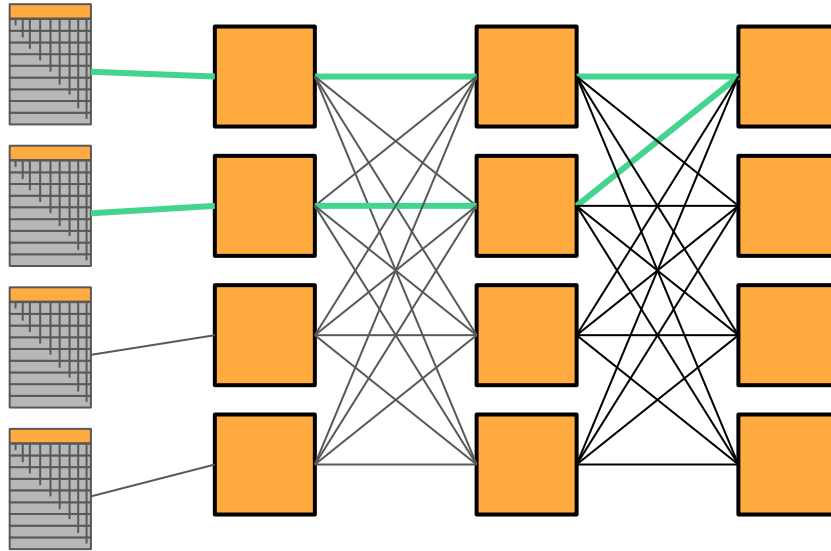# Routing in Datacenters

Autumn 2024
[cs168.io](cs168.io)

Rob Shakir

Recall

- Datacenter networks are single-organisation networks that are typically in a single location.

- Utilise meshy topologies (e.g., folded Clos) to be able to maximise bisection bandwidth.

Datacenter Topologies

Recall: previously, we assumed that a routing protocol will pick a
<u>single path</u> between some source and destination.
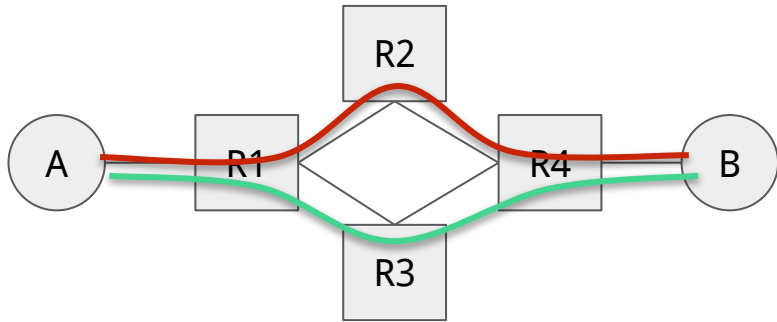
Datacenter Topologies
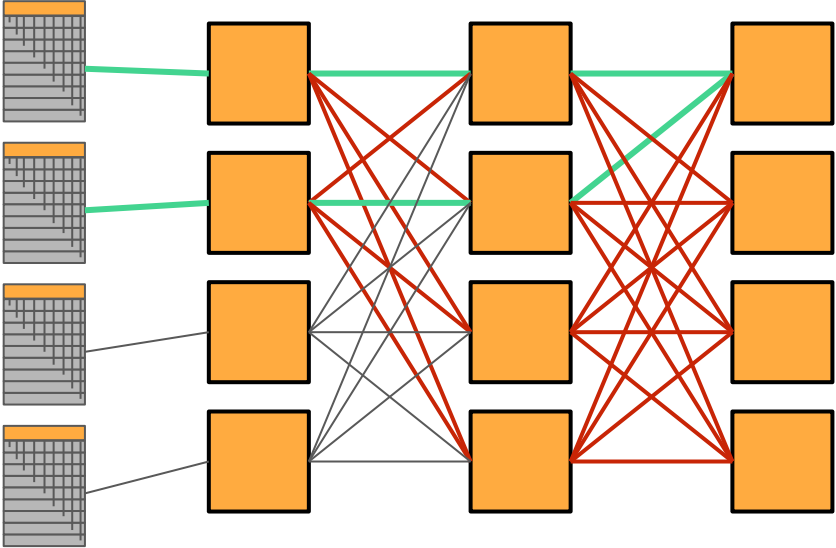
Picking only one path → not using all of the capacity available, and assumes coordinated selection of paths.

Equal Cost Multi-Path (ECMP)

- We, ideally, want to be able to use multiple paths between two sources.
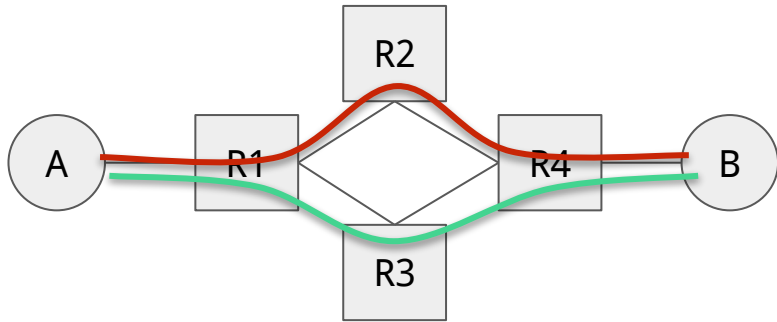- Why?

Equal Cost Multi-Path (ECMP)

- We, ideally, want to be able to use multiple paths between two hosts.
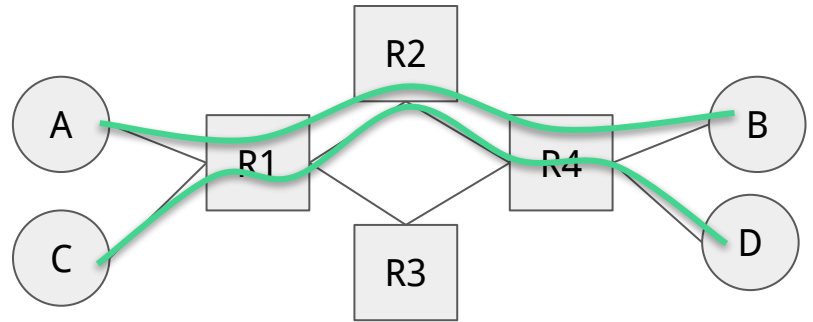
- Why? **Bandwidth.**

A → B bandwidth = link line rate
Red path unused.

Equal Cost Multi-Path (ECMP)

- We, ideally, want to be able to use multiple paths between two sources.
- Why? **Bandwidth.**

A → B bandwidth = link line rate
Red path unused.

A → B + C → D compete.
< line rate available for A → B

Equal Cost Multi-Path (ECMP)

- Recall: previously, we assumed that we pick a <u>single path</u> between some source a destination when routing.

- We need to be able to use multiple paths within the network topology *where they are equal cost.*

- Equal Cost Multi-Path.
    - Select all the paths that are the same cost.
    - Load-balance packets across these links.

Questions?

ECMP - Hashing

- How do we choose how to load balance packets across different forwarding paths?

- We need $f(packet) \rightarrow link.$
  - $f(packet)$ can run at each point that we need to load-balance traffic.

ECMP - Hashing

- What's a good f?

- Just round-robin - send each packet to a different link.
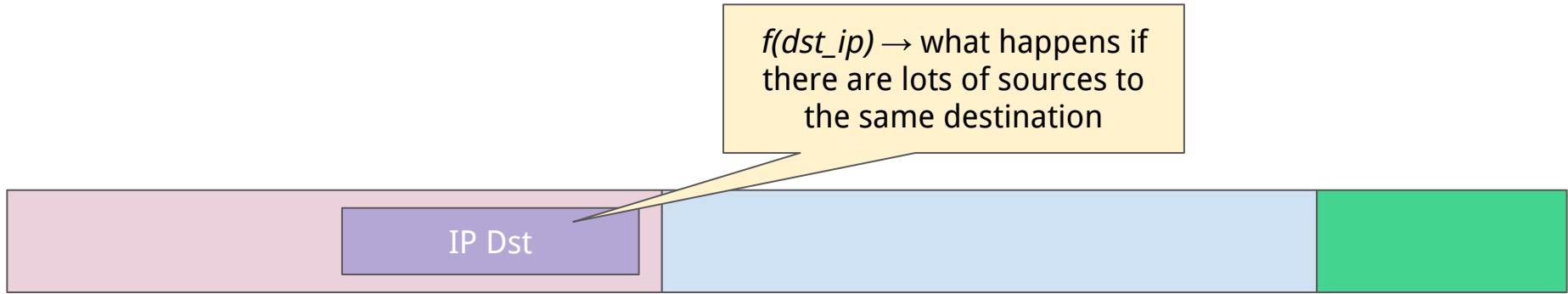
ECMP - Hashing

- What's a good f?

- Just round-robin - send each packet to a different link.

Ignores payload - does TCP care about re-ordering?

ECMP - Hashing

- What's a good f?

$f(dst\_ip) \rightarrow$ what happens if there are lots of sources to the same destination

IP Dst

ECMP - Hashing

- What's a good *f*?

f(src_ip) → what happens if there is one source to lots of destinations?

f(dst_ip) → what happens if there are lots of sources to the same destination
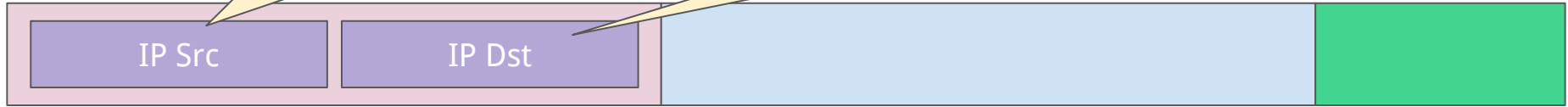
IP Src

IP Dst

ECMP - Hashing

- What's a good *f*?

f(src_ip) → what happens if there is one source to lots of destinations?

f(dst_ip) → what happens if there are lots of sources to the same destination

IP Src	IP Dst		

f(src_ip, dst_ip) → gives sufficient *entropy* for hashing.

ECMP - Hashing

- What's a good *f*?

f(src_ip) → what happens if there is one source to lots of destinations?

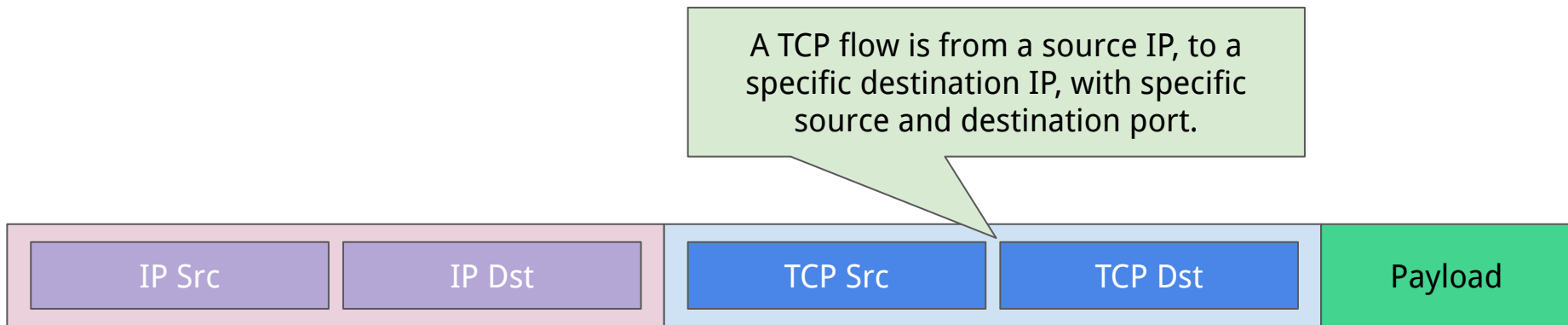*f(dst_ip)* → what happens if there are lots of sources to the same destination

IP Src	IP Dst

f(src_ip, dst_ip) → gives sufficient *entropy* for hashing.

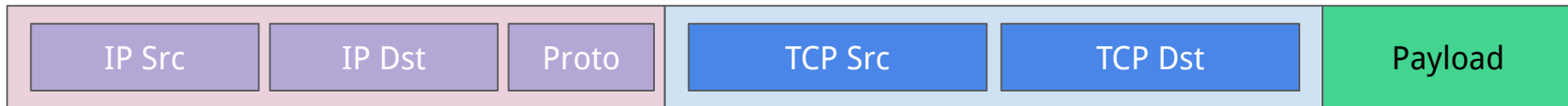What happens if there multiple elephant flows between the same <src,dst>?

ECMP - Hashing

- What's a good *f*?

A TCP flow is from a source IP, to a specific destination IP, with specific source and destination port.

IP Src	IP Dst	TCP Src	TCP Dst	Payload

f(src_ip, dst_ip, src_port, dst_port)

ECMP - Hashing

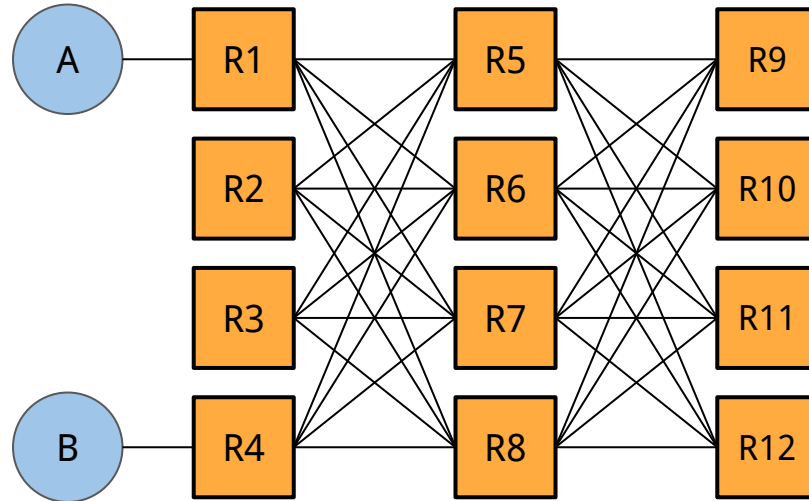| IP Src	IP Dst	Proto	TCP Src	TCP Dst	Payload

f(src_ip, dst_ip, proto, src_port, dst_port)

- This series of fields is referred to as a 5-tuple.

- Gives sufficient entropy for load-balancing whilst not resulting in re-ordering.

- We refer to this as *per-flow* load-balancing.
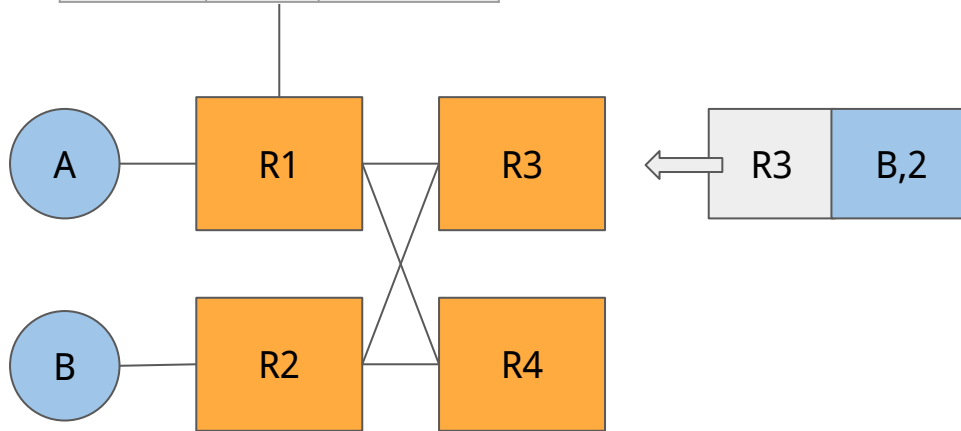
Questions?

Recall: Clos Topologies

Routing in Clos Topologies

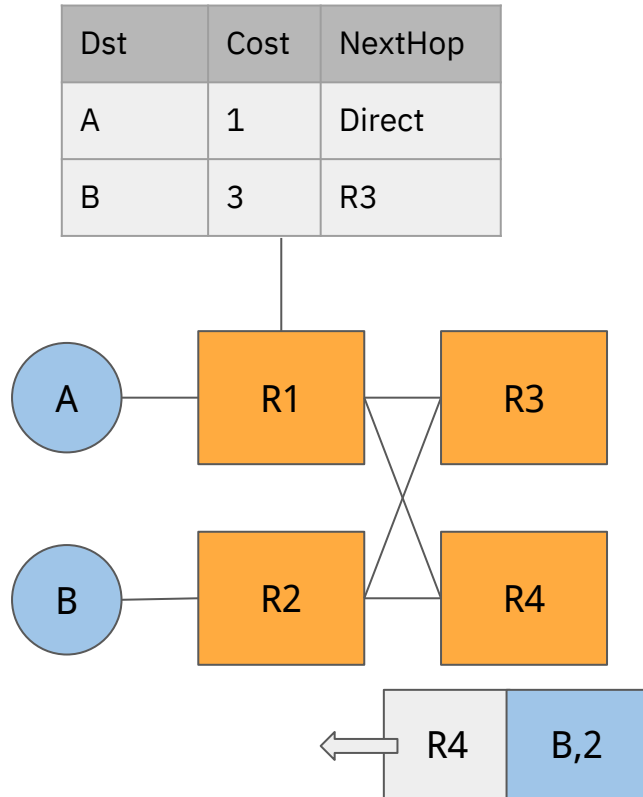- How do we adjust our routing protocols to work in Clos topologies?

Routing in Clos Topologies - DV

Dst	Cost	NextHop
A	1	Direct
B	**3**	**R3**

- **Distance-Vector**

Routing in Clos Topologies - DV

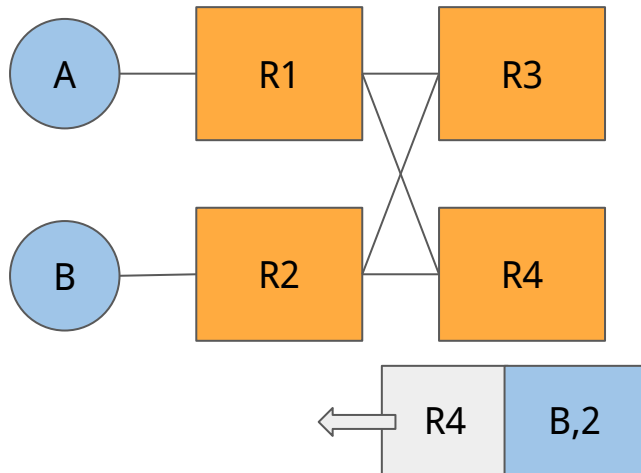| Dst	Cost	NextHop
A	1	Direct
B	3	R3

- **Distance-Vector** – R1 receives advertisement from R4 with cost of 2 to B (R4-R2, R2-B).

- Cost is *the same* as existing route B via R4.
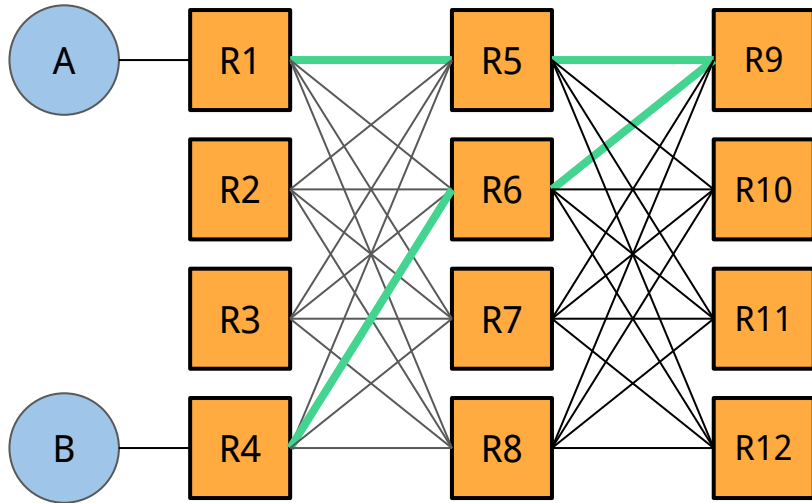
- Route is not accepted!

Routing in Clos Topologies - DV

| Dst	Cost	NextHop
A	1	Direct
B	**3**	**R4**
	3	R3

- Must extend routing table to store multiple next-hops per destination.

- Update logic – if the cost is the same, then add as an additional route.

- Forwarding can then have entries to allow for ECMP.

Routing in Clos Topologies - LS

- **Link State**

- Recall: each node stores the entire topology graph.

- Calculates shortest path to each destination through graph.

Routing in Clos Topologies - LS
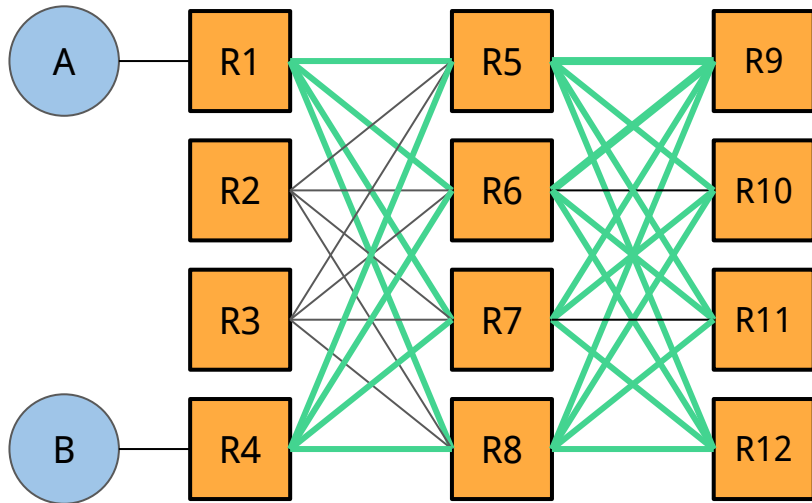
- Must extend protocol to calculate <u>all</u> shortest paths between source and destination.

- Multiple paths now programmed into the forwarding table.
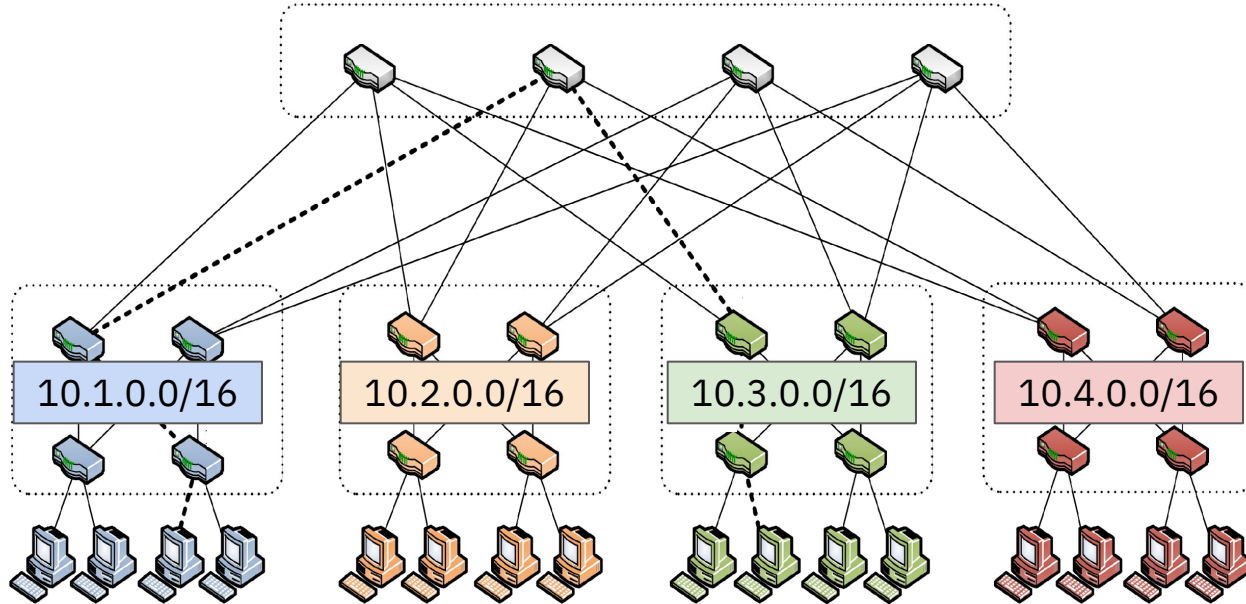
Questions?

Routing in Clos - Scaling.

- Distance-Vector:
  - Per-destination advertisements means 100,000+ destinations being advertised.

- Link-State:
  - Many link state advertisements with 10,000+ links.

- Memory, CPU and forwarding table resources are limited on commodity switches.
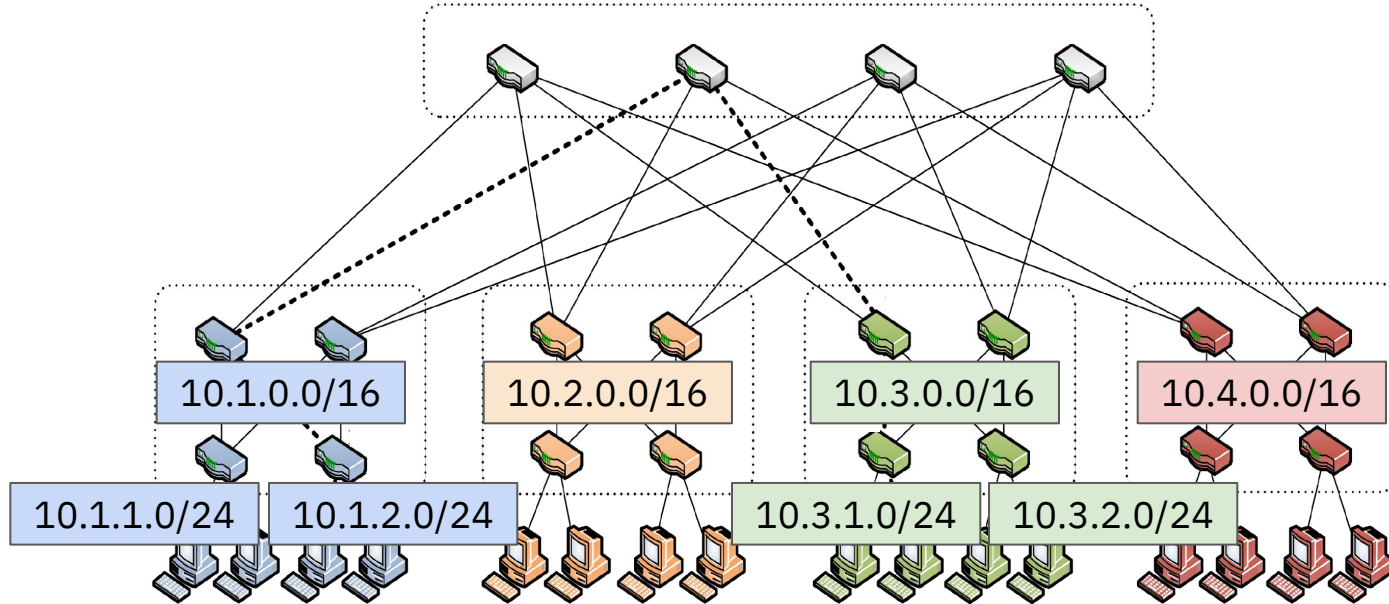
Topology-Aware Addressing and Routing

- Basic idea: the node (host) address tells us **where** in the topology it is.

- Exploits the idea that the topology is **regular**.
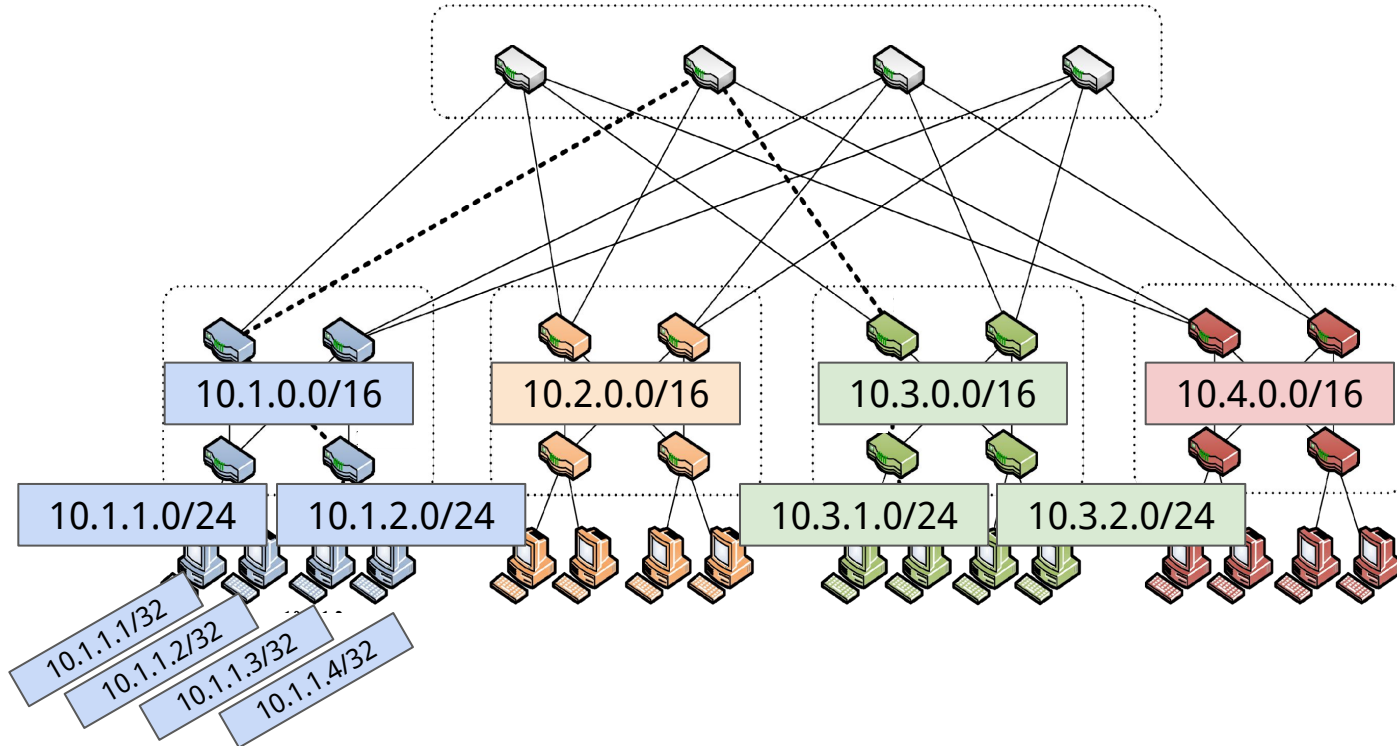
Topology-Aware Routing and Addressing

10.1.0.0/16   10.2.0.0/16   10.3.0.0/16   10.4.0.0/16

Topology-Aware Routing and Addressing

10.1.0.0/16 10.2.0.0/16 10.3.0.0/16 10.4.0.0/16

10.1.1.0/24 10.1.2.0/24 10.3.1.0/24 10.3.2.0/24

Topology-Aware Routing and Addressing

10.1.0.0/16

10.2.0.0/16

10.3.0.0/16

10.4.0.0/16

10.1.1.0/24

10.1.2.0/24

10.3.1.0/24

10.3.2.0/24

10.1.1.1/32

10.1.1.2/32

10.1.1.3/32

10.1.1.4/32

Topology-Aware Routing and Addressing

Dst	Port
10.1.0.0/16	1
10.2.0.0/16	2
10.3.0.0/16	3
10.4.0.0/16	4

10.1.0.0/16

10.2.0.0/16

10.3.0.0/16

10.4.0.0/16

10.1.1.0/24

10.1.2.0/24

10.3.1.0/24

10.3.2.0/24

10.1.1.1/32

10.1.1.2/32

10.1.1.3/32

10.1.1.4/32
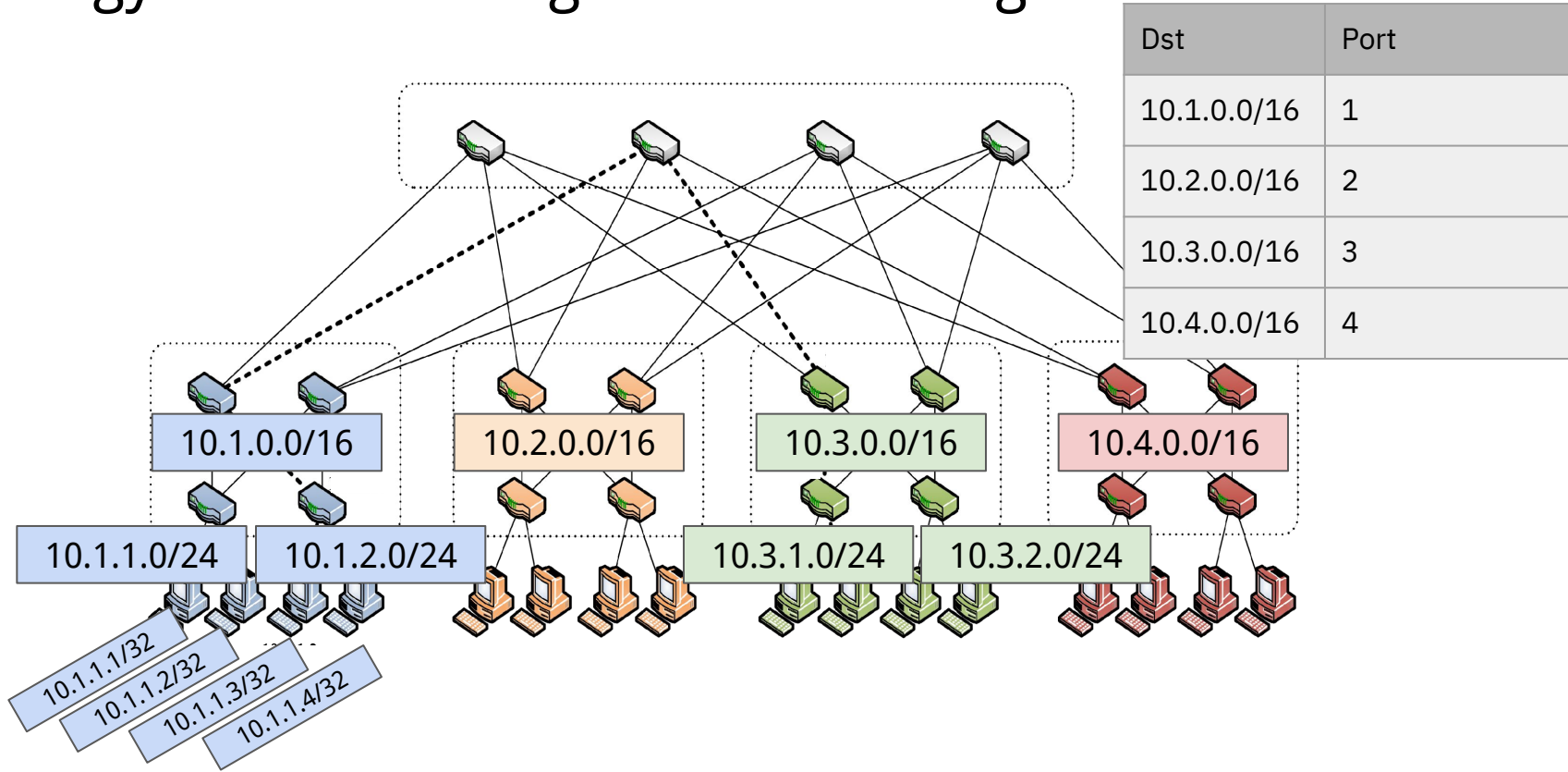
Topology-Aware Addressing and Routing

- Routing aggregation (summarisation) results in:
    - Fewer entries.
    - More stability.

- Updates are generally to address link or switch failures.
    - No need to advertise when a host goes away.

- Nice example of what can be achieved in a controlled network.
   - But… we can't deal with "random" addressing turning up!

Questions?

A challenge to our datacenter routing approach…

- Traditional datacenters required servers to be installed or uninstalled to change routing requirements.
  - Machine was either there or it wasn't!

- Virtualisation (virtual machines, containers) means that hosts can move often and quickly.

- Introduces new network control-plane scaling considerations.
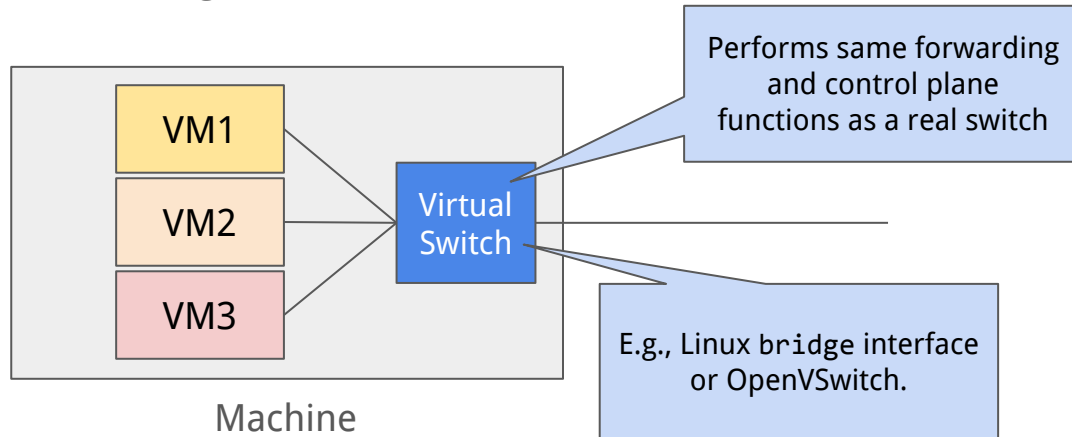 - More updates.
  - More calculations.

What is a virtual machine?

- A way of running a host inside another host.
- Allows for separation of environments whilst making efficient use of compute resources.
  - Allows for different administrators.
  - Allows for separation for security.
- Numerous hypervisors available.
  - e.g., VMWare, KVM.
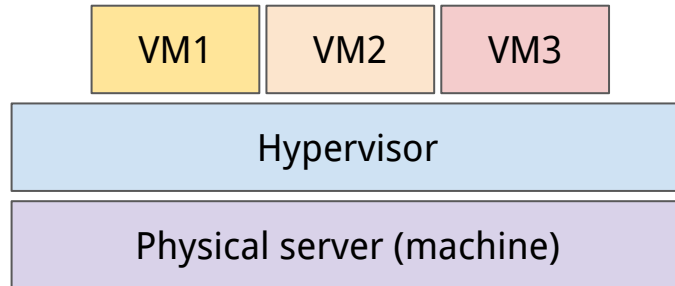
Virtual Switches

- Virtual machines need network connectivity.

- This means our routers/switches might have multiple hosts connected per port.

- We need our machines to have some form of switch.
 - A virtual switch running in software.

Performs same forwarding and control plane functions as a real switch

VM1

VM2

VM3

Virtual Switch

E.g., Linux `bridge` interface or OpenVSwitch.

Machine

Questions?

What do virtual machines mean for our DC?

- More hosts!

- Hosts that exist independently of the number of ports on our "leaf" switches.

- We need some way to refer to, and reason about, networking with this new virtual layer.

The Underlay Network

- The underlay network handles routing between physical machines.
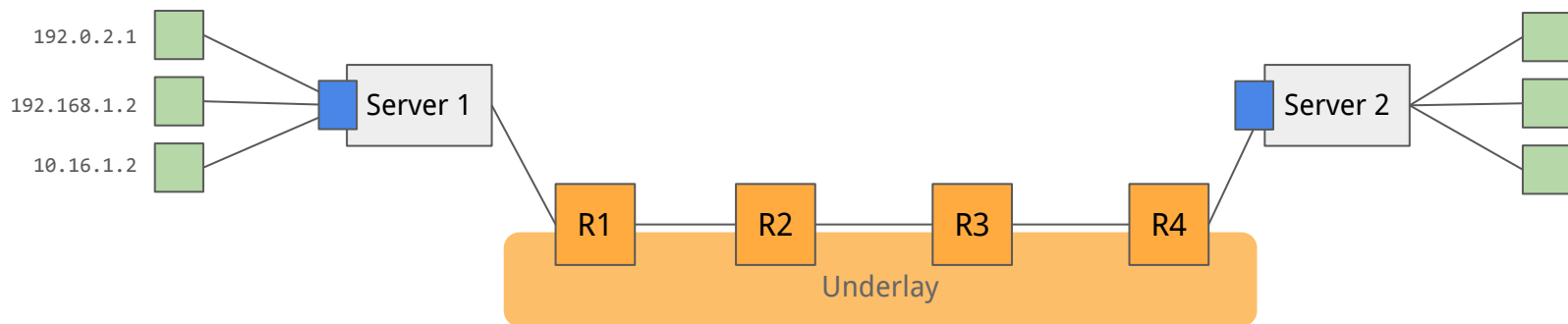 - We can exploit our addressing tricks to reduce scale.
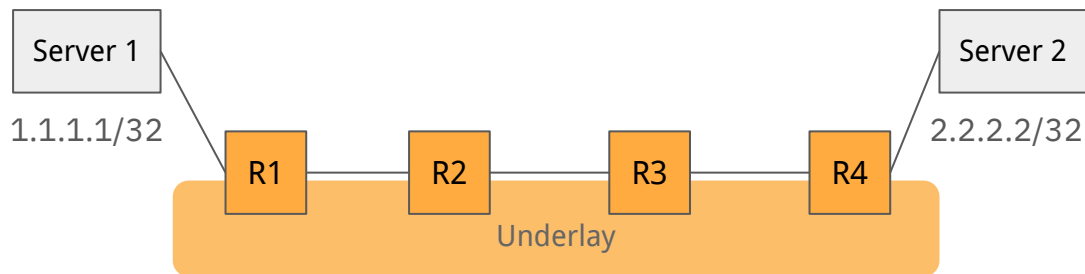
Dst	NextHop
2.0.0.0/8	R2
1.0.0.0/8	Direct

Dst	NextHop
2.0.0.0/8	Direct
1.0.0.0/8	R3

Impact of Virtual Machines

- We now have new hosts that turn up into our network rapidly!
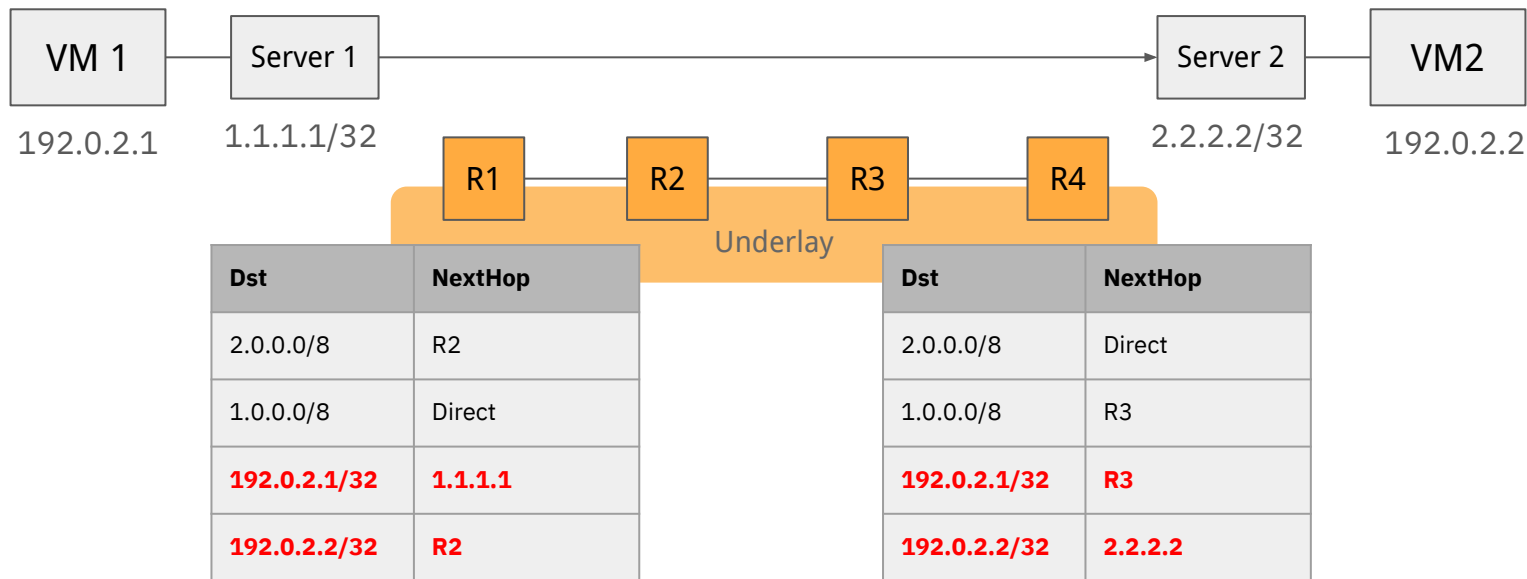- They may not be constrained to our "scalable" addressing scheme.

192.0.2.1
192.168.1.2
10.16.1.2

Server 1

R1   R2   R3   R4

Underlay

Server 2

Dst	NextHop
2.0.0.0/8	R2
1.0.0.0/8	Direct

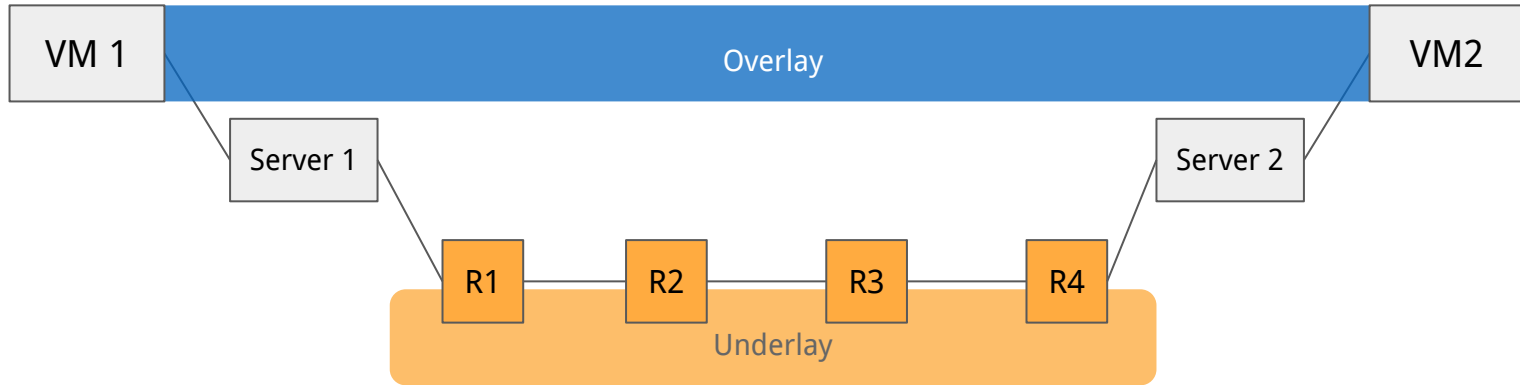| Dst	NextHop
2.0.0.0/8	Direct
1.0.0.0/8	R3

Scaling Concern

- Each virtual machine is one /32.
  - We can't necessarily guarantee that we can aggregate.

- Aggregation was our scaling trick.

| VM 1 | | Server 1 | | | | | | Server 2 | | VM2 |

| 192.0.2.1 | 1.1.1.1/32 | | | | | | 2.2.2.2/32 | 192.0.2.2 |

Underlay

Dst	NextHop
2.0.0.0/8	R2
1.0.0.0/8	Direct
192.0.2.1/32	**1.1.1.1**
192.0.2.2/32	**R2**

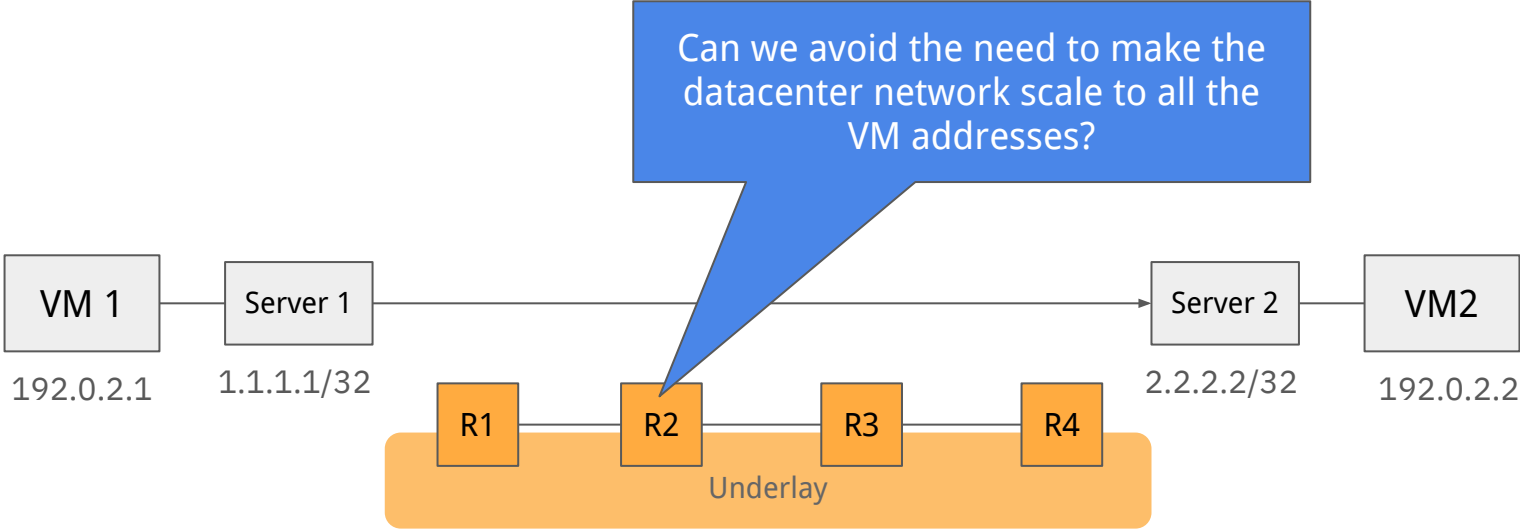| Dst	NextHop
2.0.0.0/8	Direct
1.0.0.0/8	R3
192.0.2.1/32	**R3**
192.0.2.2/32	**2.2.2.2**

Overlay and Underlay Networks

Concept: <u>overlay</u> network – a network topology that exists on top of the physical topology.

How can we scale and support overlay networks?

Questions?
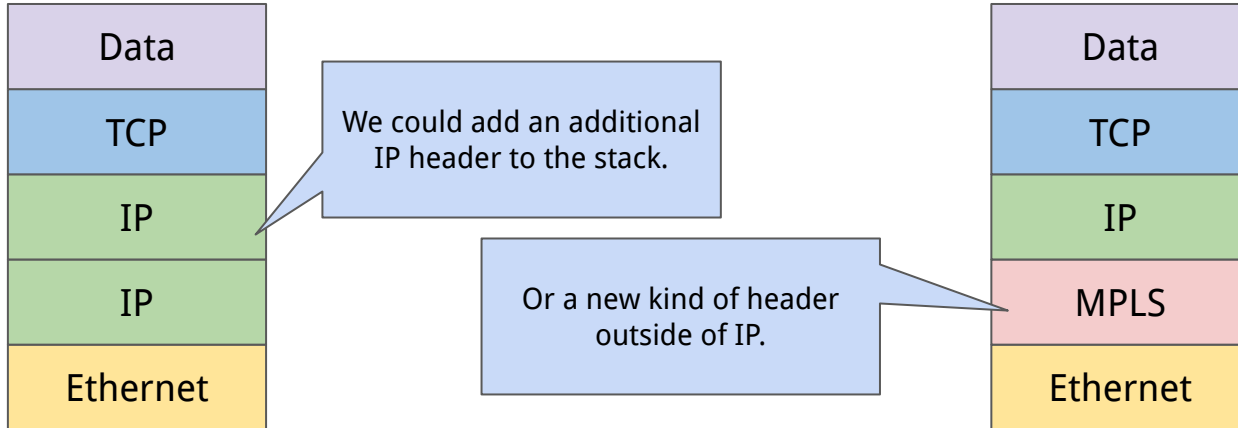
How can we scale and support overlay networks?

- VM1 → VM2 is a flow that is from 192.0.2.1 to 192.0.2.2.

- With destination based forwarding – we only look at the IP destination.

- Problem: The underlay network needs to understand every address in the overlay network.
  - Defining overlay networks didn't help us scale!

How can we scale and support overlay networks?

- VM1 → VM2 is a flow that is from 192.0.2.1 to 192.0.2.2.

- With destination based forwarding – we only look at the IP destination.

- Problem: The underlay network needs to understand every address in the overlay network.
  - Defining overlay networks didn't help us scale!

- **Is there some way we can avoid needing to think about the VM address and rather just care about the physical machine?**
 - We'd like to keep *destination-based forwarding*.

Encapsulation

- Thus far, we have just used one header per layer.
  - One L2 (Ethernet), one L3 (IP), one L4 (TCP).

- What happens if we add additional headers into the stack?

Data
TCP
IP
IP
Ethernet

We could add an additional IP header to the stack.

Or a new kind of header outside of IP.

Data
TCP
IP
MPLS
Ethernet

What does encapsulation allow us to do?

- Tell part of the network to forward based on one destination.

- At some later point in the network - remove the "outer" encapsulation, and let the network forward based on the "inner" encapsulation.

- This allows us to hide some complexity from the underlay network.
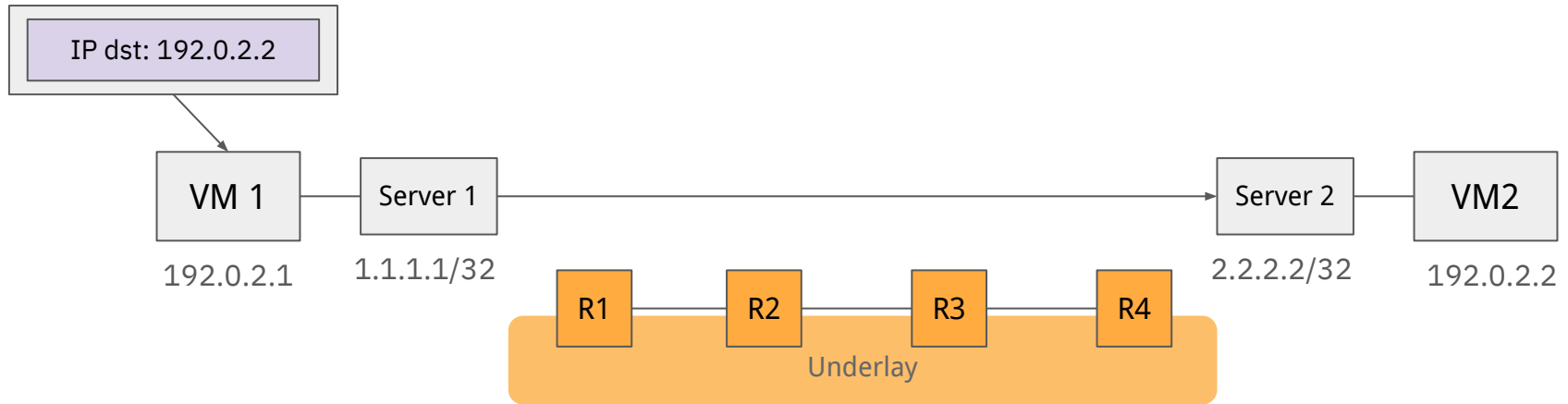
Encapsulation Actions
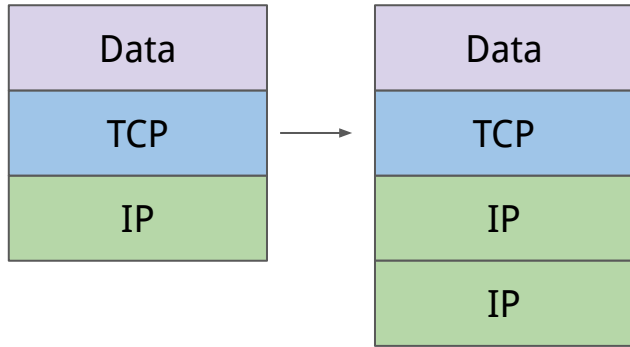
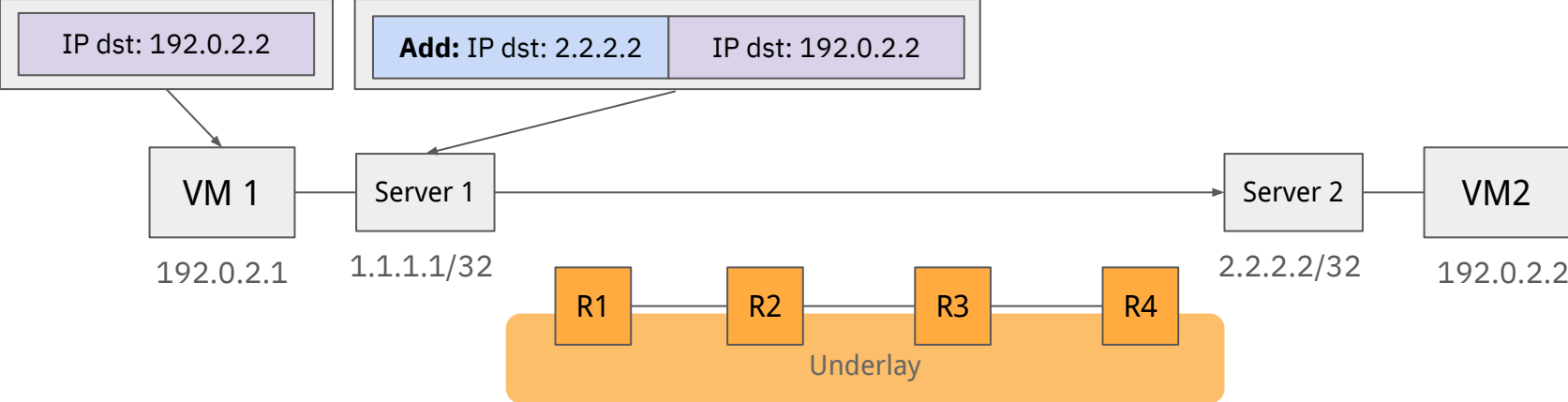**Encapsulate** packet - add another header to the original headers.
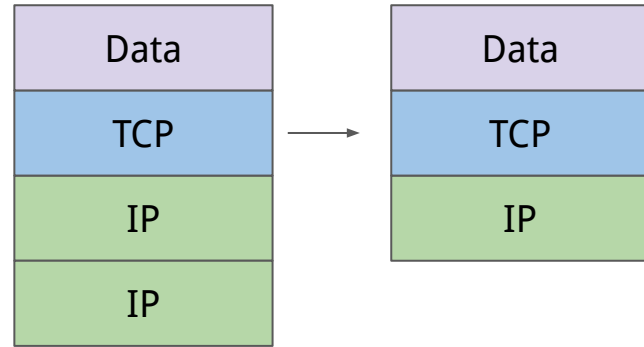
Decapsulate packet - remove a header to expose the header underneath.
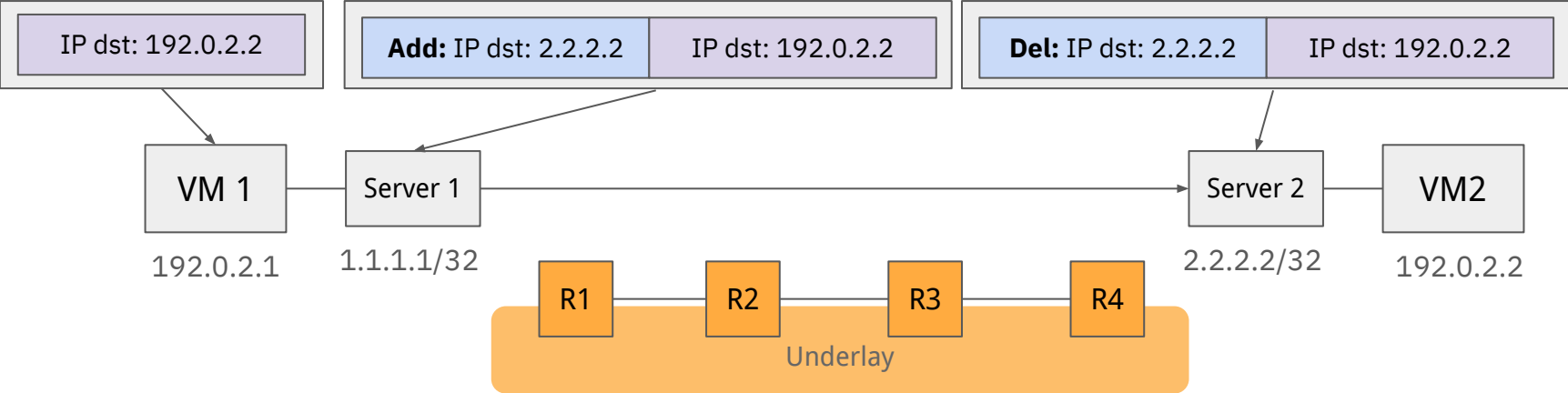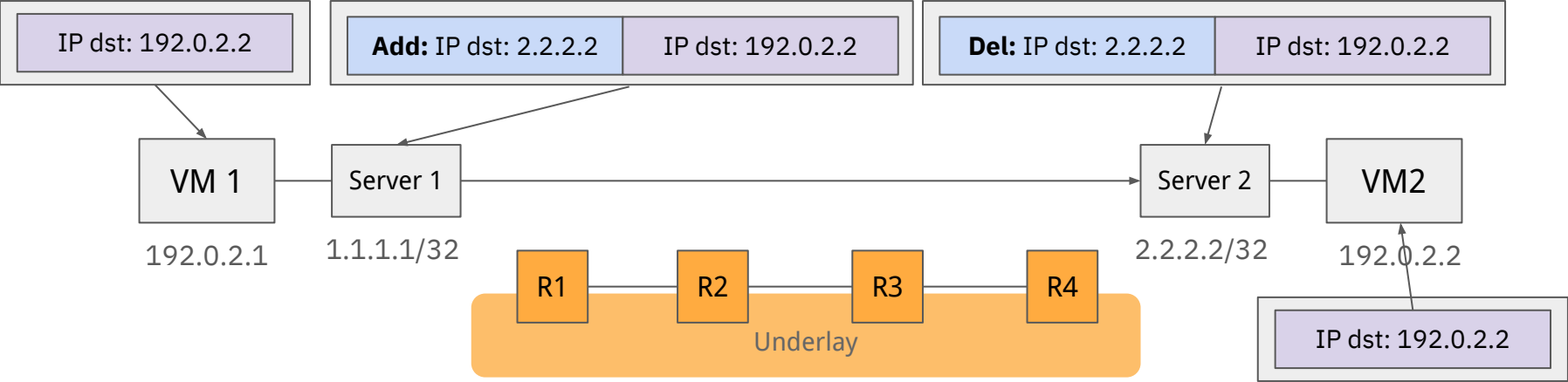
Encapsulating Packets

Using an *encapsulation* allows us to hide the "inner" overlay packet from the underlay.

Encapsulating Packets

Using an *encapsulation* allows us to hide the "inner" overlay packet from the underlay.

Encapsulating Packets

Using an *encapsulation* allows us to hide the "inner" overlay packet from the underlay.
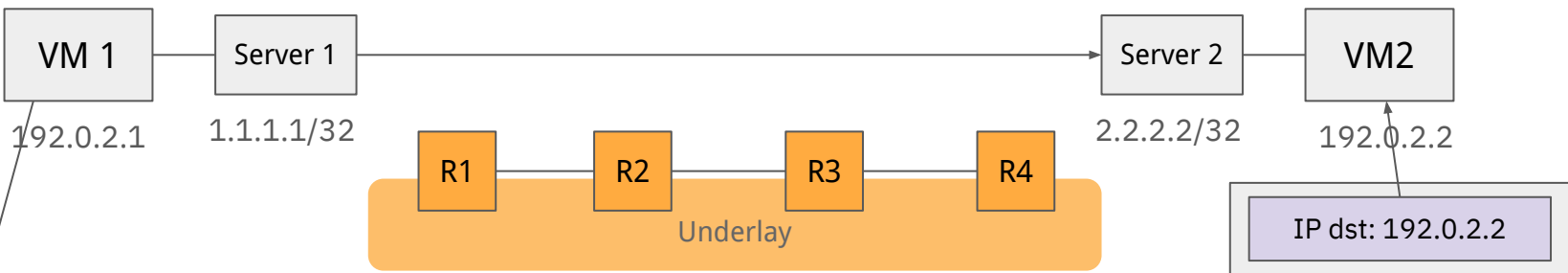
Encapsulating Packets

Using an *encapsulation* allows us to hide the "inner" overlay packet from the underlay.
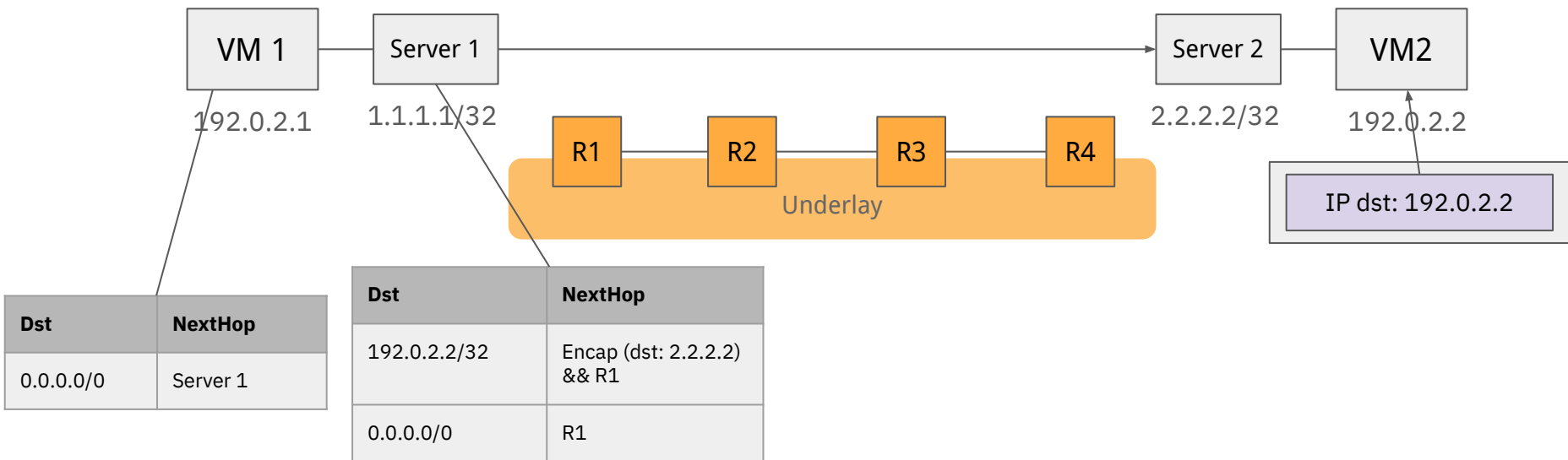
Scaling routing with encapsulation

- Encapsulation allows us to separate routing in the underlay and overlay.
- Underlay:
  - Knows only about reachability to physical machines in the datacenter (same scaling as before).
- Overlay:
  - Knows about routing to each virtual machine <u>that it needs to forward traffic to.</u>

Scaling routing with encapsulation

- Encapsulation allows us to separate routing in the underlay and overlay.
- Underlay:
  - Knows only about reachability to physical machines in the datacenter (same scaling as before).
- Overlay:
  - Knows about routing to each virtual machine that it needs to forward traffic to.
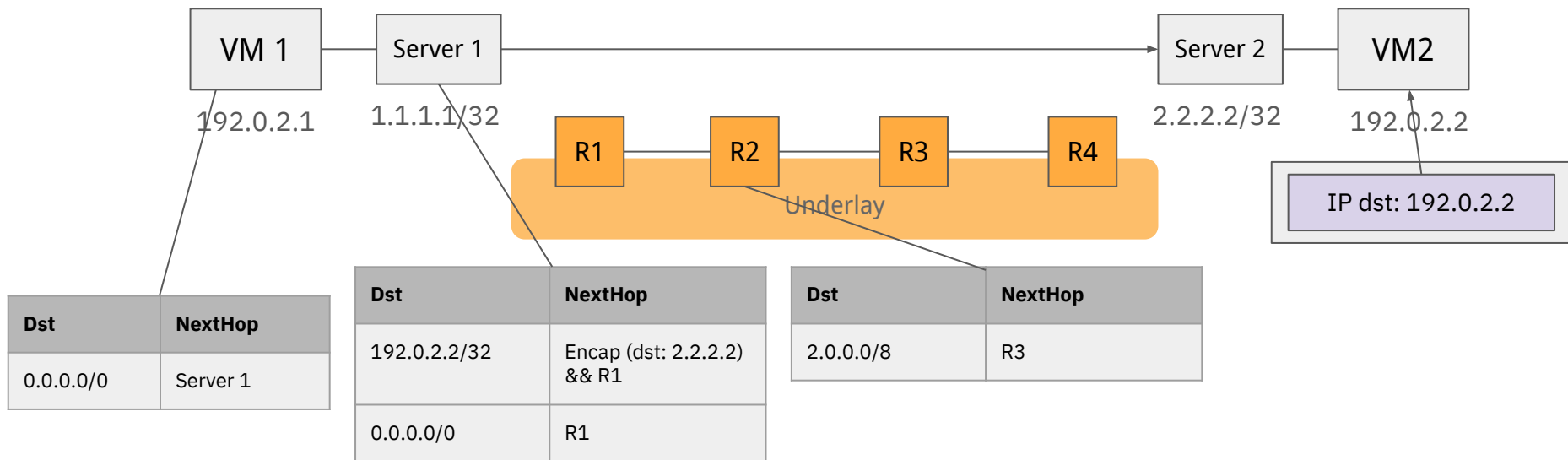
Scaling routing with encapsulation

- Encapsulation allows us to separate routing in the underlay and overlay.
- Underlay:
  - Knows only about reachability to physical machines in the datacenter (same scaling as before).
- Overlay:
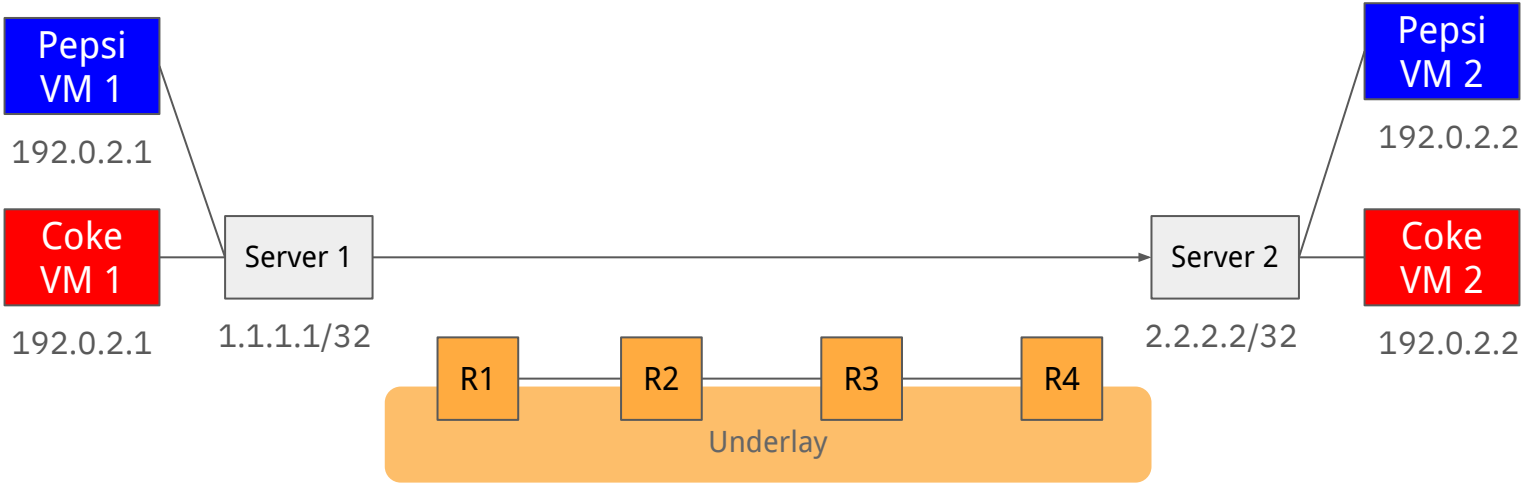 - Knows about routing to each virtual machine that it needs to forward traffic to.

Questions?

Multi-tenancy

- Datacenter networks are single owner, but multi-tenant.

- This means that there can be different companies/departments running on the same infrastructure.
  - Again, allows for efficient use of resources.

- Large case: Cloud provider.
  - Many customers with their virtual machines running inside a single datacenter.
  - e.g., Azure, AWS, GCP.

Multi-tenancy in a Datacenter

Pepsi VM 1
192.0.2.1

Coke VM 1
192.0.2.1

Server 1
1.1.1.1/32

R1 — R2 — R3 — R4
Underlay

Server 2
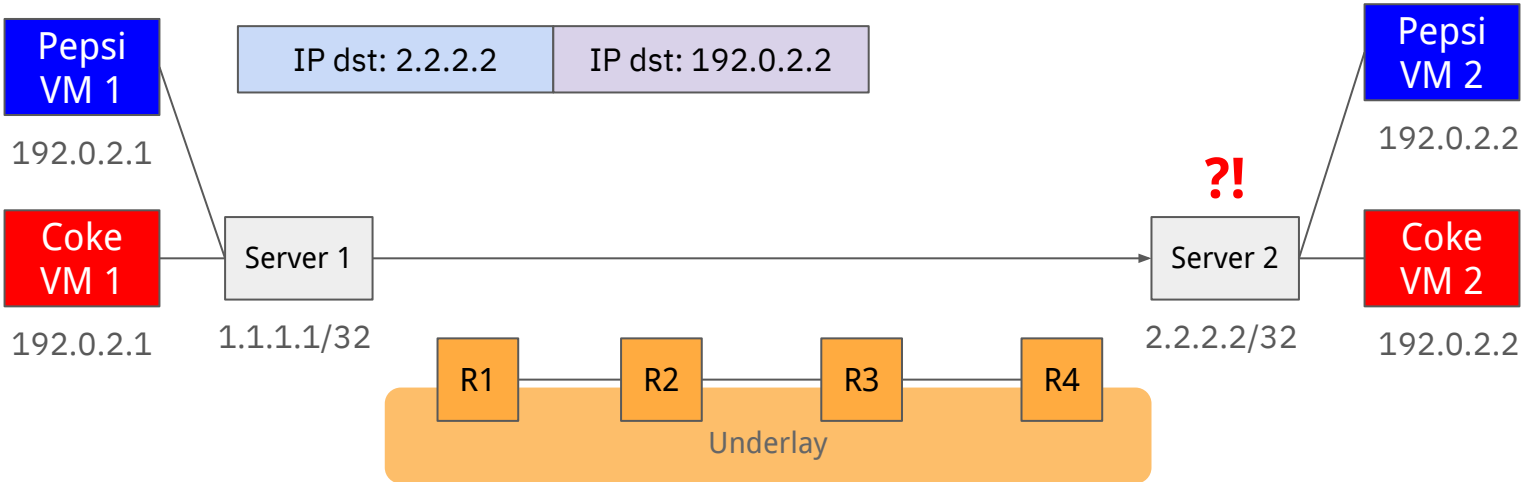2.2.2.2/32

Pepsi VM 2
192.0.2.2

Coke VM 2
192.0.2.2

Our datacenter networks need to support multiple tenant networks - who don't coordinate with each other.
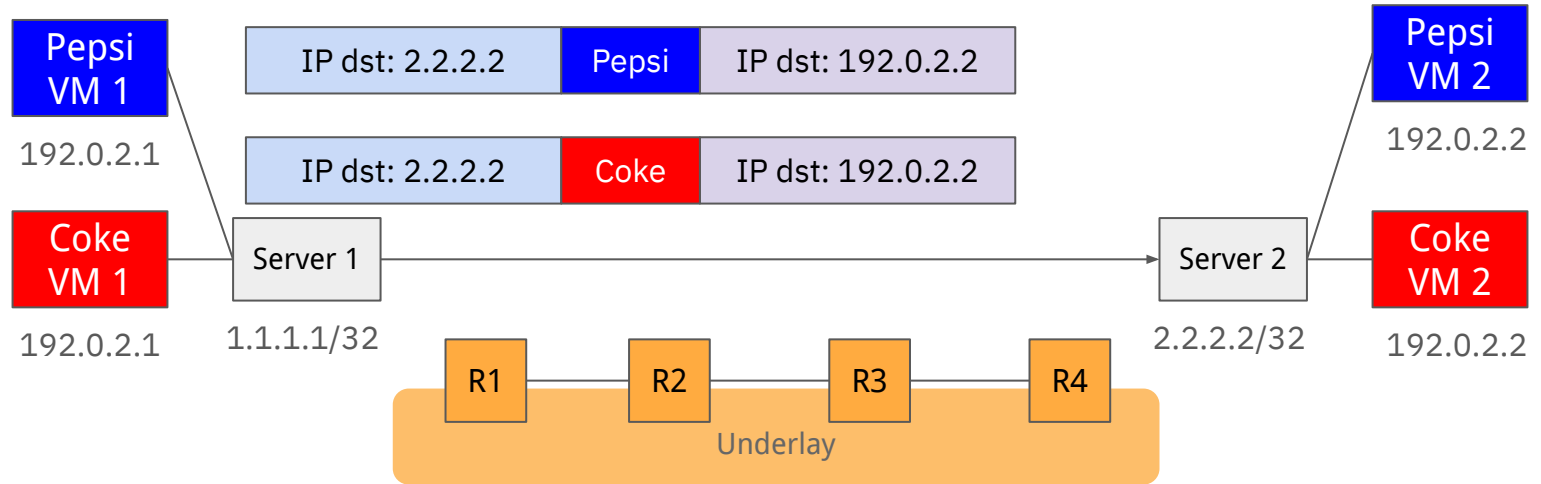
Multi-tenancy in a Datacenter

Our datacenter networks need to support multiple tenant networks - who don't coordinate with each other.
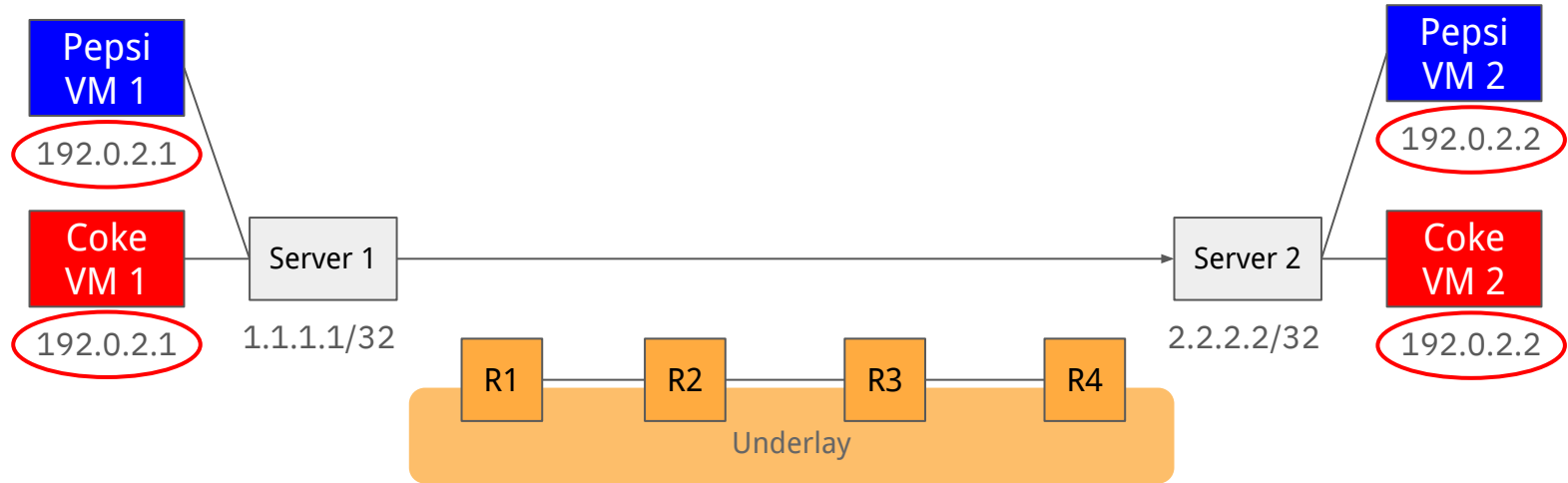
Multi-tenancy in a Datacenter

Our encapsulation just told us <u>where</u> to go - not <u>who</u> the packet was for.

Multi-tenancy in a Datacenter

Pepsi VM 1

192.0.2.1

Coke VM 1

192.0.2.1

| IP dst: 2.2.2.2 | Pepsi | IP dst: 192.0.2.2 |

| IP dst: 2.2.2.2 | Coke | IP dst: 192.0.2.2 |

Server 1

1.1.1.1/32

Server 2

2.2.2.2/32

Pepsi VM 2

192.0.2.2

Coke VM 2

192.0.2.2

R1 — R2 — R3 — R4

Underlay

Using encapsulations that allow specification of a *virtual network ID* allows multi-tenancy.
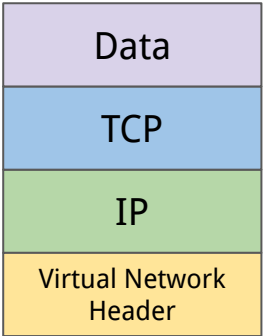
Encapsulations for Multi-Tenancy

- Carries information that isn't directly forwarding information in the packet header.

- Allows for a packet to be resolved into a specific *virtual network*.

- Each virtual network is assigned a *virtual network ID.*
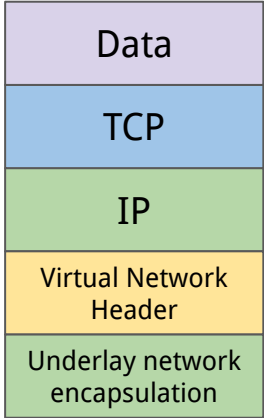
Putting it together - stacking encapsulations

Encap

Data
TCP
IP
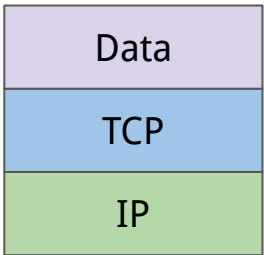
Add *virtual network* context →

Data
TCP
IP
Virtual Network Header

Add *underlay* destination →

Data
TCP
IP
Virtual Network Header
Underlay network encapsulation

Putting it together - stacking encapsulations
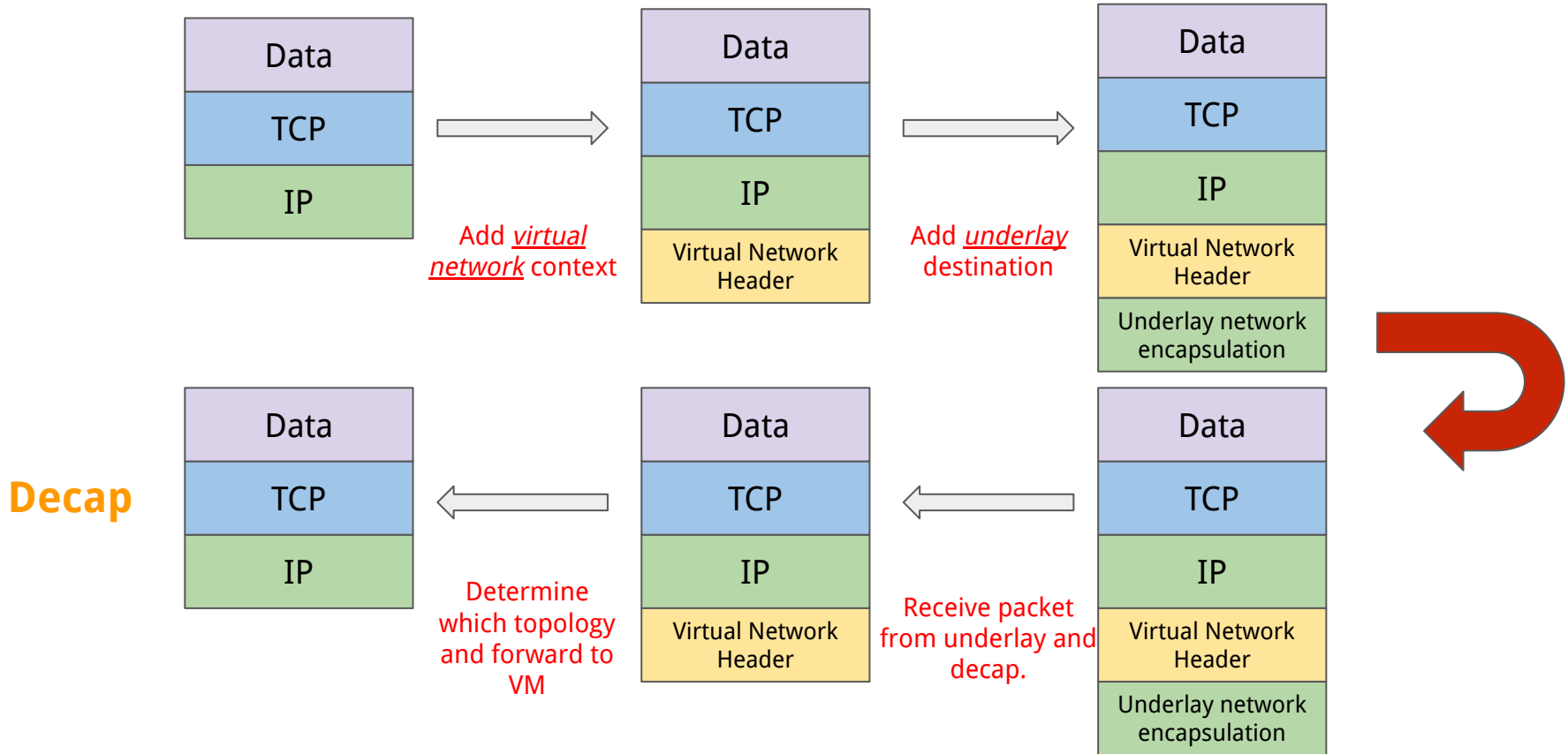
What headers do we use?

- Traditional encapsulation for multi-tenancy in telco networks has been MPLS.
  - A specific 20-bit label that identifies a <u>service</u>.
  - Historically not encapsulated in IP (but can be!)

- Many old and new(er) options as datacenters like this have grown.
  - Most work over IP (i.e., IP is the "outer" packet header that the underlay looks at).
  - e.g., GRE, VXLAN, GENEVE…

- Don't worry about the details – mainly just understand that the idea of <u>encapsulation</u> exists.

Questions?

Summary

- Datacenter networks need different treatment in routing protocols.
  - Both Link State and Distance Vector.

- 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.