(Network) Load Balancing Building Blocks

facebook
INFRASTRUCTURE

Kyle Lexmond
Traffic Applications, Seattle

SRECon EMEA 2019 – Oct 3rd 2019

Twitter: @lightweavr

Traffic @ FB

- Traffic Applications Production Engineer
- Reliability
  - Hardware
  - Deployments
  - Architecture
  - Configuration
- Matters because all bytes for Facebook, Instagram, WhatsApp flow through our services

Takeaways

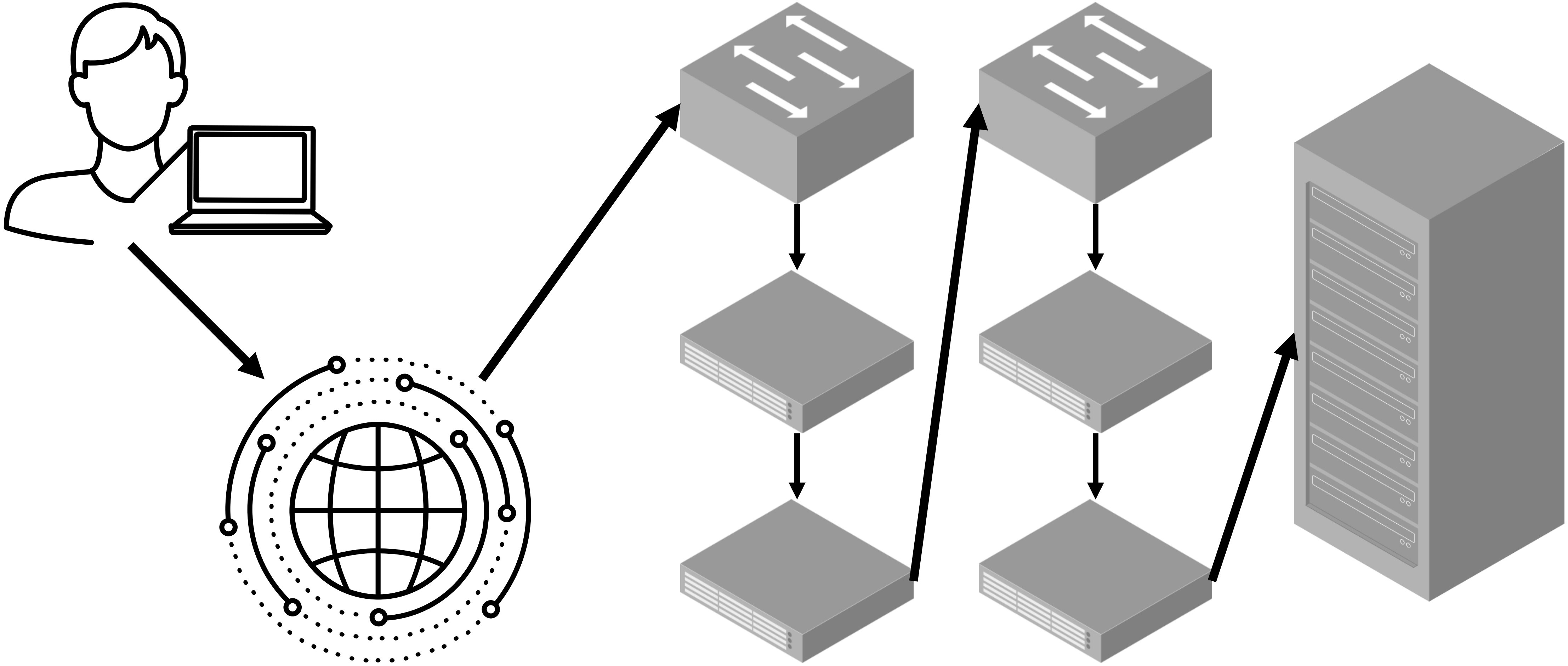
- Introduction to network load balancers
- Know what classes of load balancer is appropriate for common usecases
- Know some attributes of network architecture at scale

What we'll cover

- Why Load Balance?
- L4/L7 Load Balancers
- Directing clients to different locations
- Utility of POPs
- Global load balancing
- MagLev Load Balancers

Glimpse behind the curtain

Stack of Loadbalancers



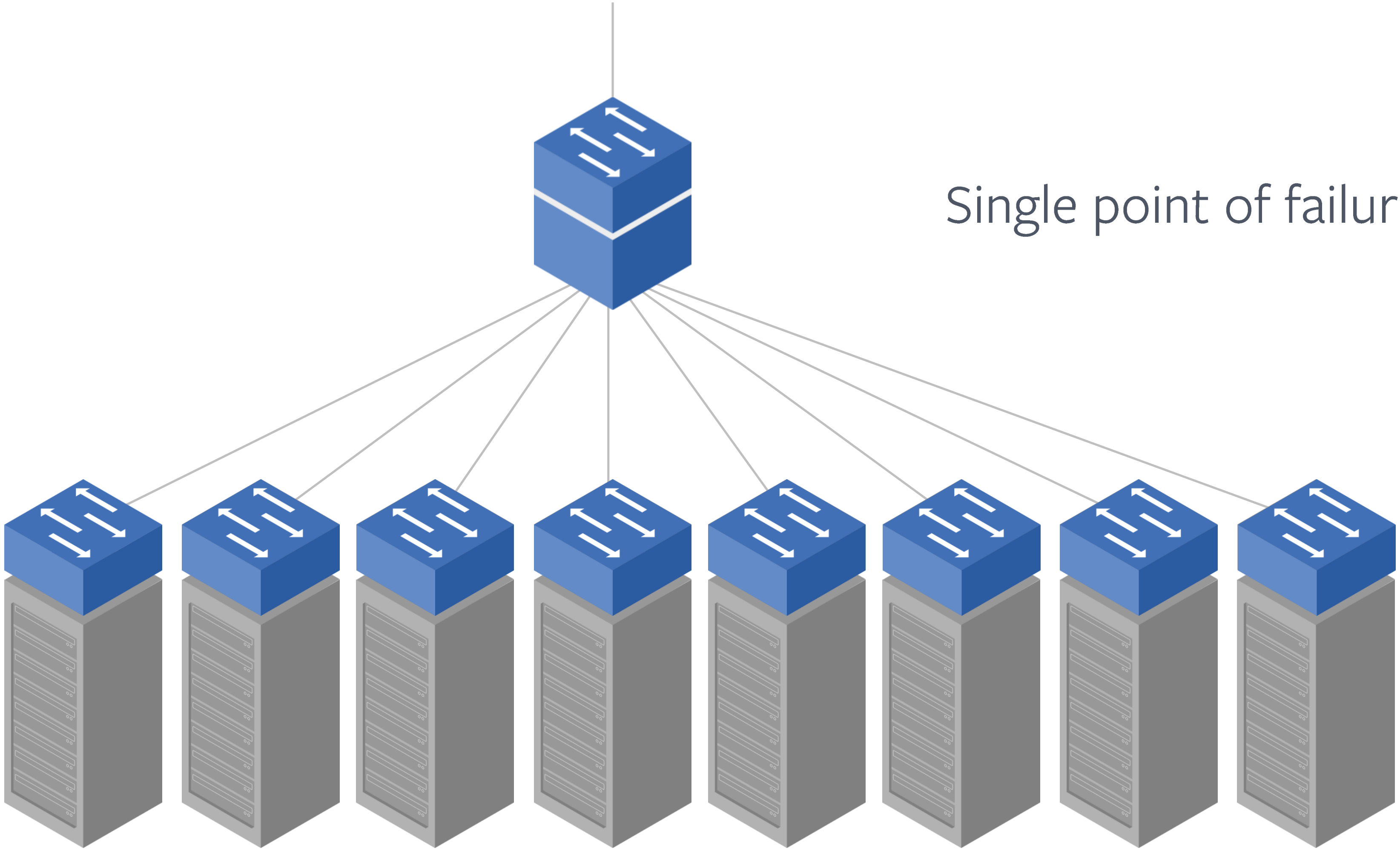
Why Load Balance?

Load Balance because…

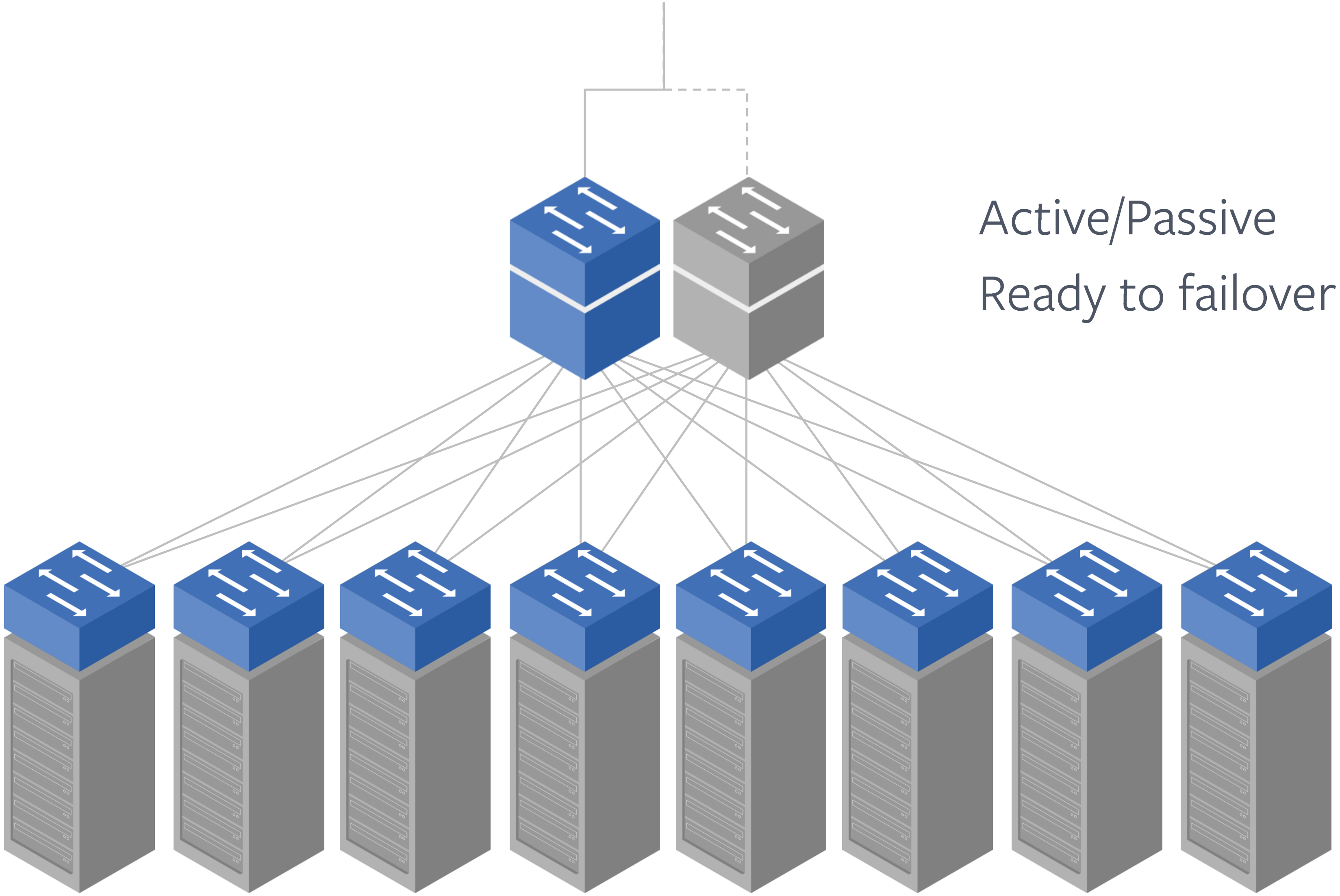
- Have too many requests for a single server to handle
- Defending against the failure of an individual server
- Want to be able to gracefully add and remove capacity
- … without service interruption

- Intelligently control where traffic gets directed

“Proxy” L4/L7 Load Balancers



Single point of failure



Active/Passive

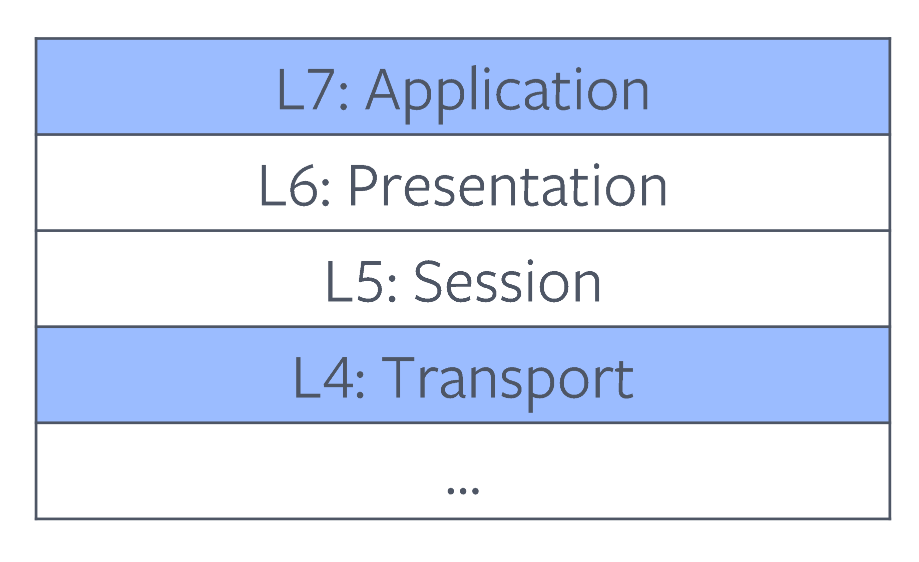
Ready to failover

How?

- Healthchecks the backends
- Incoming connections (usually) routed to same backend
- Backend failure will get existing connections TCP reset
    - New connections don’t get sent there

Difference is the OSI layer

- L4: transport and network protocol
  - Individual packets
- L7: application protocol
  - Requests



L7: Application

L6: Presentation

L5: Session

L4: Transport

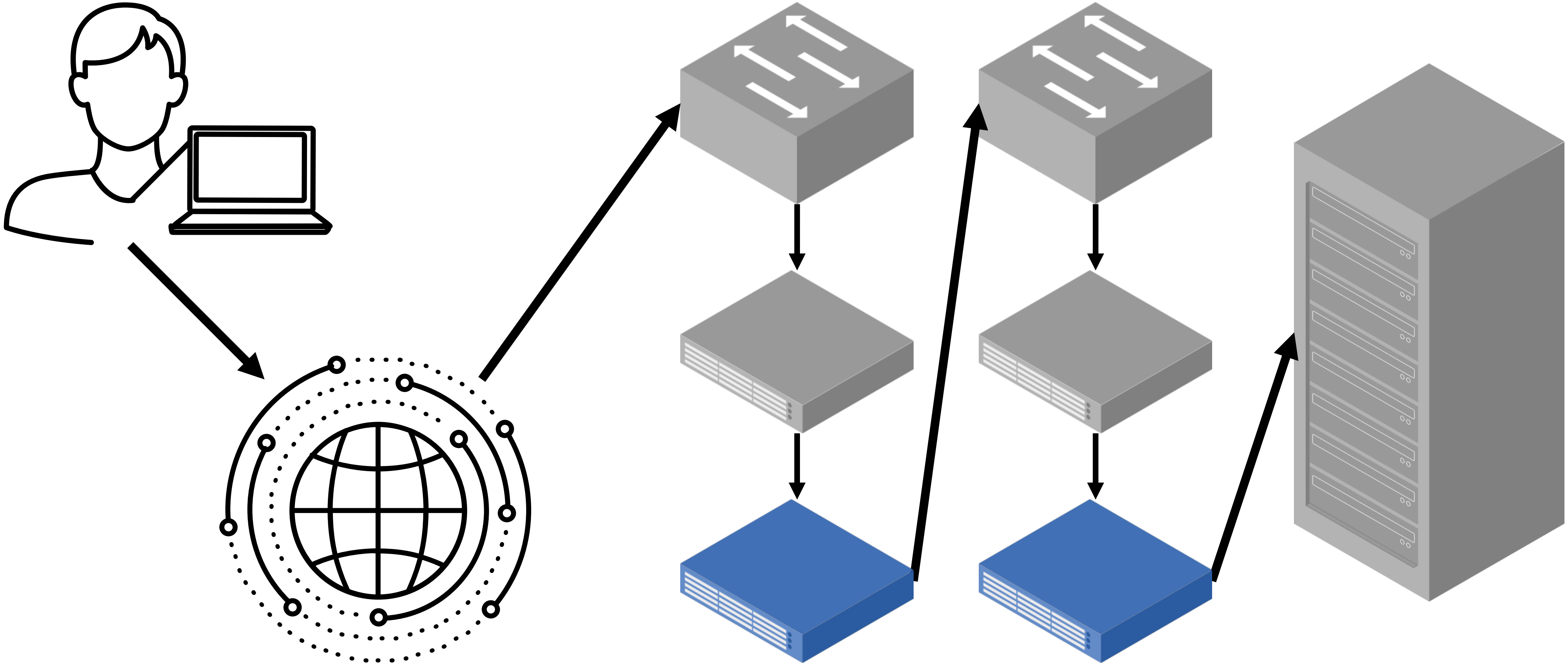
...

History

- L4 because CPU bottleneck before network
- CPUs got better faster than networks
- Increased CPU power allowed L7 features

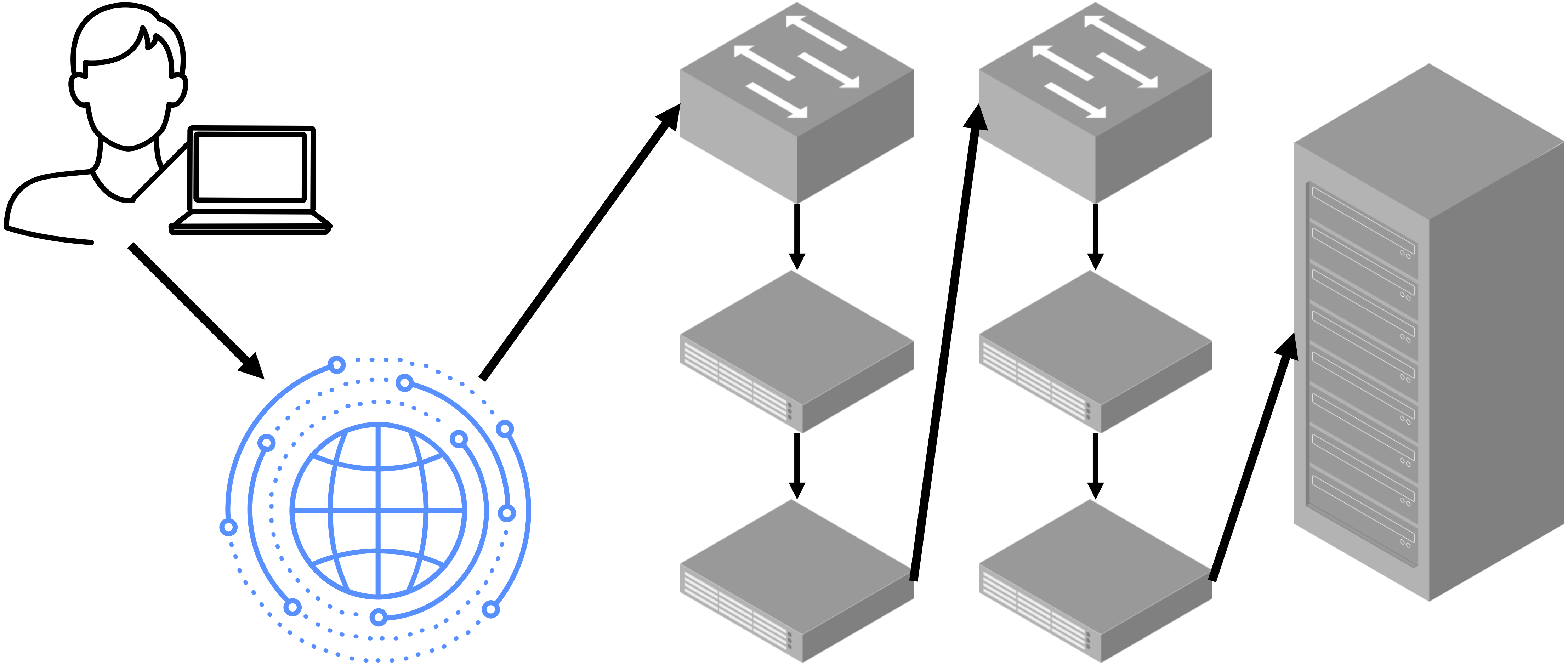
- Network is the bottleneck now
  - Stack Overflow architecture (link)

Where are we?



Getting to the datacenters

Where are we?



DNS Flavours

- Round robin DNS
  - Multiple servers for one domain
- Anycast DNS
  - One IP, shared across multiple locations
  - Routed by the intermediate networks

Even More DNS Flavours

- Geoaware DNS
  - Records specific to geographical areas
- Network aware DNS
  - Records specific to peering/networks the user is connected to
- Latency aware DNS
  - Records specific to the latency from the user to datacenter

Problems with DNS

- Caching
  - Multiple devices along the request path
  - Short TTL expiry doesn’t fix everything
  - Traffic imbalance
- Anycast routing not always optimal
  - Eg Twitter’s experience (link)

Problems with DNS

- Limited information
  - Recursive resolver doesn’t pass data on
  - EDNS client subnet RFC as a solution
  - But... not always following it

Steering users

- None of these decisions have to be made once and only once
- Initial target can be best guess
- Everything past the initial connection is based on what you send
    - TCP connection has the user’s address

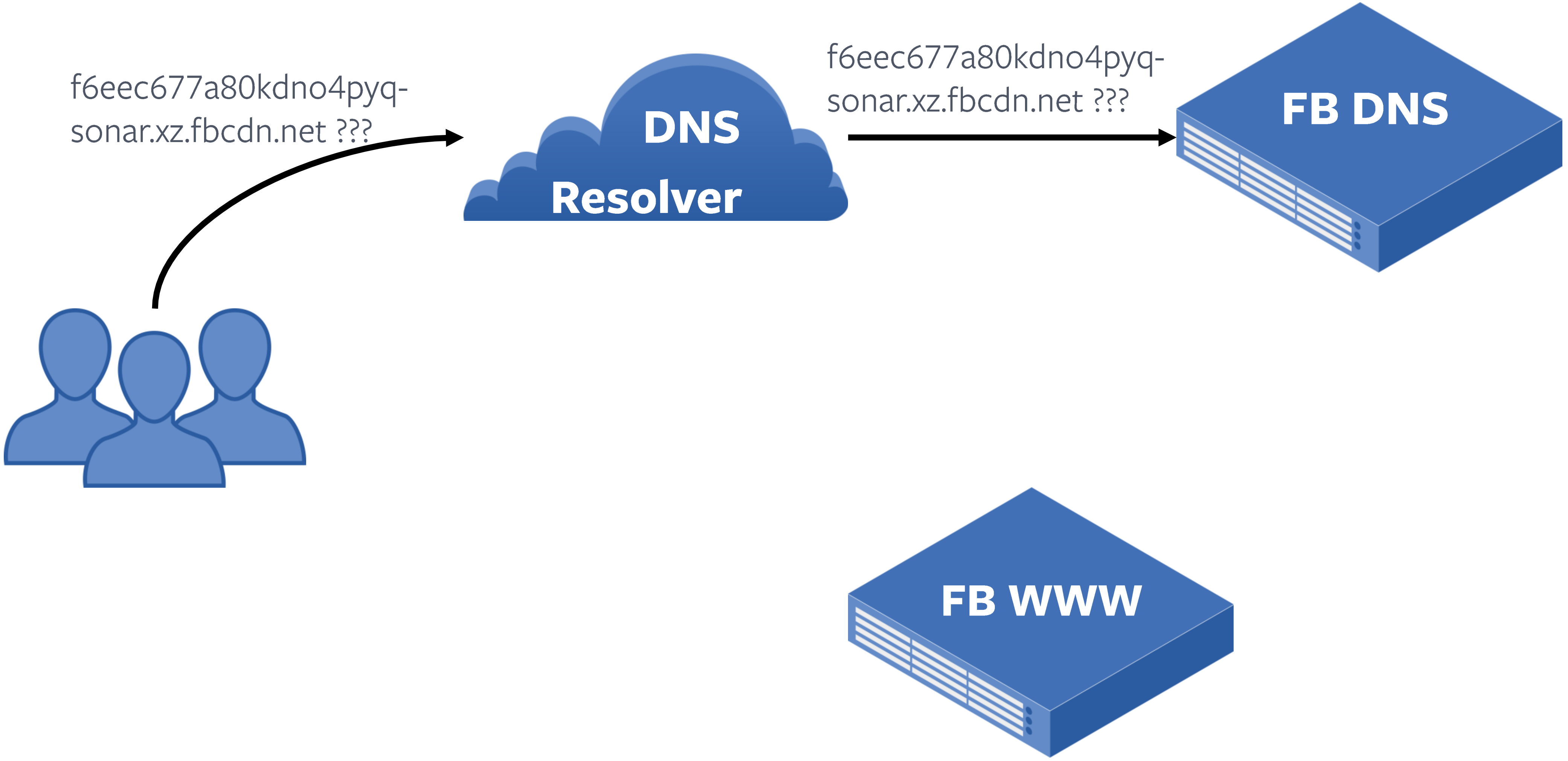
Data sources

- Geo IP data
- Peering databases
- Latency measurements

DIY latency measurement

- Fraction of profile pictures are loaded from random location
- DNS logs (unique record, **resolver address**)
- Webserver logs (unique record, **latency measurements**)
- Join to get (**resolver address**, **latency measurements**)

- More details: Sonar @ APNIC 44 talk (link)



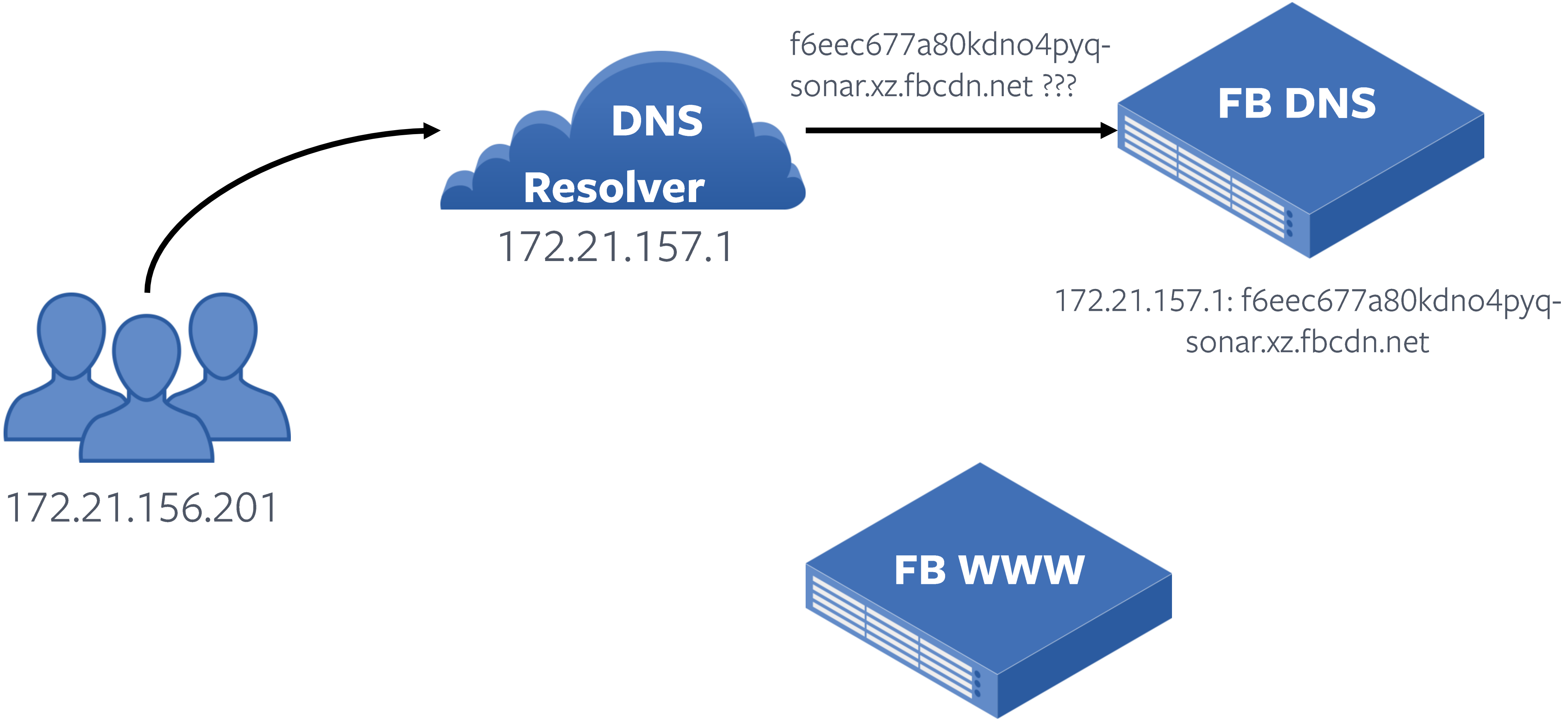
f6eec677a80kdno4pyq-sonar.xz.fbcdn.net ???

DNS Resolver

f6eec677a80kdno4pyq-sonar.xz.fbcdn.net ???

FB DNS

FB WWW



f6eec677a80kdno4pyq-sonar.xz.fbcdn.net ???

DNS Resolver

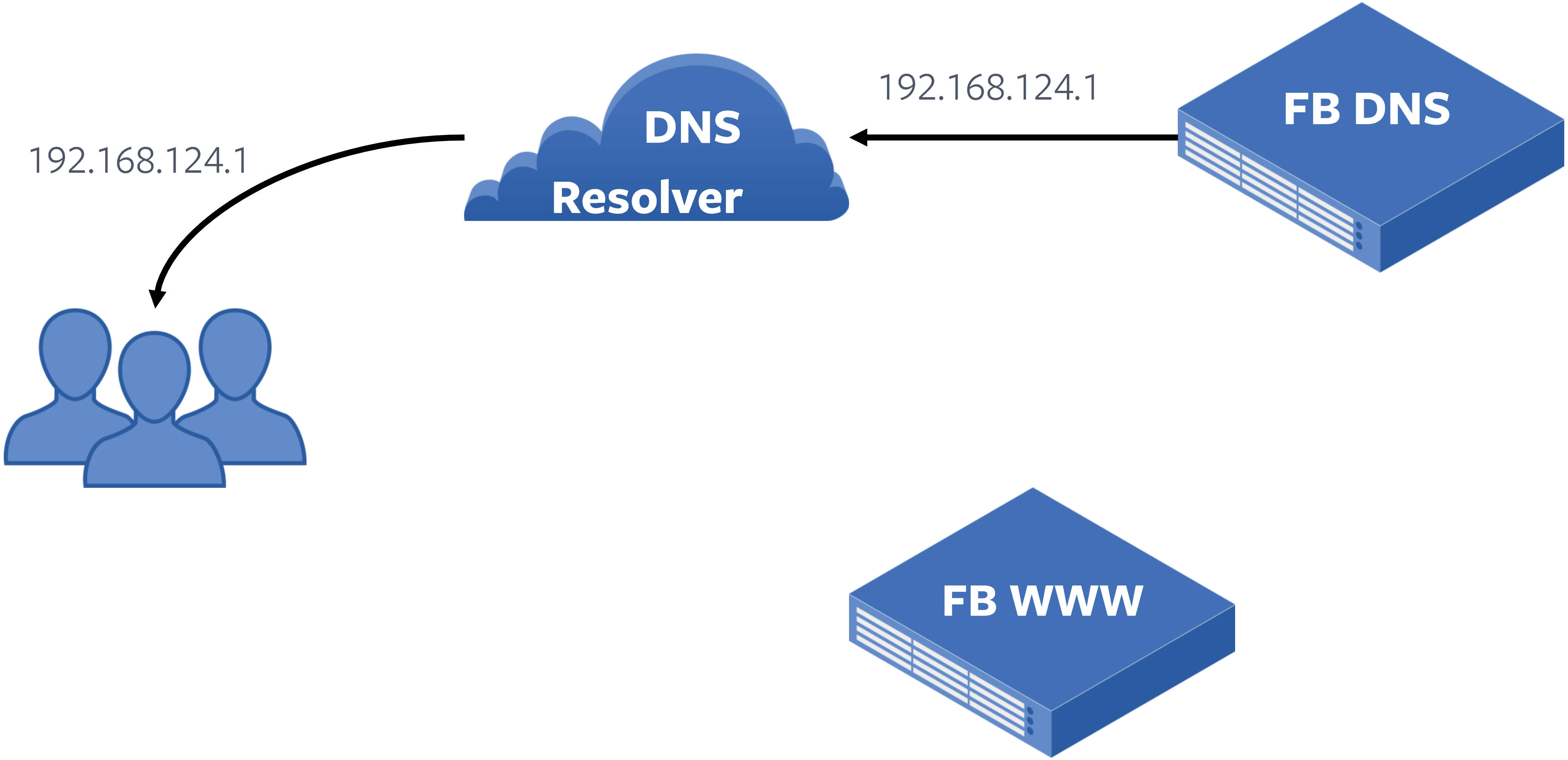
172.21.157.1

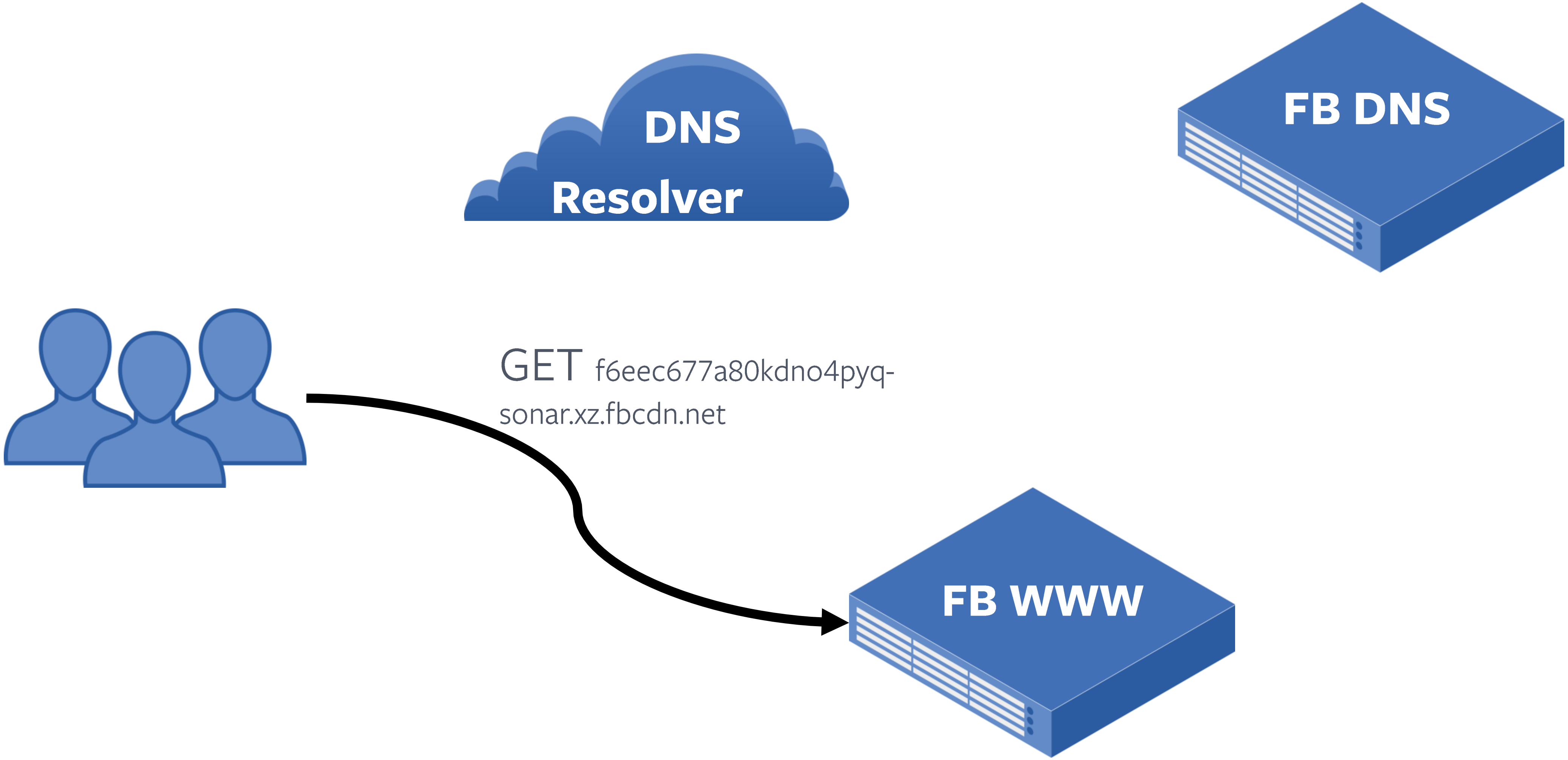
FB DNS

172.21.157.1: f6eec677a80kdno4pyq-sonar.xz.fbcdn.net

172.21.156.201

FB WWW


192.168.124.1
DNS
Resolver
192.168.124.1
FB DNS
FB WWW

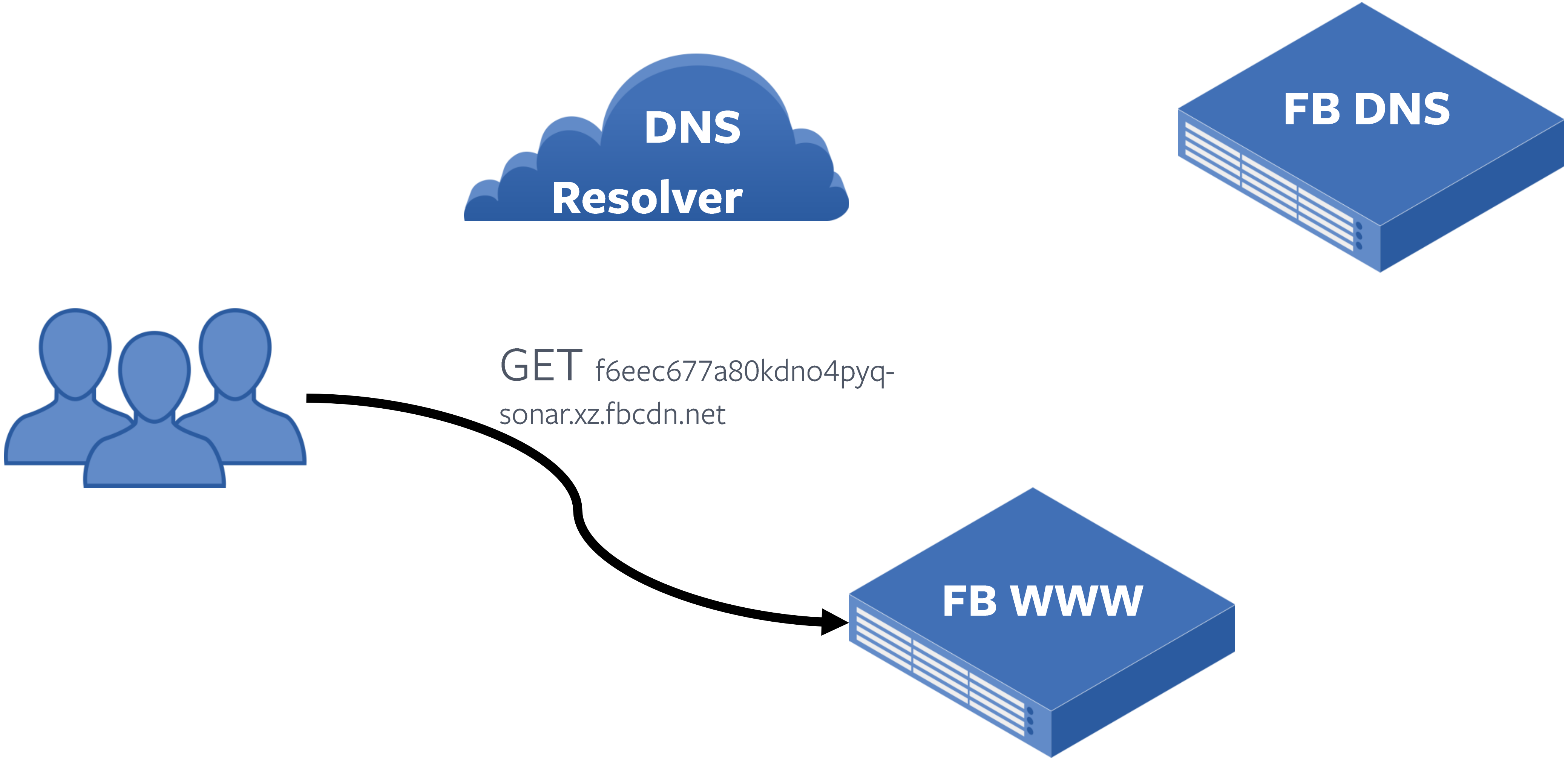


DNS Resolver

FB DNS

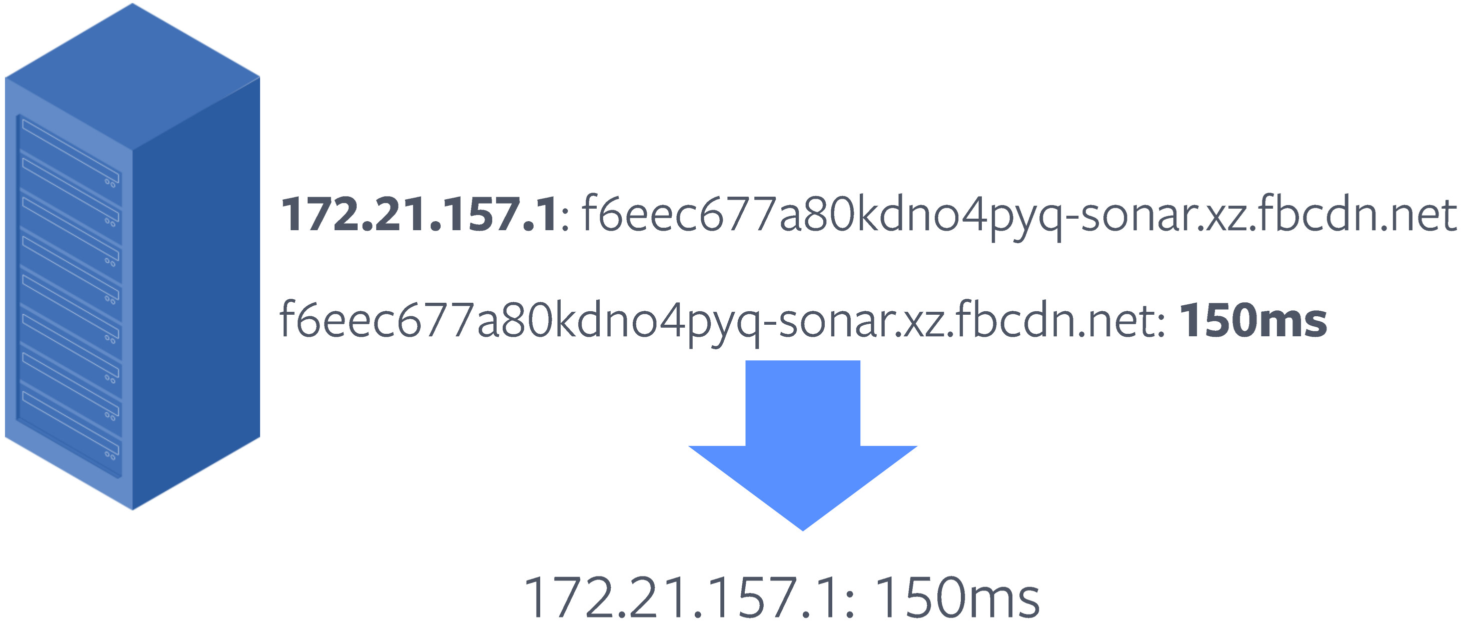
GET f6eec677a80kdno4pyq-sonar.xz.fbcdn.net

FB WWW



GET f6eec677a80kdno4pyq-sonar.xz.fbcdn.net

f6eec677a80kdno4pyq-sonar.xz.fbcdn.net: 150ms



172.21.157.1: f6eec677a80kdno4pyq-sonar.xz.fbcdn.net

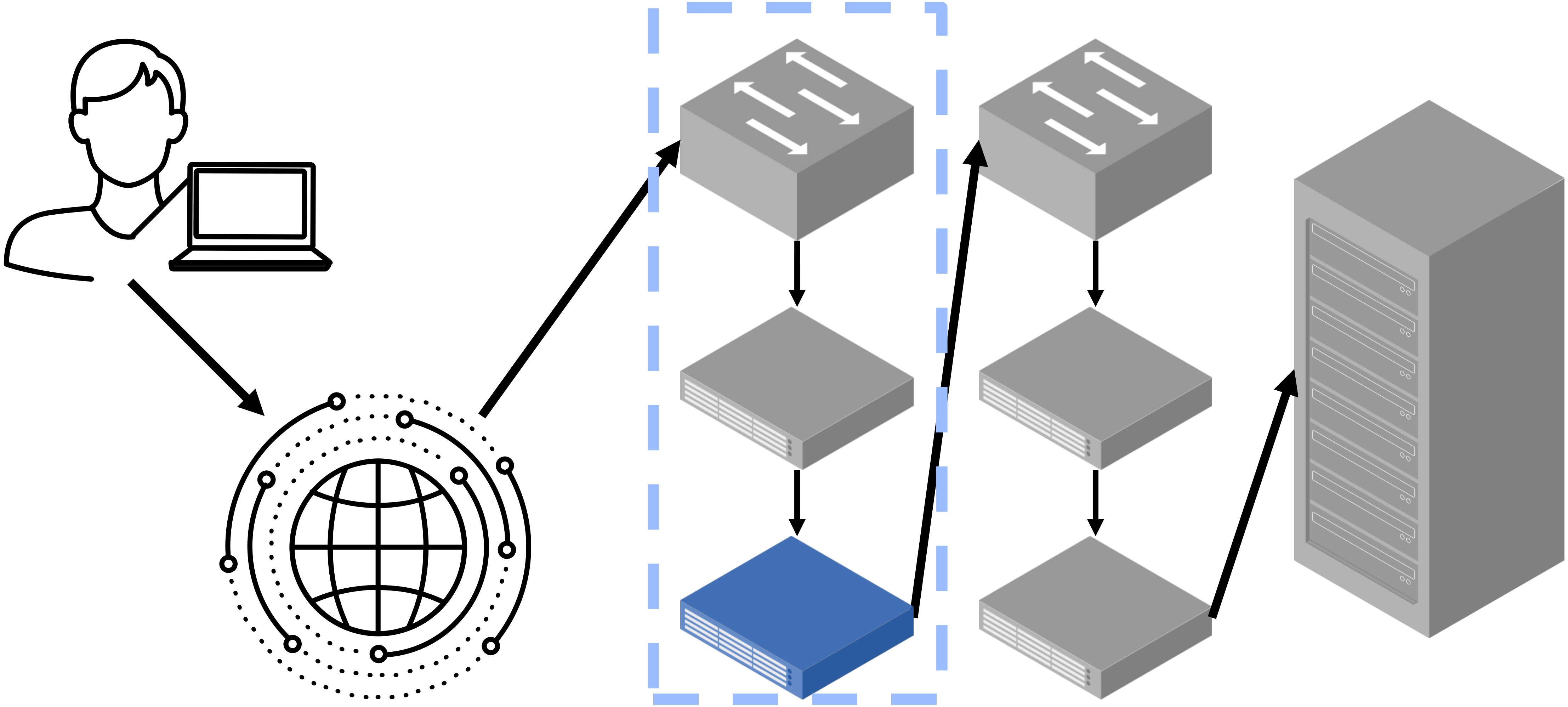
f6eec677a80kdno4pyq-sonar.xz.fbcdn.net: **150ms**

172.21.157.1: 150ms

Material is recycled and flame retardant
Material is recycled and flame retardant
Material is recycled and flame retardant

Points of Presence (POPs)

Where are we?



Why PoPs

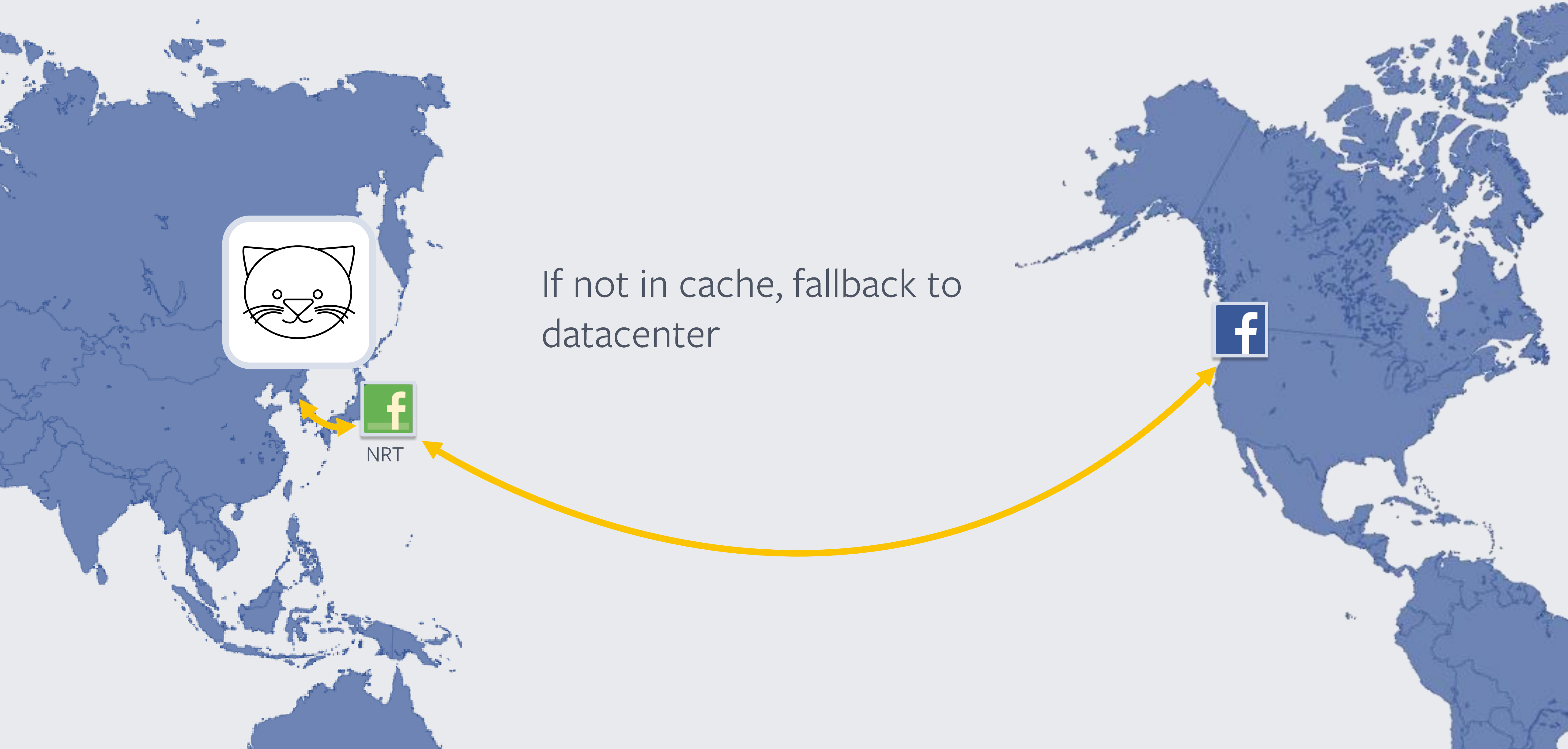
- PoPs are useful for more than CDNs
- Static vs Dynamic content

- Less complex than an entire new region because it only serves web traffic

Static vs Dynamic

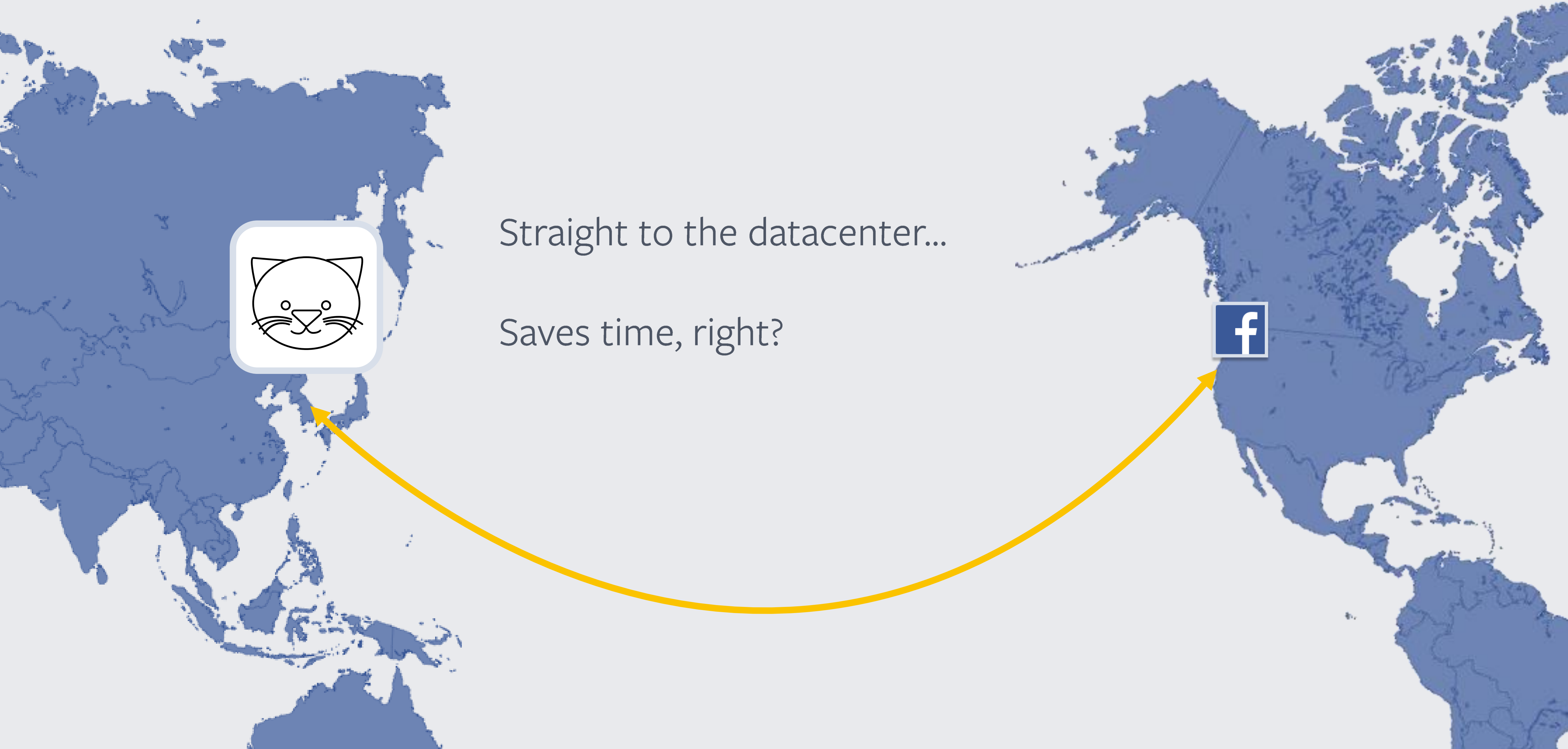
- Static
  - Easily cacheable
  - Not changing between users
  - Video, pictures
- Dynamic
  - Per-user customizations
  - Messenger chats

Typical CDN: Static Content + Fallback



If not in cache, fallback to datacenter

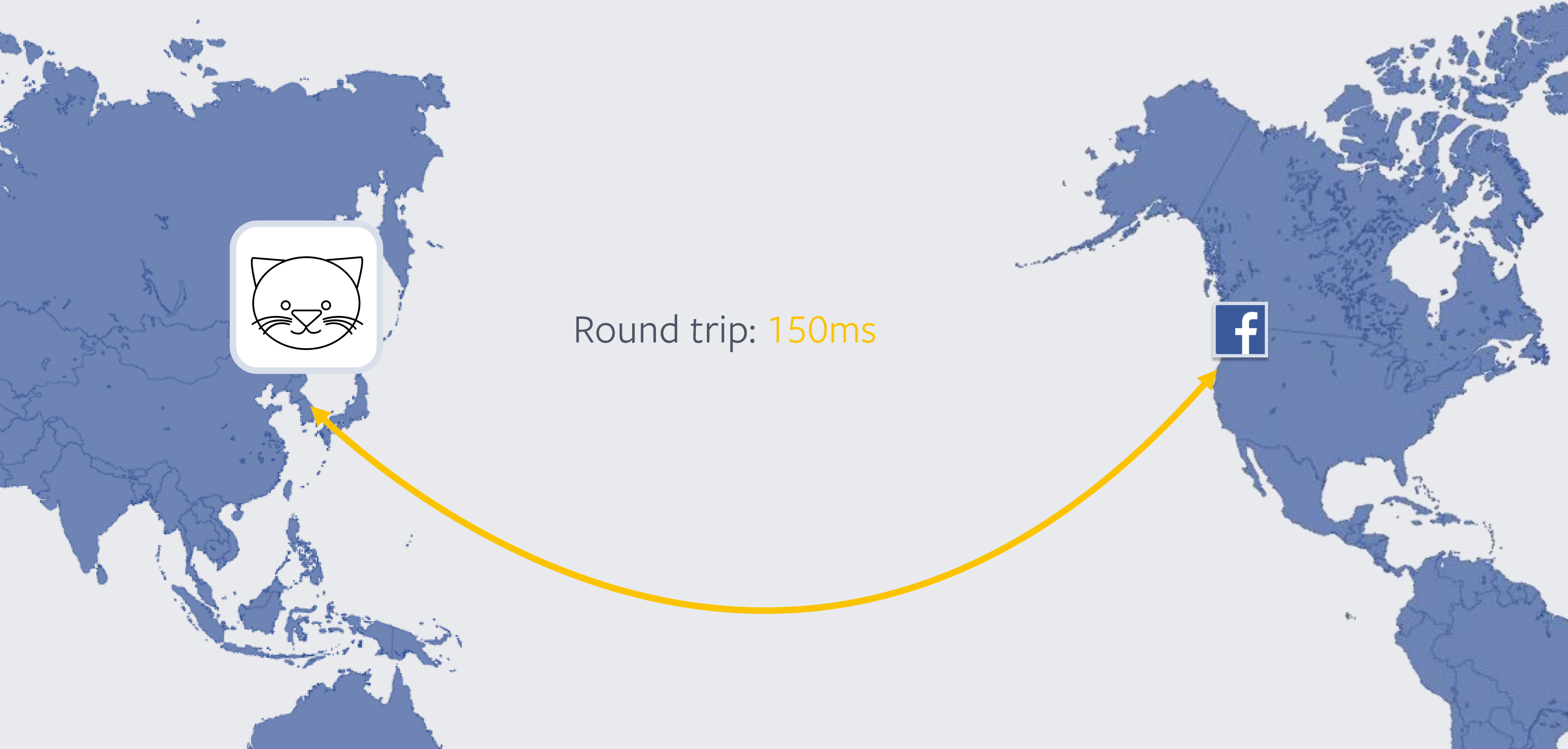
Dynamic Content



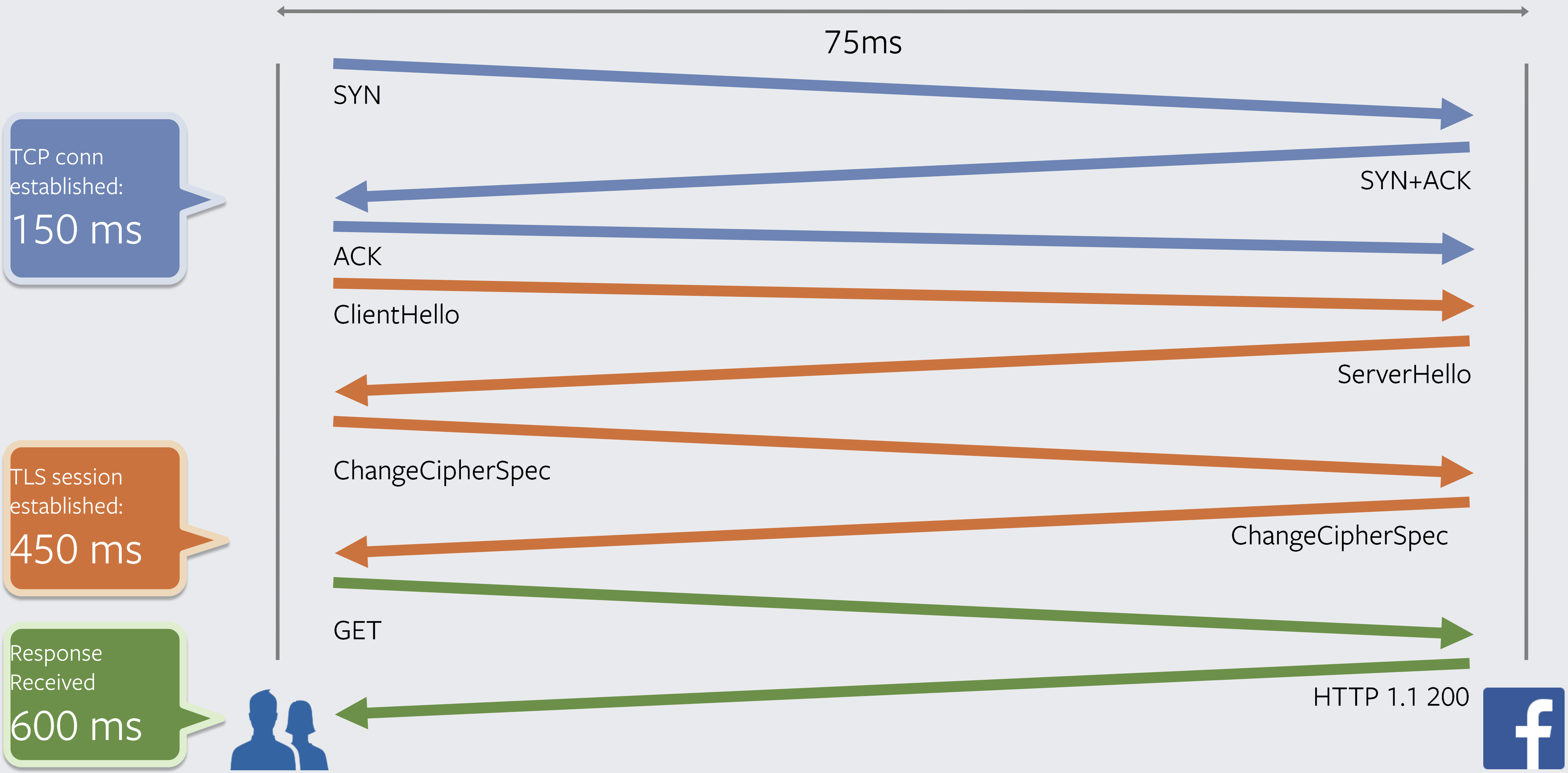
Straight to the datacenter...

Saves time, right?

Seoul -> Oregon



Round trip: 150ms

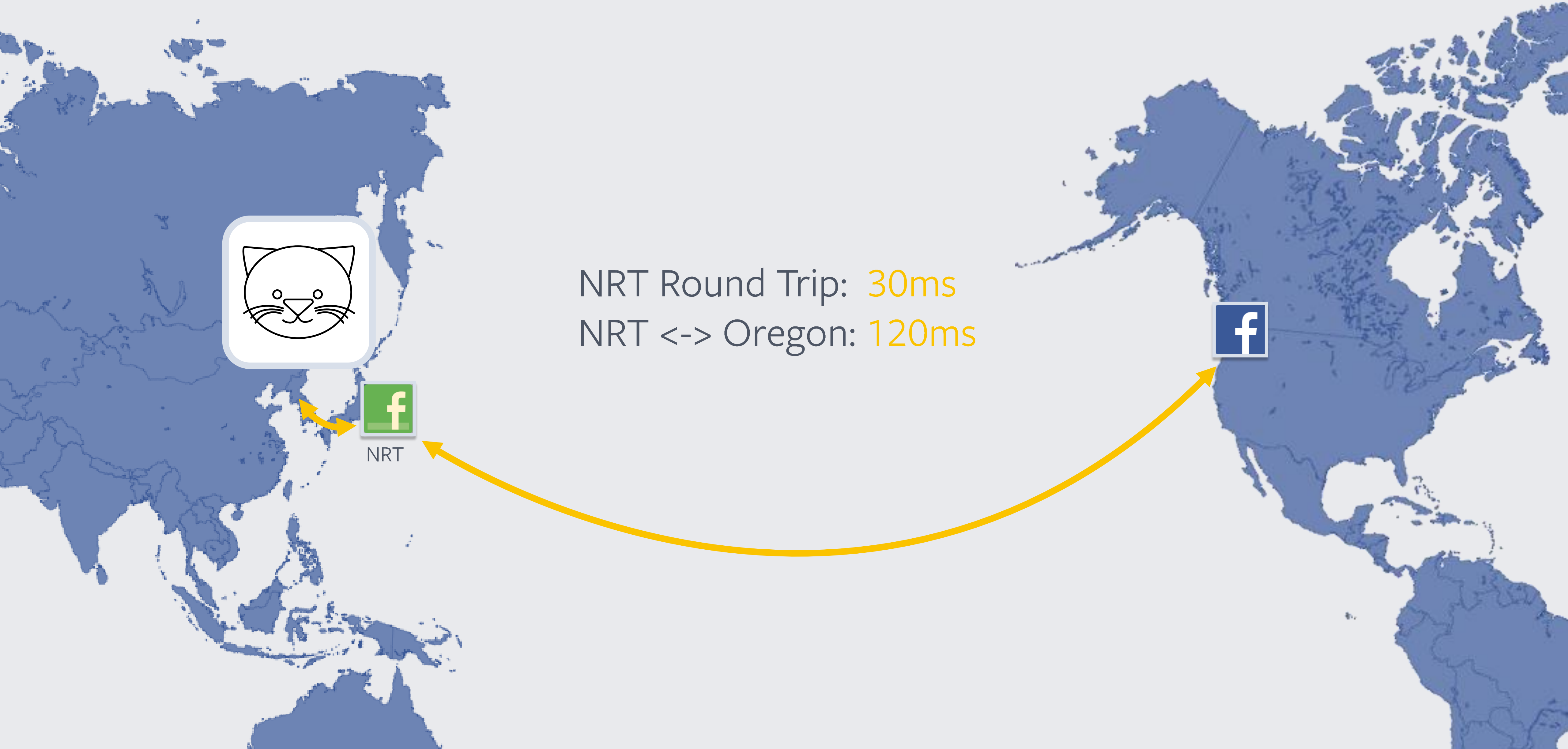


TCP conn established:
150 ms

TLS session established:
450 ms

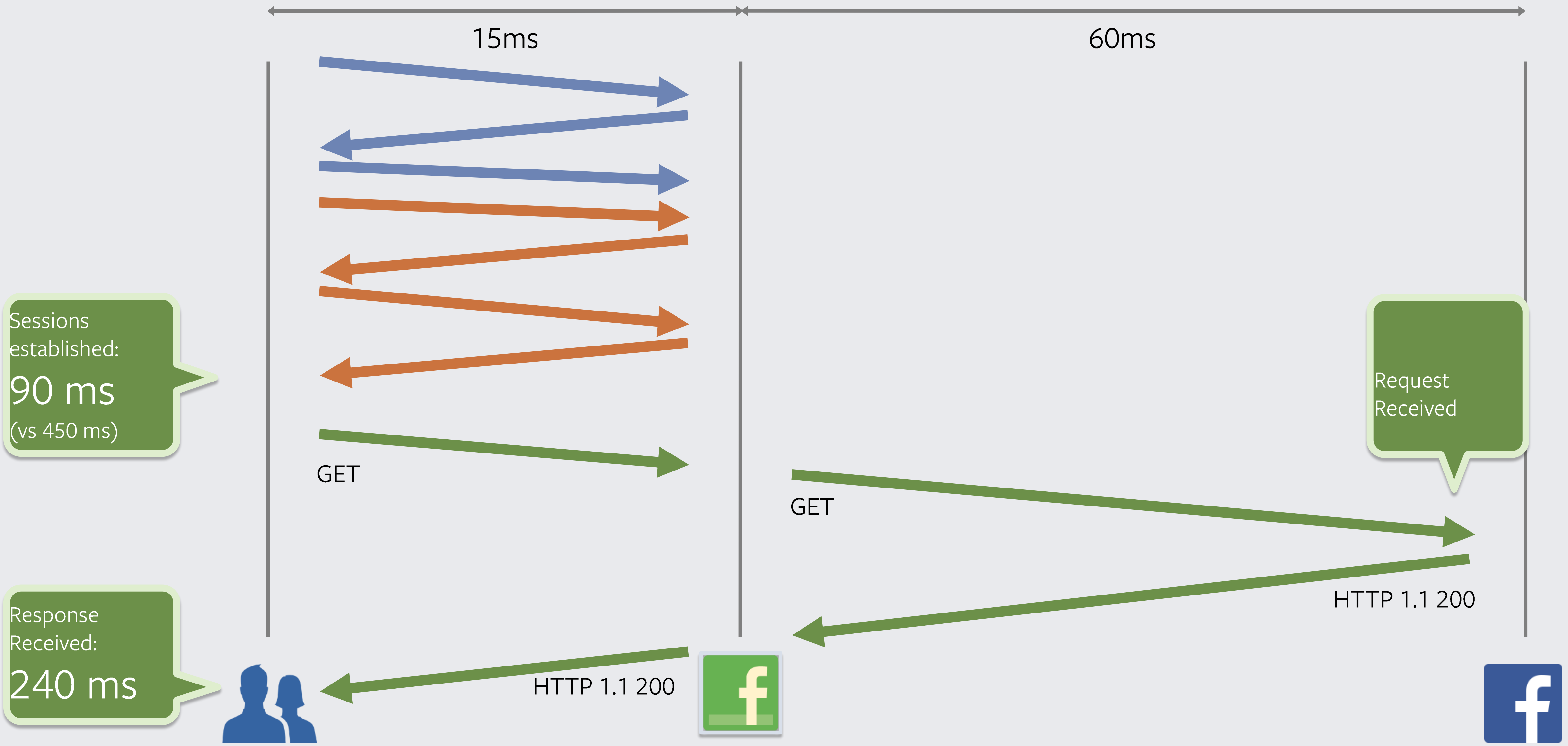
Response Received
600 ms

Seoul -> Narita -> Oregon



NRT Round Trip: 30ms

NRT <-> Oregon: 120ms

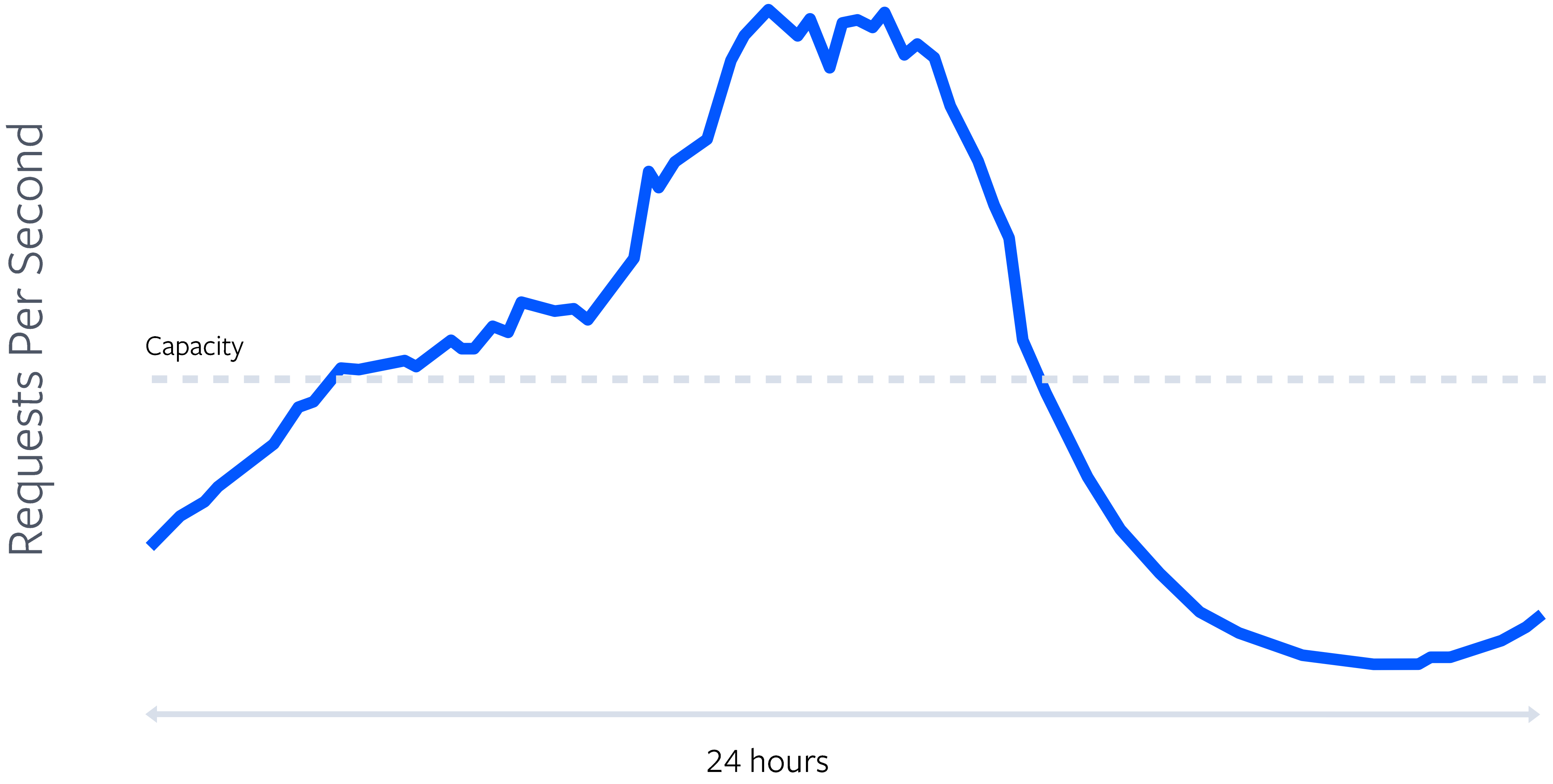

15ms
60ms
Sessions established:
90 ms
(vs 450 ms)
GET
GET
Request Received
HTTP 1.1 200
Response Received:
240 ms
HTTP 1.1 200

Global Load Balancing

Why is it useful?

- 1.5 billion daily active users
- Network has to account for special events
  - Can’t autoscale fiber laying
- How do we send users to available capacity?

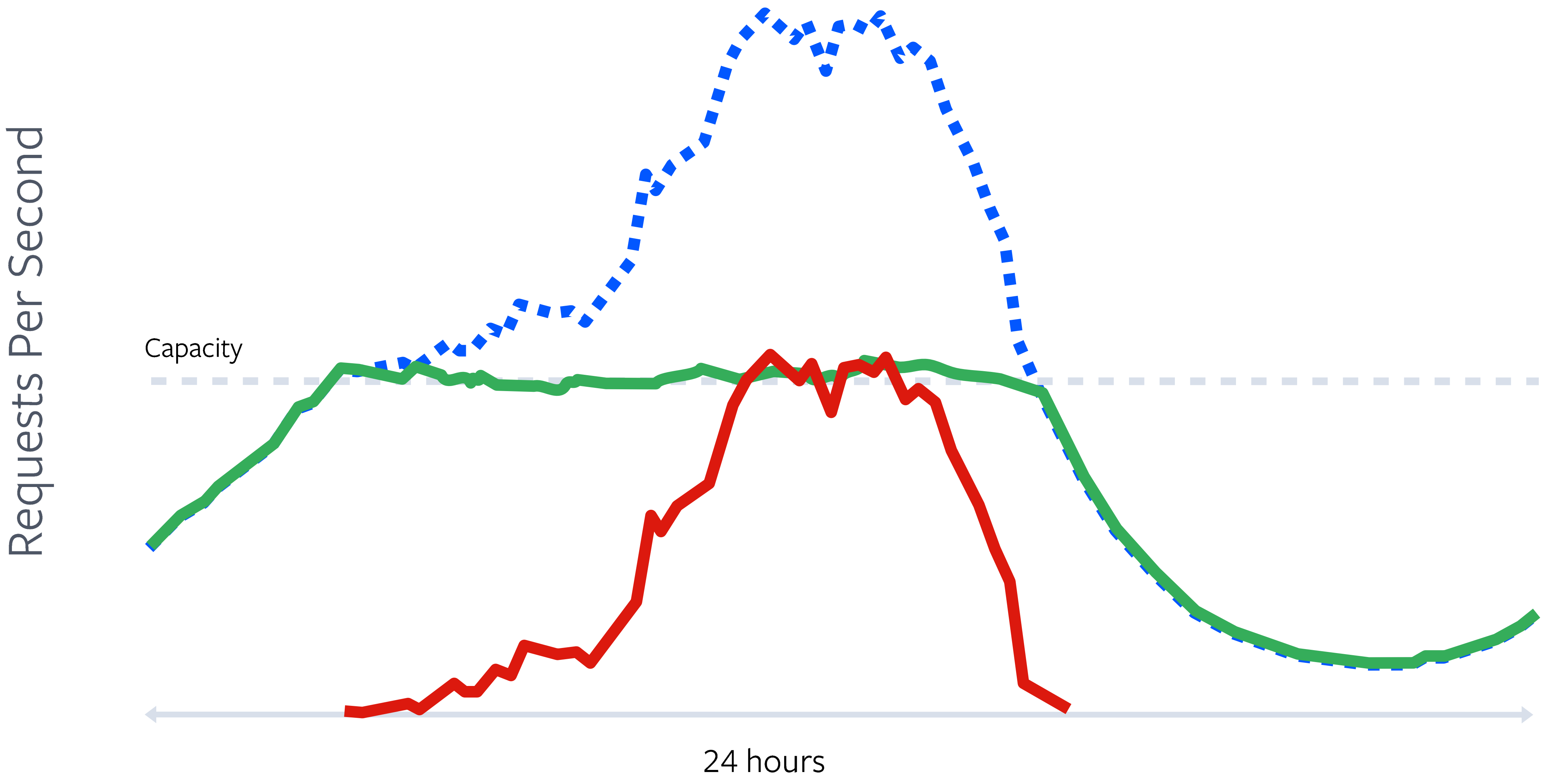
Requests exceed capacity



Cartographer

- Look at what a specific PoP/DC can handle.
- Direct traffic to the PoP/DC until it approaches the capacity limit
- Divert excess traffic elsewhere, continue serving up to the limit
- When requests go down, stop diverting

Use non-optimal unused capacity elsewhere



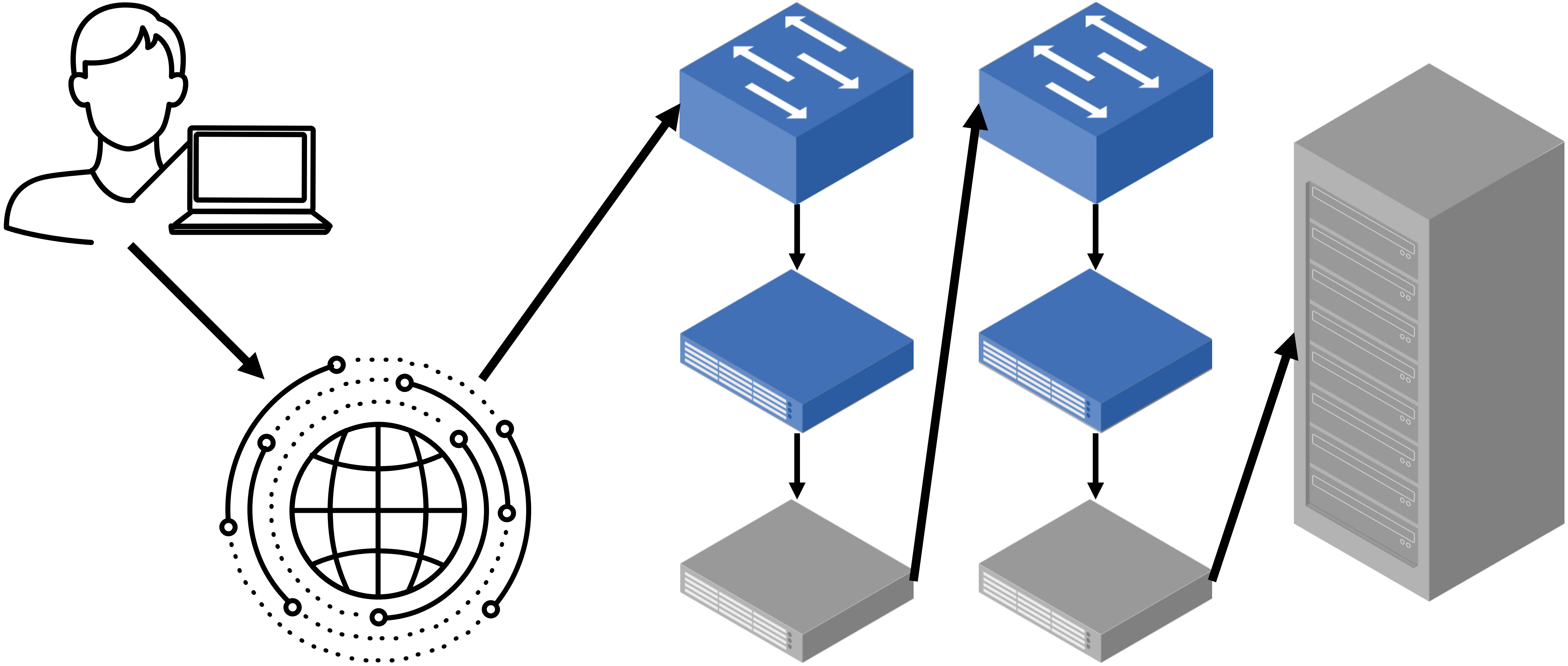
38
37
36
35
34
33
32
31
28
27
26
23
22
21
20
19
17
16
15

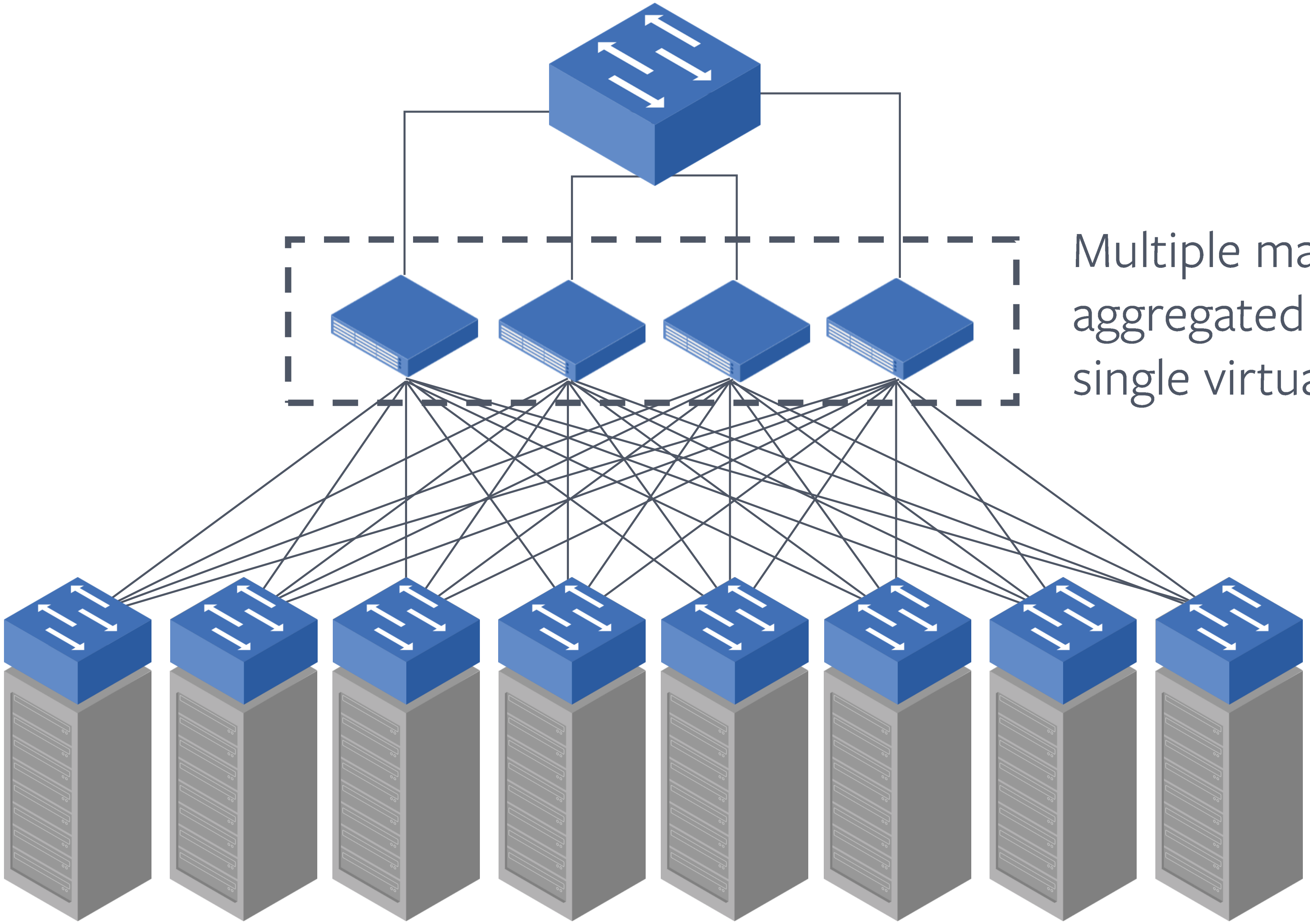
MagLev Load Balancers

What.

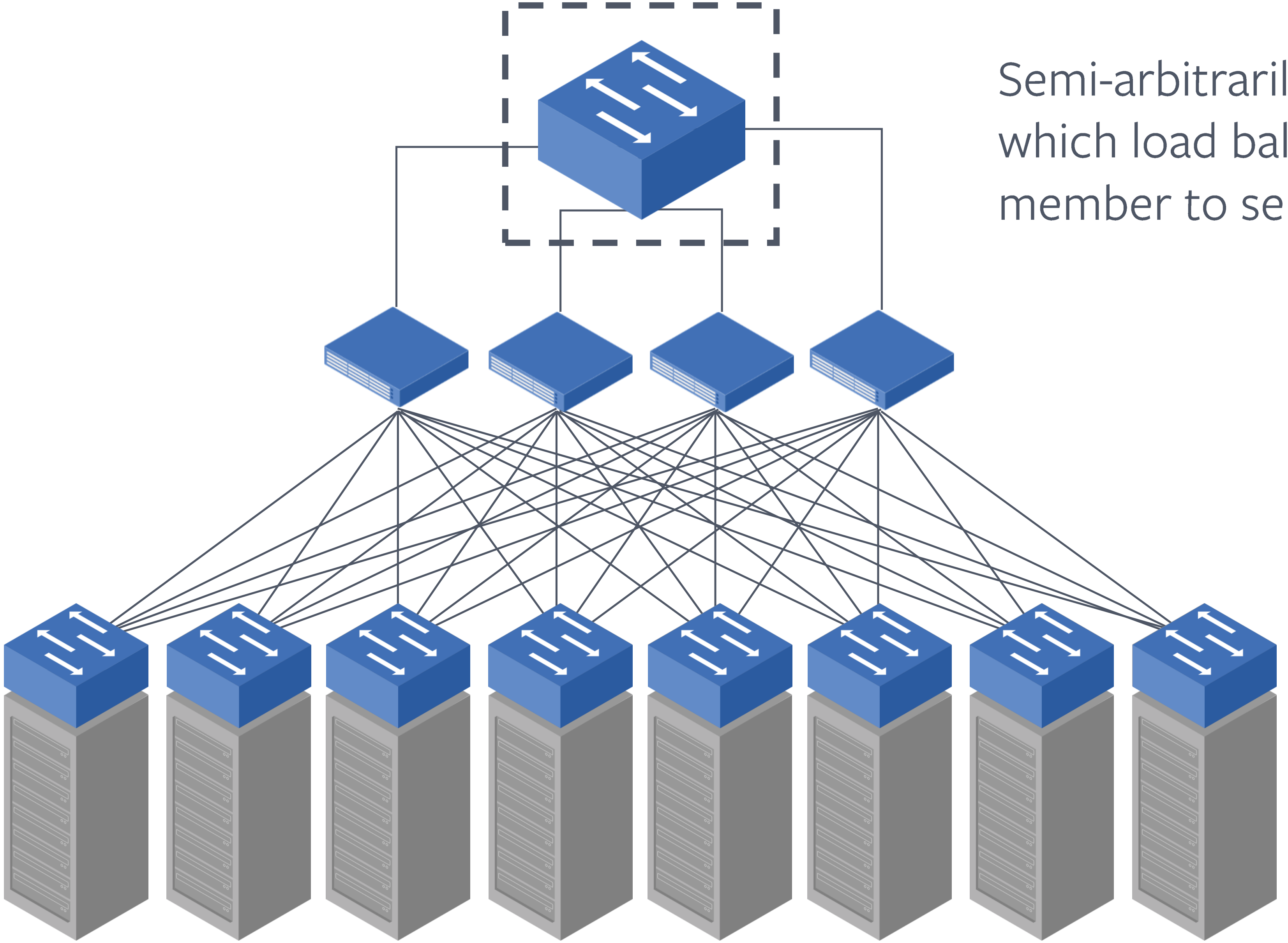
- Name from Google paper
- An answer to “How do you spread packets for a single IP across multiple load balancers while making sure the flow ends up at the same backend machine”

Where are we?





Multiple machines aggregated into single virtual LB



Semi-arbitrarily decides which load balancer member to send it to

Why not just use the first layer?

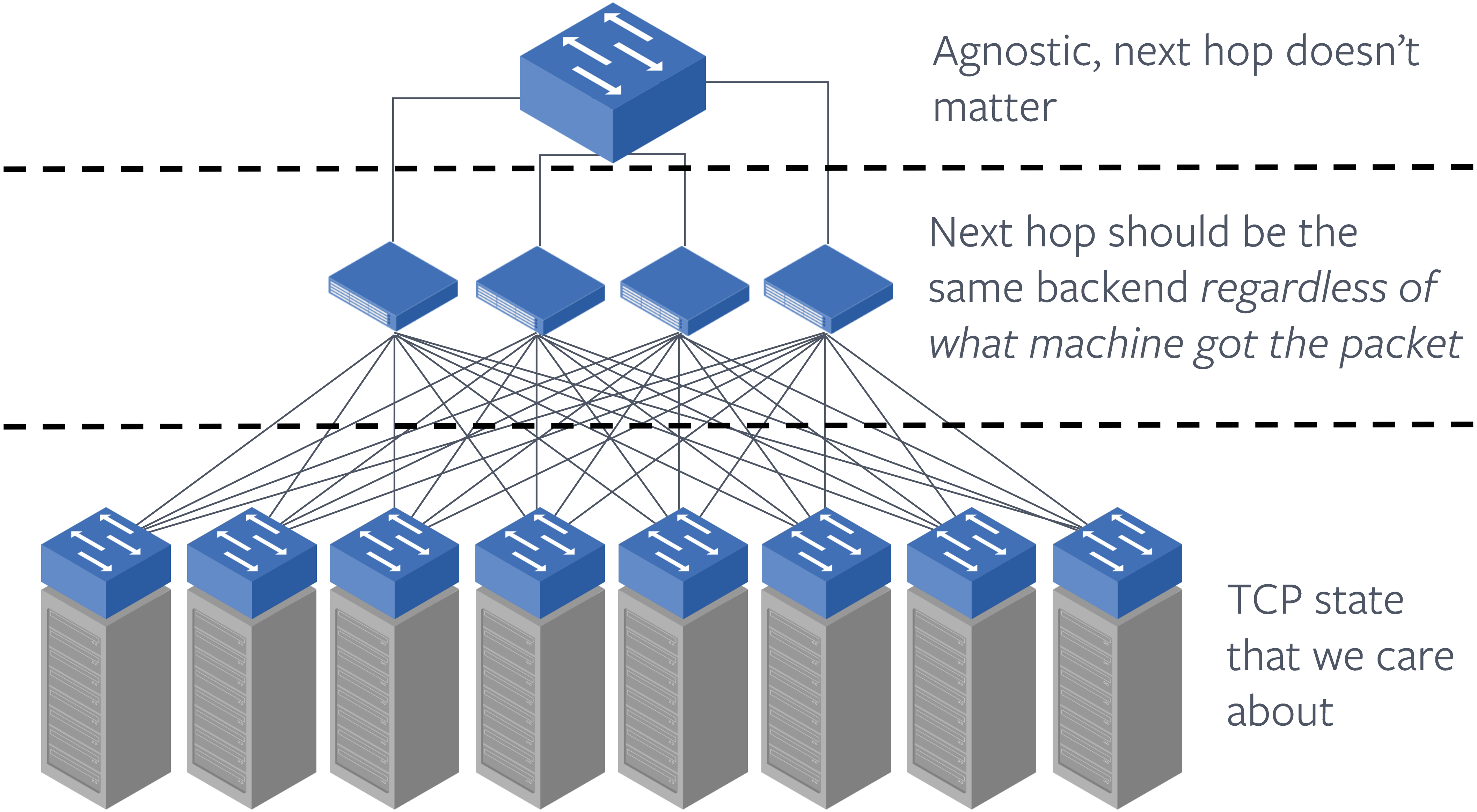
Why is stability needed?

- Adding or removing capacity shouldn’t induce a reshuffling of where packets ultimately get sent
- TCP is stateful
  - Need to end up at the same backend server

Why is stability needed?

- Adding or removing capacity shouldn’t induce a reshuffling of where packets ultimately get sent
- TCP is stateful
    - Need to end up at the same backend server

- ***Goal: Keep sending the same flow to the same backend host, regardless of what network path it takes***

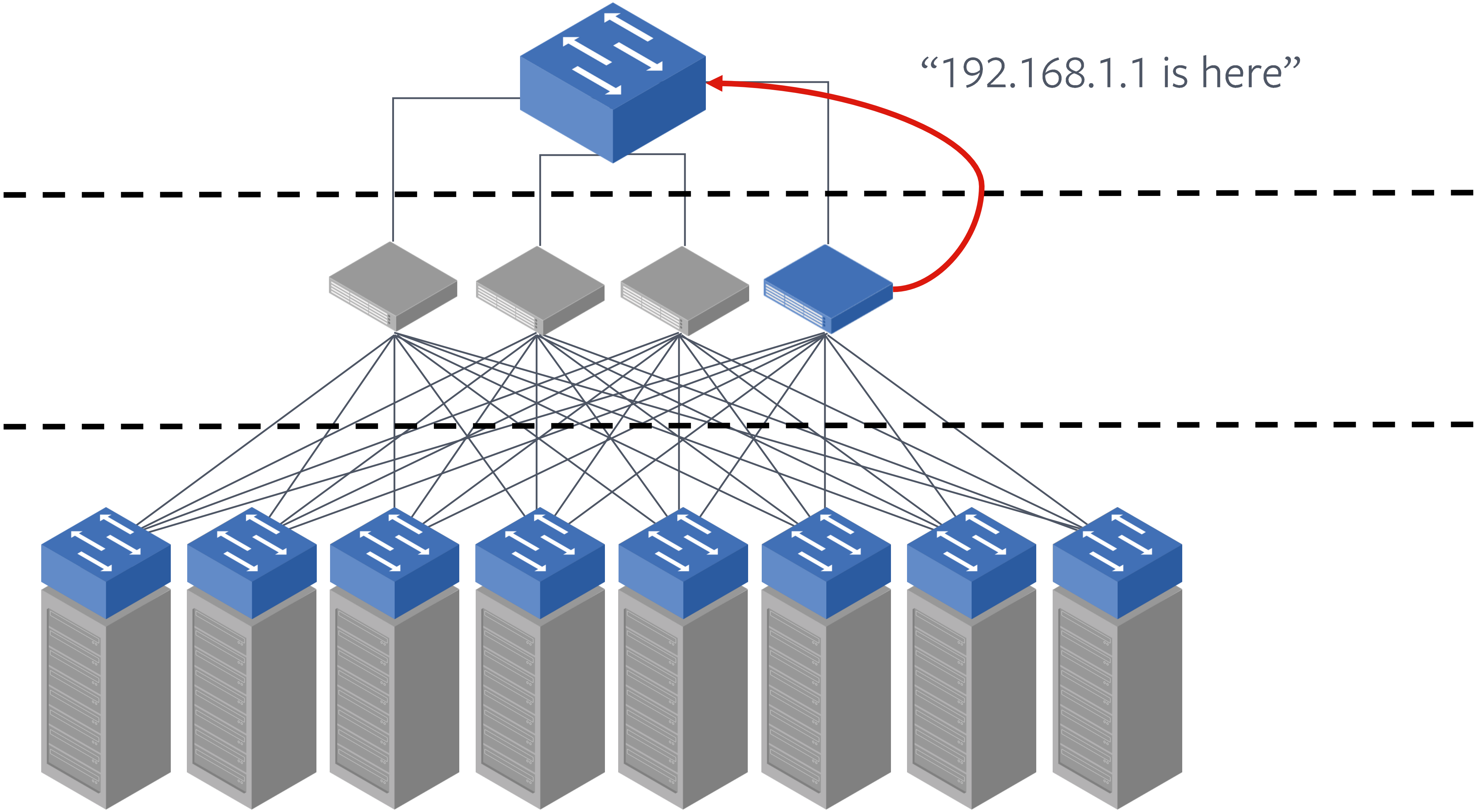


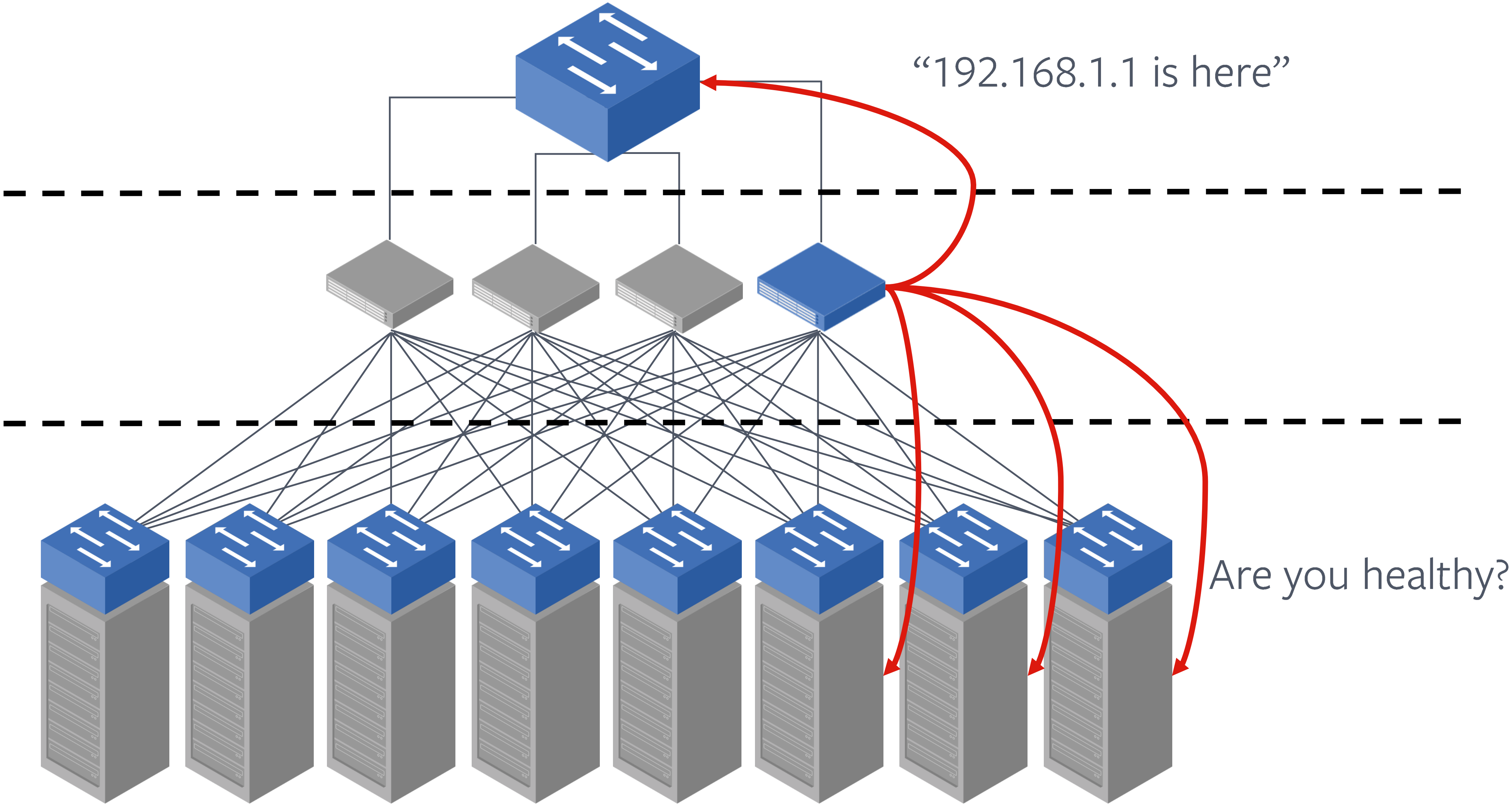
Agnostic, next hop doesn't matter

Next hop should be the same backend *regardless of what machine got the packet*

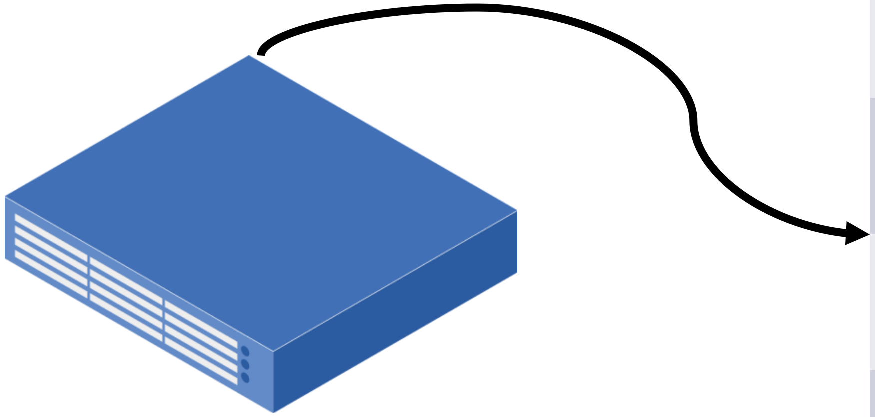
TCP state that we care about

RPP-SB-1C-8
RPP-SB-1C-7

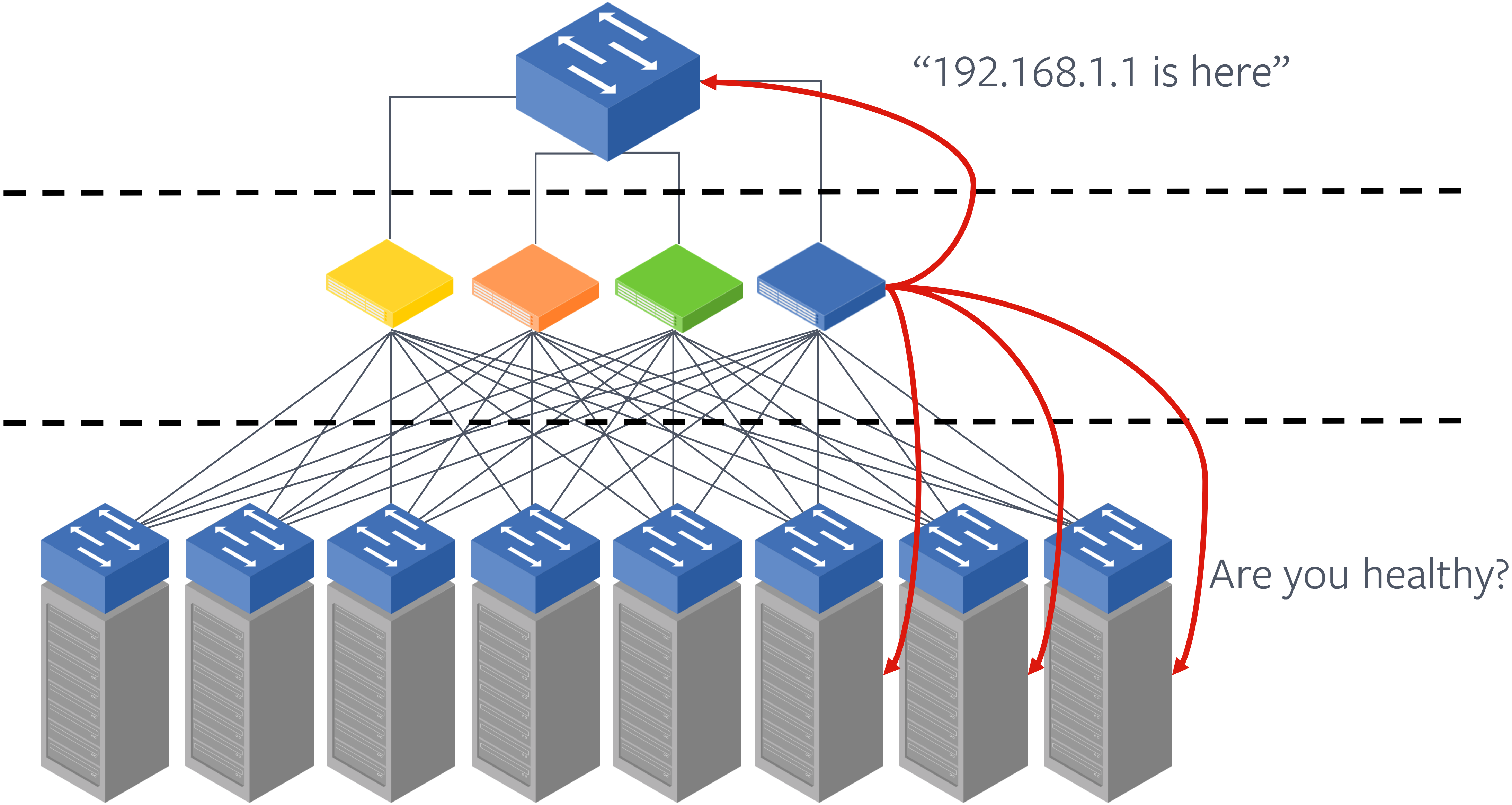

“192.168.1.1 is here”


“192.168.1.1 is here”
Are you healthy?

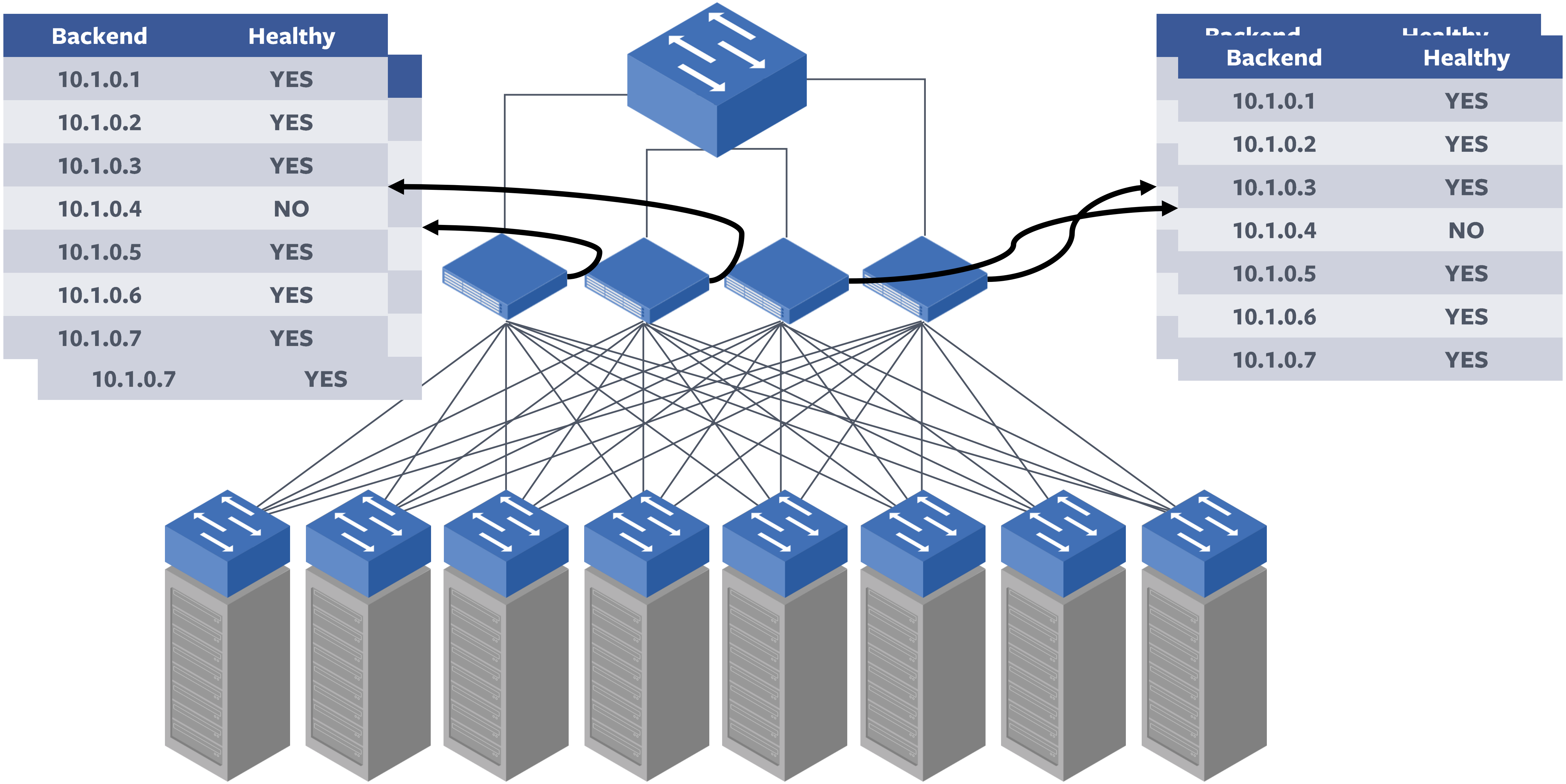
Healthchecks are simple but frequent



Backend	Healthy
10.1.0.1	YES
10.1.0.2	YES
10.1.0.3	YES
10.1.0.4	NO
10.1.0.5	YES
10.1.0.6	YES
10.1.0.7	YES



“192.168.1.1 is here”
Are you healthy?



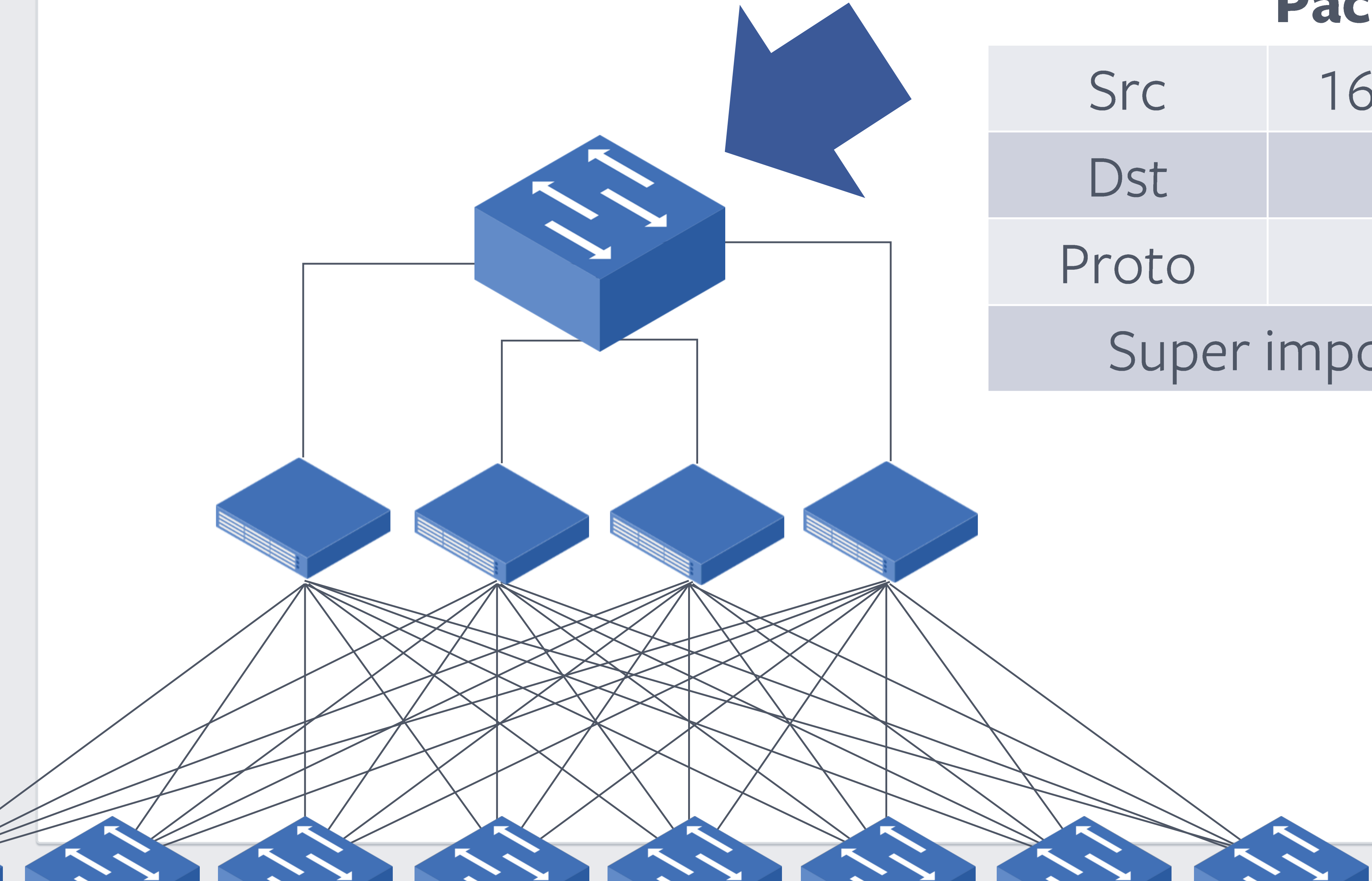
Backend	Healthy
10.1.0.1	YES
10.1.0.2	YES
10.1.0.3	YES
10.1.0.4	NO
10.1.0.5	YES
10.1.0.6	YES
10.1.0.7	YES

10.1.0.7 YES

Backend Healthy

Backend	Healthy
10.1.0.1	YES
10.1.0.2	YES
10.1.0.3	YES
10.1.0.4	NO
10.1.0.5	YES
10.1.0.6	YES
10.1.0.7	YES

CYCLED AND FLAME RETARDANT
MATERIAL IS RECYCLED A

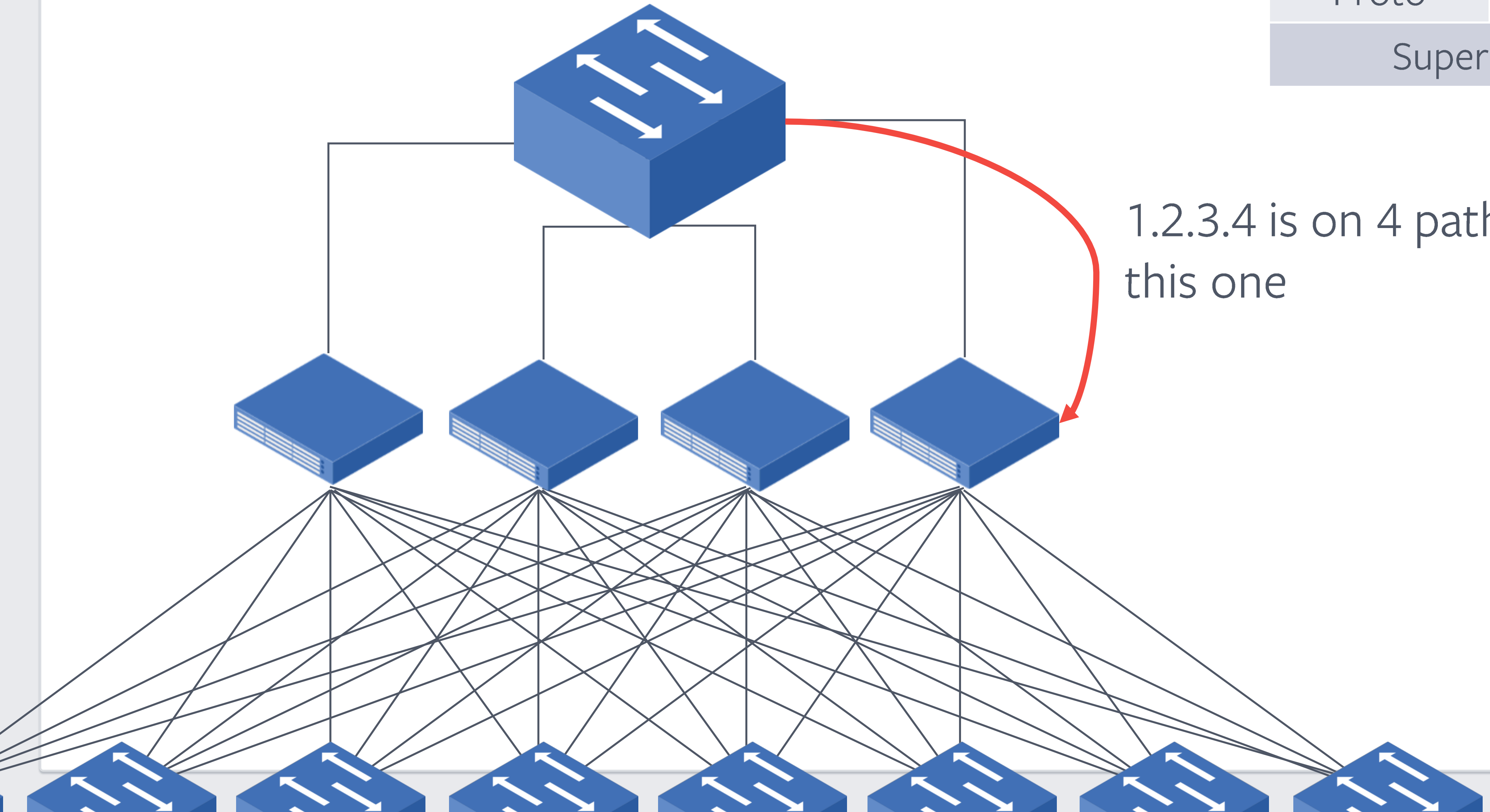


Packet

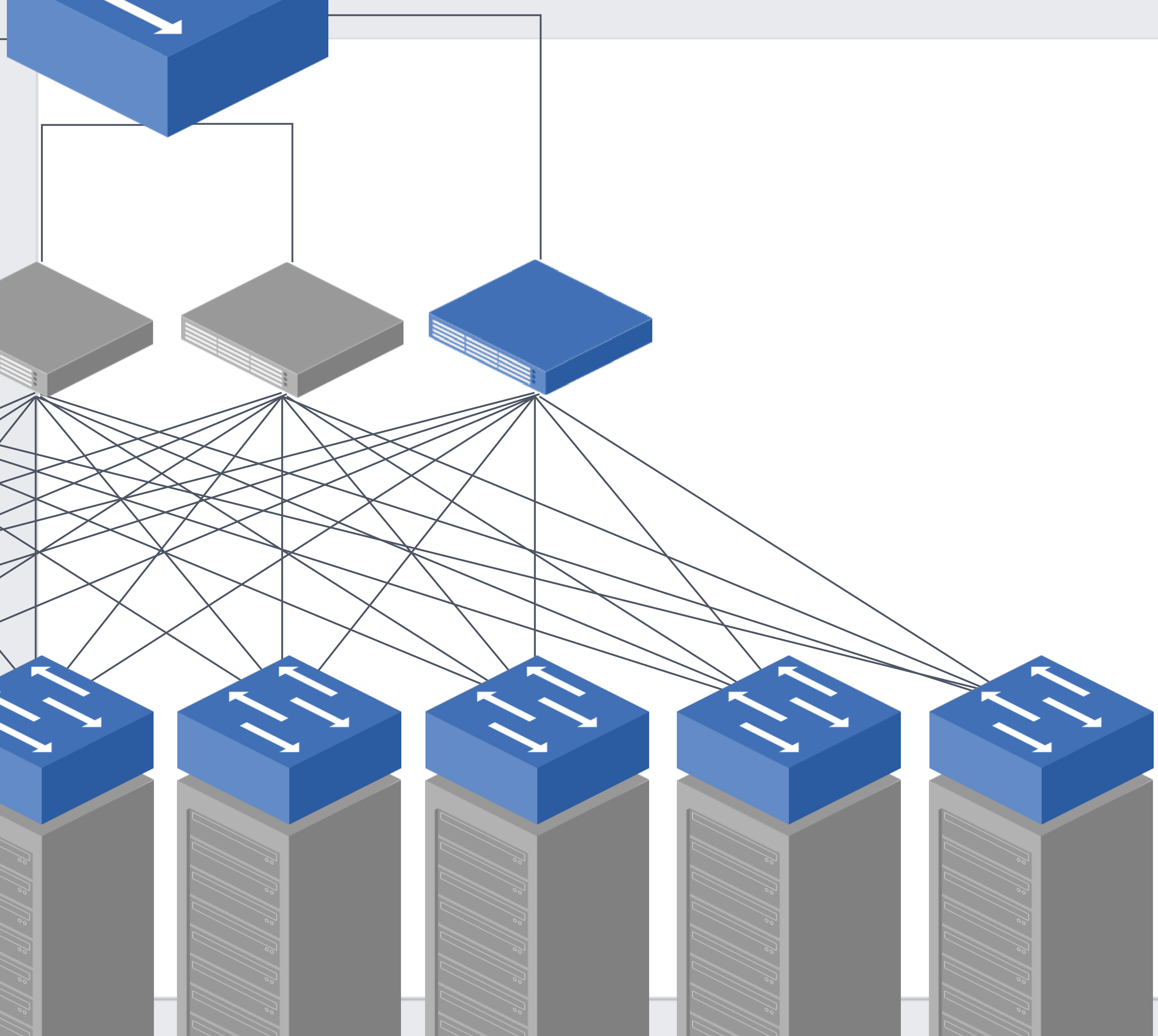
Src	162.8.7.6:12345
Dst	1.2.3.4:443
Proto	TCP
Super important data	

Packet

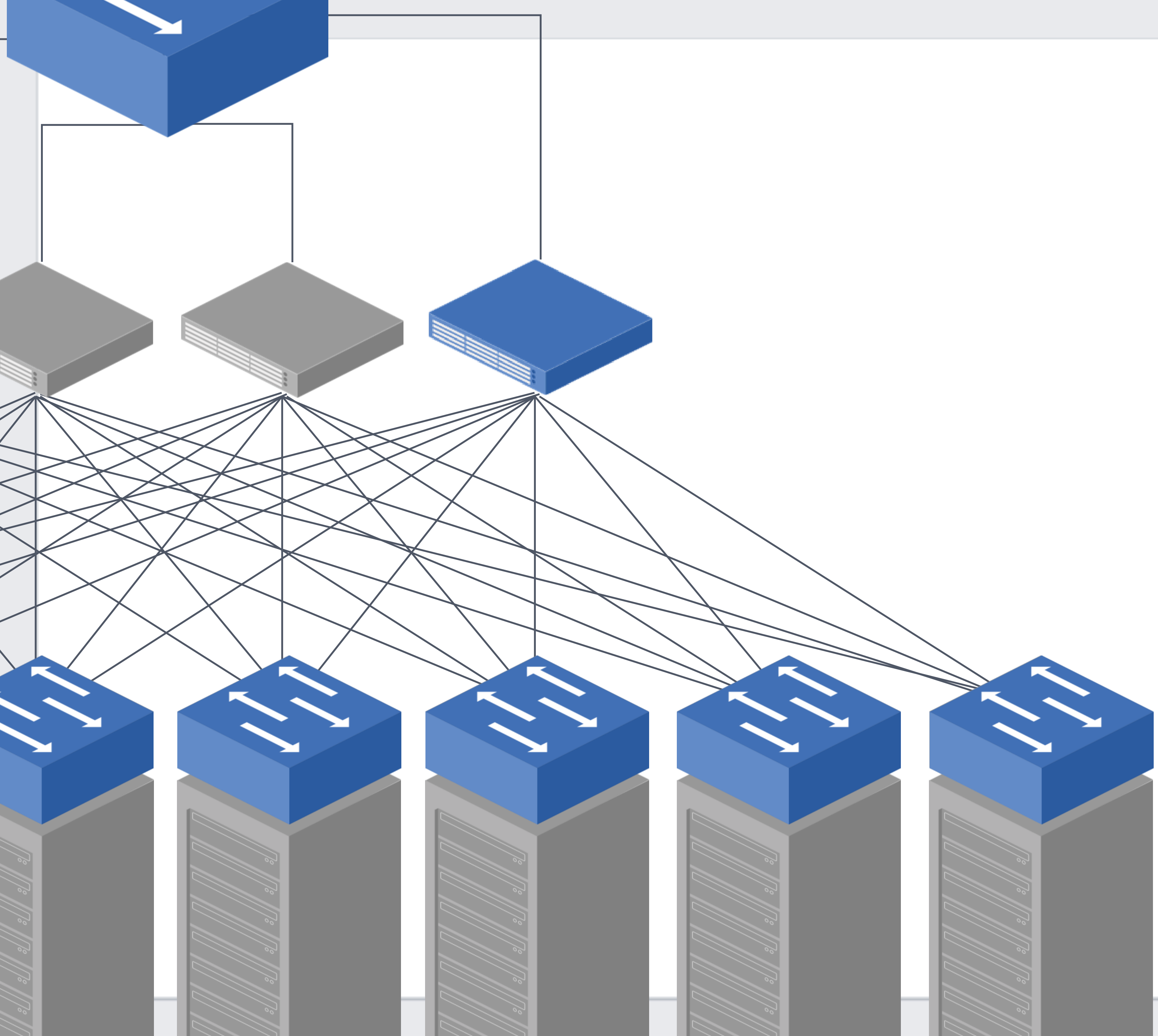
Src	162.8.7.6:12345
Dst	1.2.3.4:443
Proto	TCP
Super important data	



1.2.3.4 is on 4 paths, I’ll choose this one

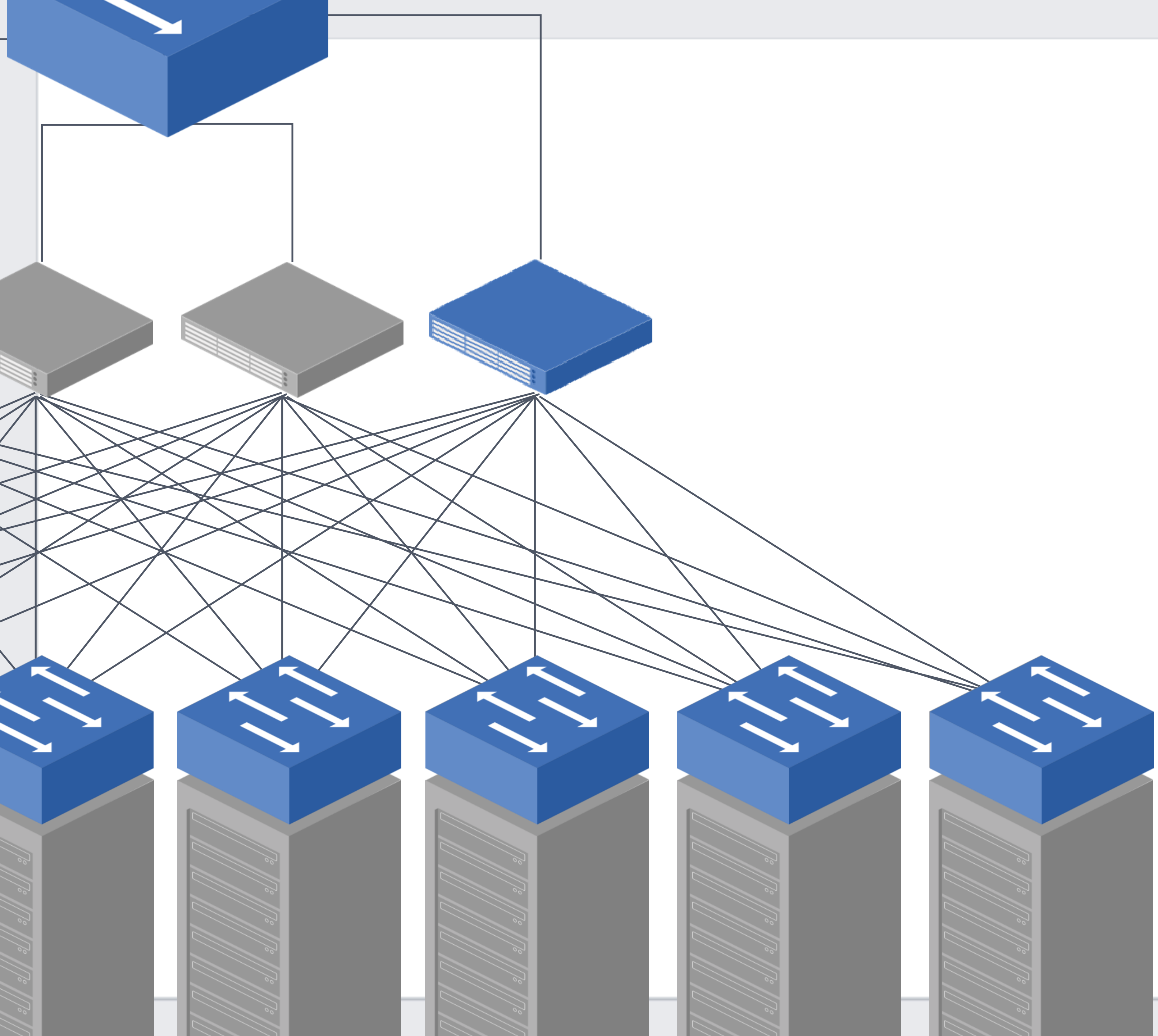


1.2.3.4 is for this set of backends...



1.2.3.4 is for this set of backends...

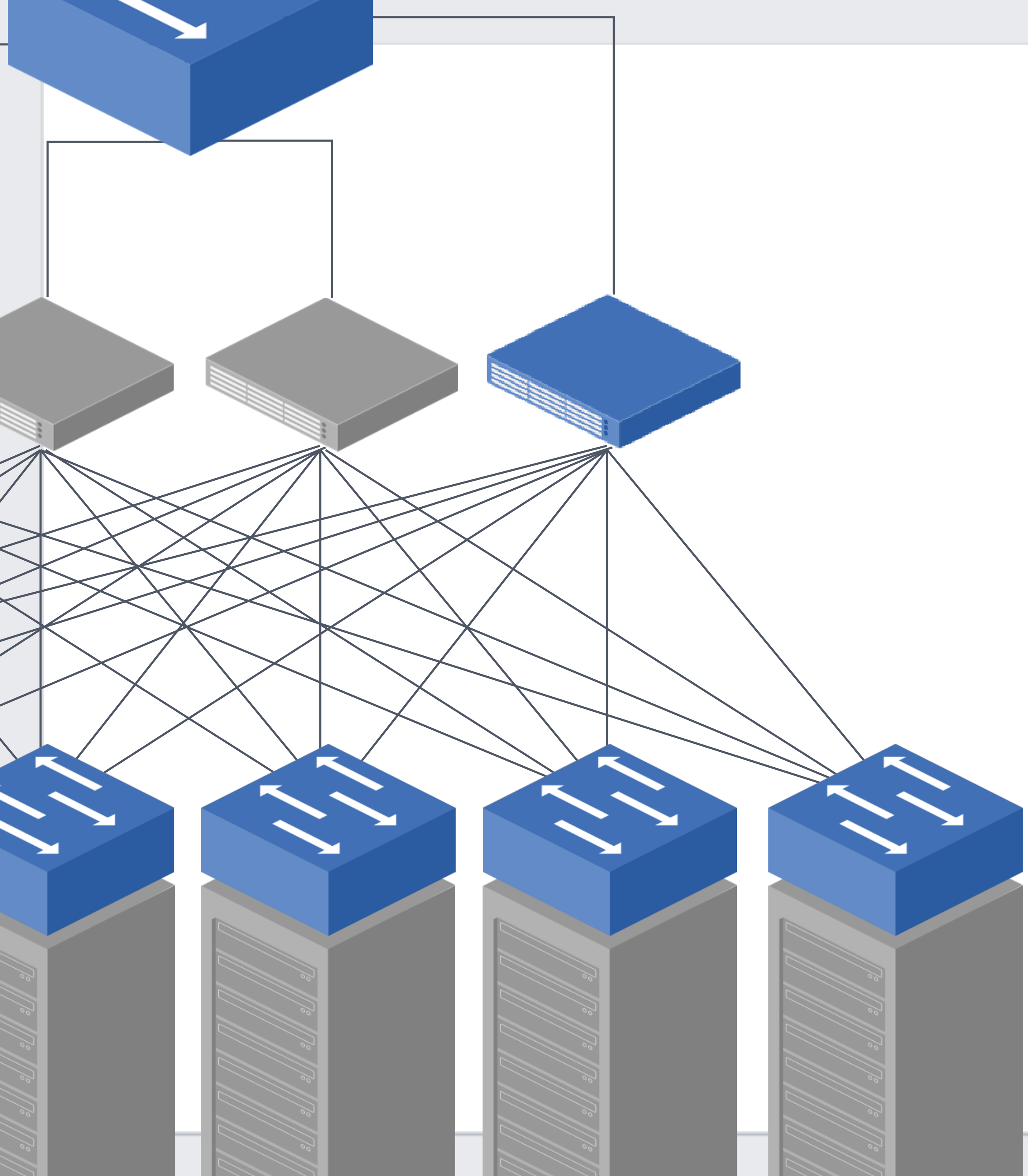
It's not in my LRU cache, recalculate the hash



1.2.3.4 is for this set of backends...

It's not in my LRU cache, recalculate the hash.

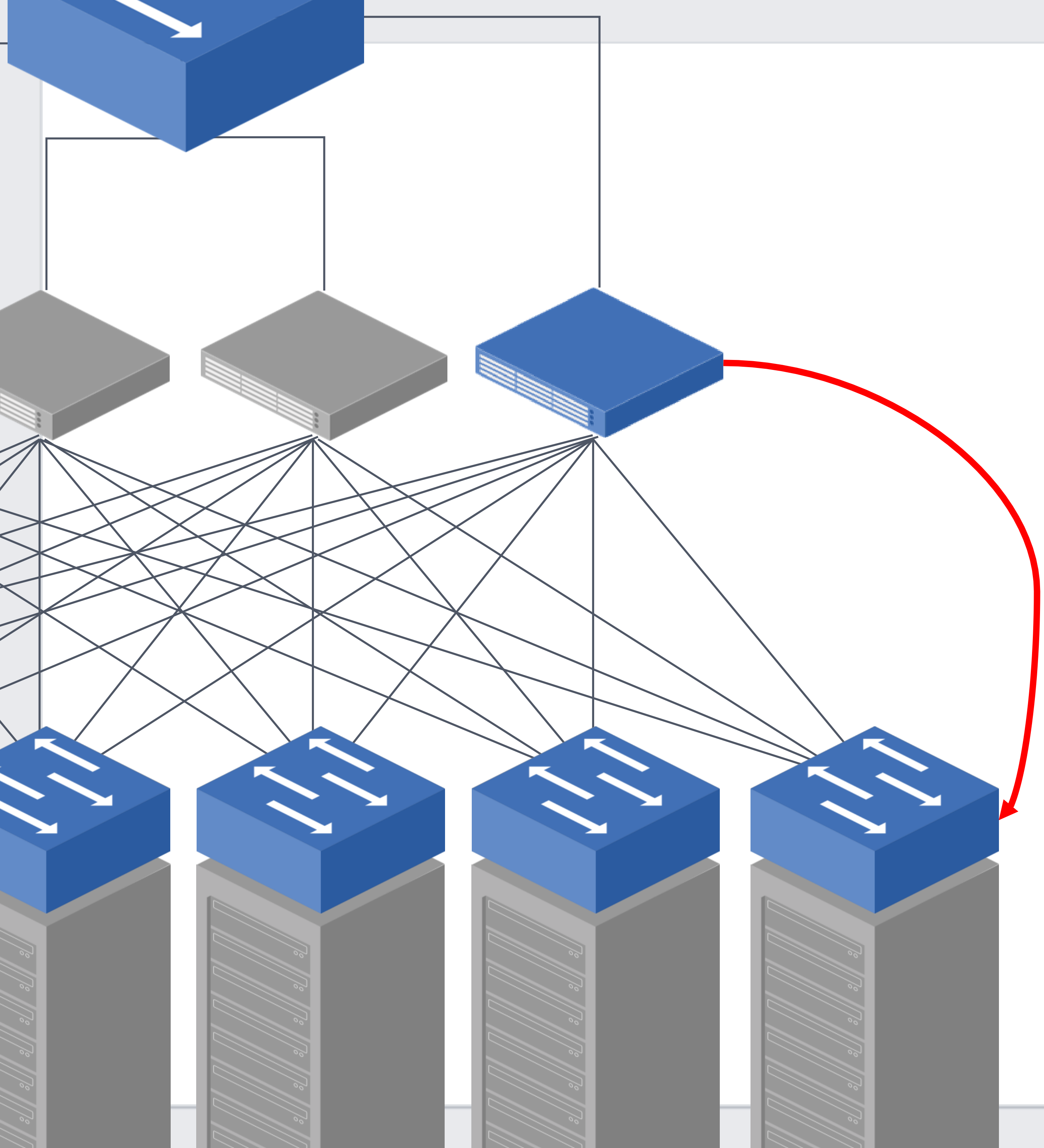
It should go to 2604::ae!



New headers!

Src	2604::42:12345
Dst	2604::ae:443
Proto	TCP
Src	162.8.7.6:12345
Dst	1.2.3.4:443
Proto	TCP

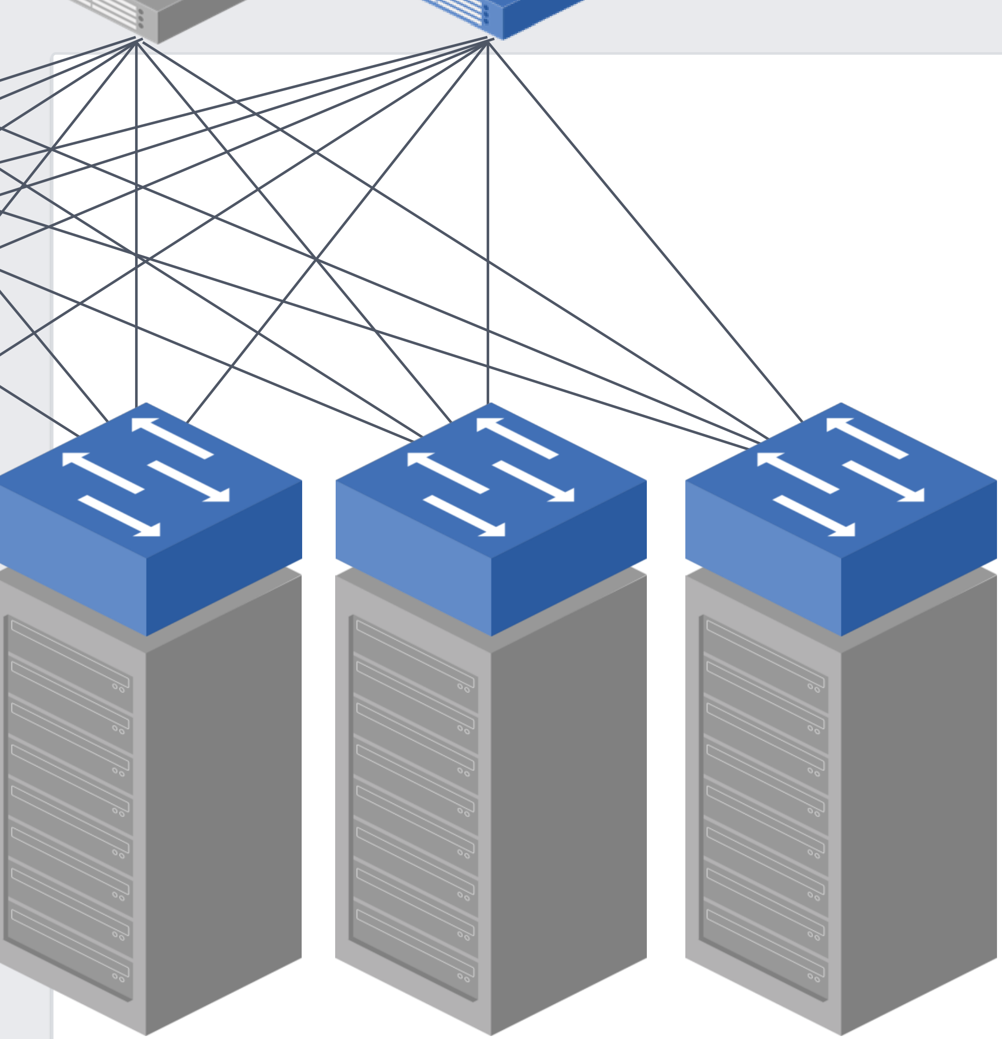
Super important data



Packet

Src	2604::42:12345
Dst	2604::ae:443
Proto	TCP
Src	162.8.7.6:12345
Dst	1.2.3.4:443
Proto	TCP

Super important data



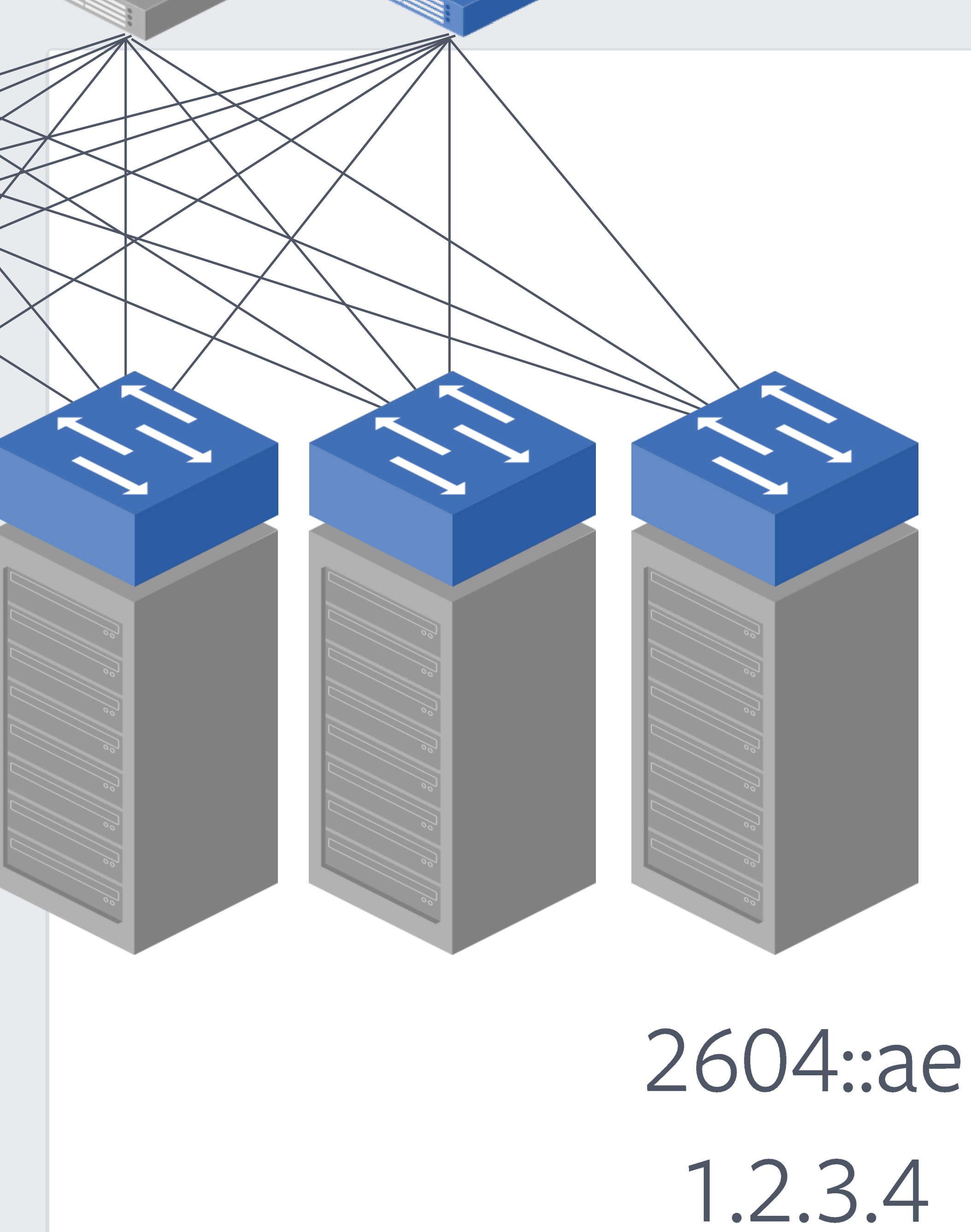
2604::ae

1.2.3.4

Packet

Src	2604::42:12345
Dst	2604::ae:443
Proto	TCP
Src	162.8.7.6:12345
Dst	1.2.3.4:443
Proto	TCP

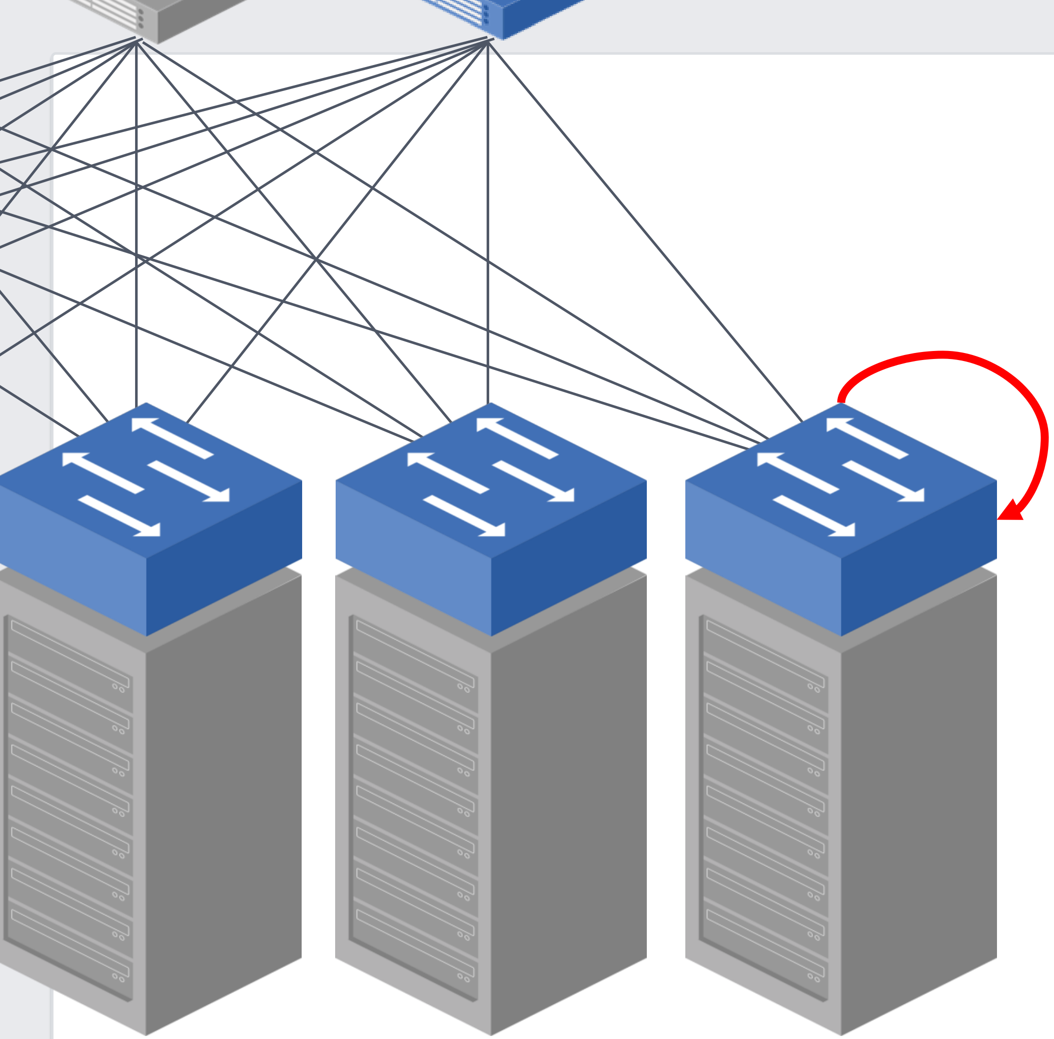
Super important data



Packet

Src	162.8.7.6:12345
Dst	1.2.3.4:443
Proto	TCP

Super important data



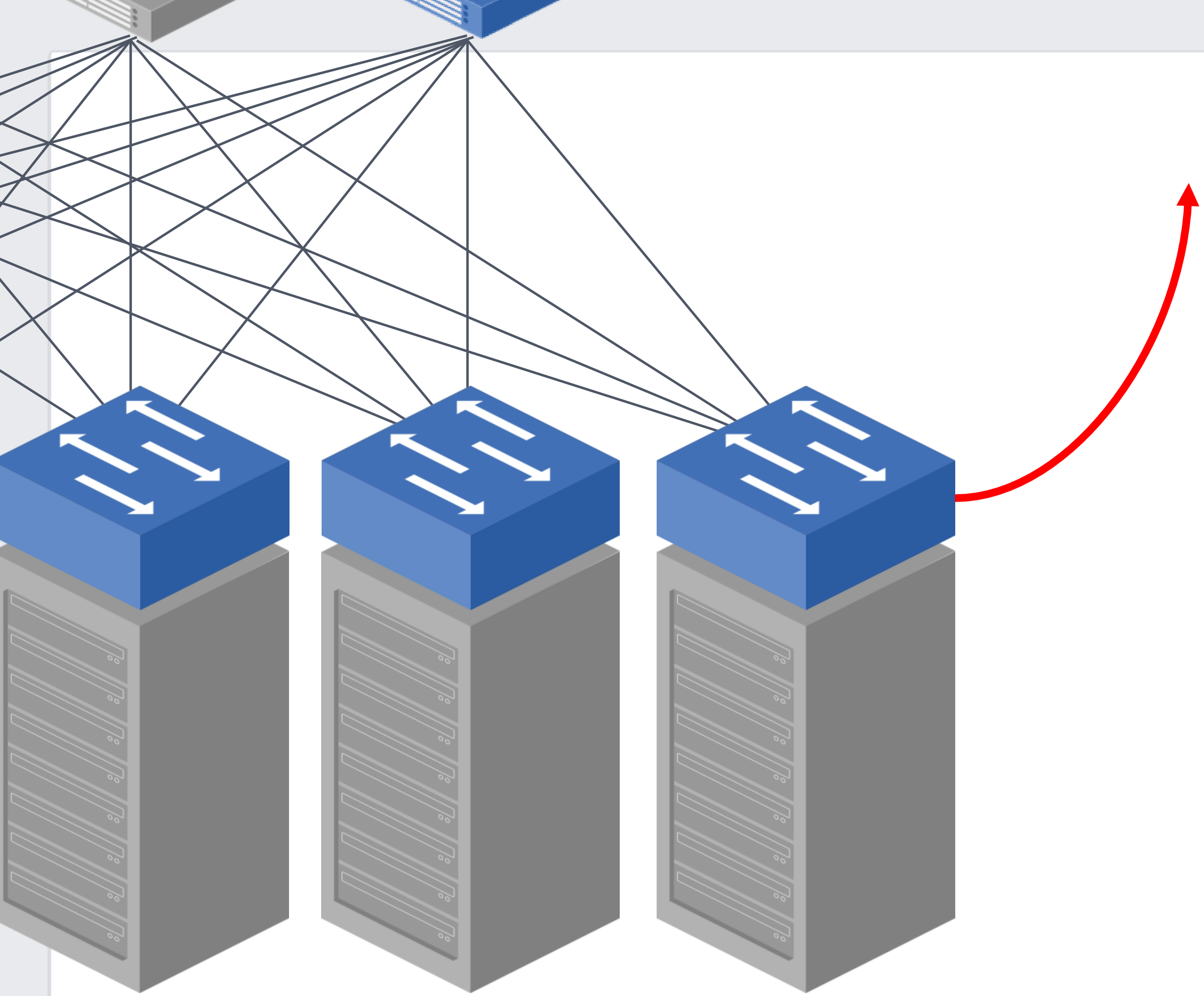
2604::ae

1.2.3.4

Packet

Src	162.8.7.6:12345
Dst	**1.2.3.4**:443
Proto	TCP

Super important data



2604::ae

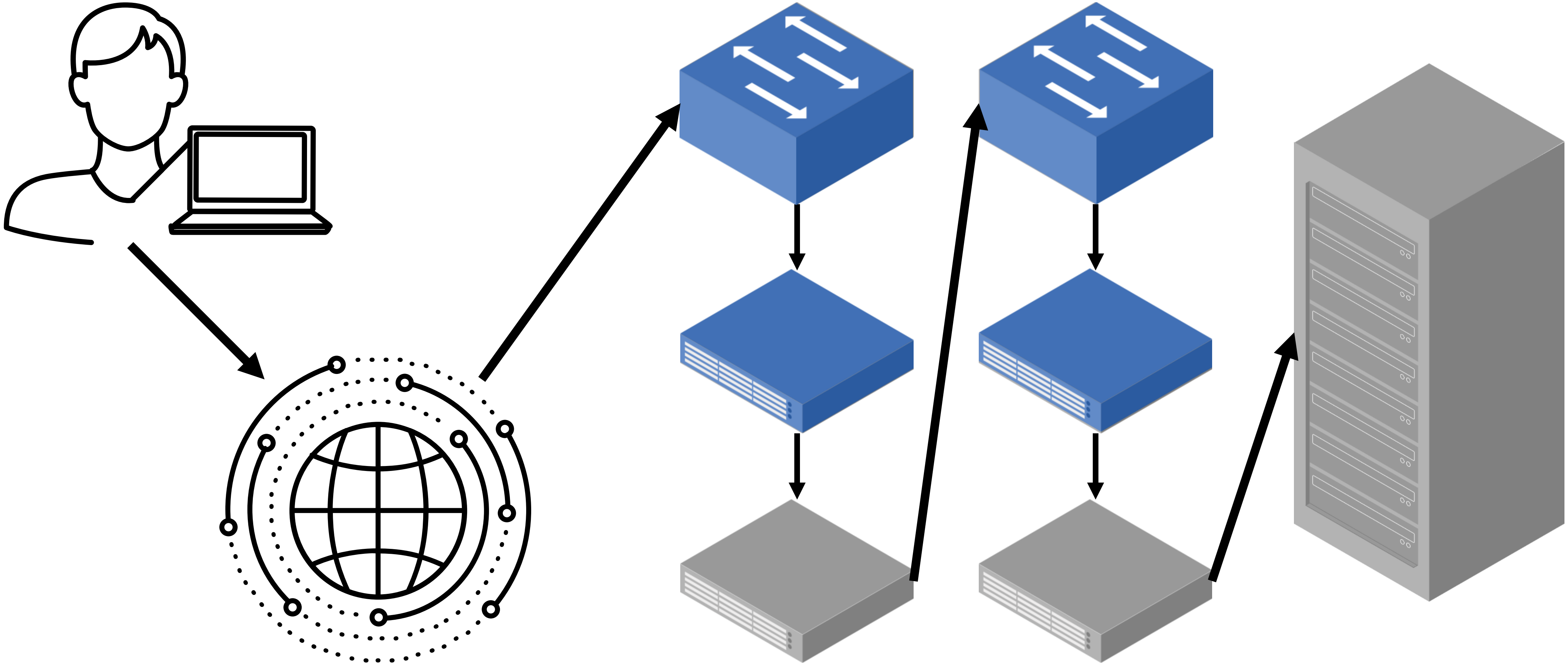
1.2.3.4

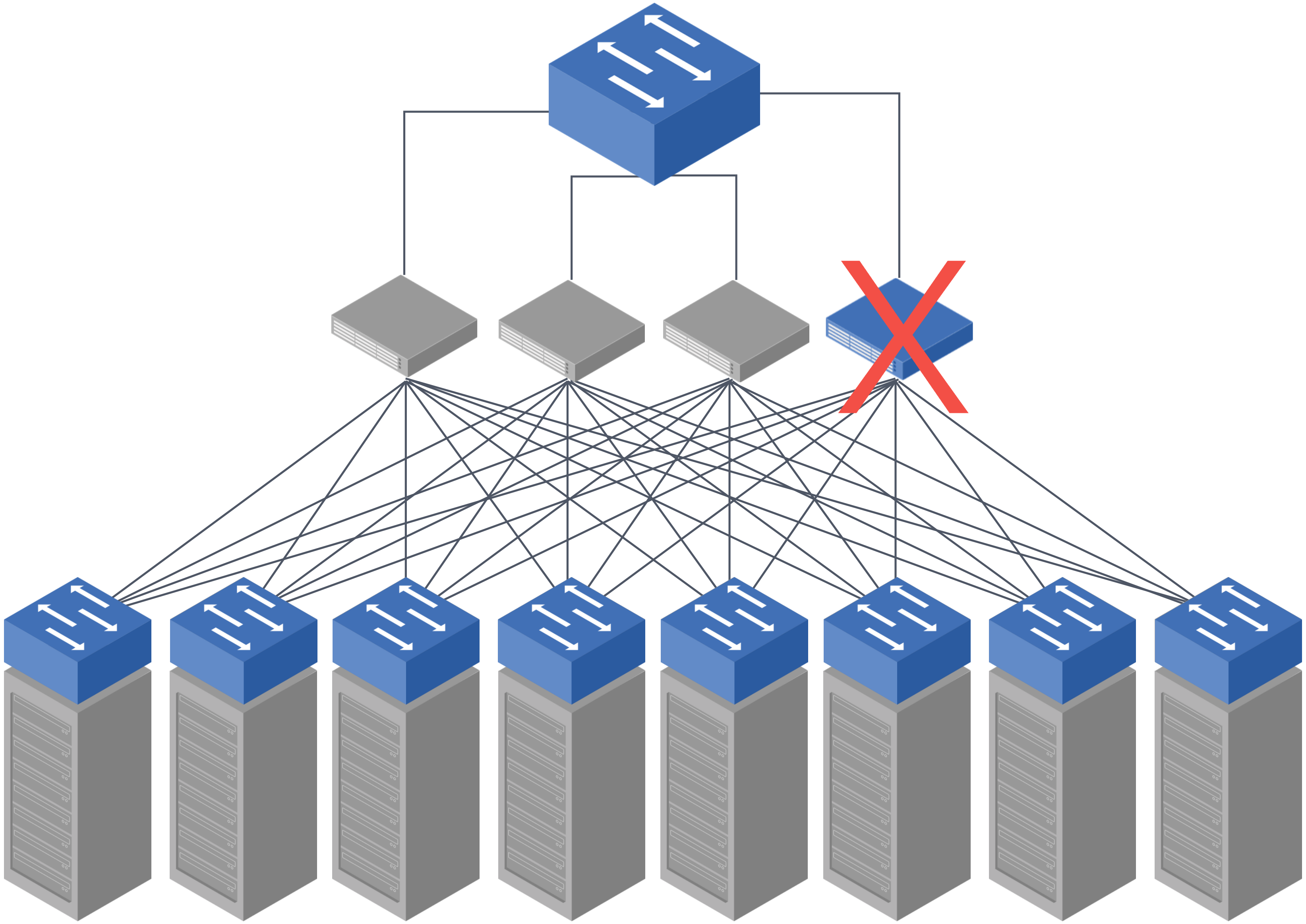
Packet

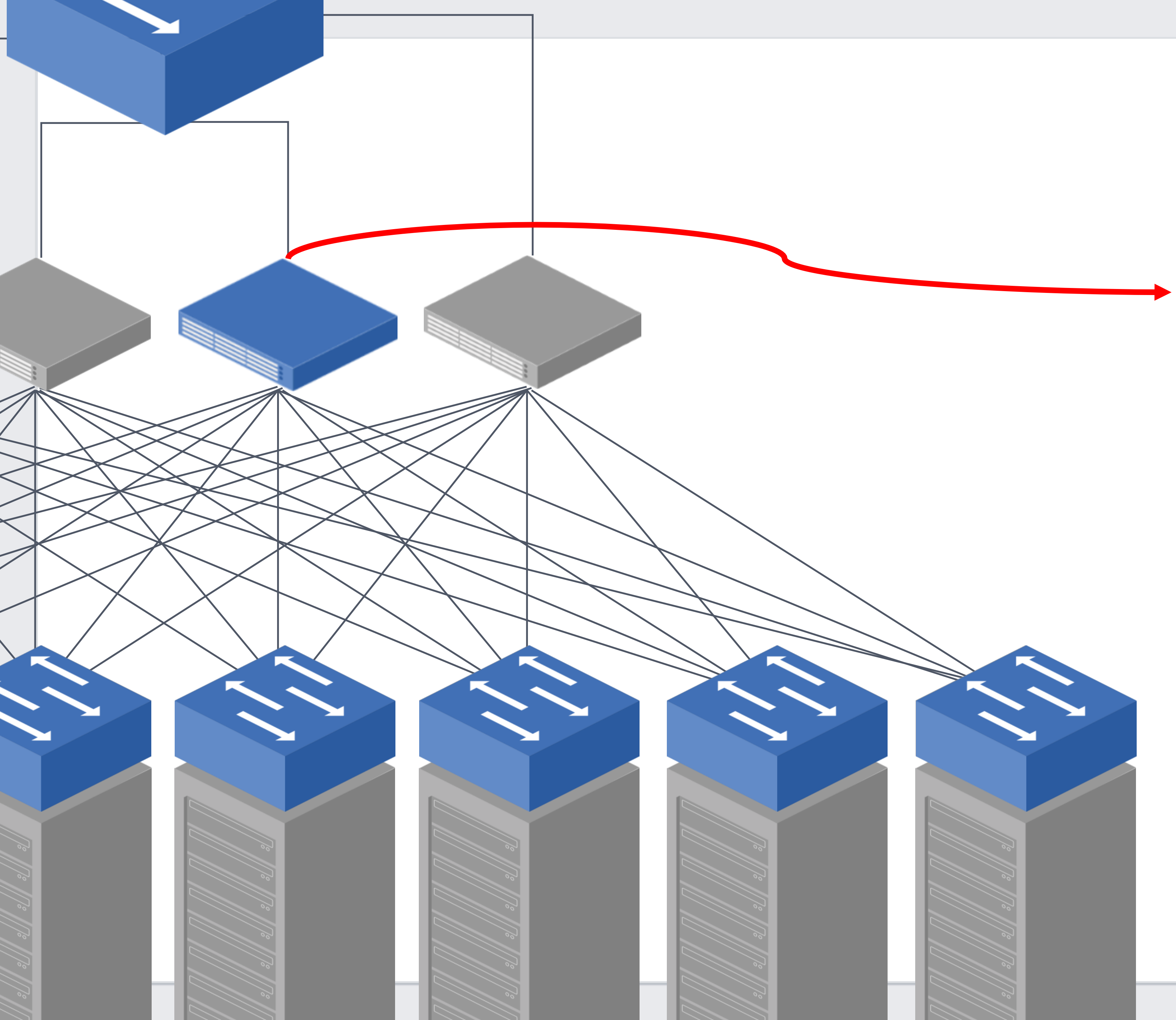
Src	**1.2.3.4**:443
Dst	162.8.7.6:12345
Proto	TCP

Super important data

Where are we?

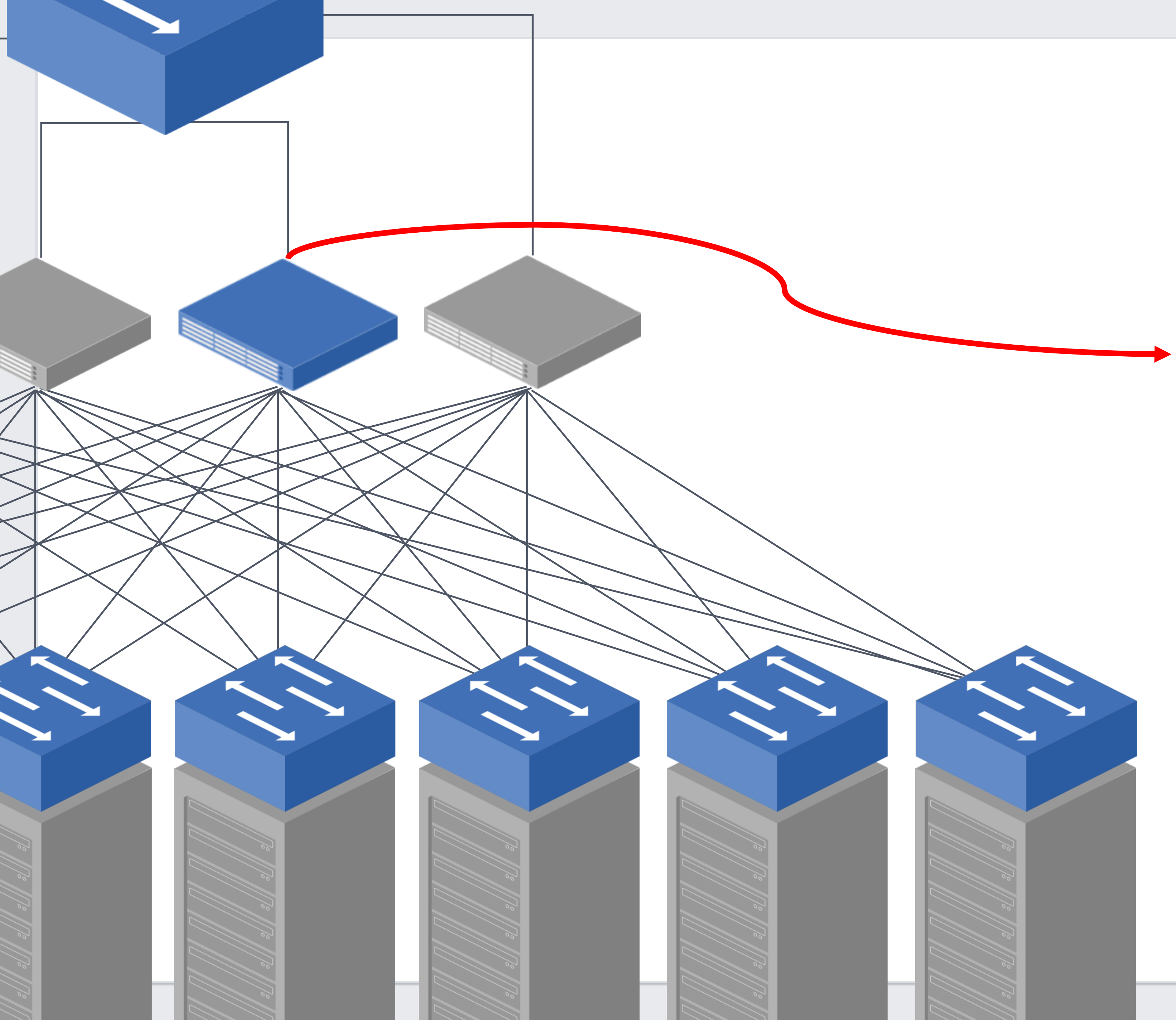






1.2.3.4 is for this set of backends…

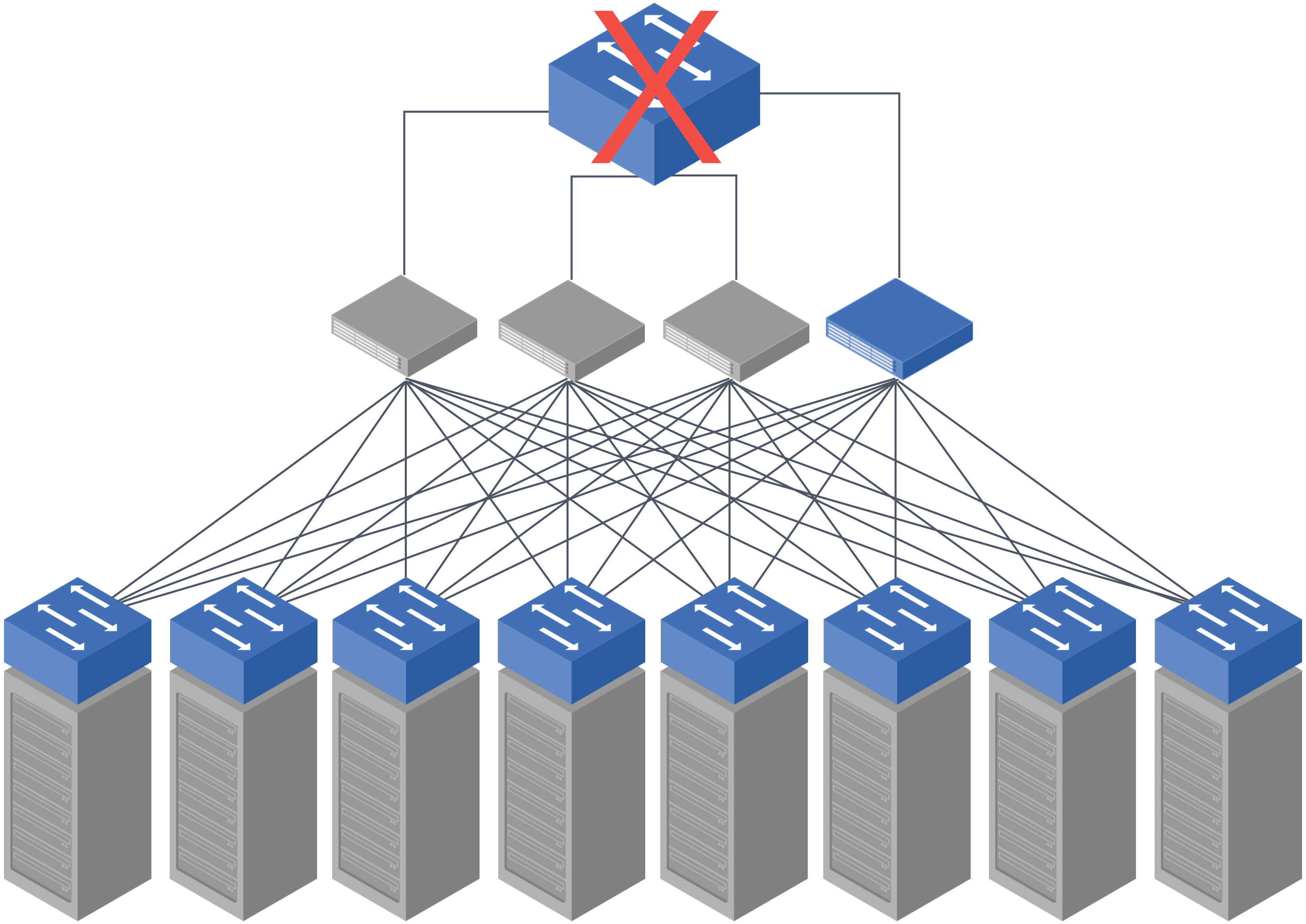
It’s not in my LRU cache, recalculate the hash

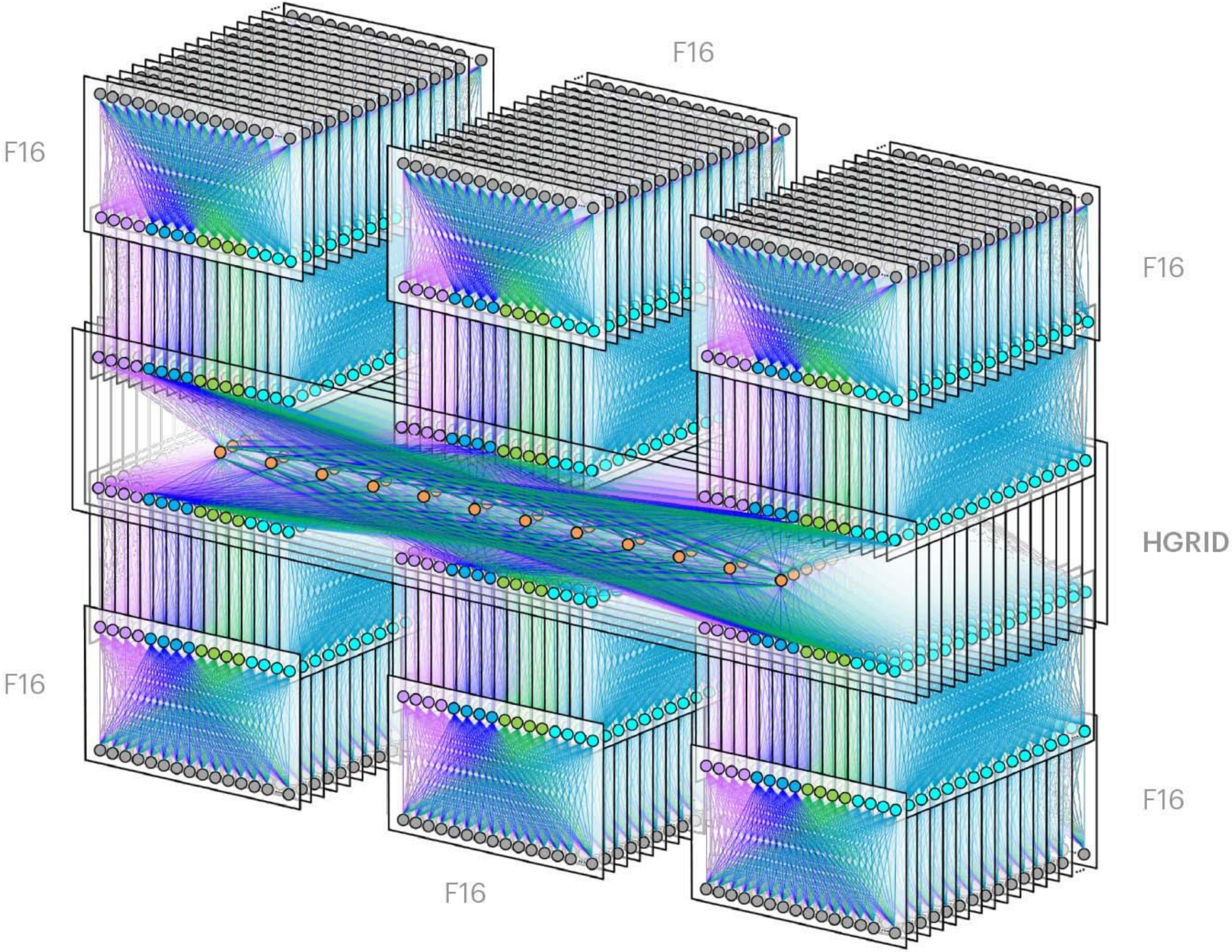


1.2.3.4 is for this set of backends...

It’s not in my LRU cache, recalculate the hash

It should go to 2604::ae!





https://engineering.fb.com/data-center-engineering/f16-minipack/

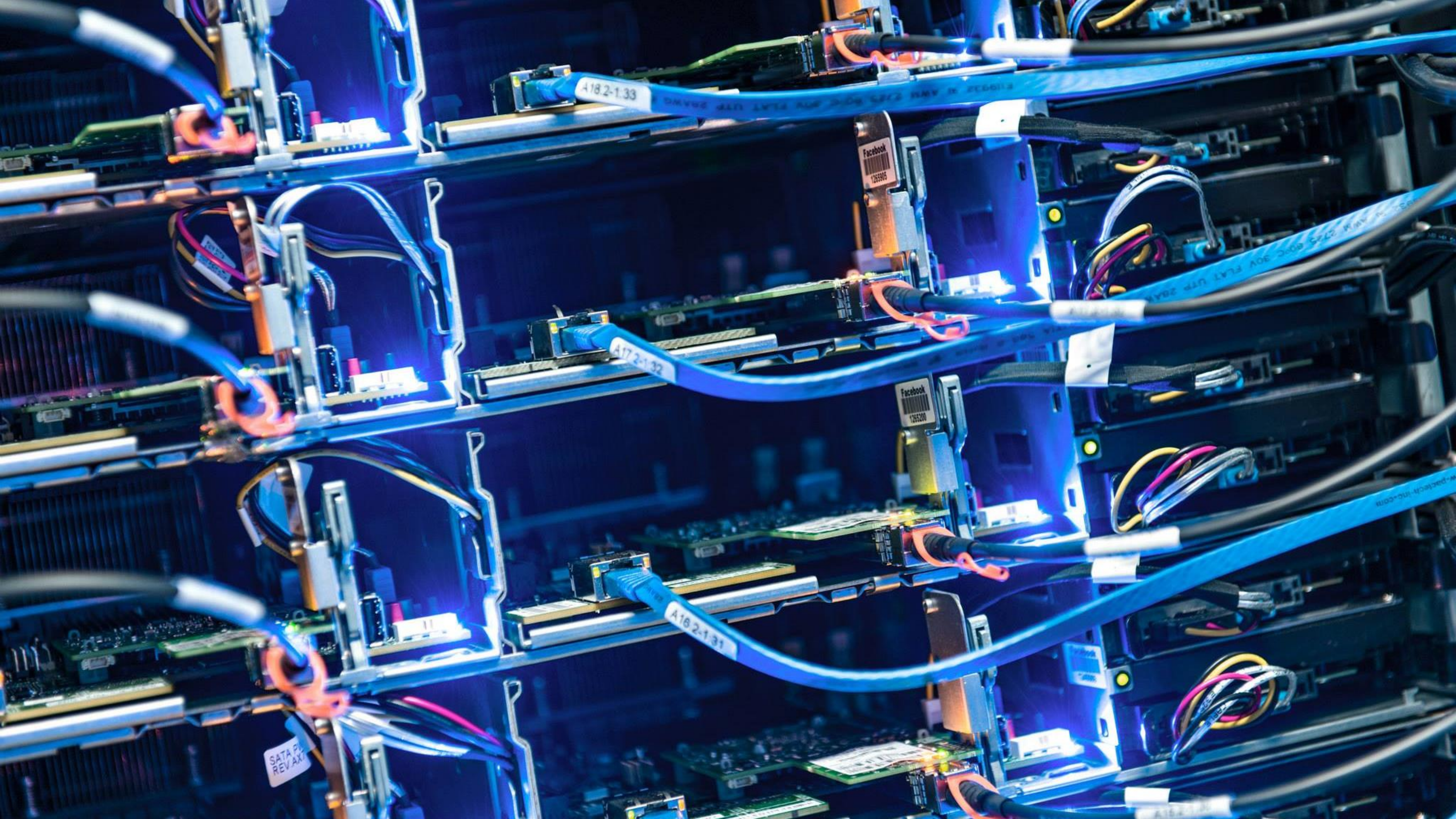
Differences with ‘standard’ L4

- Remove a single point of failure
- Replaced with requirement to keep backend state roughly in common
- Invasive changes to network
    - Network needs to support multiple machines announcing same IP
    - Backend systems need to add same IP on loopback

Special sauce

- Katran on github - facebookincubator/katran
- eBPF + eXpress Data Path means packets get processed as early as possible

- Open sourcing blog post (link) + conference talk (link)

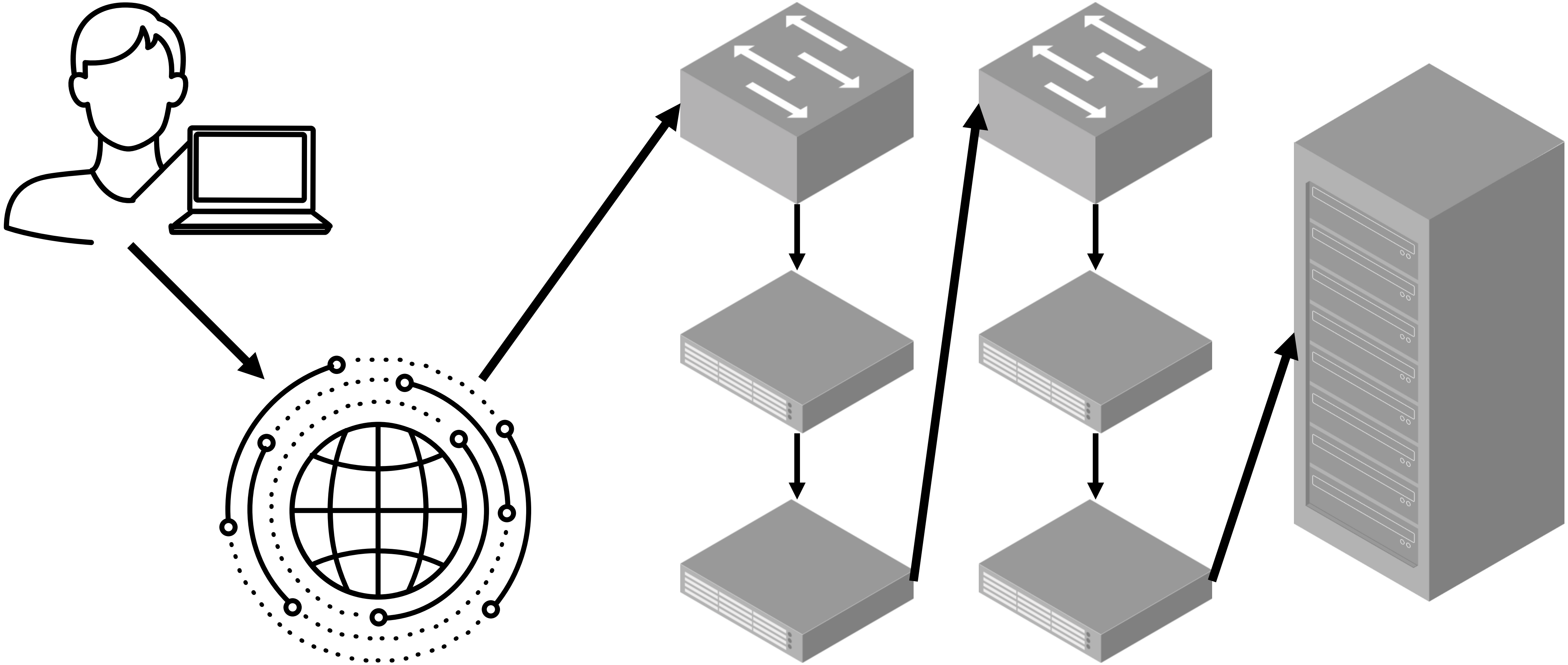

A18.2-1:33
Facebook
A17.2-1:32
Facebook
A16.2-1:31
SATA P
REV AX

Comparison

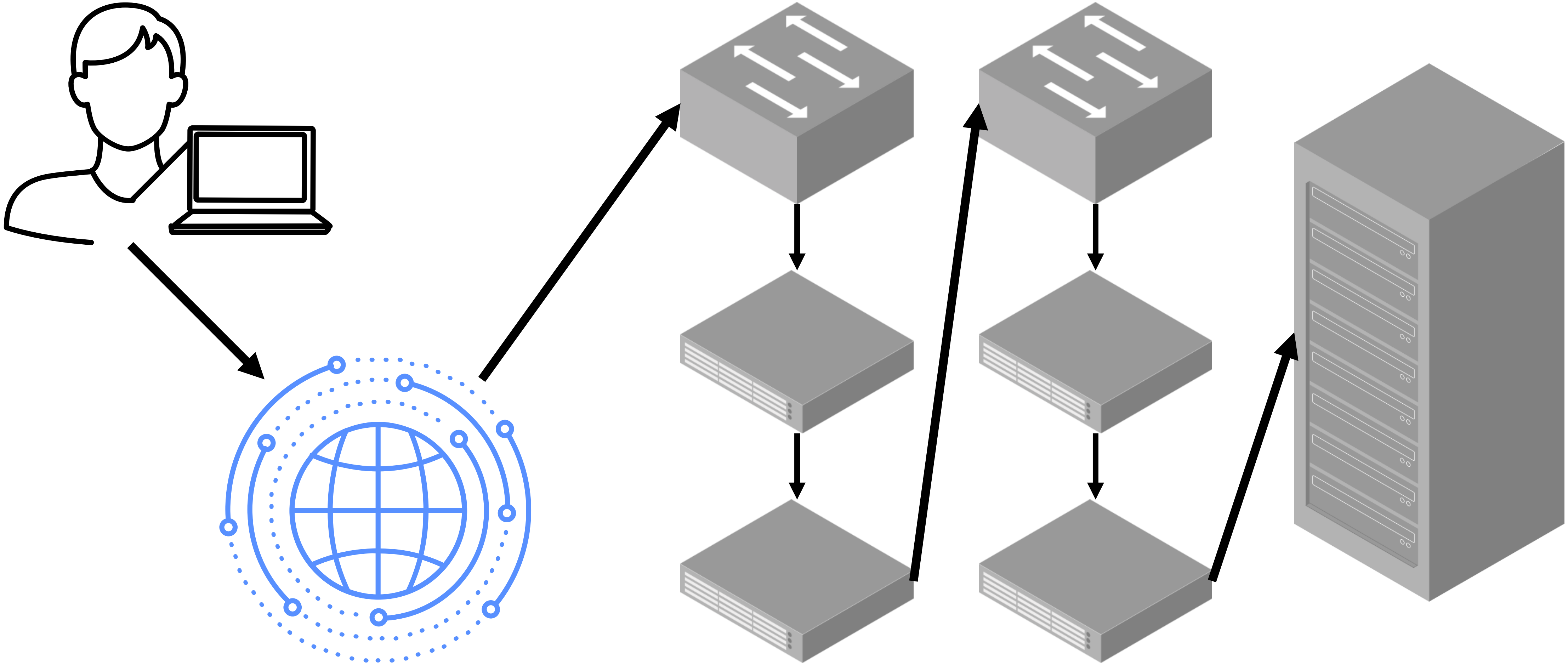
- Many L7 Load Balancers
  - Maintain IPs and DNS records for each?
- DNS
  - MagLev reacts quicker to a dead L7

Putting it together

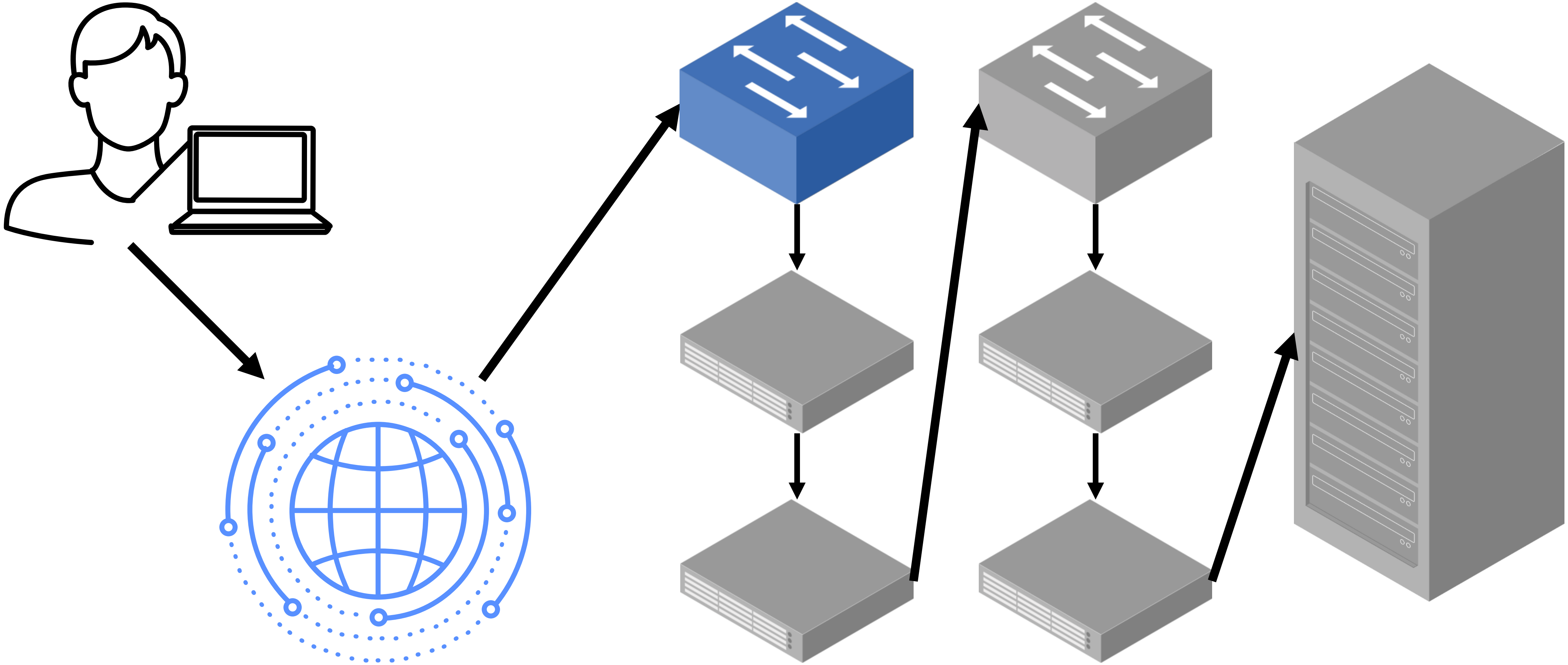
What happens when I type “facebook.com” and hit enter



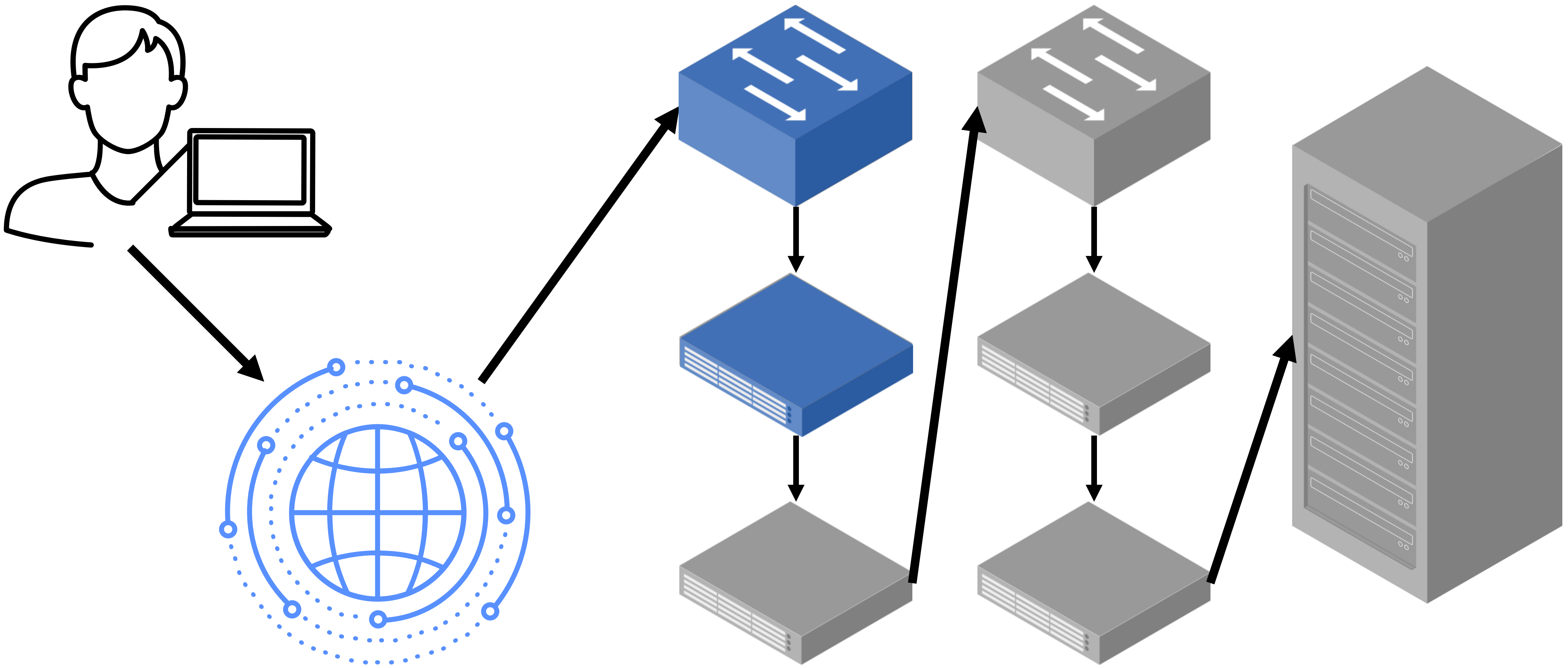
What happens when I type “facebook.com” and hit enter



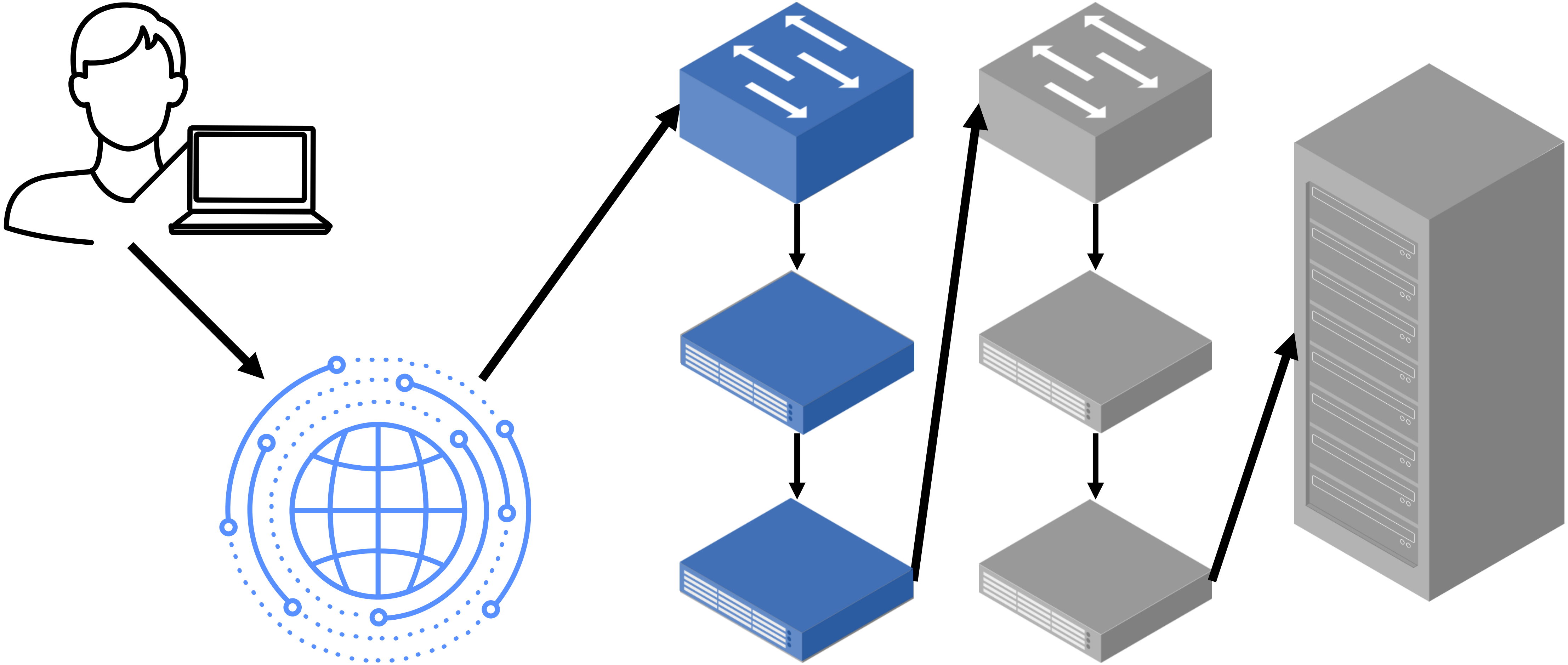
What happens when I type “facebook.com” and hit enter



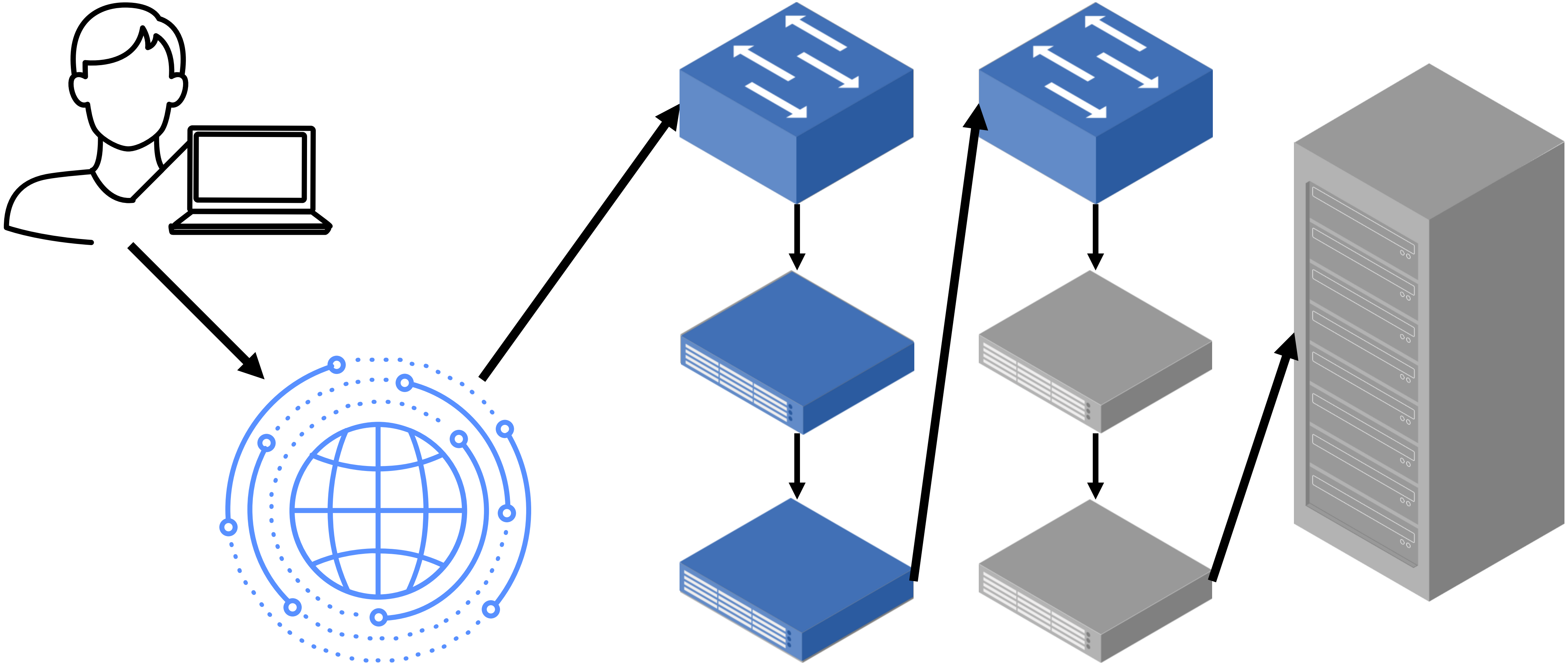
What happens when I type “facebook.com” and hit enter



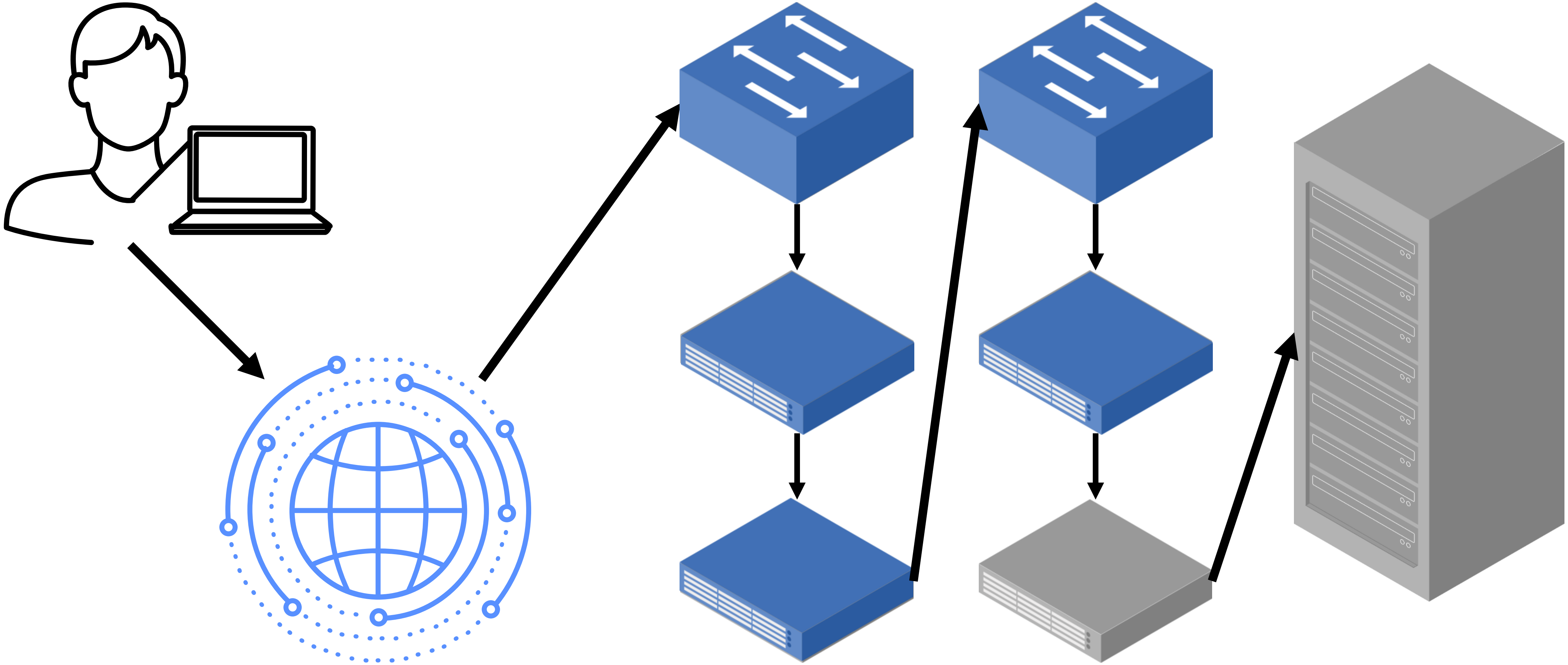
What happens when I type “facebook.com” and hit enter



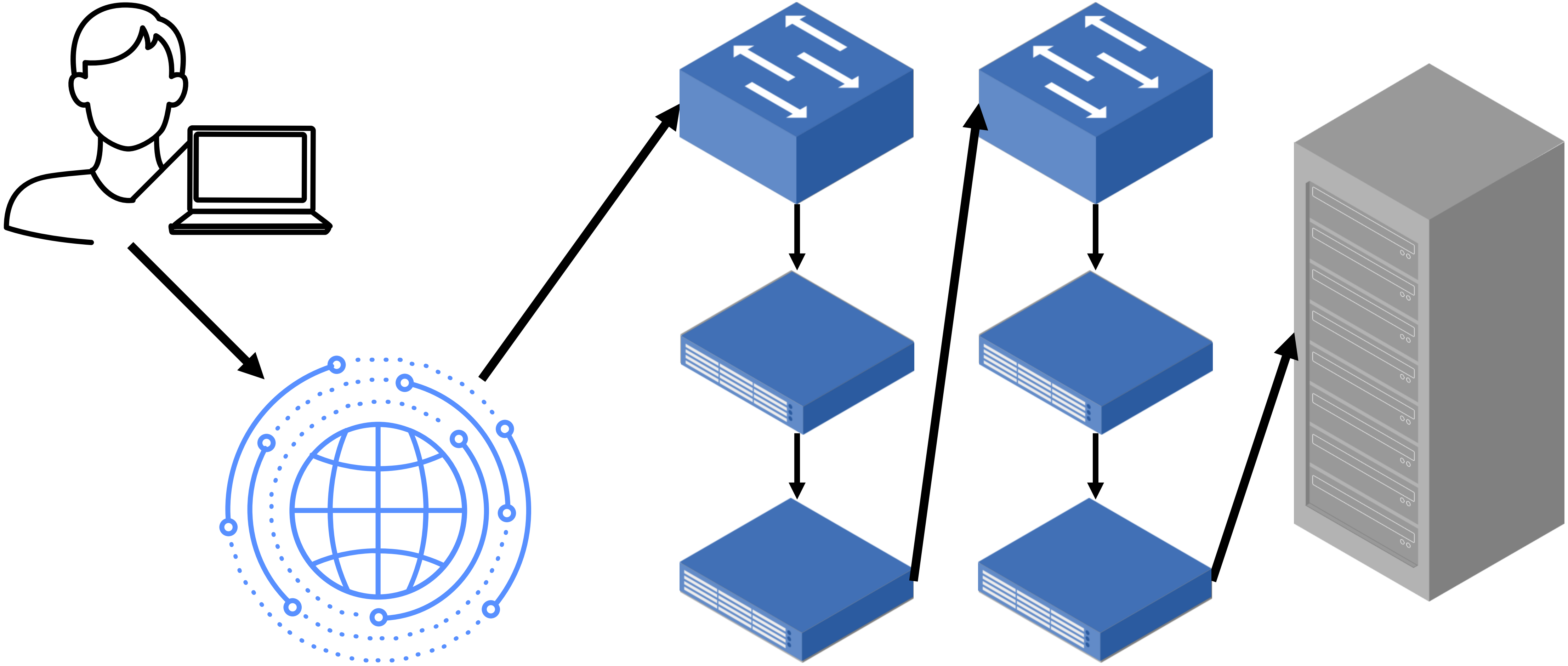
What happens when I type “facebook.com” and hit enter



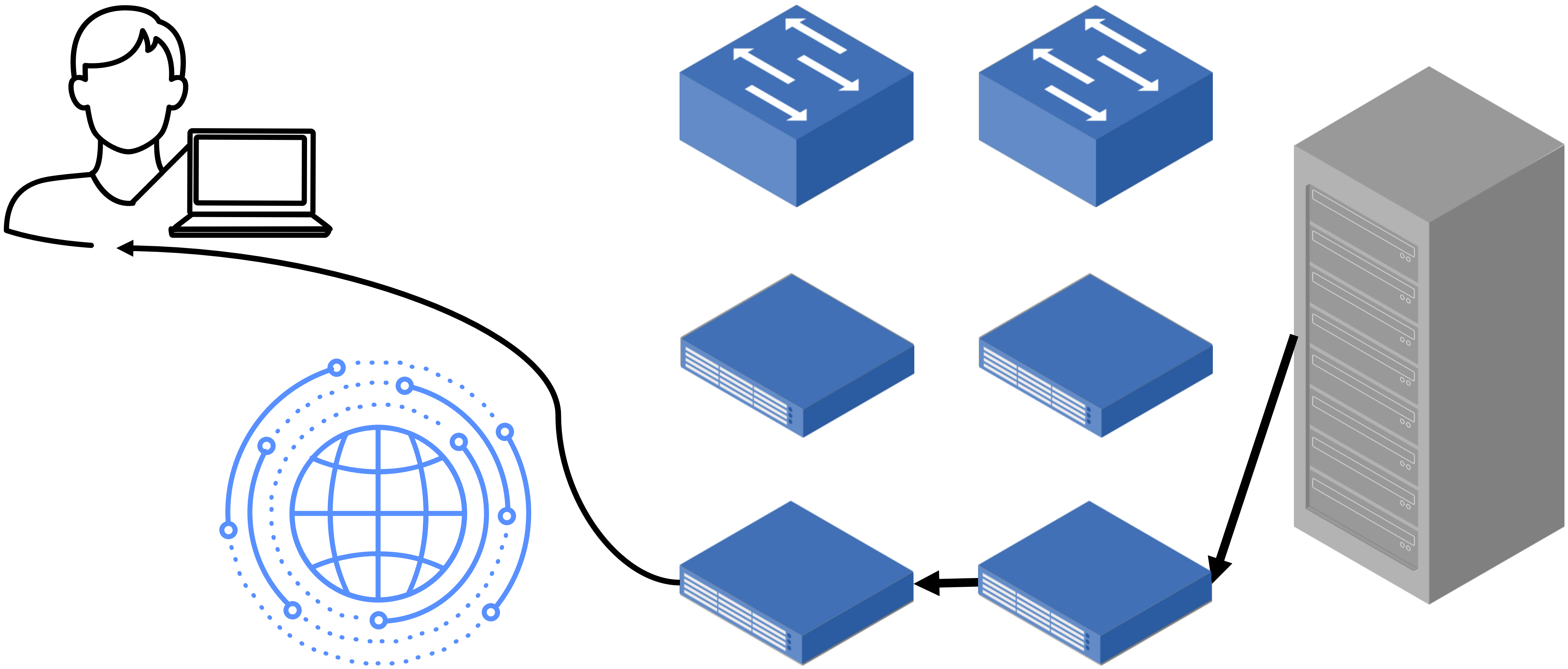
What happens when I type “facebook.com” and hit enter



What happens when I type “facebook.com” and hit enter



What happens when I type “facebook.com” and hit enter



Wrapping up

- Introduction to network load balancers
- Know what classes of load balancer is appropriate for common usecases
- Know some attributes of network architecture at scale

facebook

Thank You