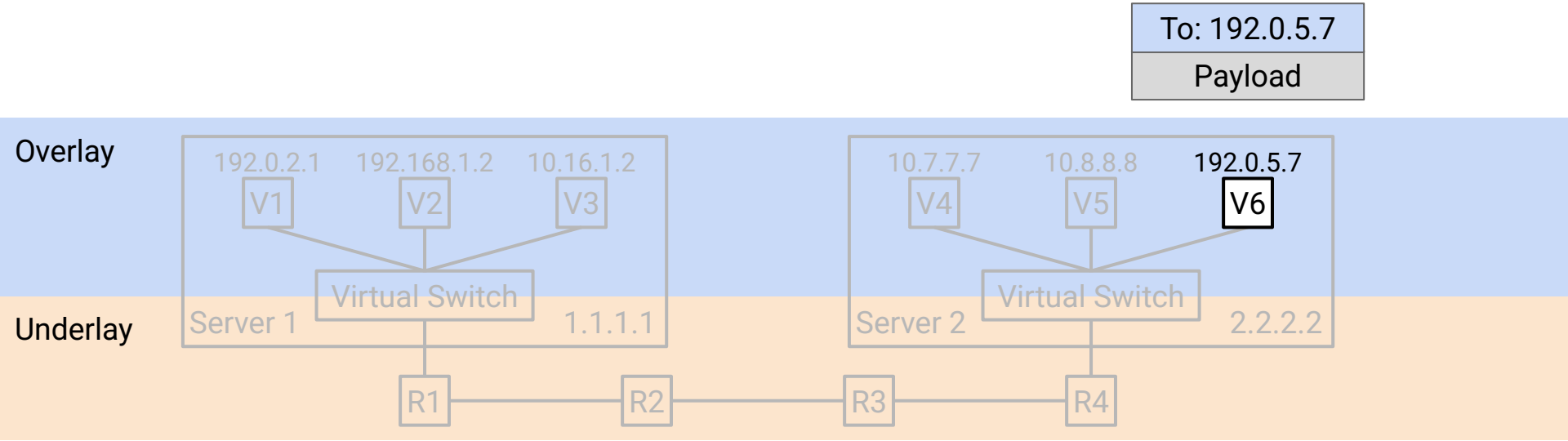




Then, the virtual switch sends the packet to the VM with virtual address 192.0.5.7.

Encapsulation and Decapsulation

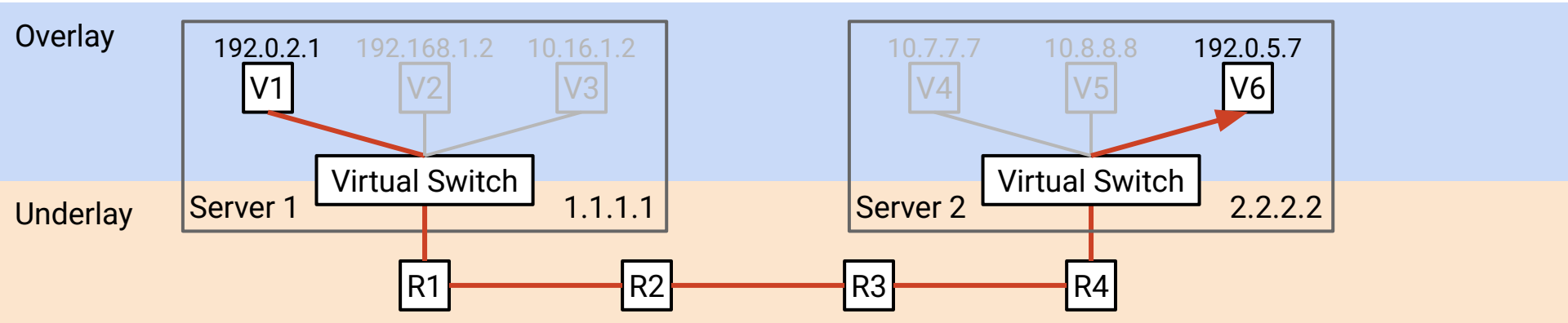
Success – our packet reached VM6!



Encapsulation and Decapsulation

Why did this work?

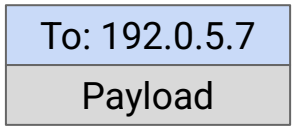
- The overlay network (VM1 and VM6) only thought about virtual addresses.
- The underlay network (R1–R4) only thought about physical addresses.
- The virtual switches acted as a bridge between the two layers.



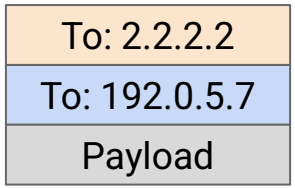
Encapsulation and Decapsulation

Encapsulation: Adding the extra header.

Decapsulation: Removing the extra header, exposing the original header underneath.

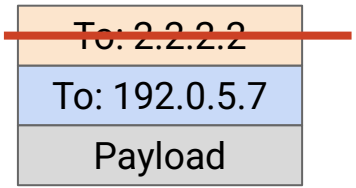


Original packet only has *virtual* (overlay) address.



We add the *physical* (underlay) address.

The extra header helps the packet travel through the underlay network.



Eventually, we remove the extra header.

The packet travels based on the *virtual* (overlay) address the rest of the way.

Encapsulation and Decapsulation

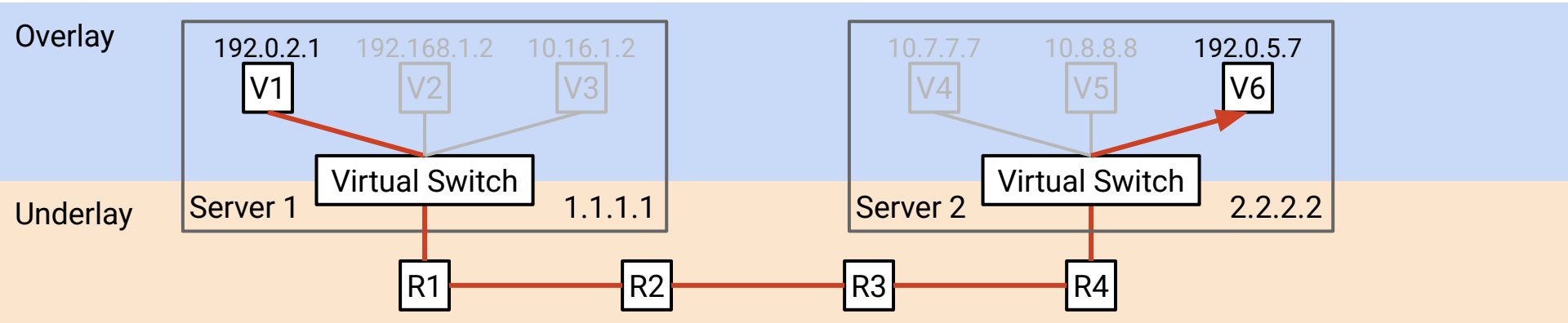
VM1's forwarding table	
To:	Next hop:
Anywhere	Virtual switch

Virtual switch's forwarding table	
To:	Next hop:
192.0.5.7	Add header: 2.2.2.2 Then, send to R1

Haven't discussed how to map 192.0.5.7 → 2.2.2.2 yet.
For now, it's magic.

R2's forwarding table	
To:	Next hop:
1.1.1.1	R1
2.2.2.2	R3

Only includes physical addresses (which can be aggregated!)



Multi-Tenancy and Private Networks

Lecture 19, CS 168, Spring 2026

Datacenter Routing

Datacenter Addressing

Virtualization and Encapsulation

- Virtualization
- Overlay and Underlay Networks
- **Multi-Tenancy and Private Networks**

Multi-Tenancy

Datacenters are owned by a single operator, but they can have multiple tenants.

Different companies/departments are running on the same infrastructure.

- Example: Google Maps, Gmail, YouTube can all run in a Google datacenter..
- Example: In cloud provider datacenters (e.g. AWS, GCP), customers can start up a VM, and destroy it when they're done.
- Allows for efficient use of resources.

(AWS = Amazon Web Services. GCP = Google Cloud Platform.)

Private IP Addressing

Different tenants don't coordinate when choosing addresses. Why is this a problem?

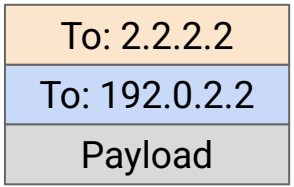
Two hosts (from two different tenants) could have the *same* IP address.

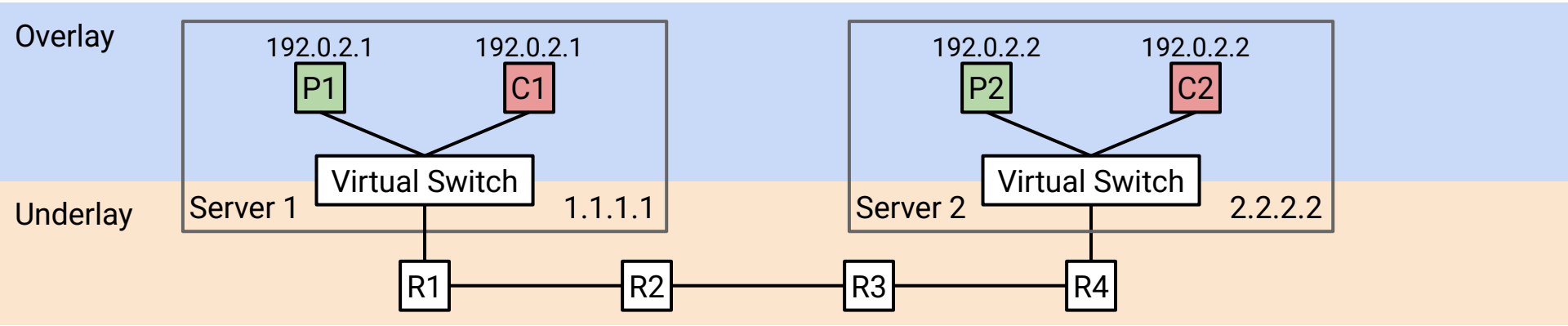
- Remember, we only have 2^{32} IPv4 addresses.
- Some hosts never want to be contacted from the Internet, so they don't need a unique address.
- Recall we can save addresses by having private addresses that can be used in multiple networks.
  - RFC 1918 defines specific ranges of IPs for private use.
  - Common examples: 192.168.0.0/16 and 10.0.0.0/8.

Routing with Multi-Tenancy

In this datacenter, Coke and Pepsi are two separate tenants.

- Both tenants have a VM with IP address 192.0.2.2.
- In this packet, it's ambiguous which 192.0.2.2 we're trying to contact.

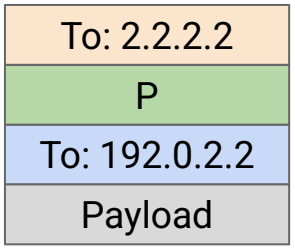


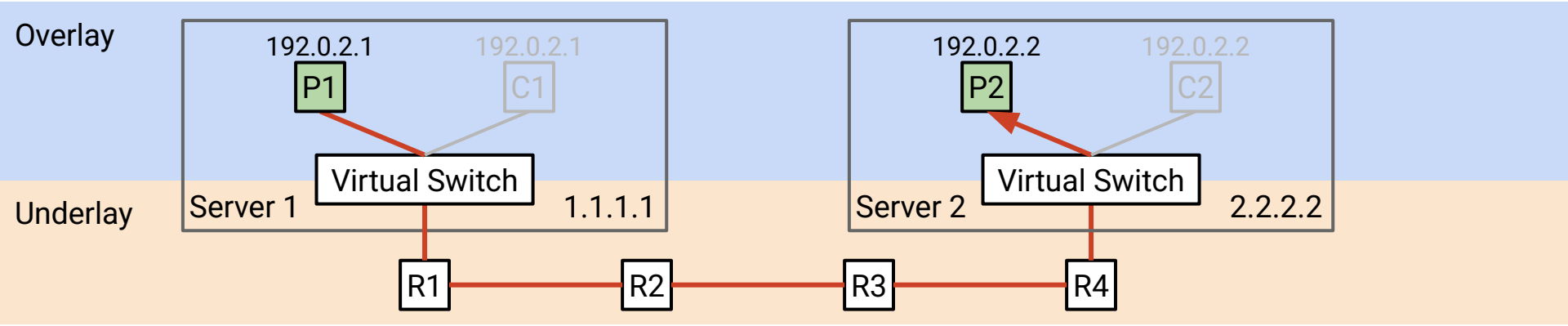


Encapsulations for Multi-Tenancy

Solution: Use encapsulation again!

- Add an extra header specifying which tenant this packet is meant for.

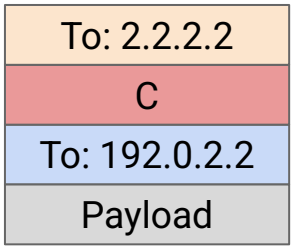


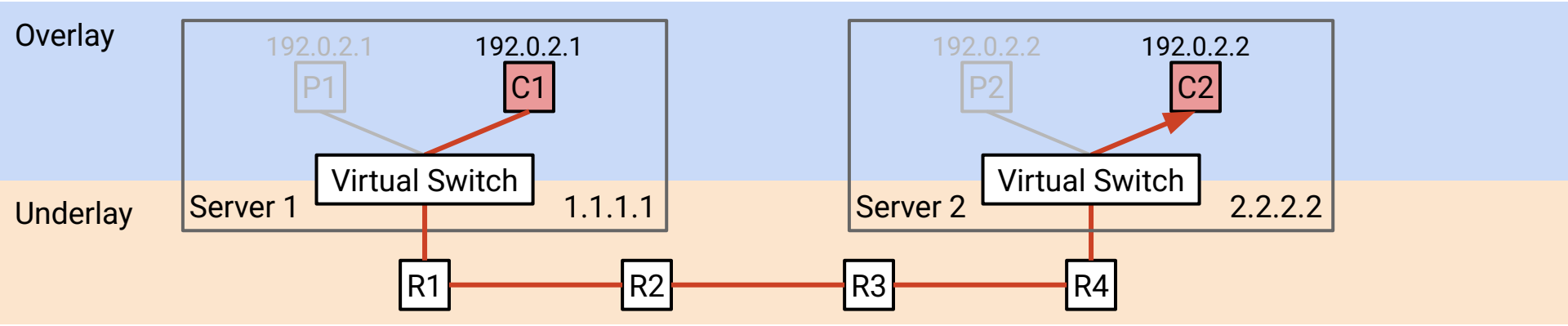


Encapsulations for Multi-Tenancy

Solution: Use encapsulation again!

- Add an extra header specifying which tenant this packet is meant for.

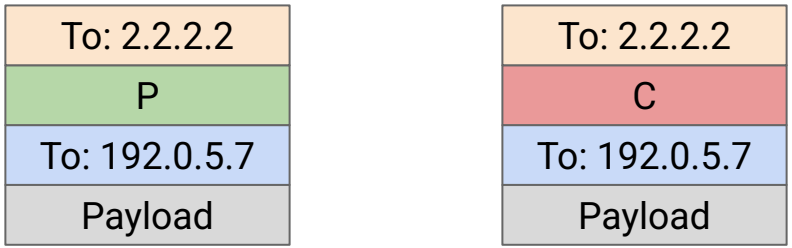




Encapsulations for Multi-Tenancy

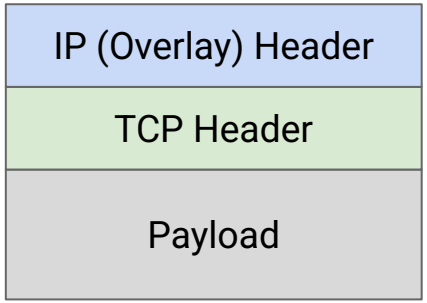
This new extra header lets us distinguish between tenants.

- Each tenant has a **virtual network ID** for its own VMs.
- The extra header includes the virtual network ID.
- The ID is not used for forwarding or routing, but provides more context to supplement the virtual IP address.

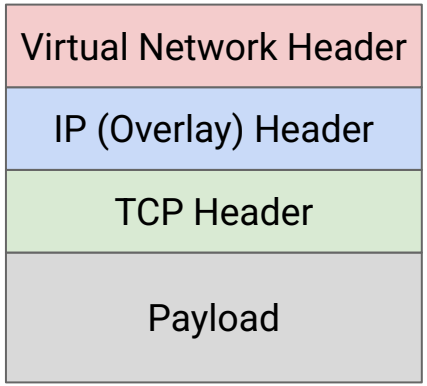


Putting It Together – Stacking Encapsulations

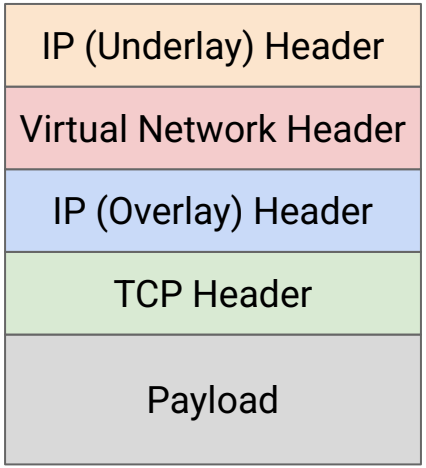
We can use encapsulation for both virtualization and multi-tenancy.



Original packet
from application.



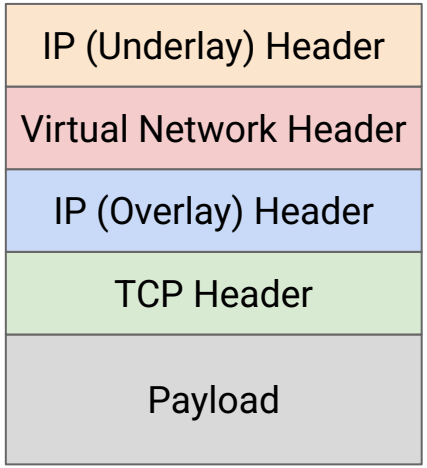
Encapsulate:
Add virtual network context.



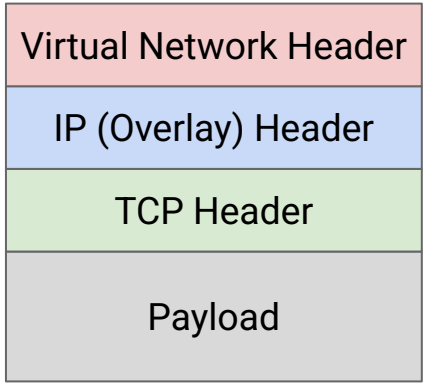
Encapsulate:
Add underlay destination.

Putting It Together – Stacking Encapsulations

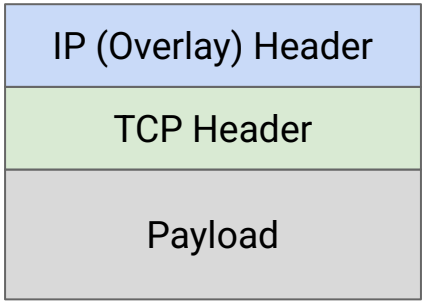
We can use encapsulation for both virtualization and multi-tenancy.



Receive packet from underlay.



Decapsulate:
Expose virtual network header, decide which tenant to forward to.



Decapsulate:
Expose overlay header, forward to corresponding VM.

Implementing Encapsulation

What real-world protocols exist for adding extra headers?

- IP-in-IP: Adds an extra IP header.
- MPLS: Adds a 20-bit label to identify a service.
  - Can be used to support multi-tenancy.
  - Can also be used at other layers.
- As datacenters have grown, more protocols have been introduced.
  - Most work over IP (IP is the "outer" packet header that the underlay looks at).
  - Examples: GRE, VXLAN, GENEVE...

Don't worry about the details – the idea of encapsulation is more important than the specific implementation.

Summary: Routing and Encapsulation in Datacenters

- Datacenter networks need different treatment in routing protocols.
  - Both distance-vector and link-state need to be modified.
- We can design addressing in our datacenters to improve scalability.
- As more dynamic, virtual servers arrived – routing could not scale!
- We use an overlay network to separate VM-to-VM networking from server-to-server.
  - This is achieved through encapsulating packets.
- Packet encapsulation also allows us to separate different tenants in a datacenter.