
facebook
INFRASTRUCTURE

Building a Billion User Load Balancer

Mikel Jimenez
Network Engineer, Network Datacenter Engineering

facebook

about: me

- Originally from Bilbao
- Joined Facebook in 2012, living in Dublin since 2011
  - Network Infrastructure Engineering
  - Bootstrap Network Datacenter Engineering in Europe
- Previously doing networking stuff @{Ibermatica,Amazon}

Agenda

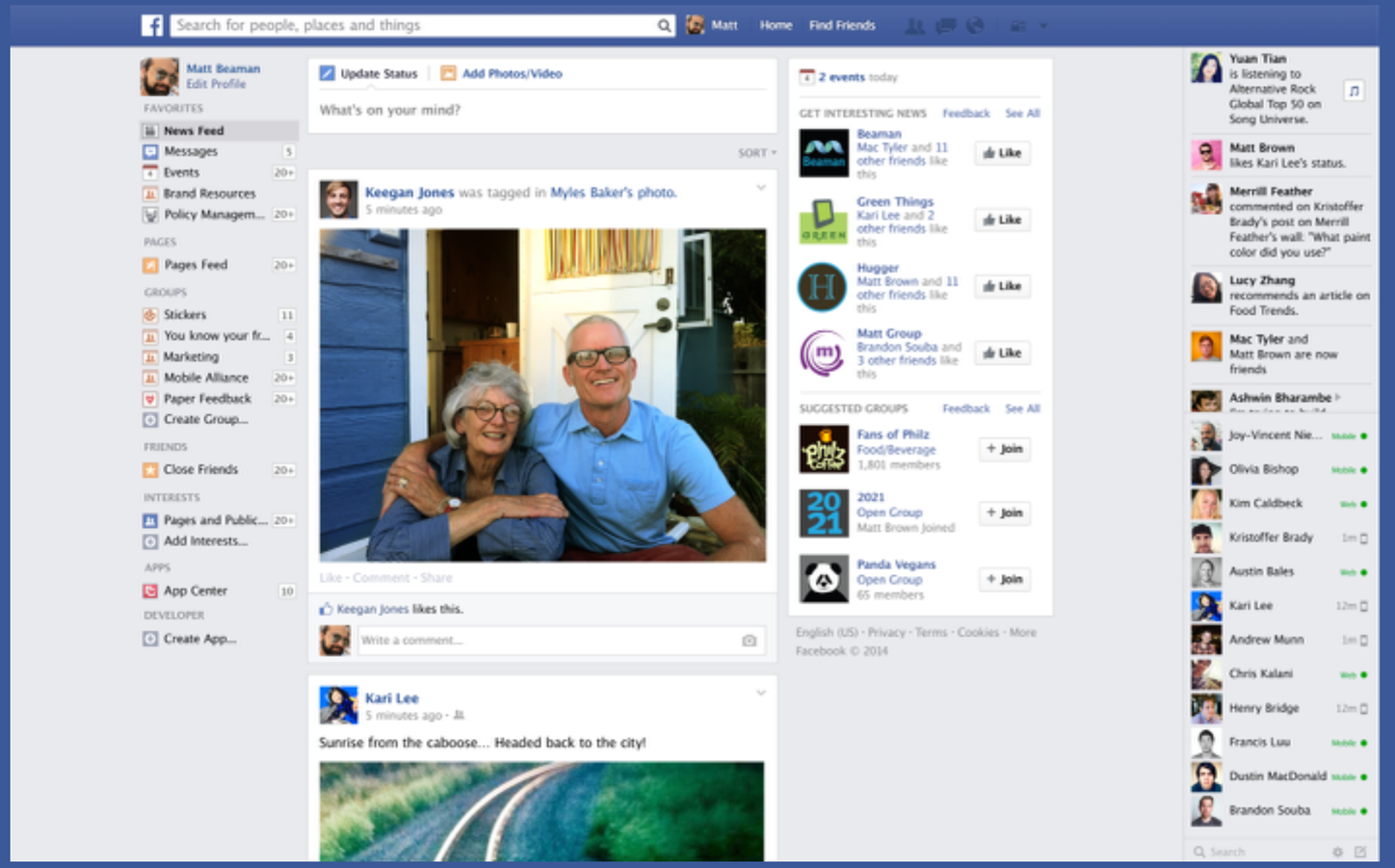
1	Serving Dynamic Facebook Requests
2	Load Balancing: L4/L7
3	Edge PoP’s & Reducing Latency
4	Global Load Balancing - DNS
5	DataCenter Networking

Facebook Traffic Overview

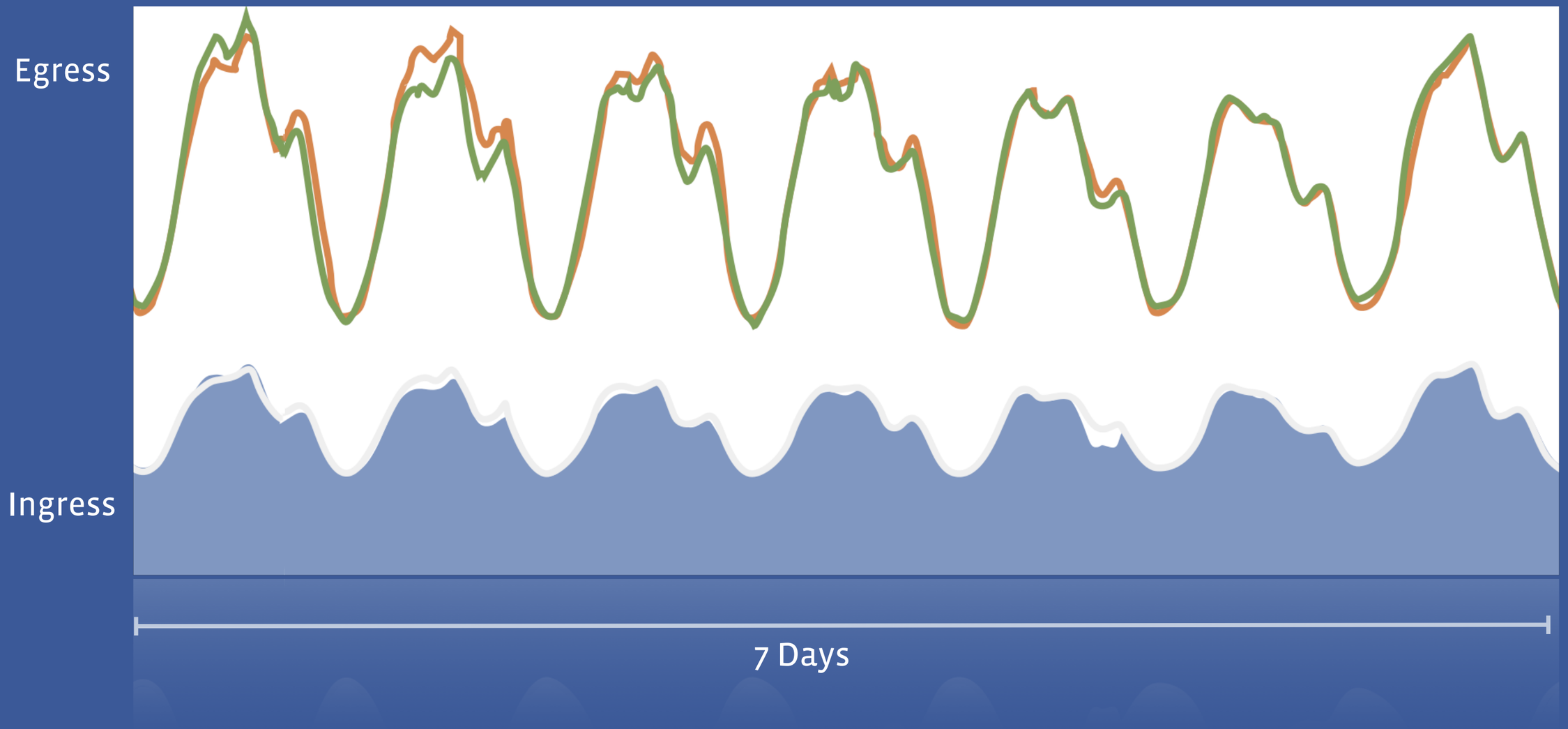
What is Facebook?

Facebook is mainly composed of two request types:

- Dynamic requests
  - newsfeed
  - likes
  - status updates
- Static requests
  - Images
  - Video
  - js/css

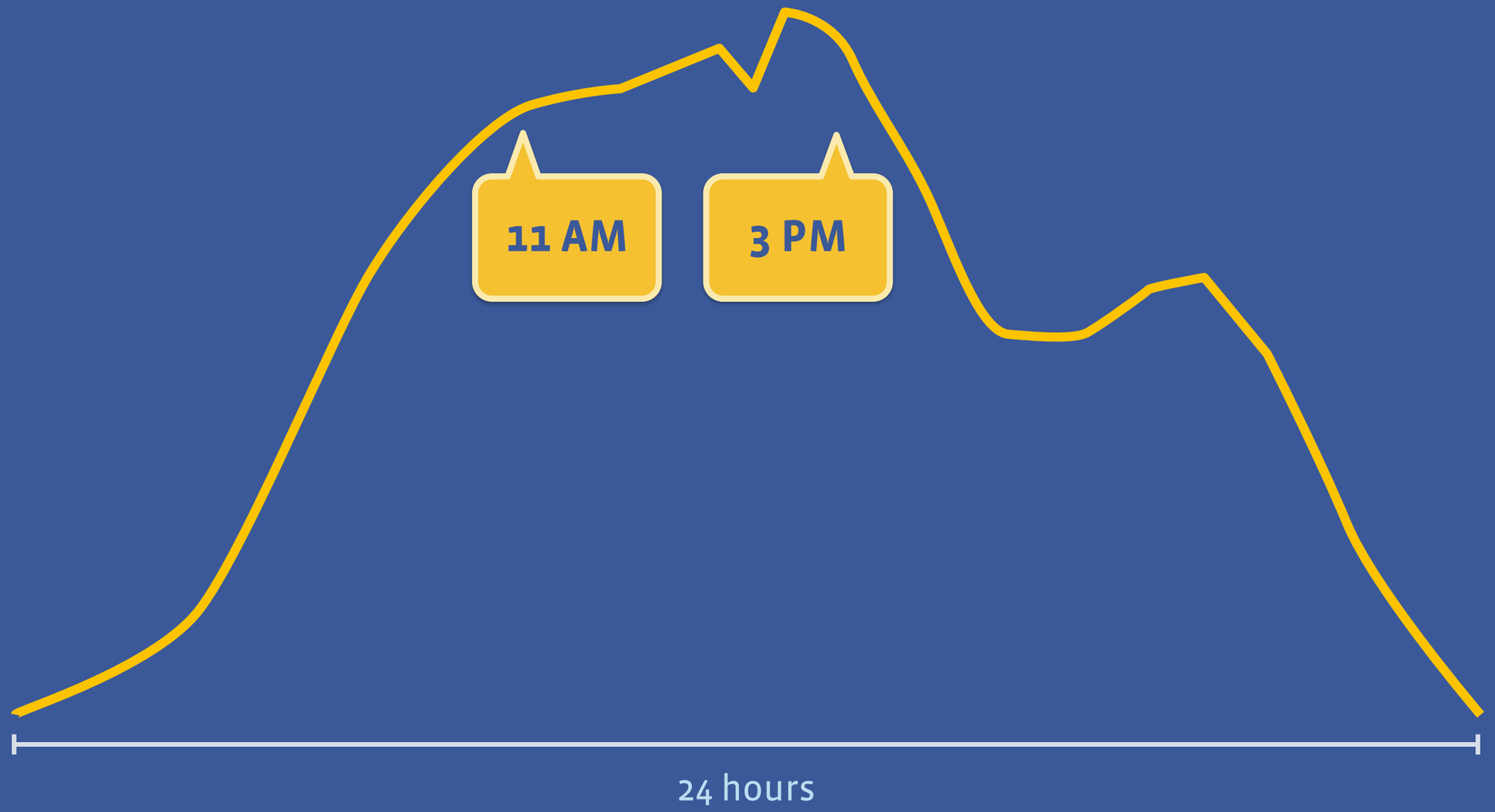


Weekly Cycle

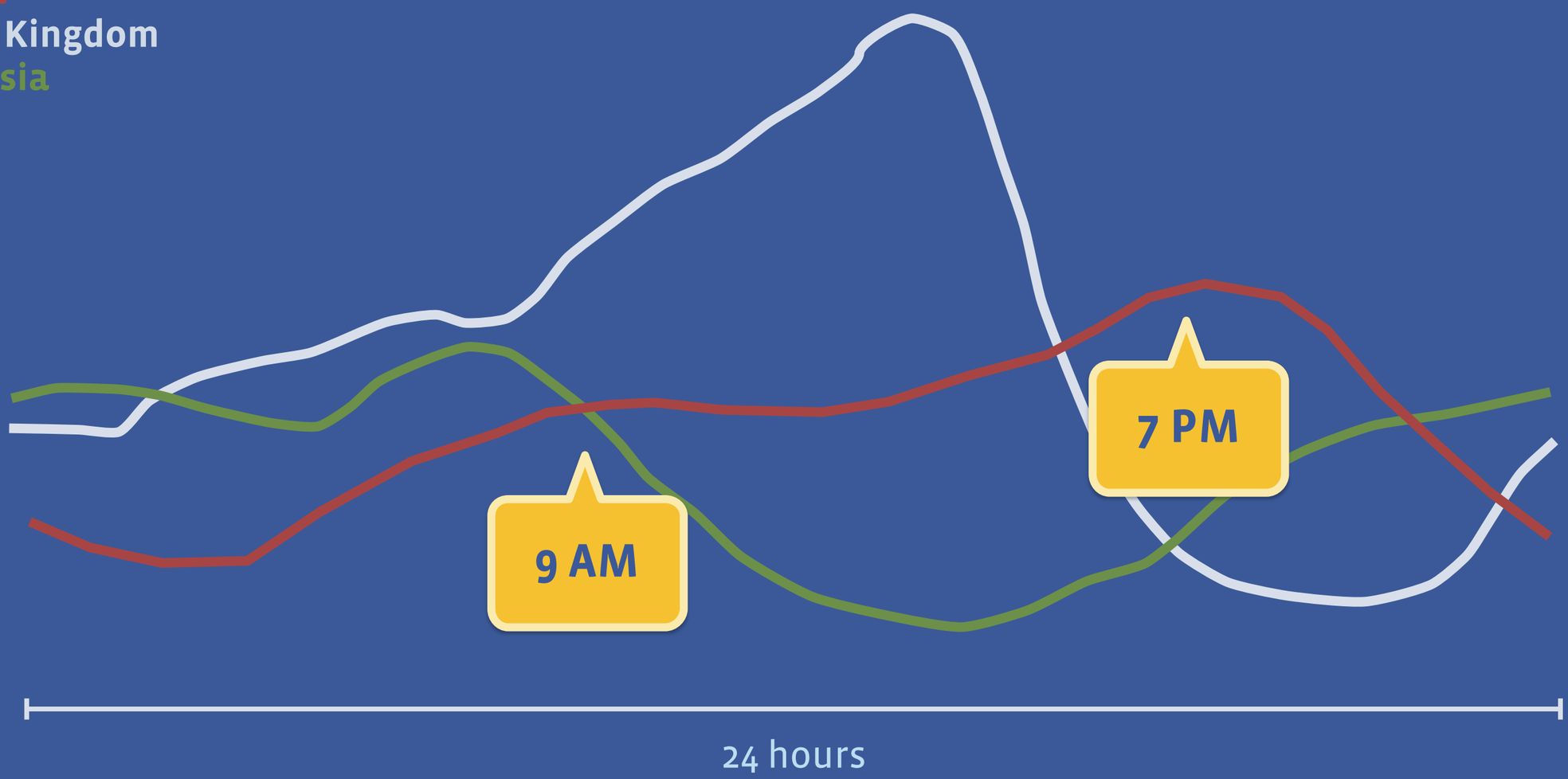


Diurnal Cycle

Egress



Sum of timezones



Cool Traffic Stats (June 2015)

- 1.49+ billion MAP (monthly active people)
- 1 billion DAP (Daily active people)
- 83% users outside US and Canada
- 350 million new photos uploaded per day (2013)
- Terabits per second of egress

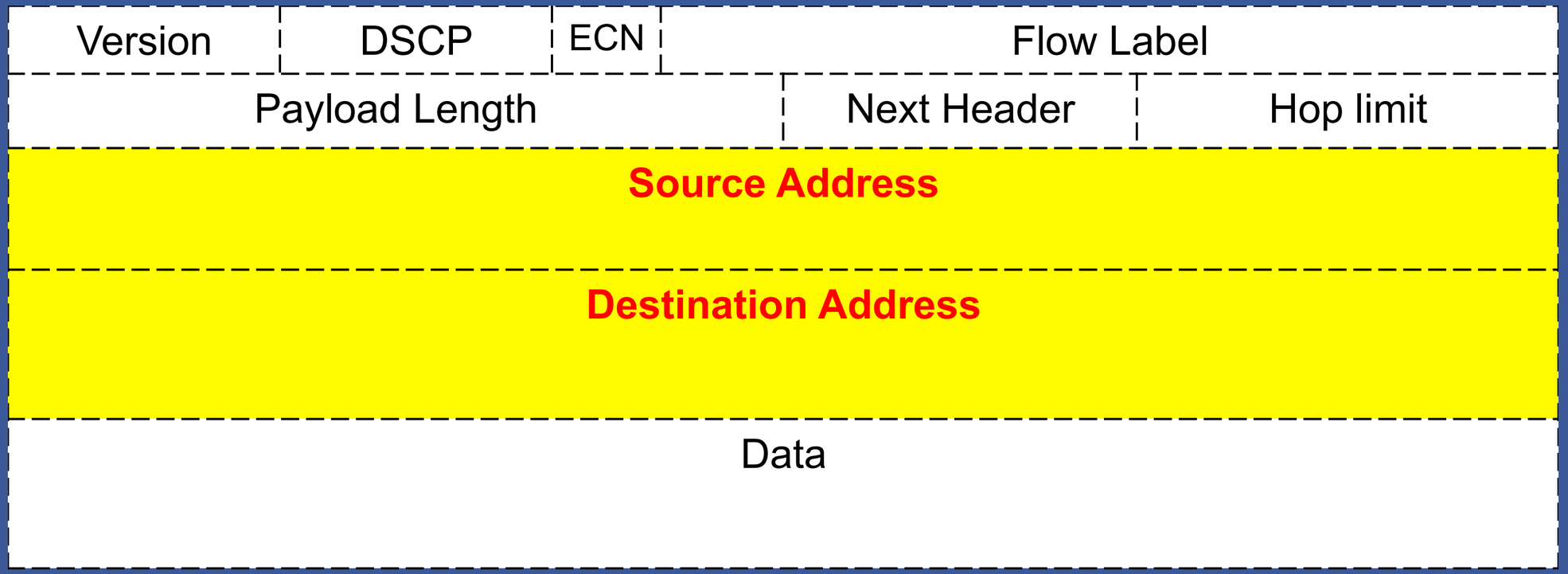
source: http://newsroom.fb.com/company-info/

TCP/IP Review

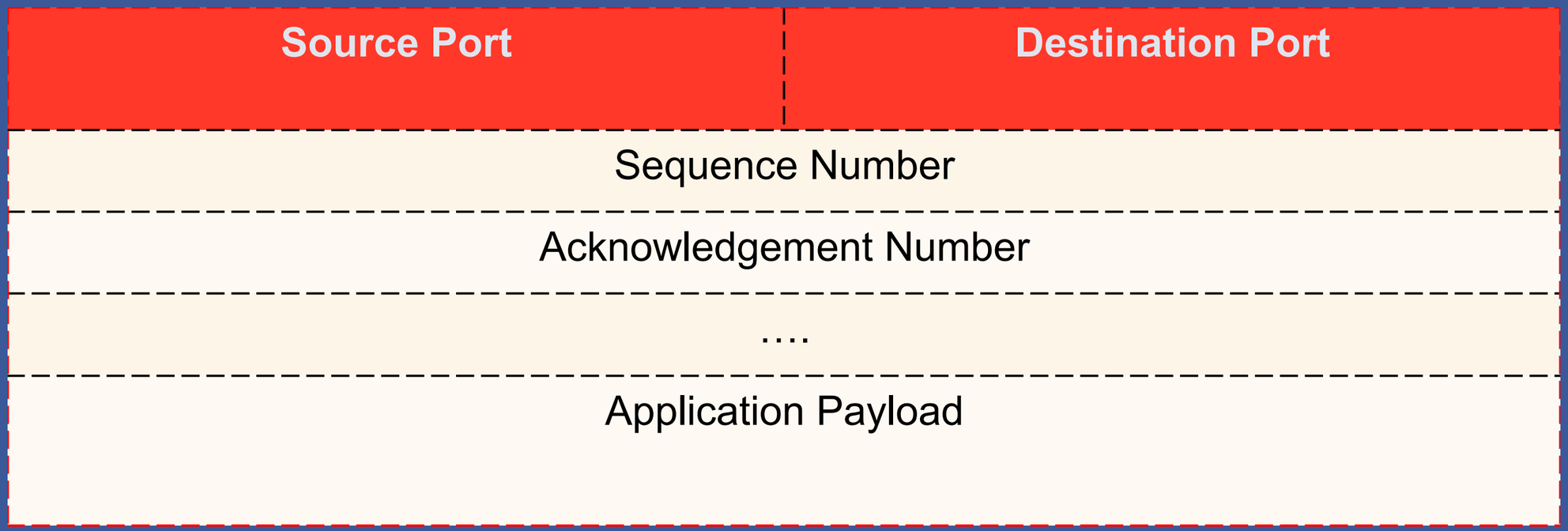
OSI Model

Layer	Purpose	Ex
7: Application	**High-Level API**	**HTTP**, SPDY, FTP
6: Presentation	Data Translation	ASCII, JPEG
5: Session	Communication Session	RPC
4: Transport	**Transmission**	**TCP**, UDP
3: Network	**Address, Routing, Flow**	**IPv4, IPv6**
2: Data Link	Reliable Physical Comm.	IEEE, 802.2
1: Physical	Raw bit transmission	DSL, USB

IP Header (OSI Layer 3)



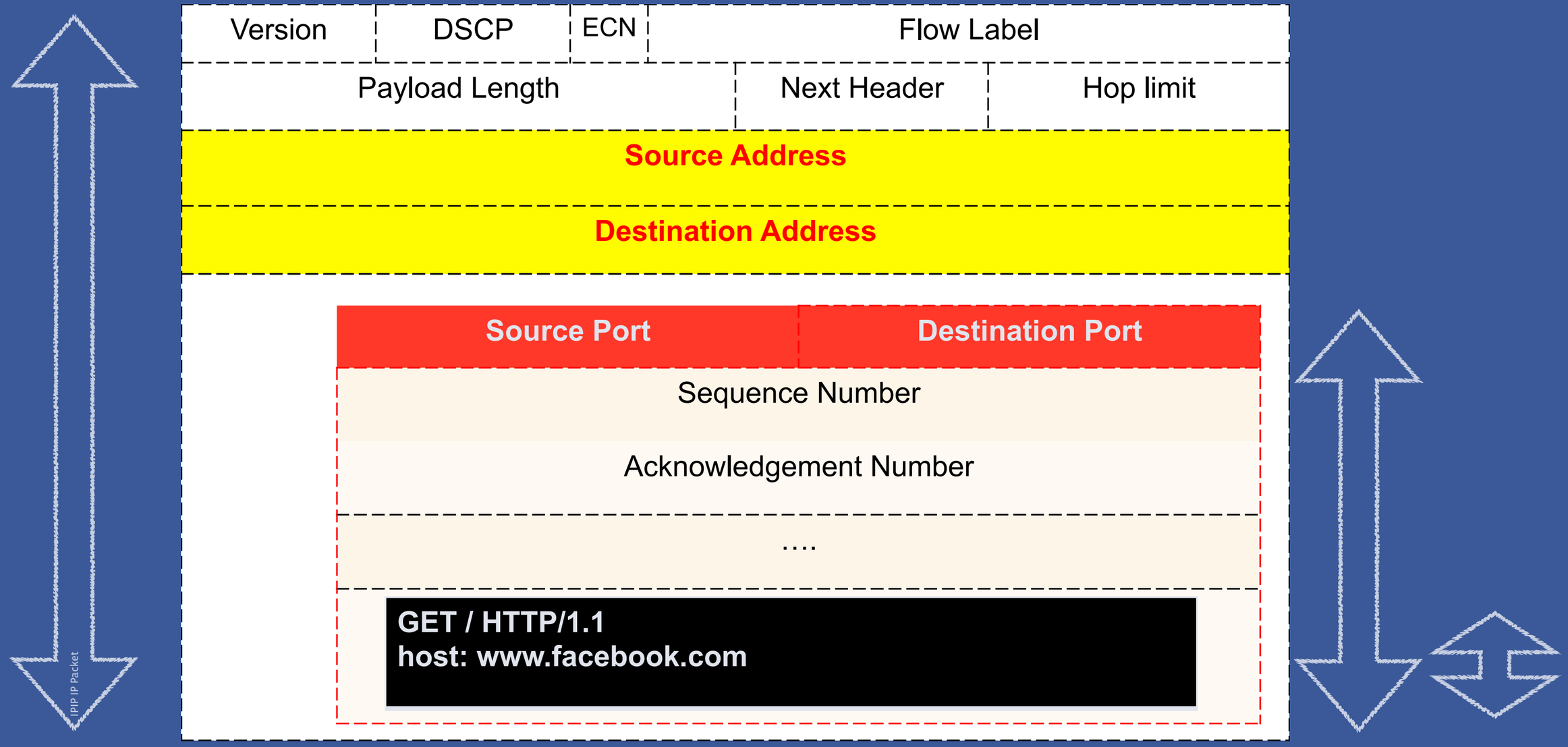
TCP Header (OSI Layer 4)



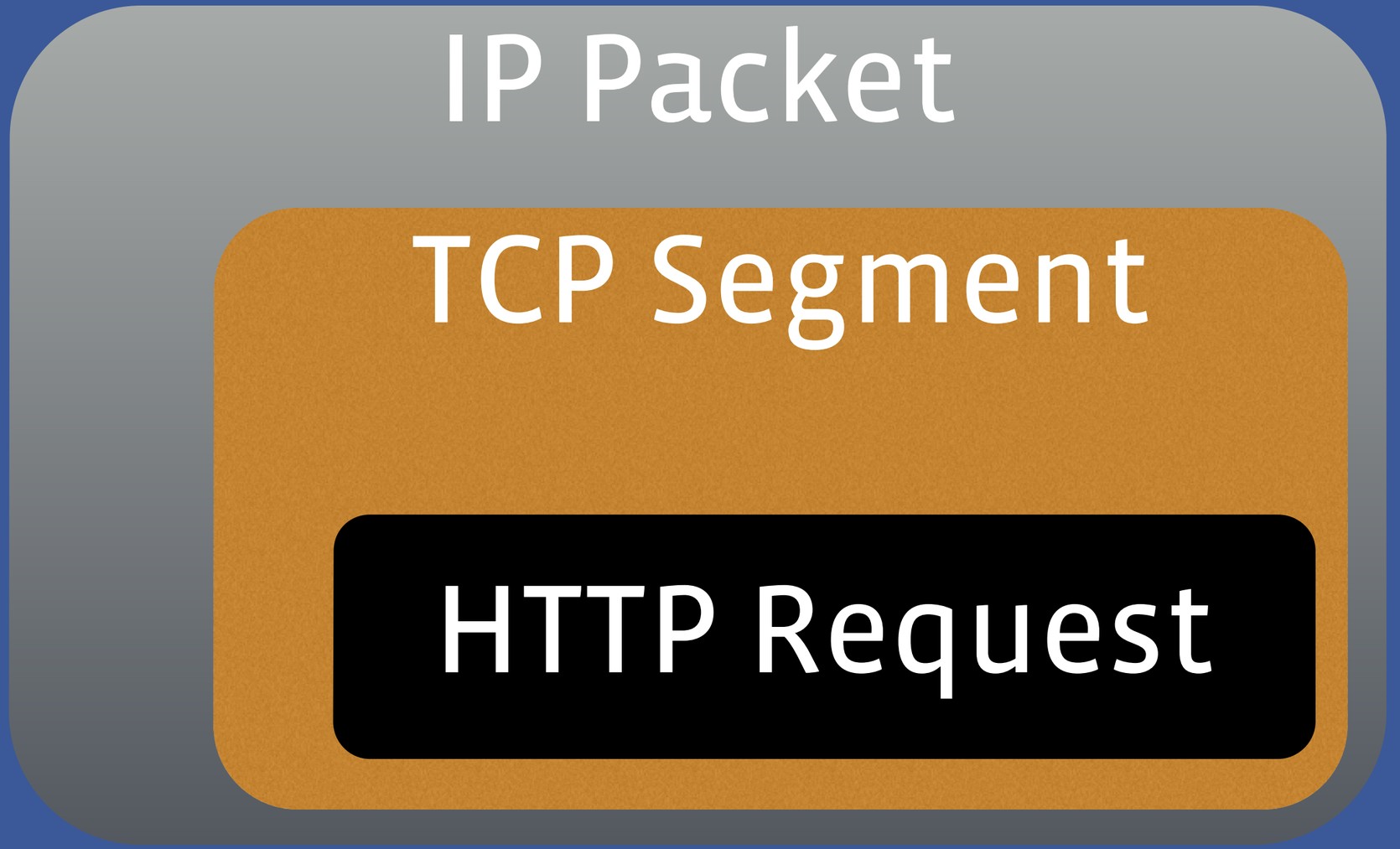
HTTP/1.1 Request

```
GET / HTTP/1.1
host: www.facebook.com
```

Putting it all together



Putting it all together



Serving Dynamic Facebook Requests

Datacenter Locations

FB Request -- one web server

rps = requests per second

www?

A 1.2.3.4

DNS

www ?

GET /

<html>...

HHVM

low rps

how do we get more rps?!

Add a load balancer!

rps = requests per second

www?

A 1.2.3.4

DNS

GET /

<html>...

L7LB
(proxygen)

GET /

<html>...

www ?

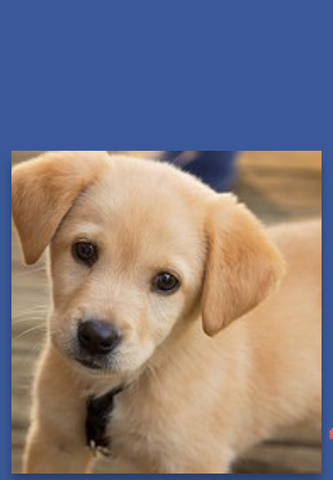
HHVM

lots more rps

low rps

how do we get more rps?!

Add another load balancer!



www?

A 1.2.3.4

DNS

www ?

GET /

<html>...

L4LB
(shiv)

GET /

<html>...

L7LB
(proxygen)

GET /

<html>...

HHVM

network bound

lots more rps

low rps

how do we get more rps?!

Add another load balancer!

www?

A 1.2.3.4

DNS

ECMP

GET /

<html>...

L4LB
(shiv)

network bound

GET /

<html>...

L7LB
(proxygen)

lots more rps

GET /

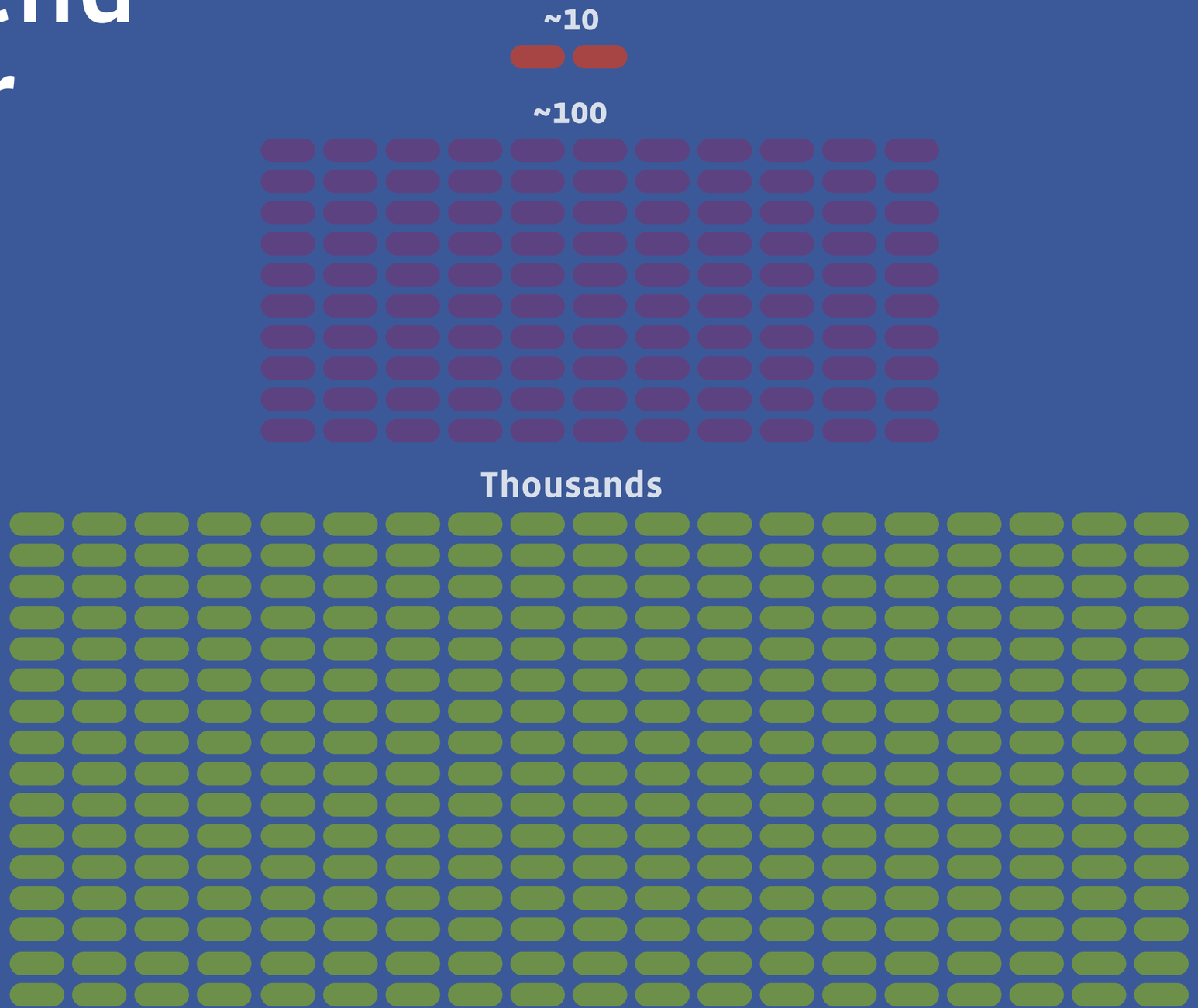
<html>...

HHVM

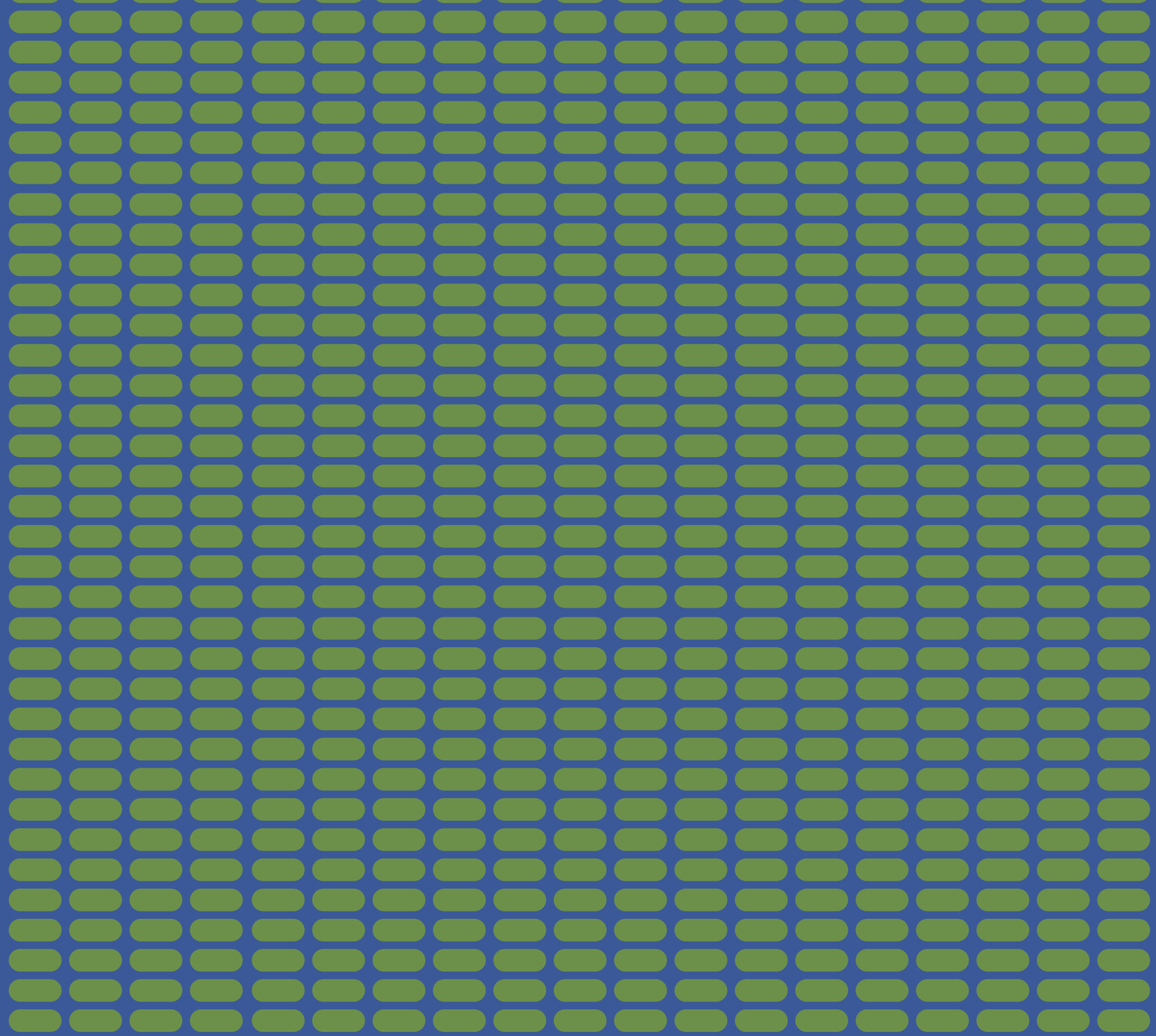
HHVM

low rps

Front end Cluster



cont.



x 10 or more

More RPS? Add another cluster!

www?

A 1.2.3.4

DNS
LB

GET /

<html>...

L4LB
(shiv)

GET /

<html>...

L7LB
(proxygen)

GET /

<html>...

HHVM

GET /

<html>...

L4LB
(shiv)

GET /

<html>...

L7LB
(proxygen)

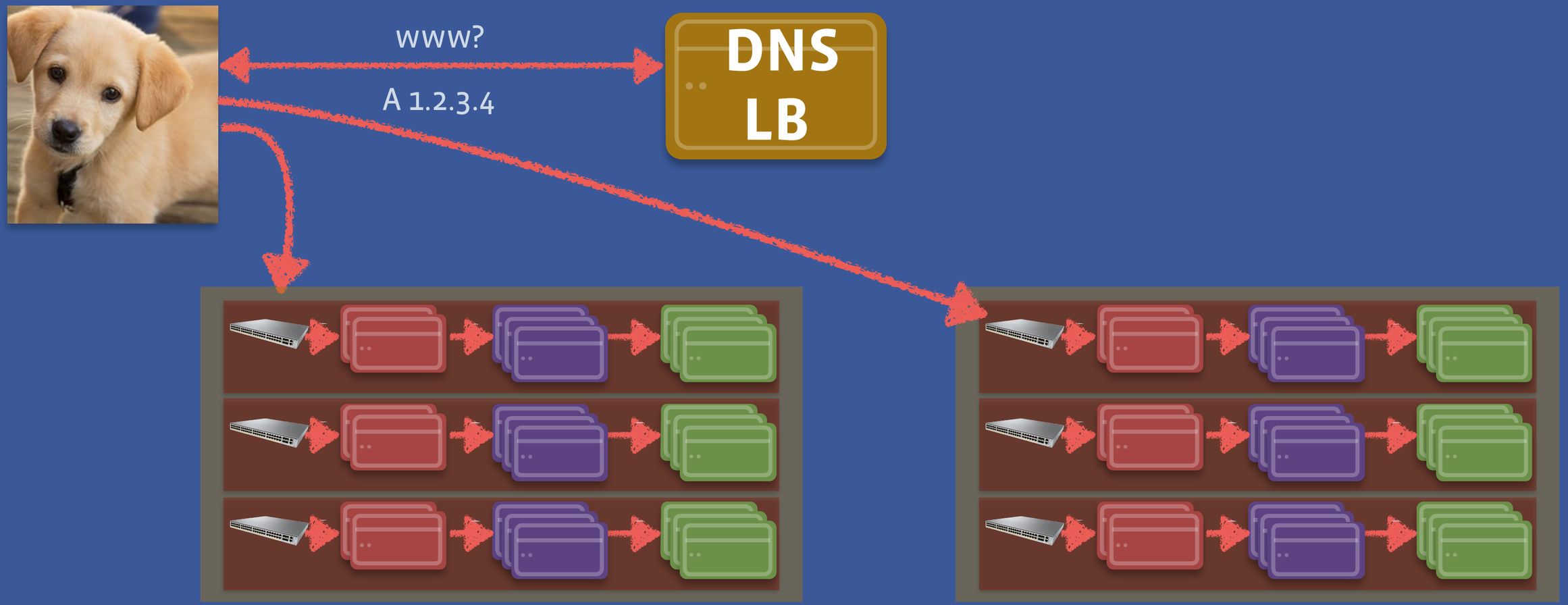
GET /

<html>...

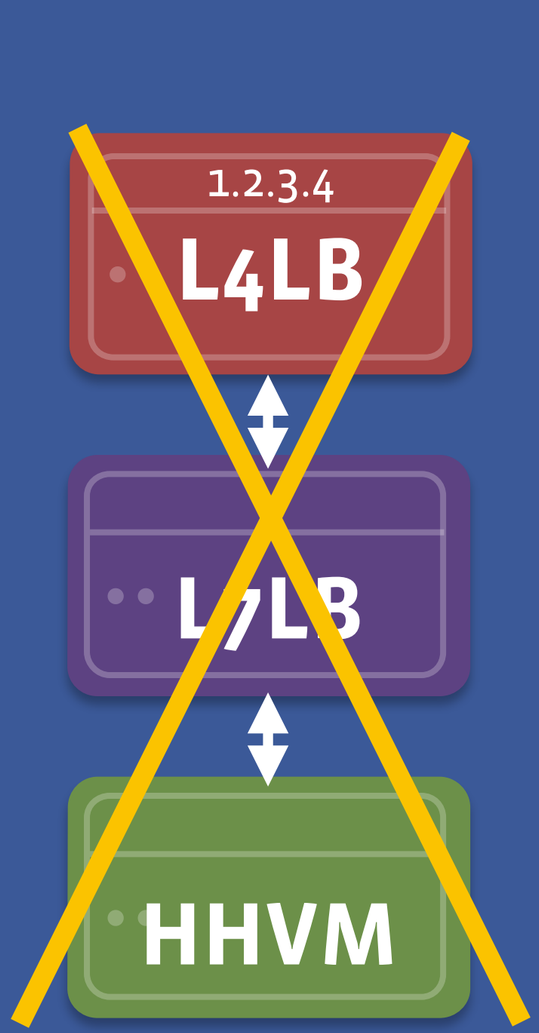
HHVM

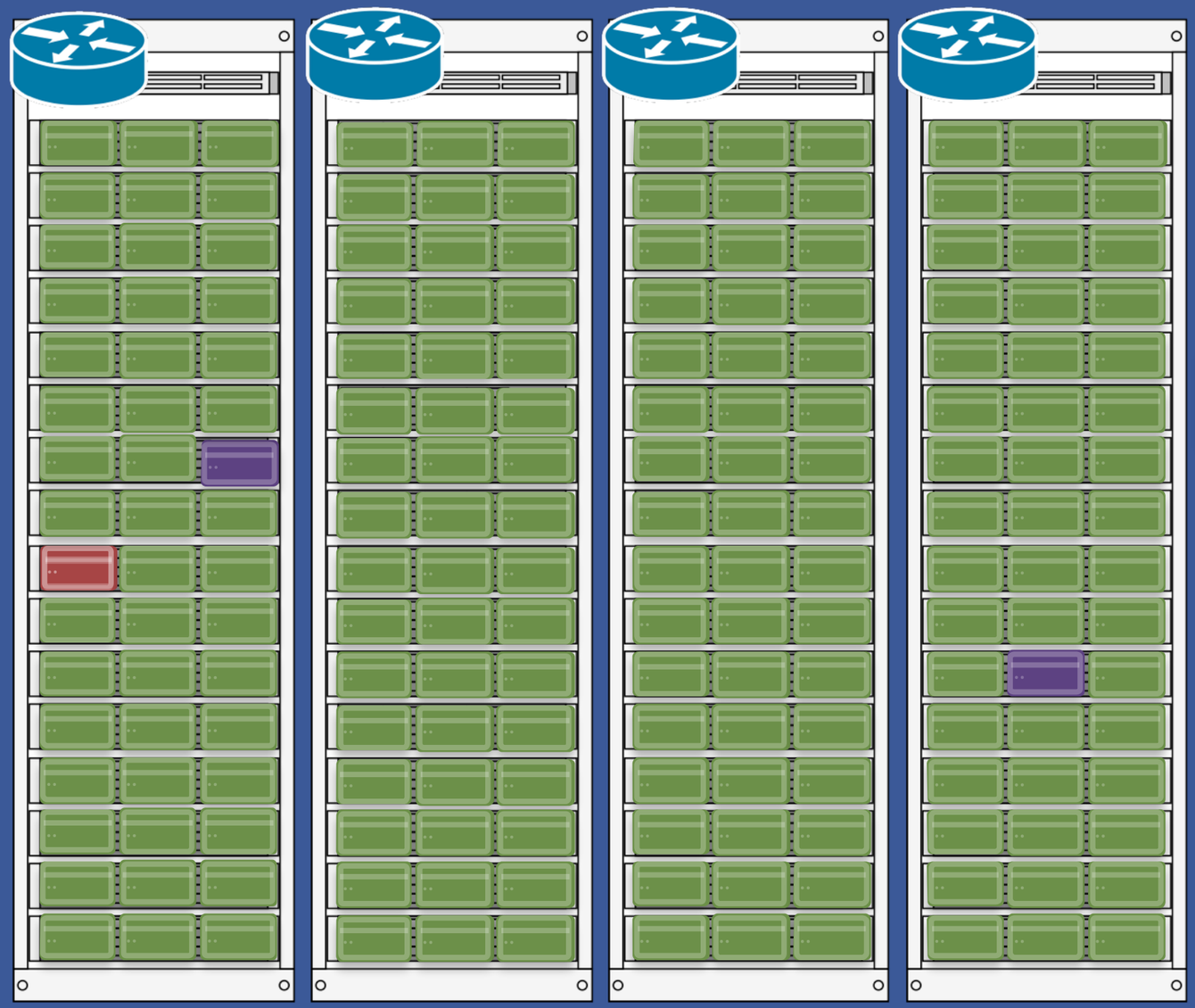
how do we get more rps?!

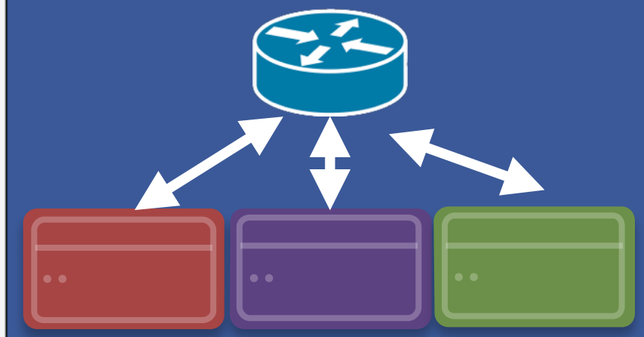
Add another datacenter!



Not really top down







Load Balancing: L4/L7

Let's break it down

www?

A 1.2.3.4

DNS

L4 = IP/port
L7 = HTTP

GET /

<html>...

L4LB
(shiv)

GET /

<html>...

L7LB
(proxygen)

GET /

<html>...

HHVM

OSI Model: What is L4/L7?

Layer	Purpose	Ex
7: Application	**High-Level API**	**HTTP**, **SPDY,** FTP
6: Presentation	Data Translation	ASCII, JPEG
5: Session	Communication Session	RPC
4: Transport	**Transmission**	**TCP**, UDP
3: Network	Address, Routing, Flow	IPv4, IPv6
2: Data Link	Reliable Physical Comm.	IEEE, 802.2
1: Physical	Raw bit transmission	DSL, USB

L7LB (proxygen)

L4LB (shiv)

ECMP

L4LB

BGP

ECMP Hash

1.2.3.4 L4LB

1.2.3.4 L4LB

1.2.3.4 L4LB

1.2.3.4 L4LB

1.2.3.4 L7LB

1.2.3.4 L7LB

1.2.3.4 L7LB

1.2.3.4 L7LB

L4LB

BGP

ECMP Hash

State Table + Hash

1.2.3.4

L4LB

1.2.3.4

L4LB

1.2.3.4

L7LB

1.2.3.4

L7LB

1.2.3.4

L7LB

1.2.3.4

L7LB

L4LB Routing

ECMP Hash

State Table + Hash

A

1.2.3.4

L4LB

1.2.3.4

L4LB

D

1.2.3.4

L7LB

A

1.2.3.4

L7LB

B

1.2.3.4

L7LB

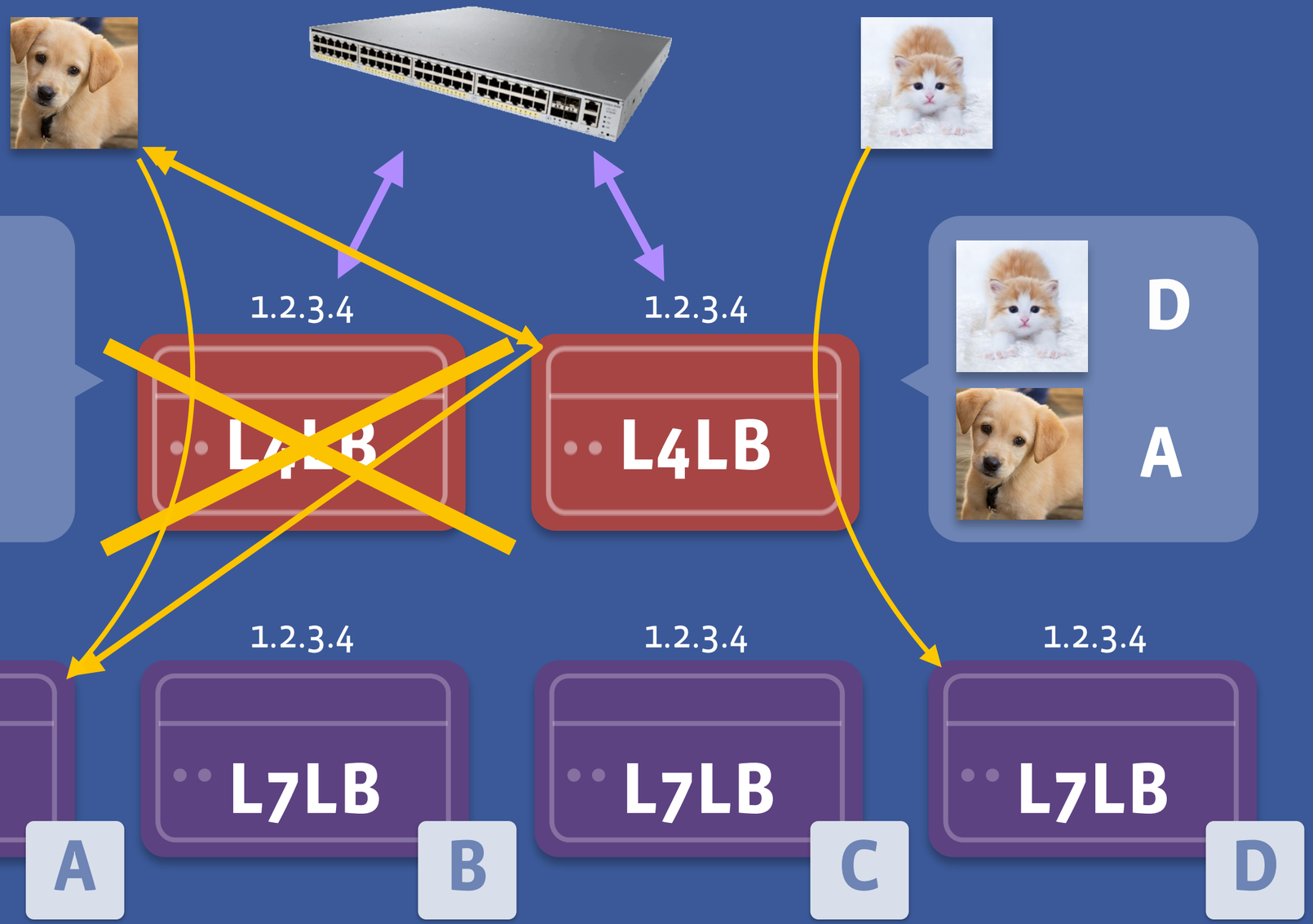
C

1.2.3.4

L7LB

D

L4LB Routing



L4LB Routing

ECMP Hash

State Table + Hash

B

1.2.3.4

L4LB

1.2.3.4

L4LB

1.2.3.4

A

1.2.3.4

L7LB

B

1.2.3.4

L7LB

C

1.2.3.4

L7LB

D

L4LB Routing

ECMP Hash

State Table + Hash

B

1.2.3.4

L4LB

1.2.3.4

L4LB

1.2.3.4

L7LB

A

1.2.3.4

L7LB

B

1.2.3.4

L7LB

C

1.2.3.4

L7LB

D

Direct Server Return

TCP Routing

TCP
SSL
HTTP

Facebook

1.2.3.4
L4LB

1.2.3.4
L7LB

1.2.3.4
L7LB

1.2.3.4
L7LB

1.2.3.4
L7LB

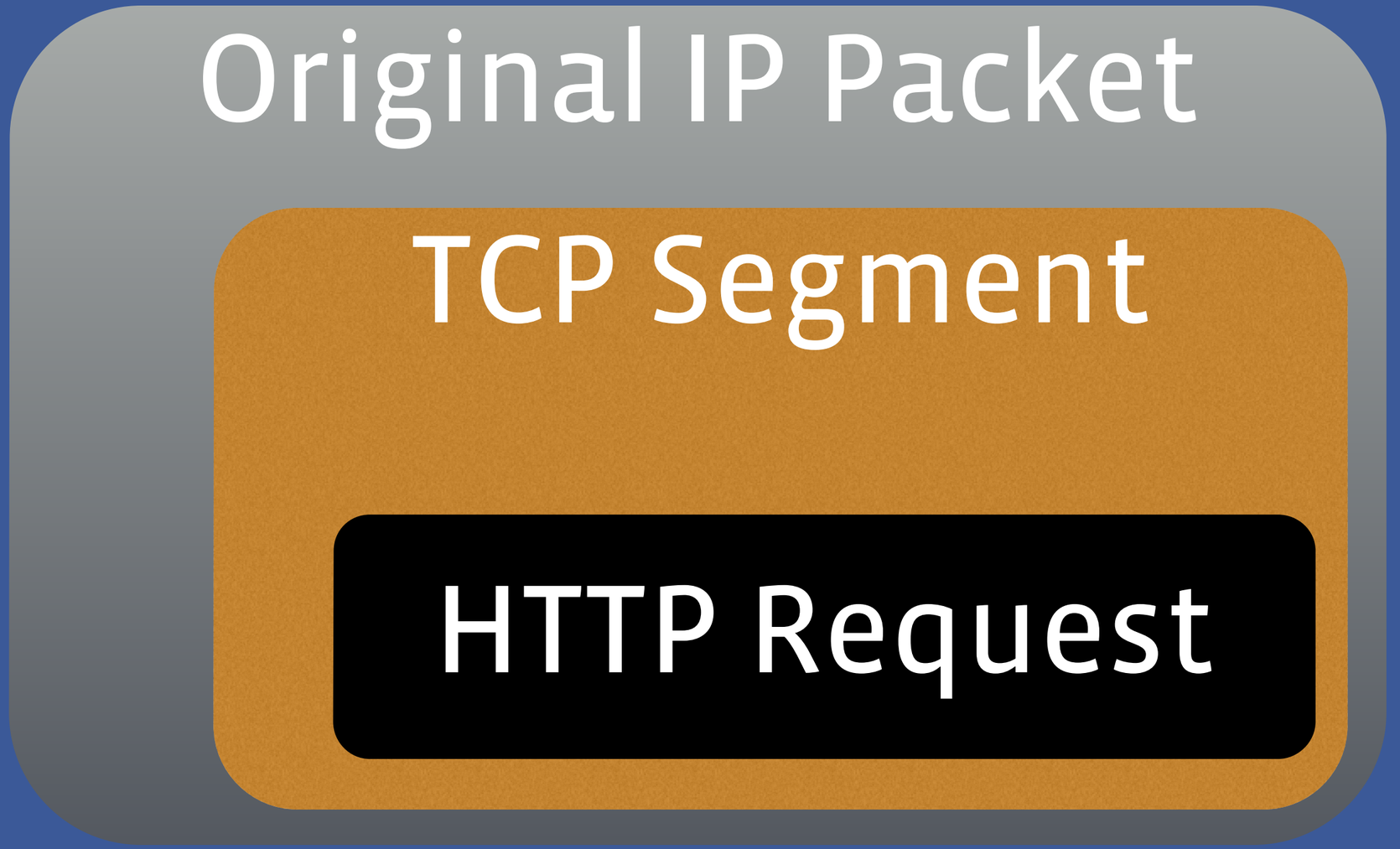
HHVM

HHVM

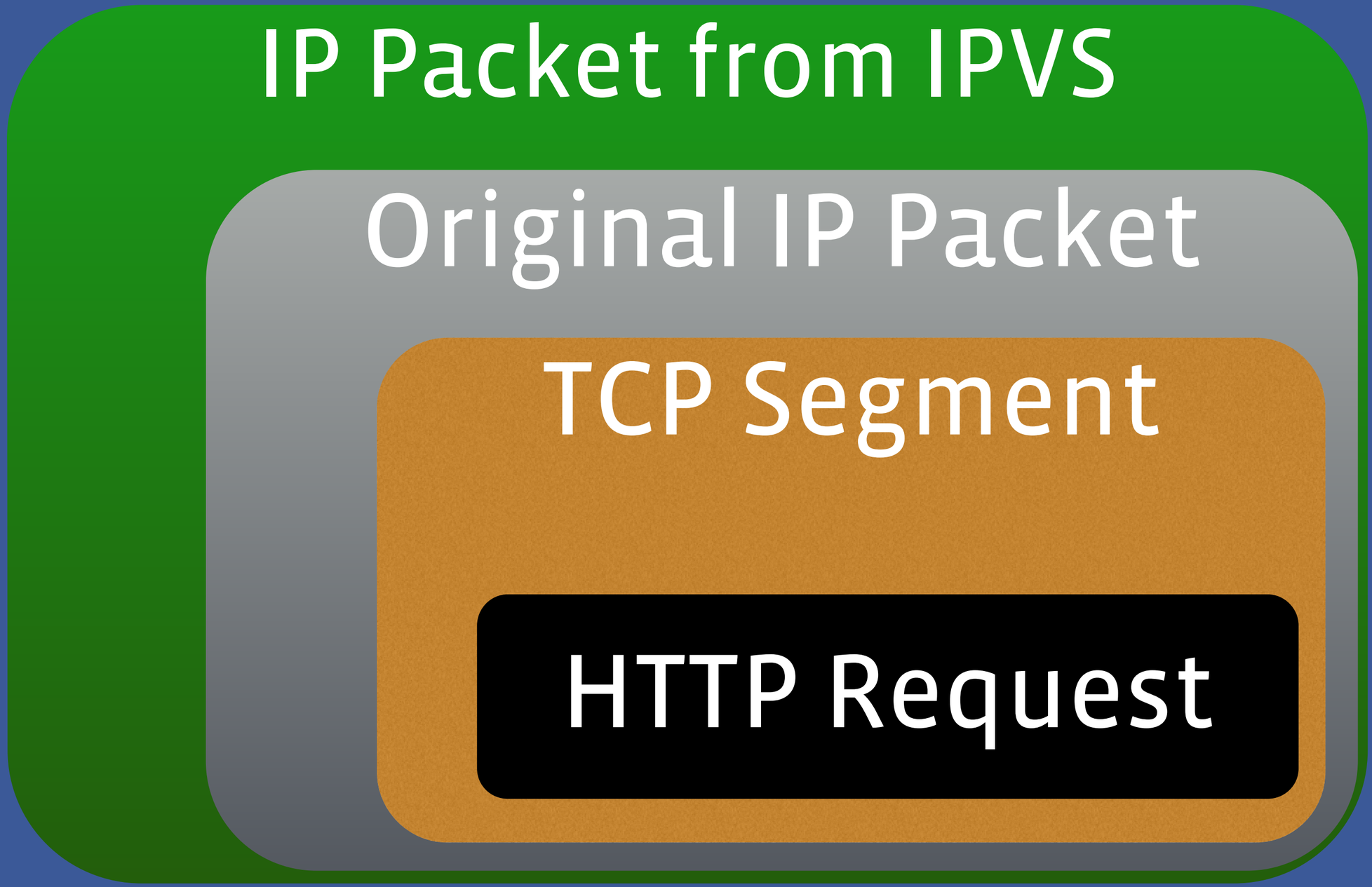
HHVM

HHVM

Remember this?



IP in IP encapsulation



Direct Server Return

Original IP Packet

TCP Segment

HTTP Request

1.2.3.4

L4LB

TCP/SSL Term

Forward HTTP Request

1.2.3.4

1.2.3.4

1.2.3.4

L4LB

IP Packet from IPVS

Original IP Packet

TCP Segment

HTTP Request

1.2.3.4

L7LB

1.2.3.4

L7LB

Response IP Pckt

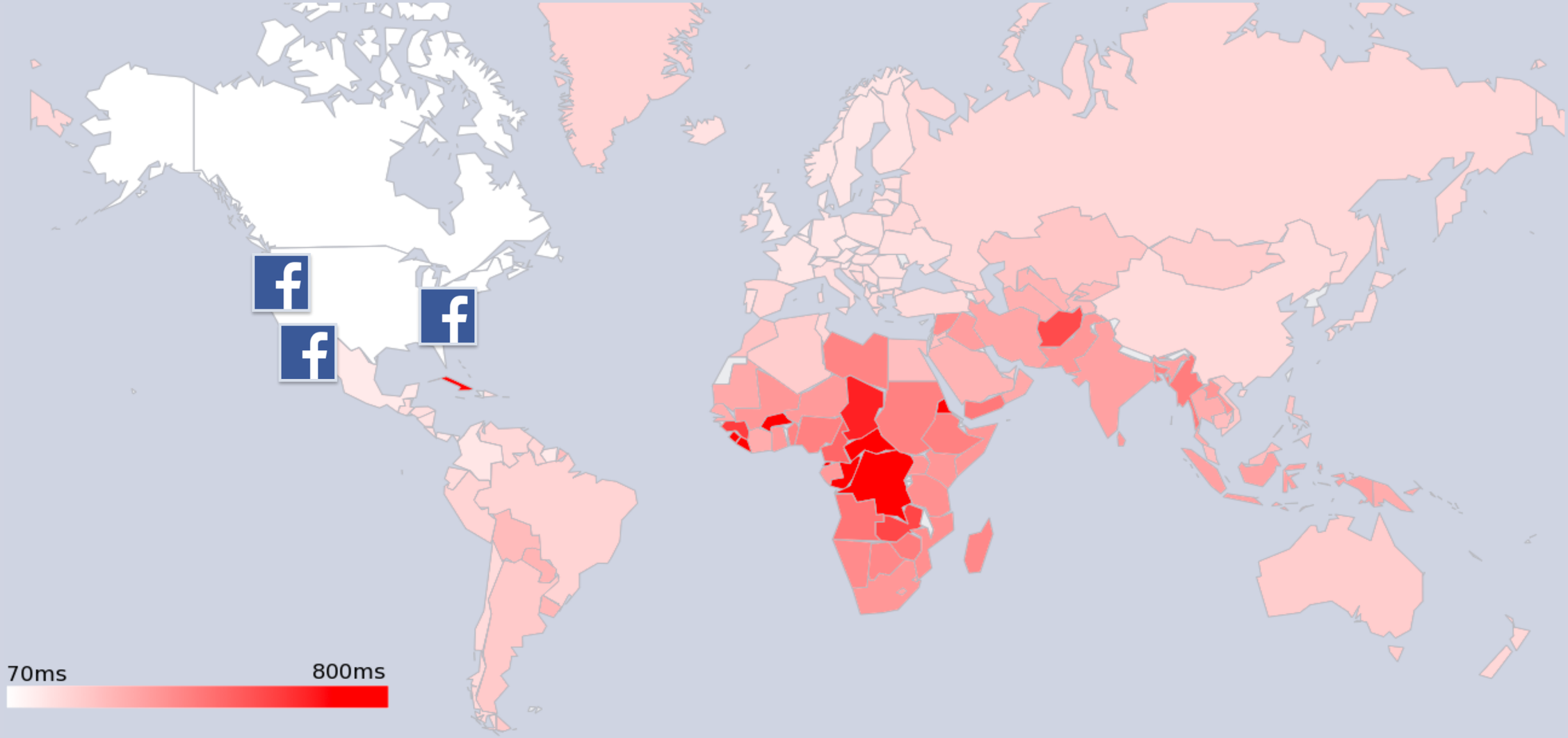
TCP Segment

HTTP Response

Edge PoP’s & Reducing Latency

International RTT

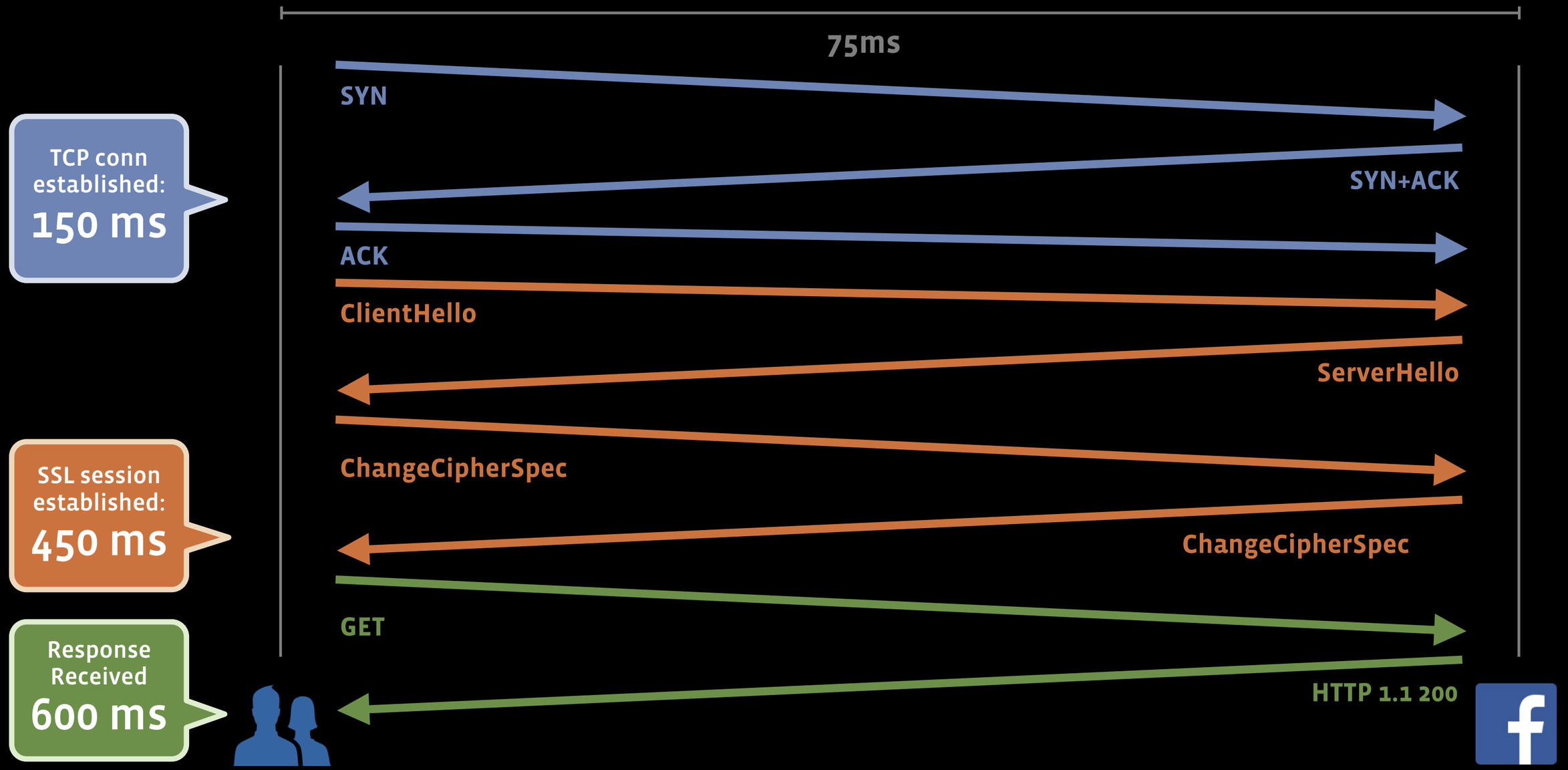
circa 11/2011



Seoul -> Oregon

TCP Connect: 150ms

HTTPS Seoul -> Oregon

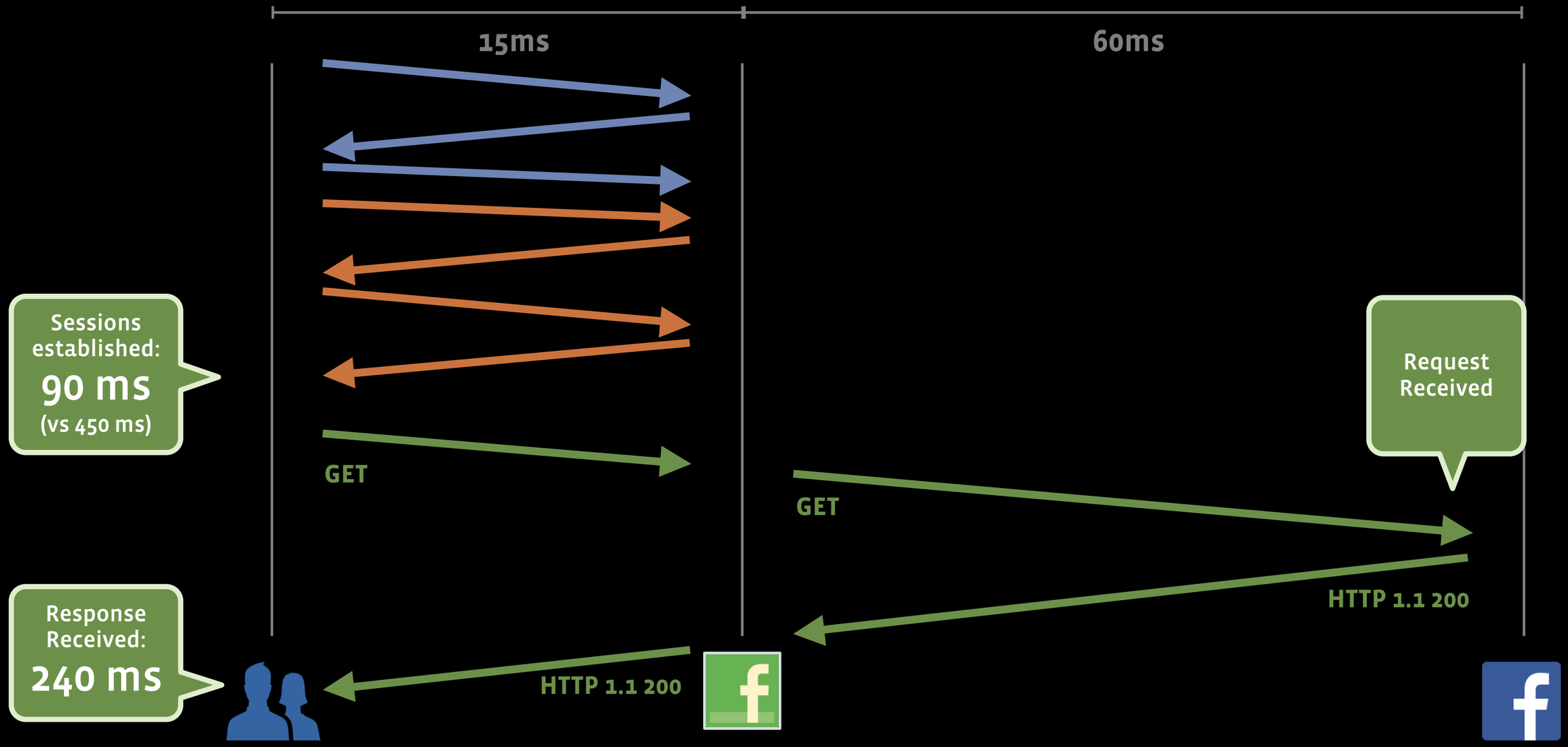


Seoul -> Tokyo -> Oregon

NRT

TCP Connect: 30ms
SSL Session: ??
HTTP Response: ??

HTTPS Seoul->Tokyo->Oregon

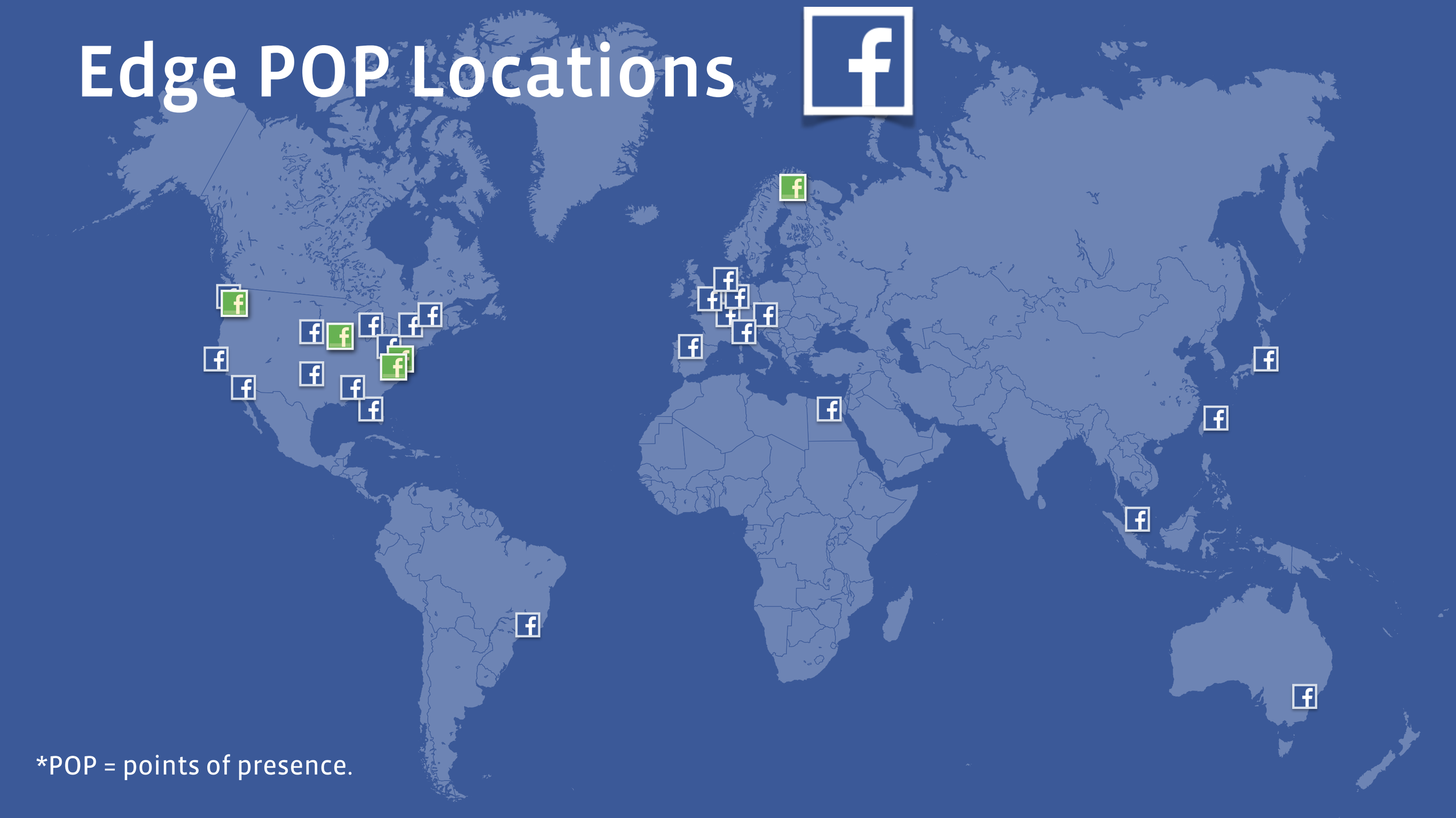


Seoul -> Oregon

NRT

TCP Connect: ~~150ms~~ 30ms
SSL Session: ~~450ms~~ 90ms
HTTP Response: ~~600ms~~ 240ms

Edge POP Locations



*POP = points of presence.

How do the LB's in PoP's work?

1.2.3.4

L4LB

10.1.2.3

L4LB

TCP Routing
(ip/port)

L7LB

L7LB

TCP/SSL
HTTP

HHVM

HHVM

Facebook

DNS LB: Cartographer

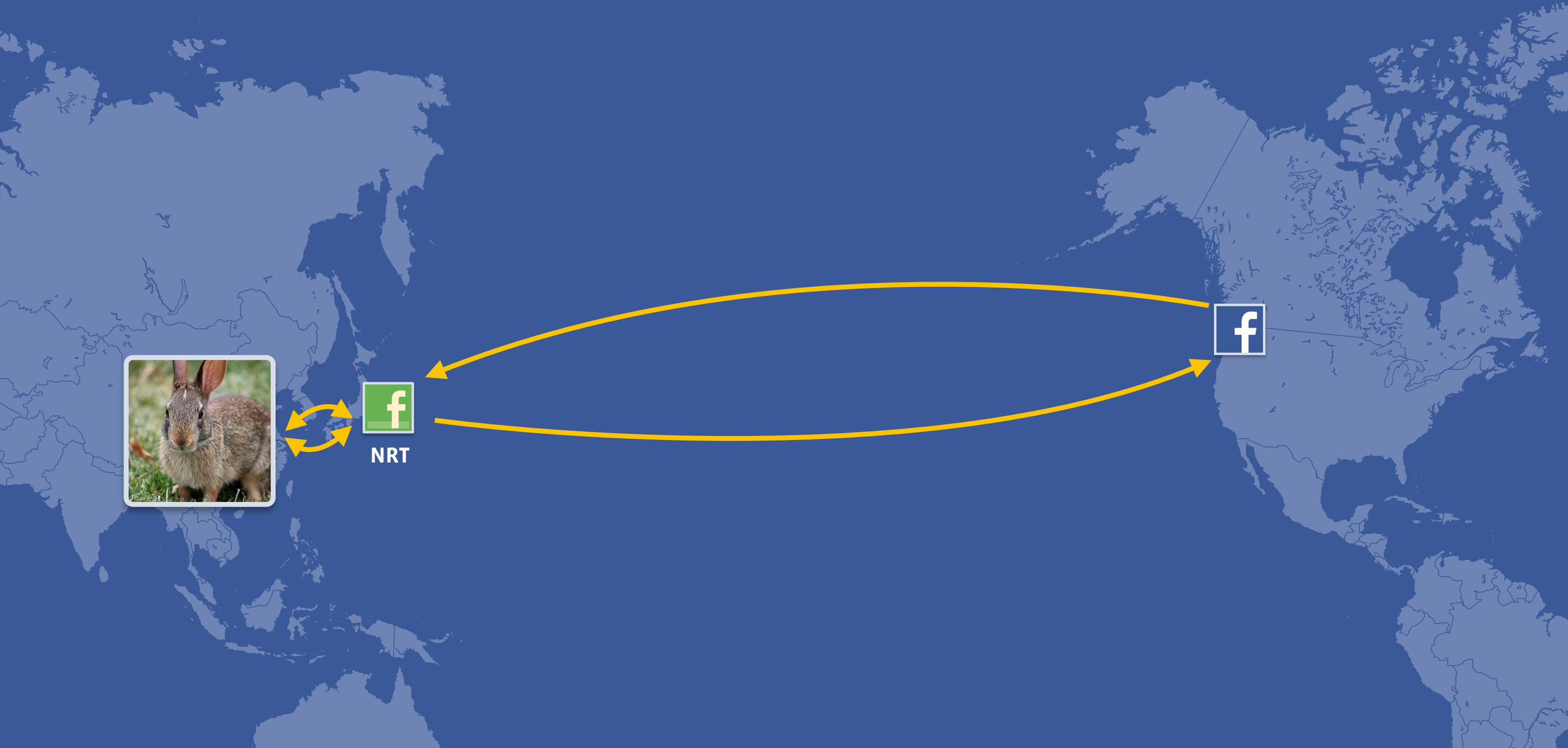
DNS LB Decision

Considerations:

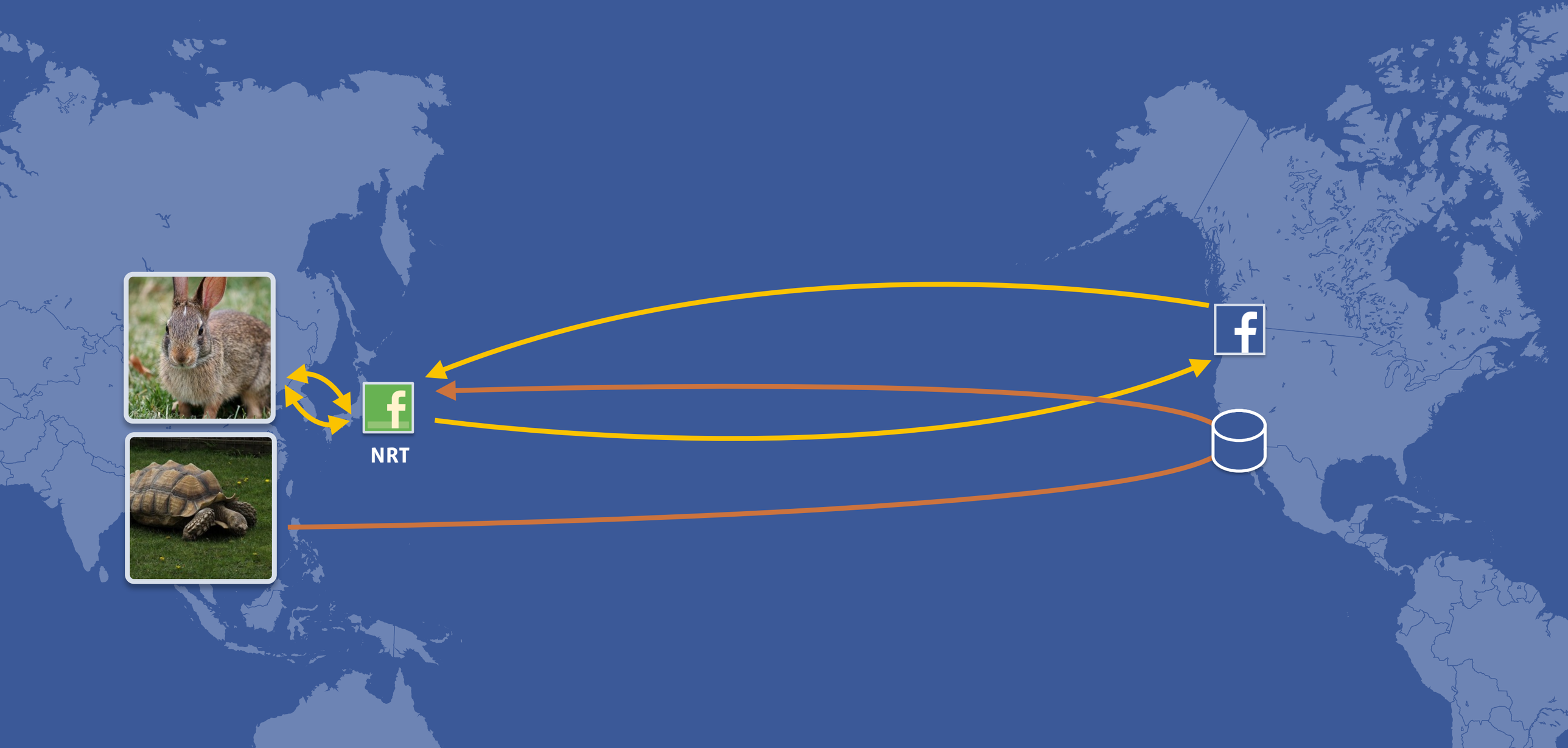
- Closest Edge to user
- Capacity

???

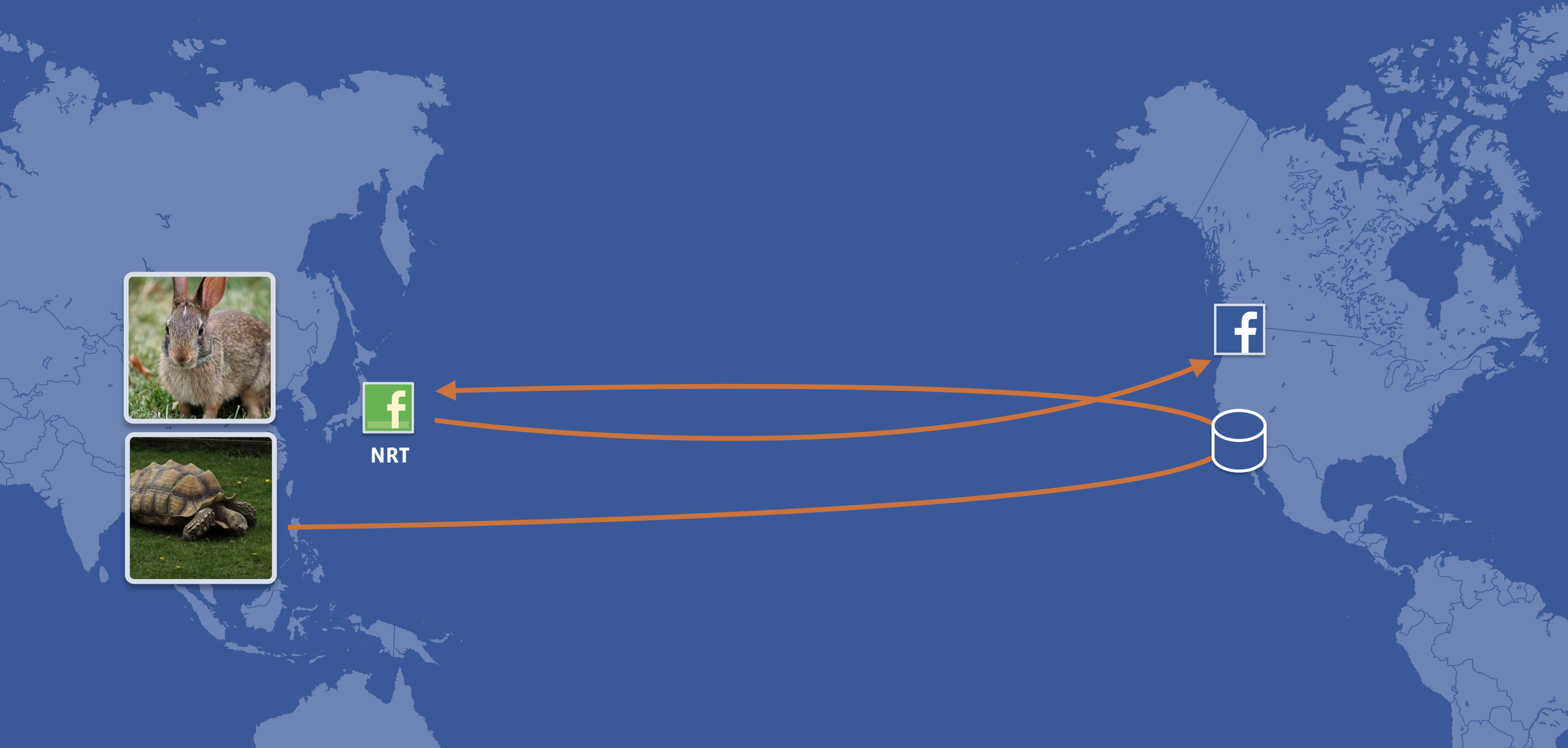
Network Topology?



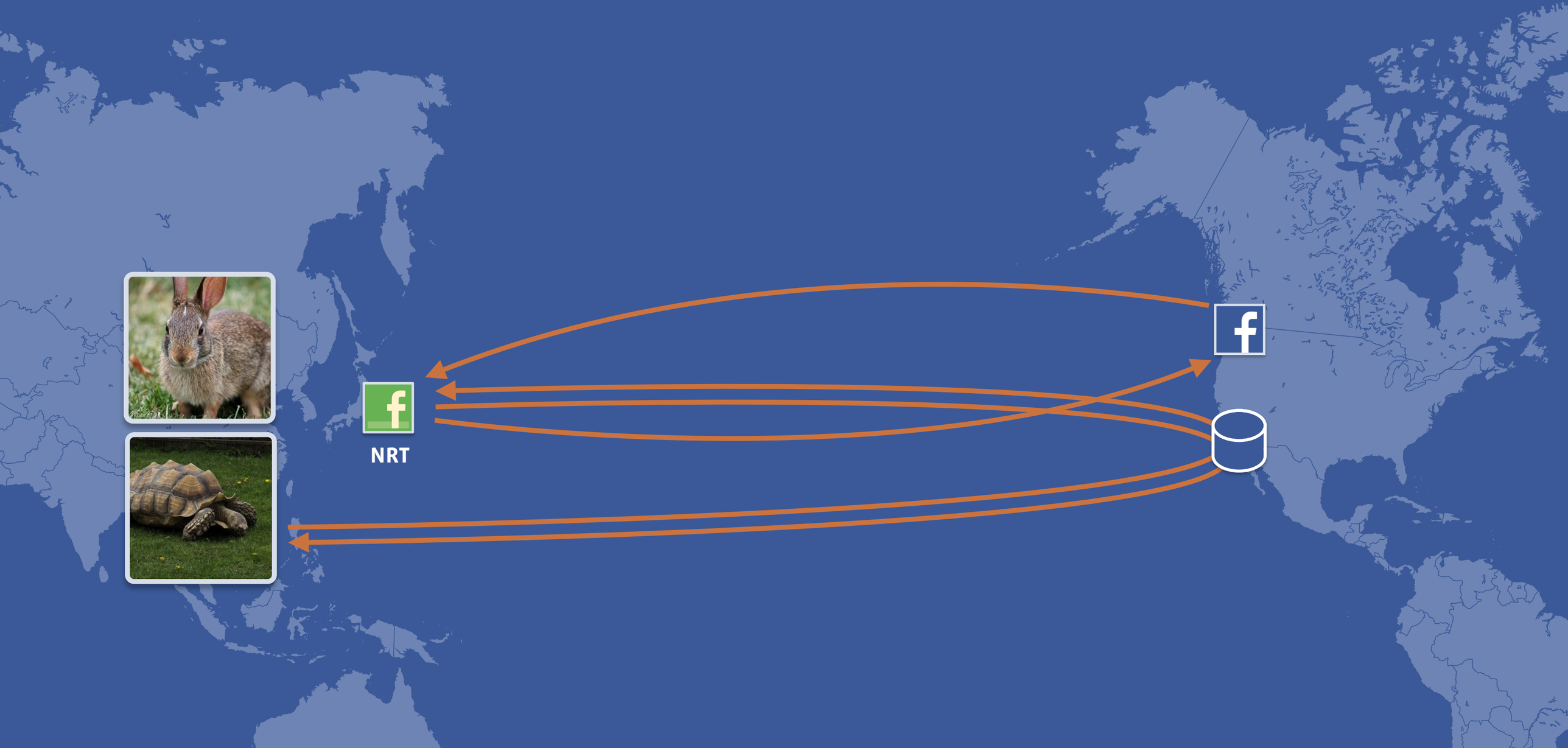
Network Topology?



Network Topology?



Network Topology?

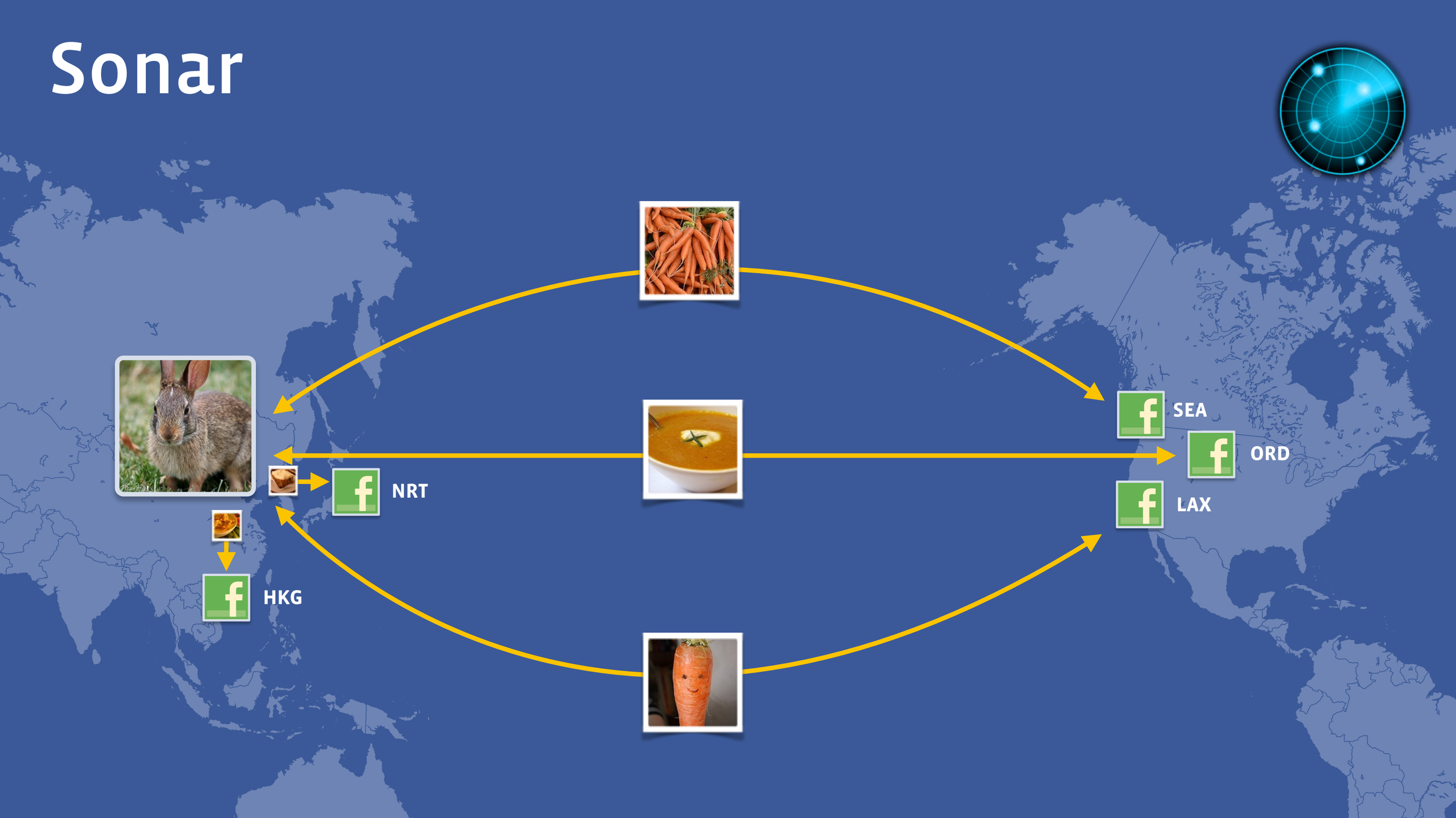


DNS LB Decision

Considerations:

- Closest Edge to user
- Capacity
- Network topology

Sonar



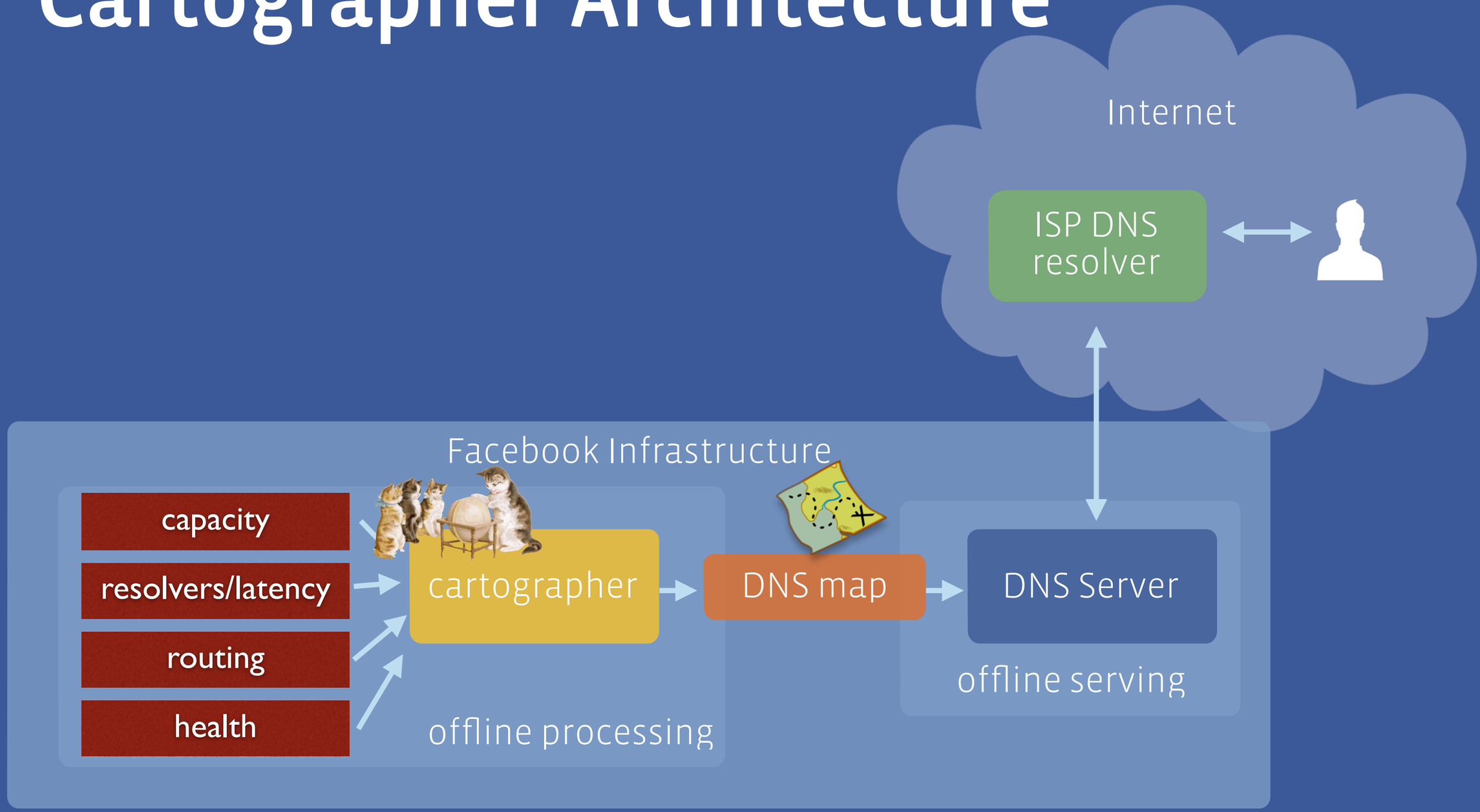
Sonar

https://alskdjflkasjdf-sonar.fbcdn.net/thumb.jpg

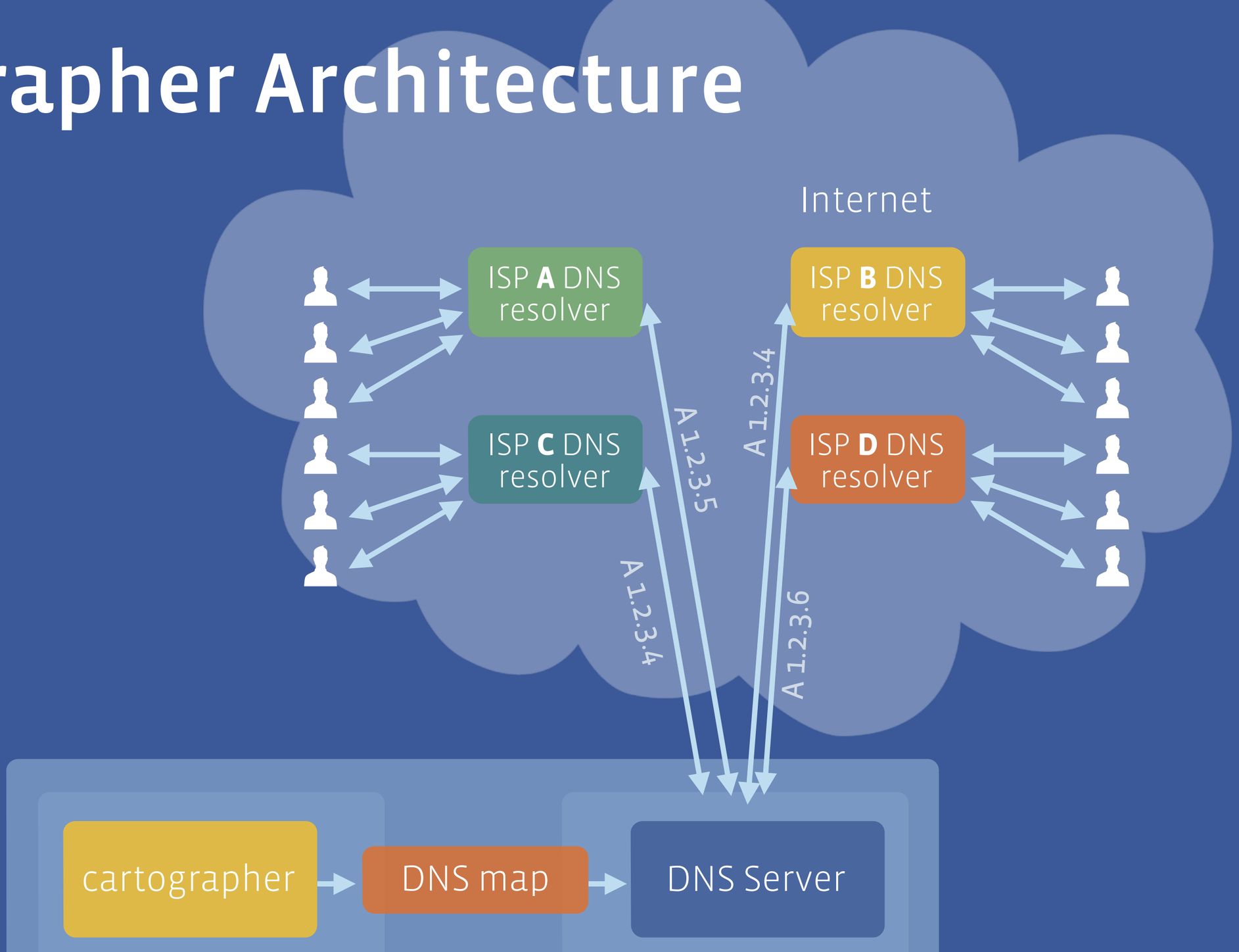
ORD

DNS - Unique Hostname, Resolver IP
HTTPS - Unique Hostname, Client IP, RTT

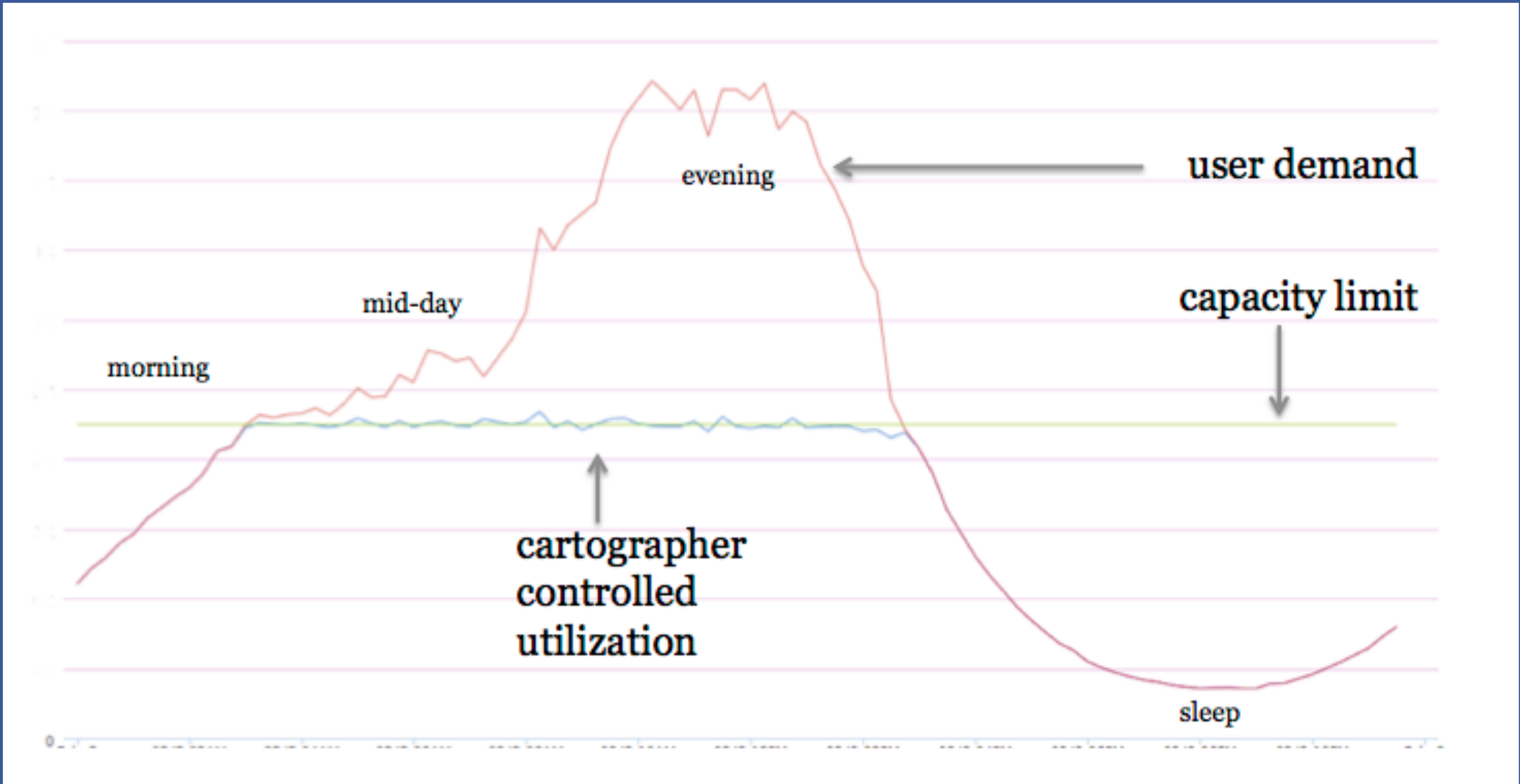
Cartographer Architecture



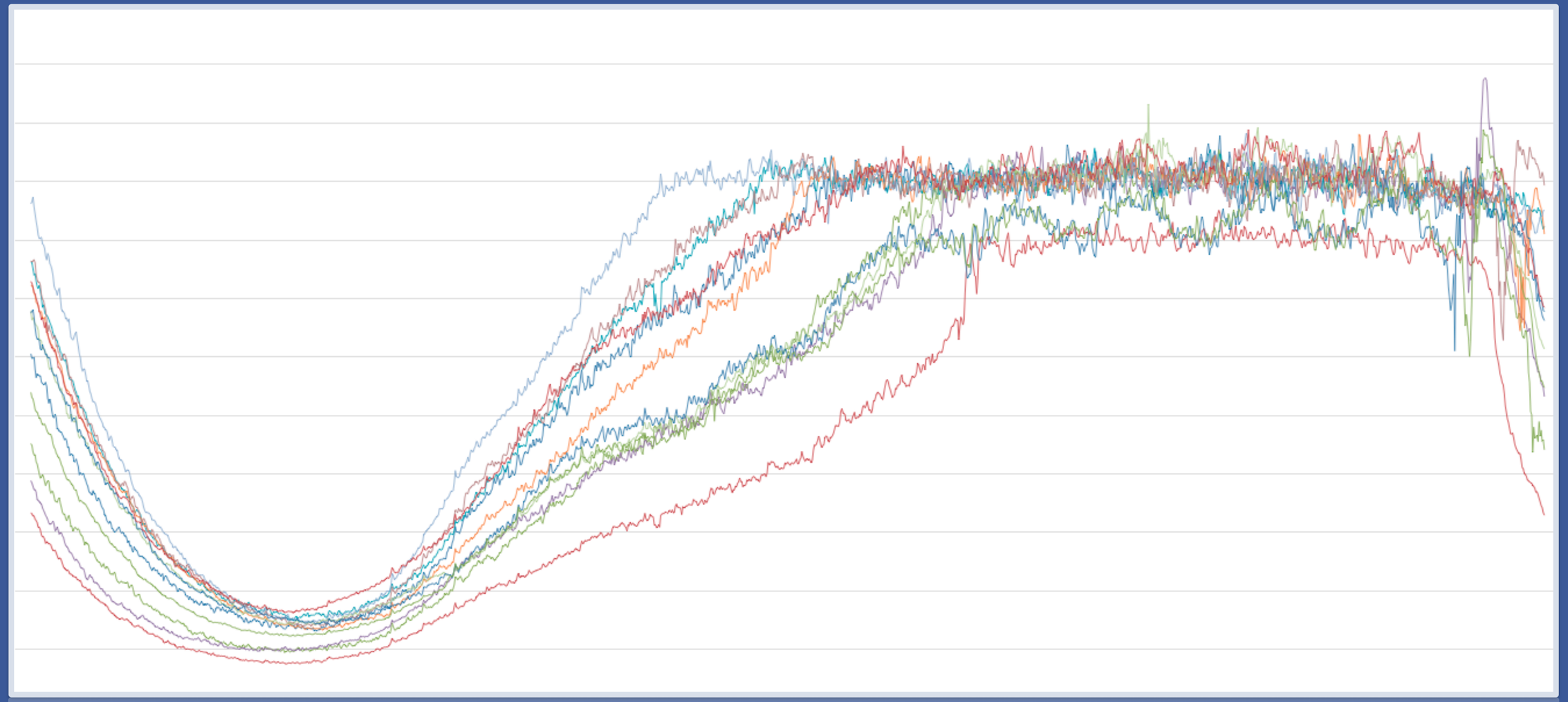
Cartographer Architecture



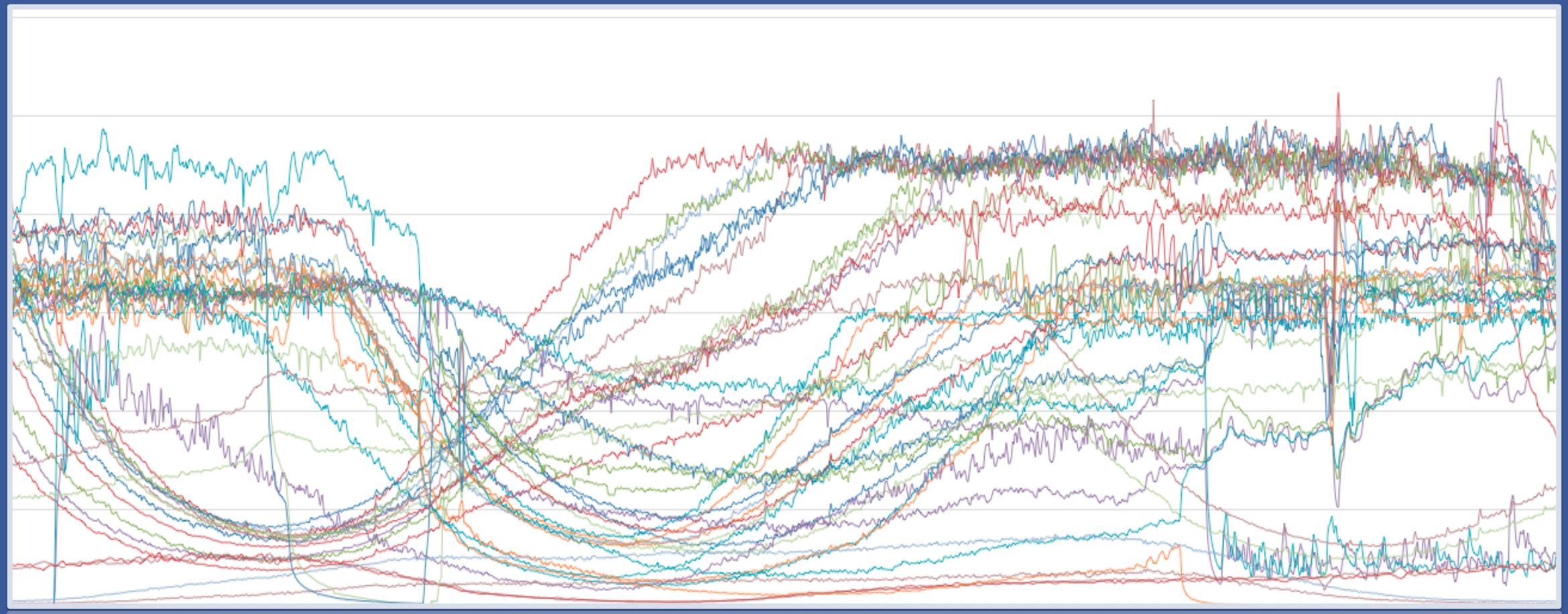
Cartographer in action

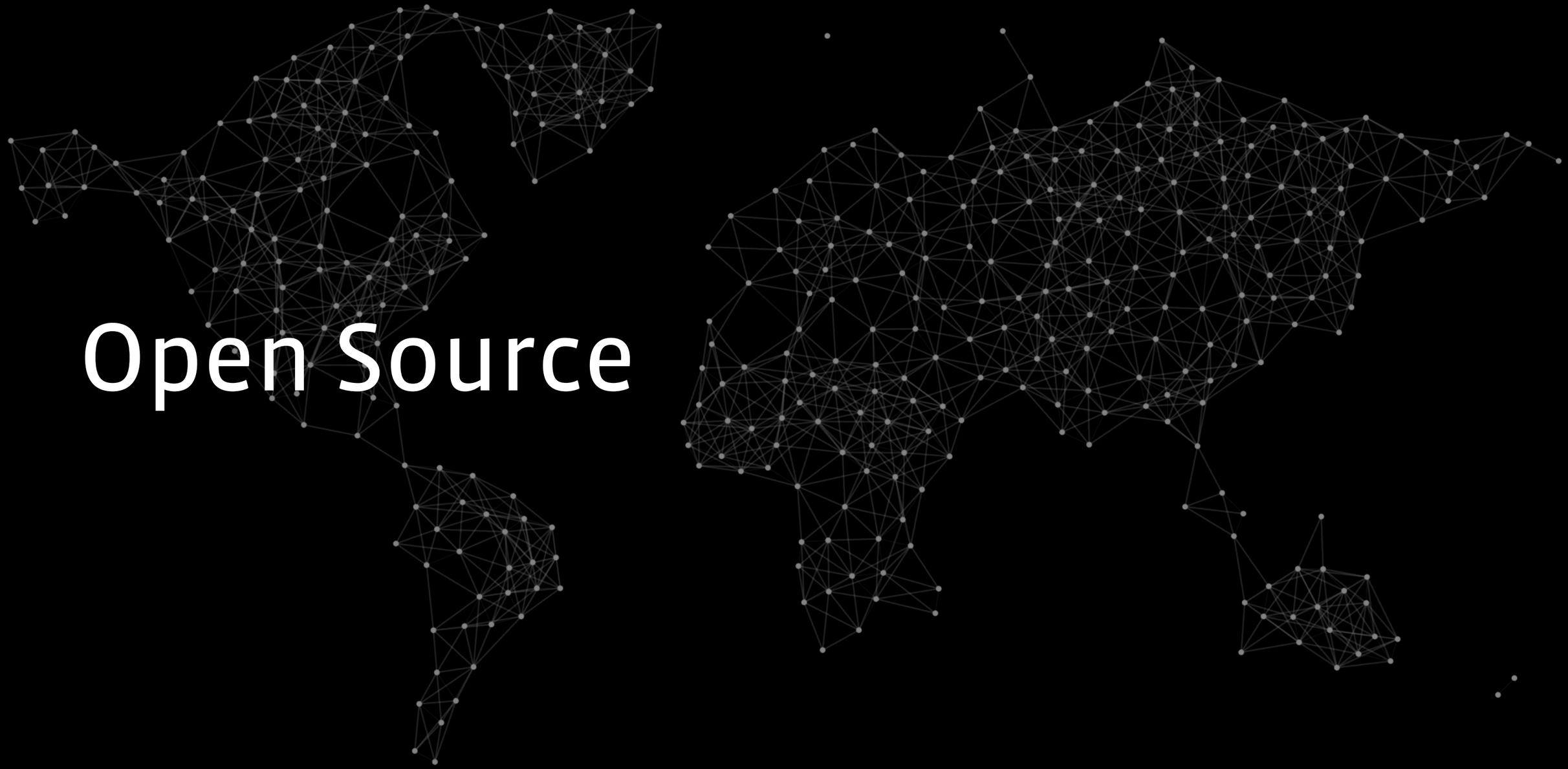


Regional Load Shedding



Global Load Shedding





Open Source

Open Source Components

- Proxygen HTTP Libs

https://github.com/facebook/proxygen

- TinyDNS

https://cr.yp.to/djbdns/tinydns.html

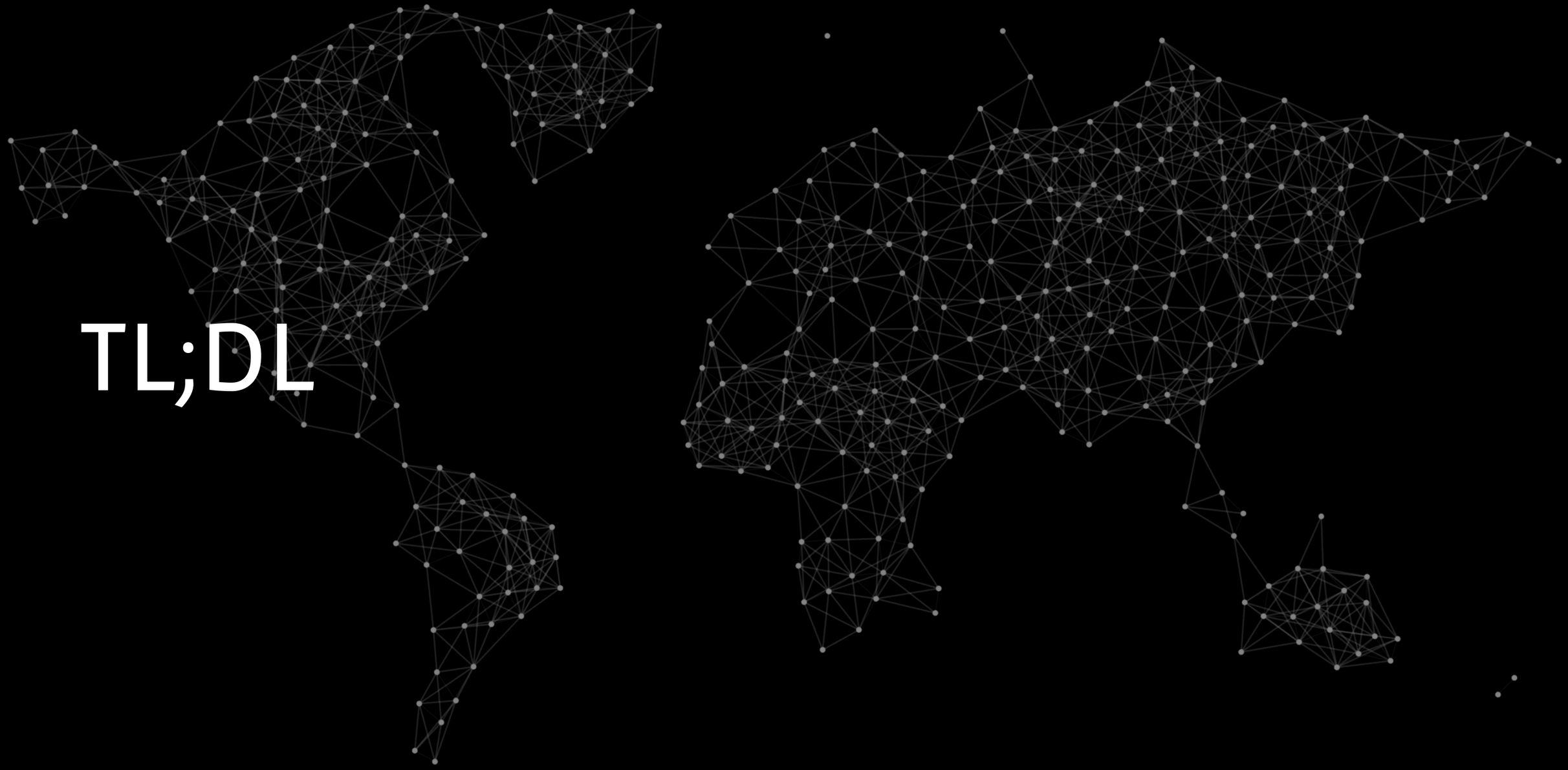
- IPVS (IP Virtual Server)

http://www.linuxvirtualserver.org/software/ipvs.html

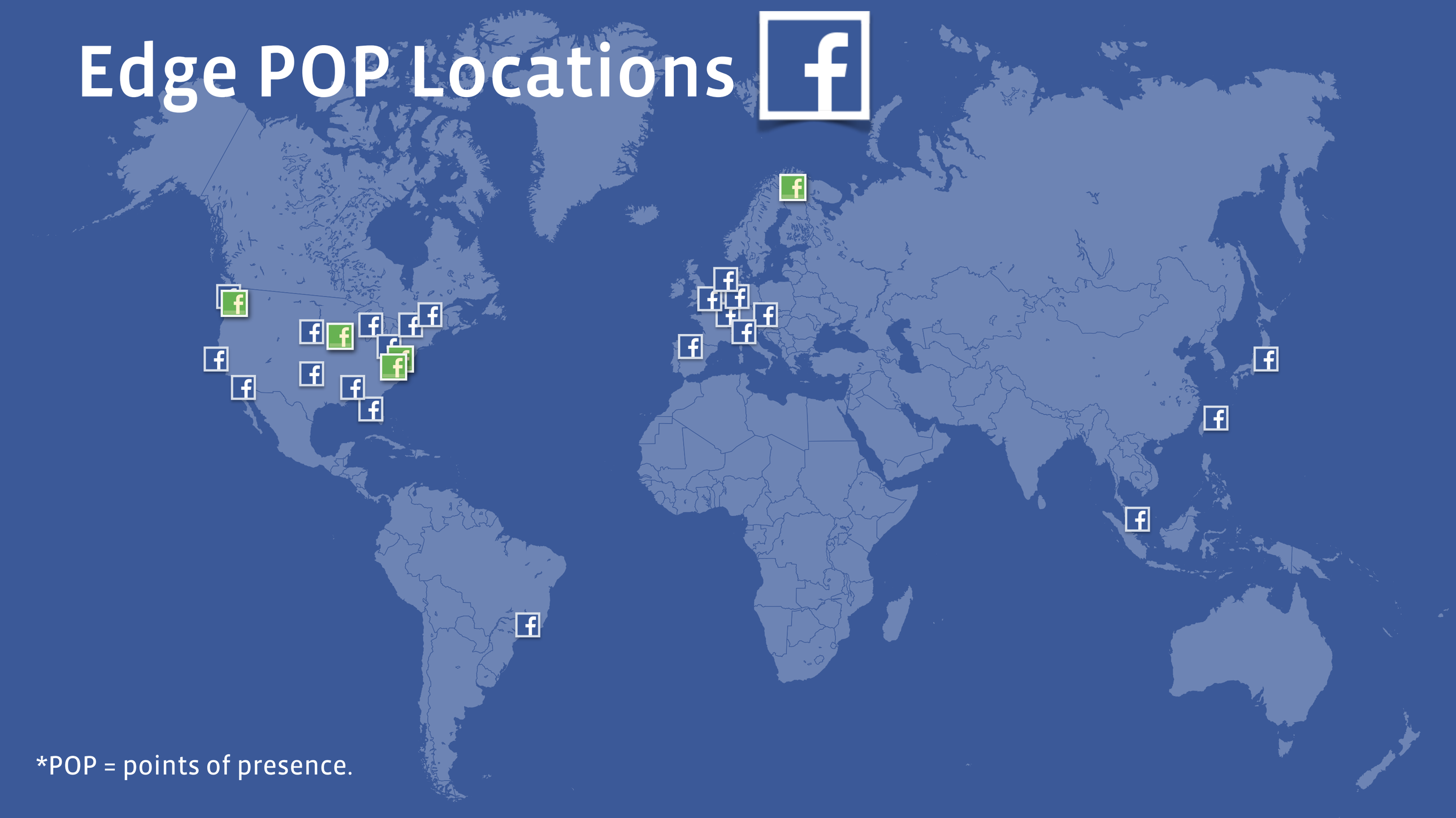
- ExaBGP

https://github.com/Exa-Networks/exabgp

- Python

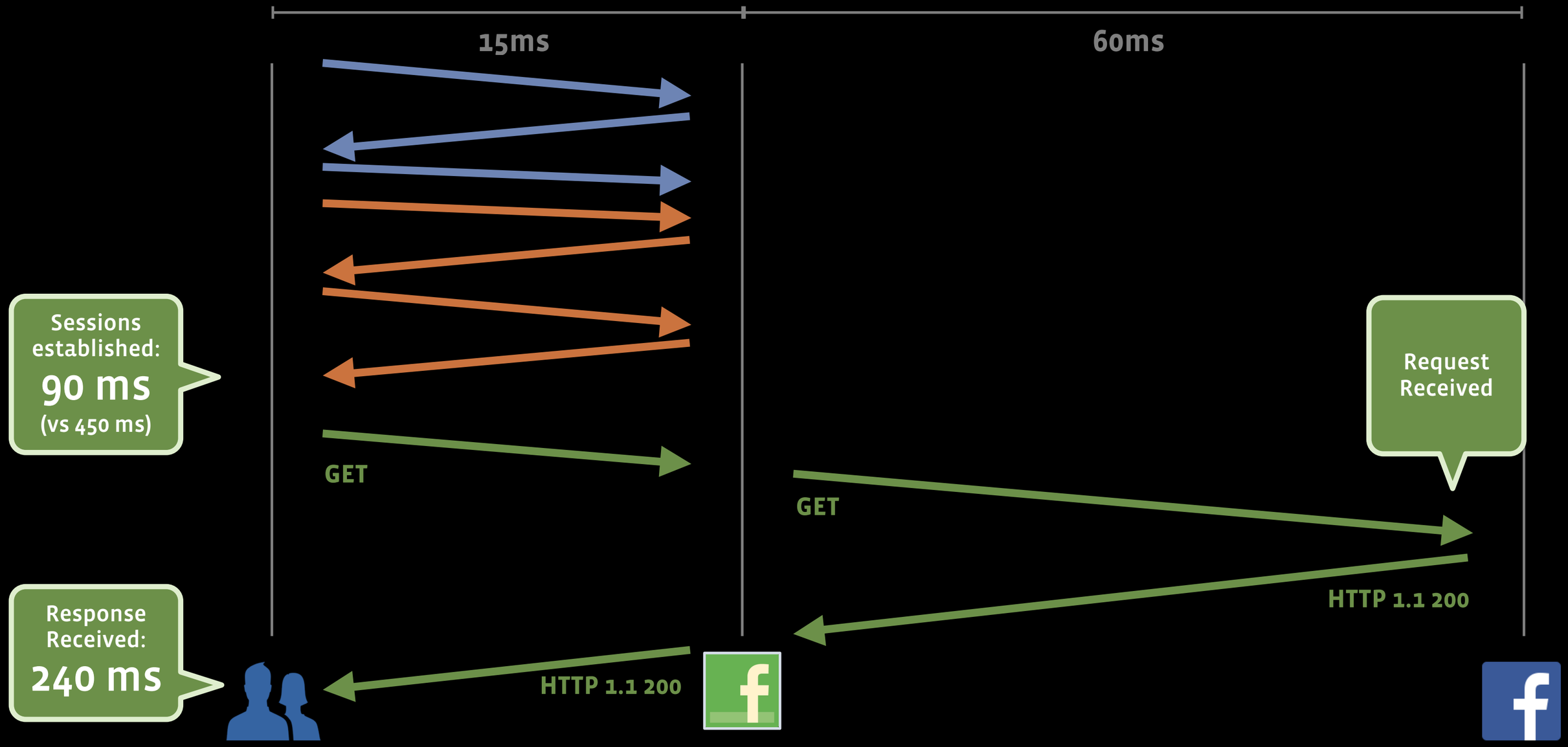

TL;DL

Edge POP Locations



*POP = points of presence.

HTTPS Seoul->Tokyo->Oregon



Direct Server Return

TCP Routing

TCP
SSL
HTTP

Facebook

1.2.3.4
L4LB

1.2.3.4
L7LB

1.2.3.4
L7LB

1.2.3.4
L7LB

1.2.3.4
L7LB

HHVM

HHVM

HHVM

HHVM

3 LBs

- DNS - Cartographer
- TCP - Shiv/IPVS
- HTTP, SPDY - Proxygen

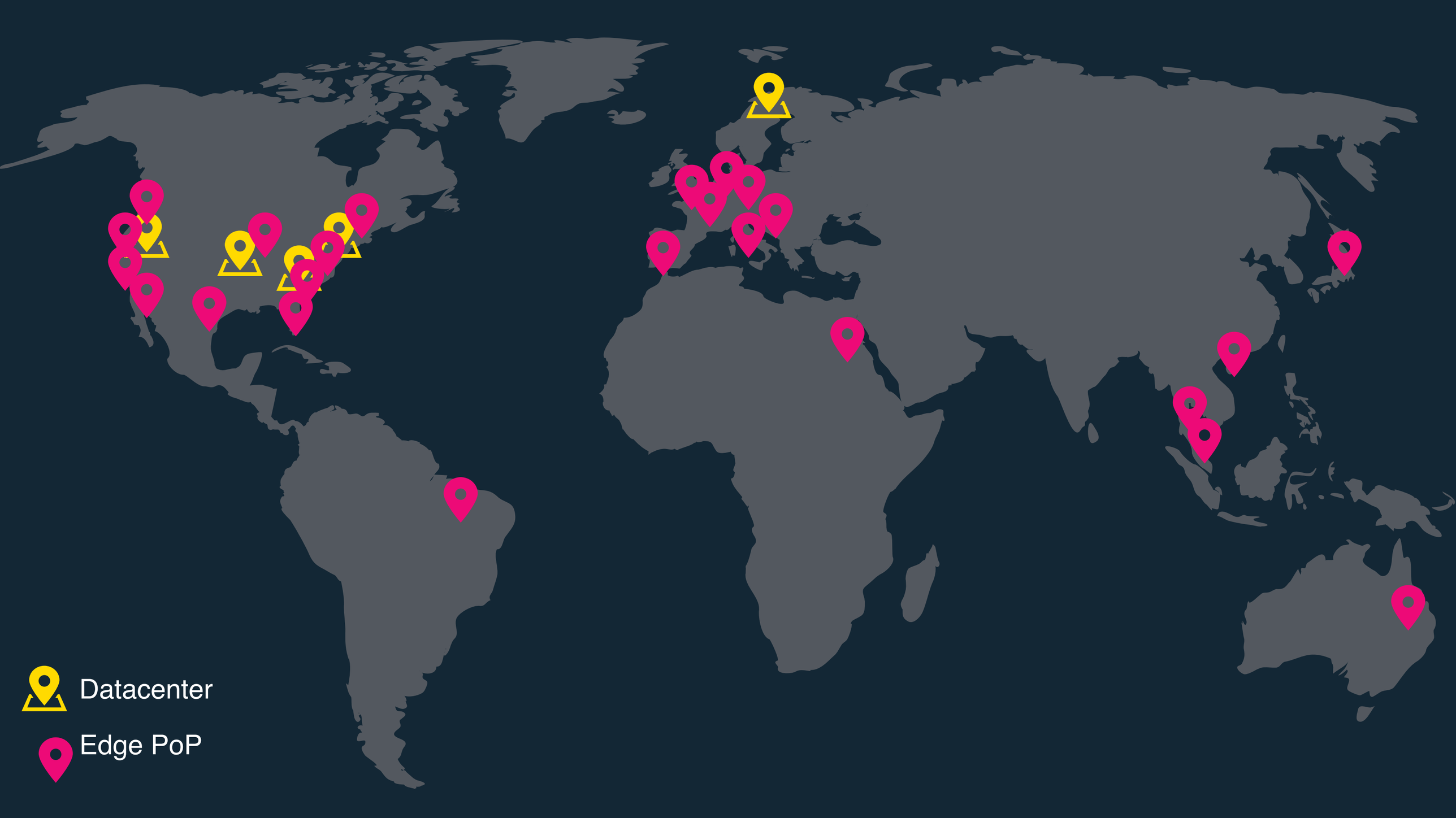
L4LB

L7LB

Did I mention “Highly Available”?


Datacenter

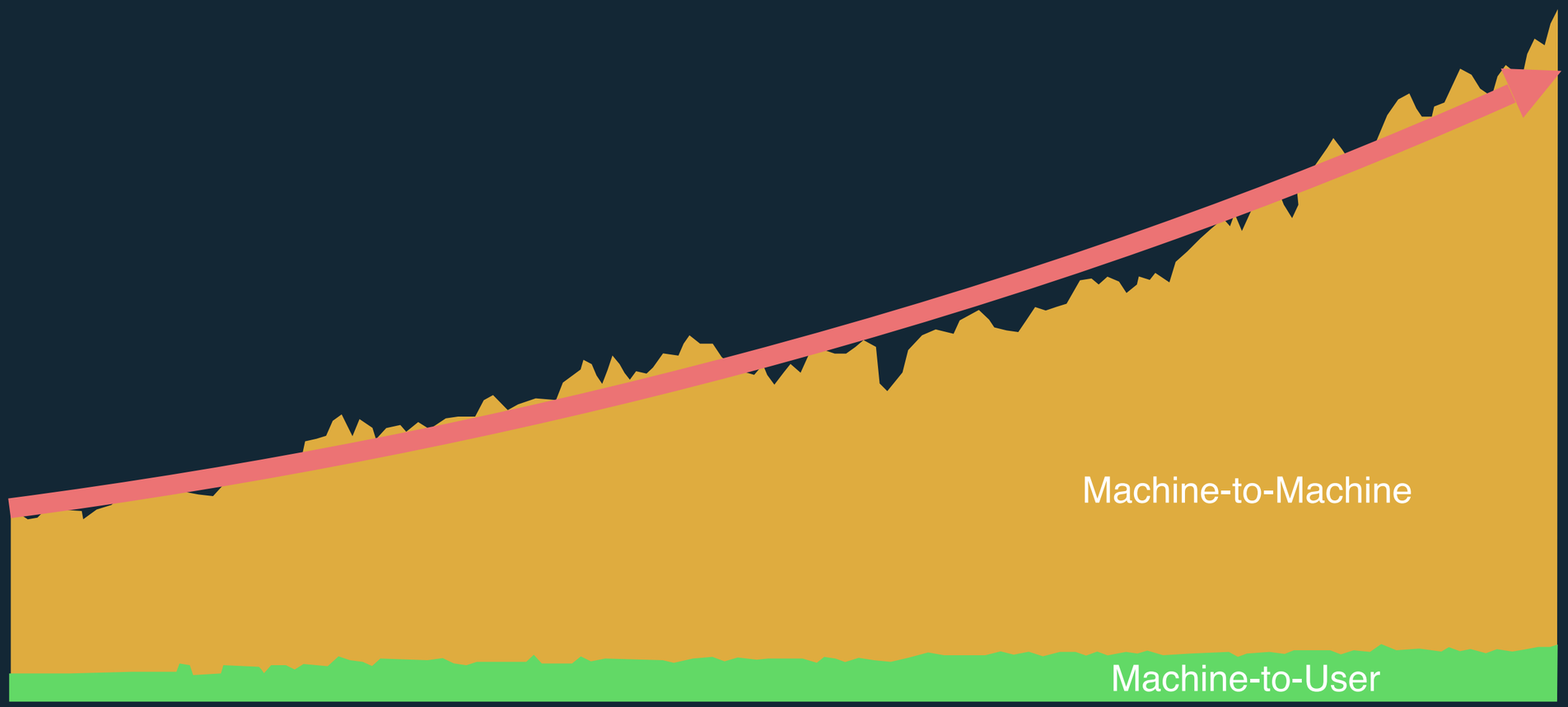
Datacenter


Datacenter
Edge PoP

Datacenter

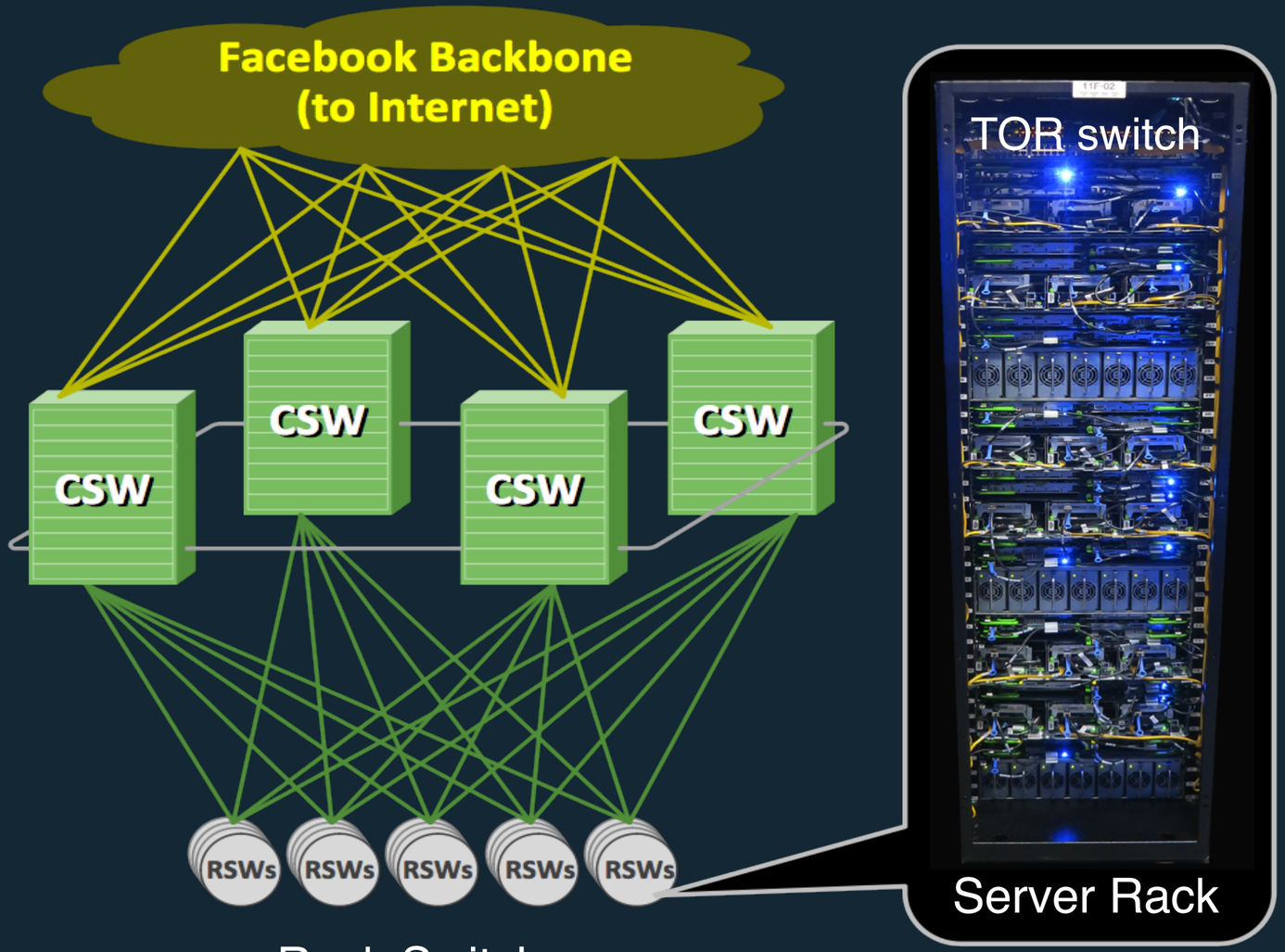
Edge PoP

Rise of the @scale data center network

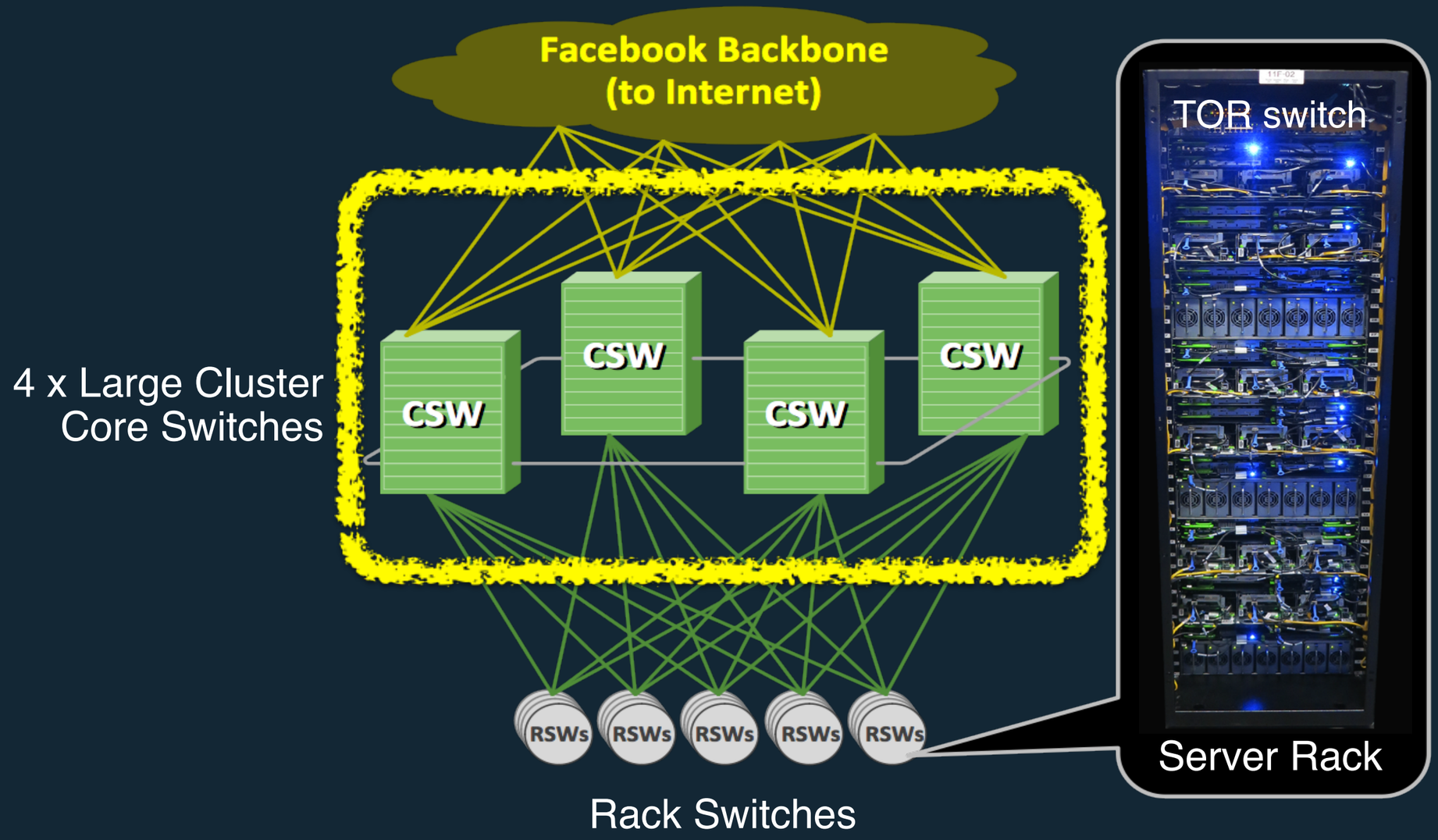


The 4-post cluster - our old design

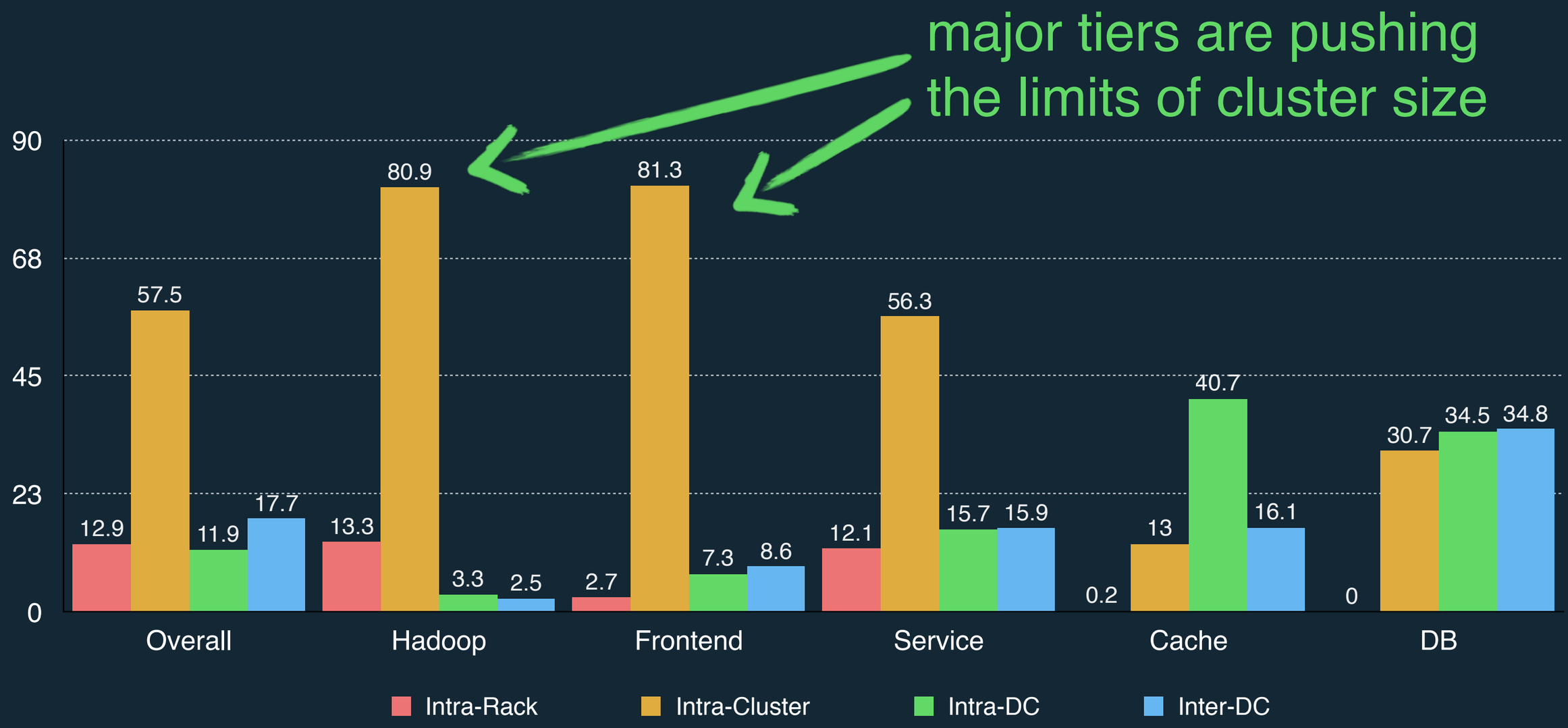
4 x Large Cluster
Core Switches



Box size limited cluster size



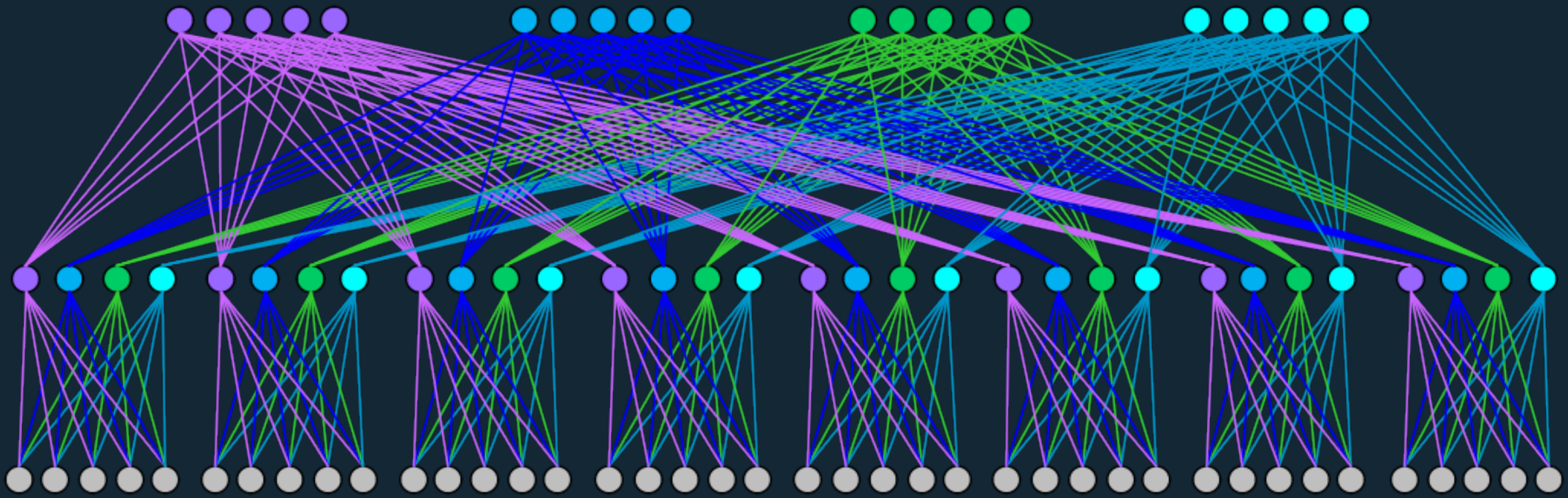
Cluster size limited application size



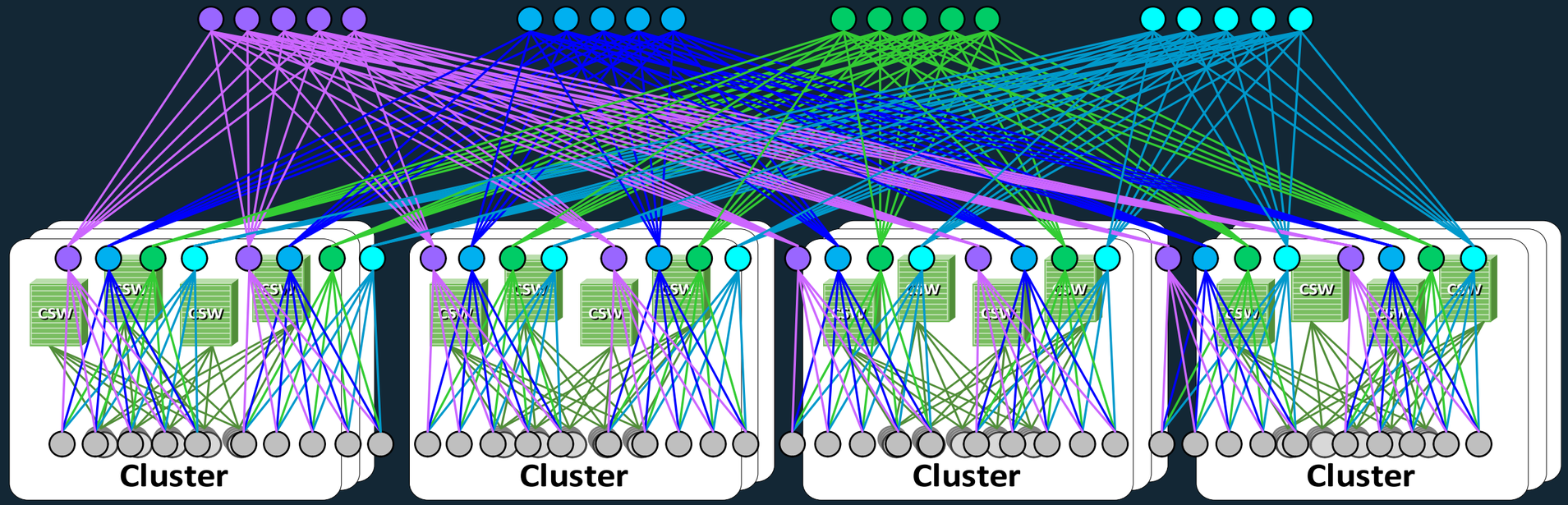
The Vision: The Whole Data Center Network, Redone.

The topology

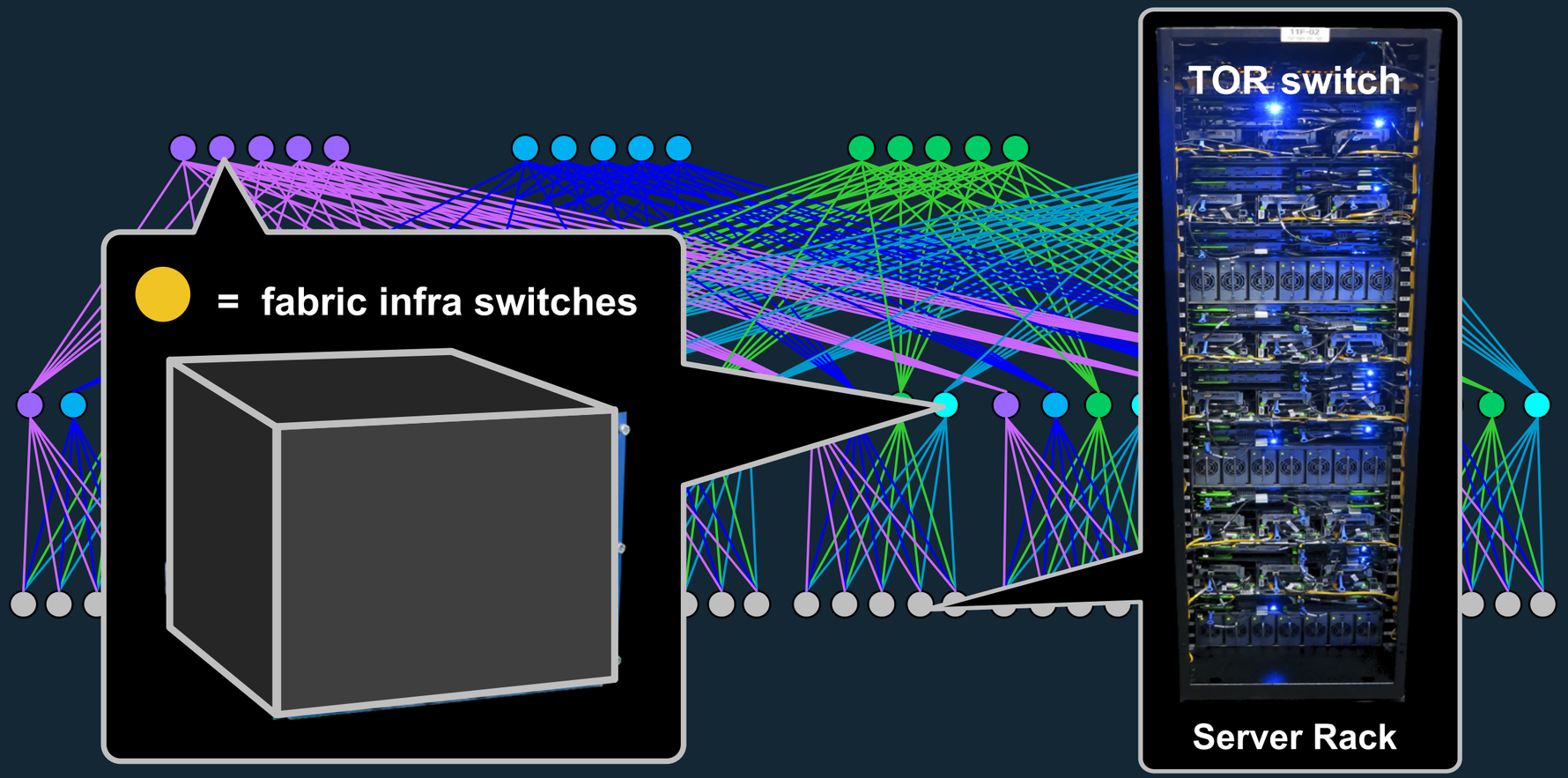
Facebook Fabric:
an innovative network
topology for data centers



The Fabric: one datacenter-wide network



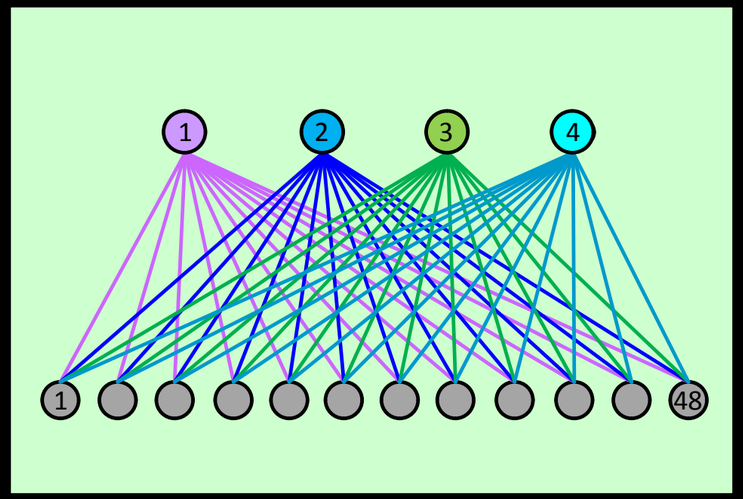
The Fabric: one datacenter-wide network



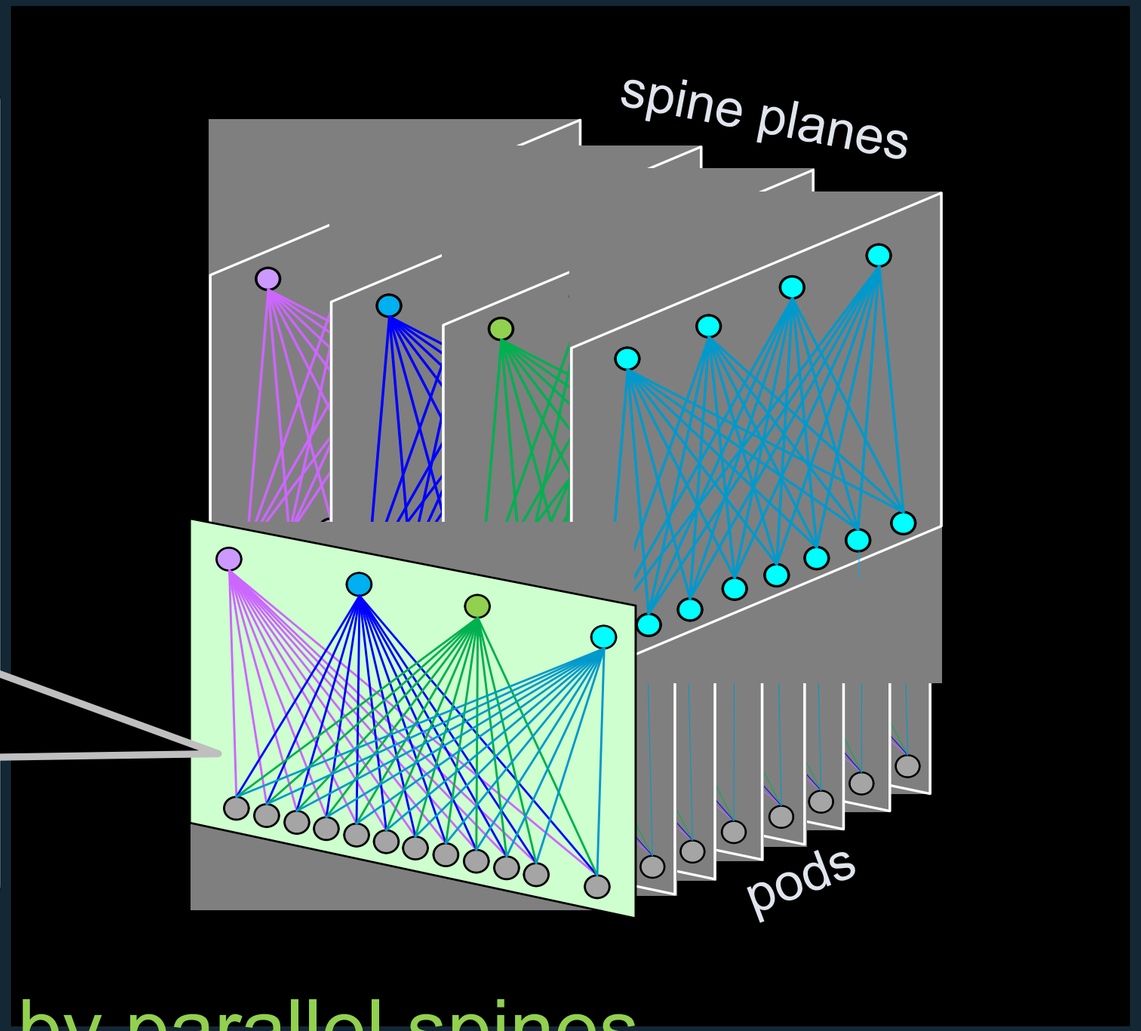
small & simple boxes

Server Pod: a [small] unit of deployment

4 fabric switches

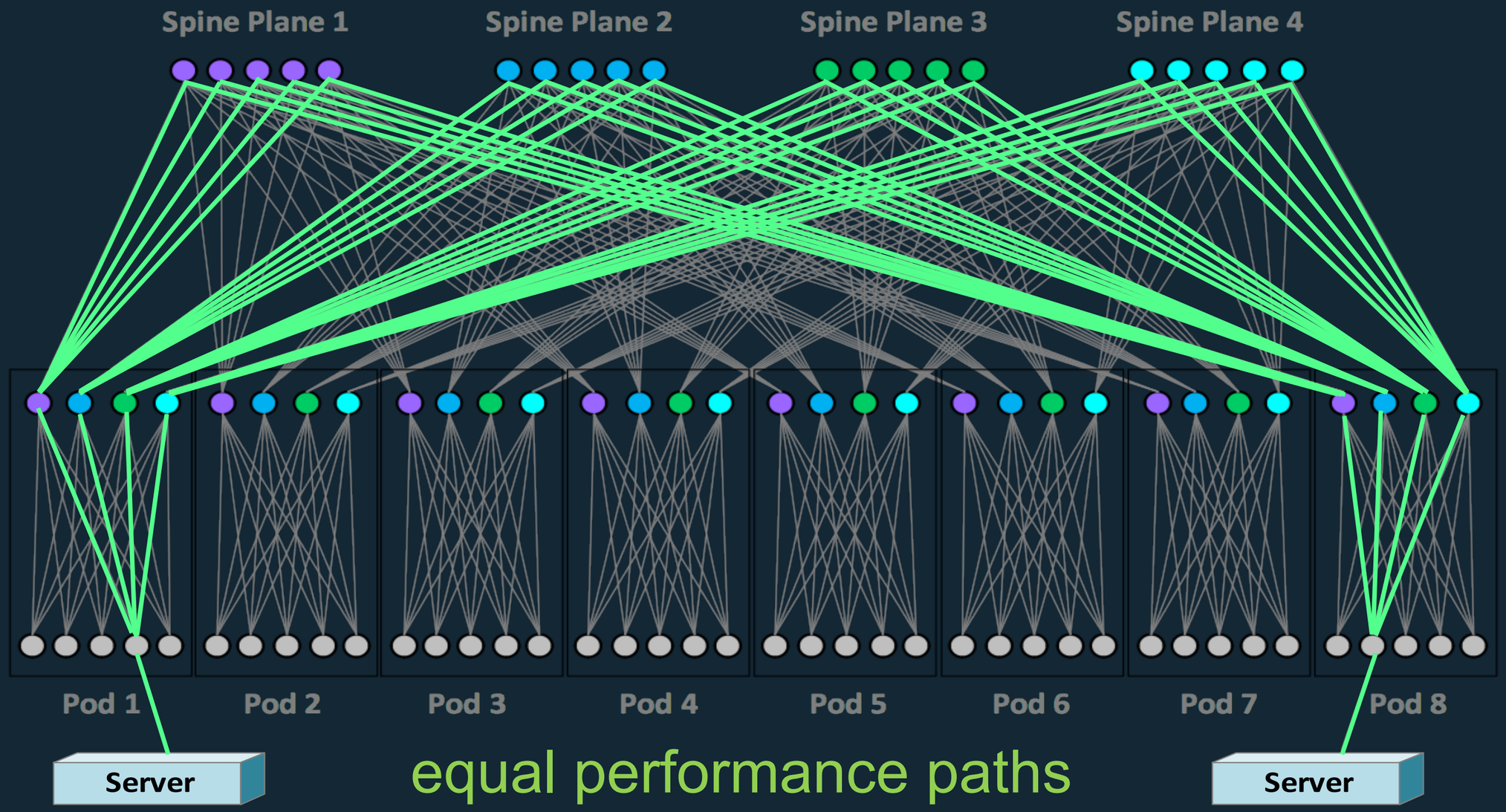


48 rack switches



pods interconnected by parallel spines

Many paths between servers



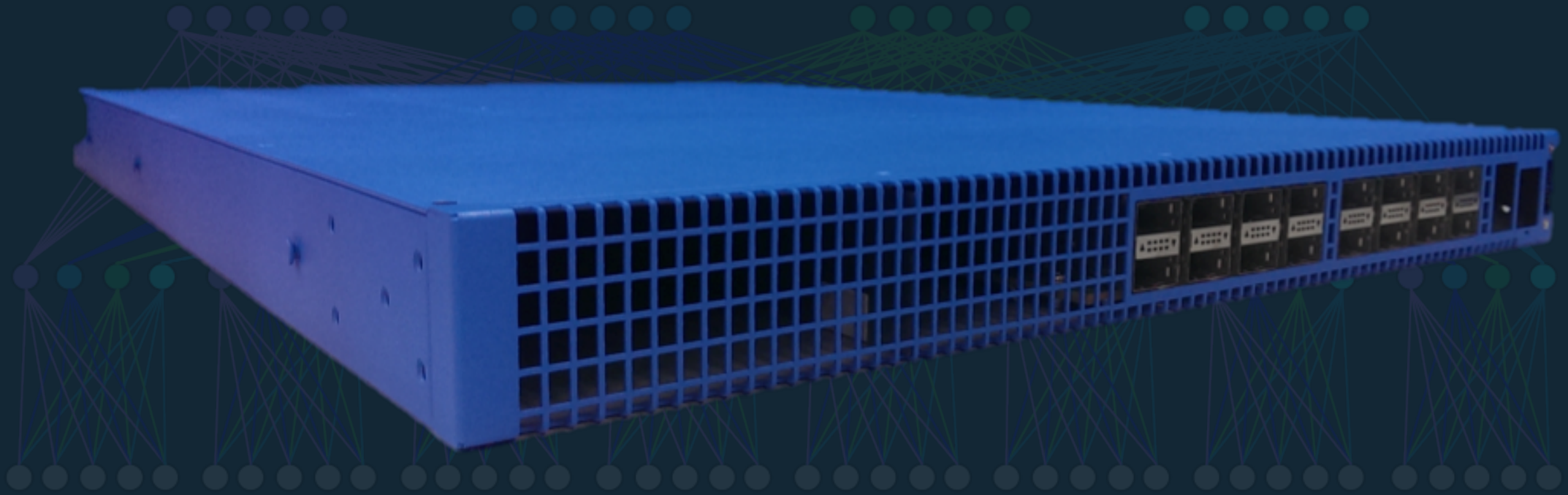
equal performance paths

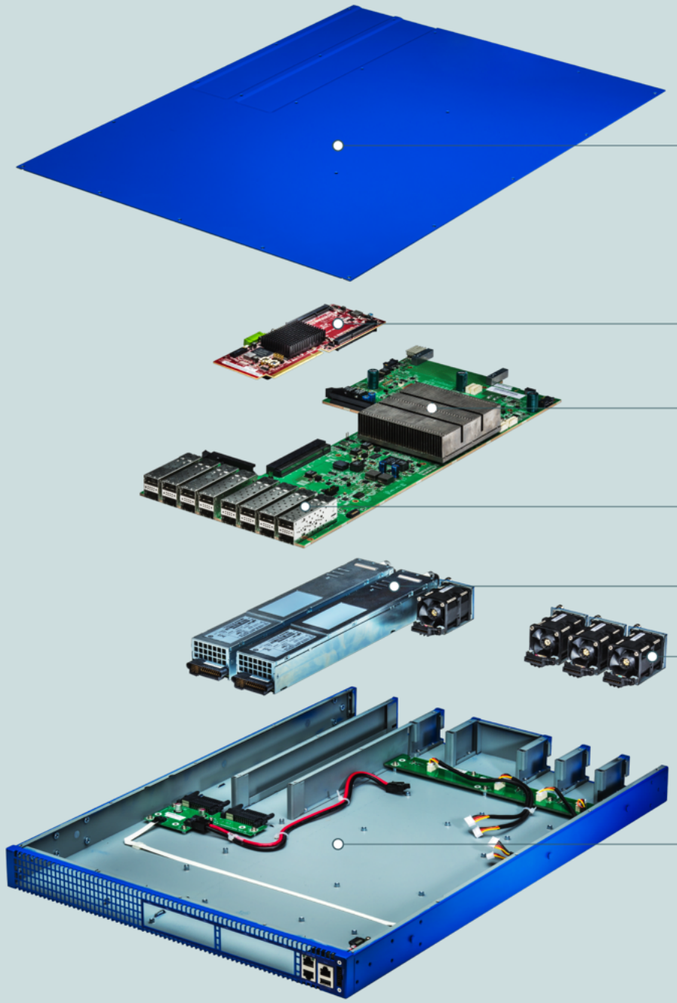
Advantages of Fabric

- Modular/scalable network building block
- More bandwidth capacity - future proof
- Distributed load
- Resilient to failures
  - Individual devices and links are not important

The top-of-rack switch

Facebook Wedge





Wedge Hardware Design

Chassis

Open Compute “Group Hug”
Micro Server

40Gb switching ASIC
Commercially available

Sixteen 40Gb network ports
spaced for optimal airflow

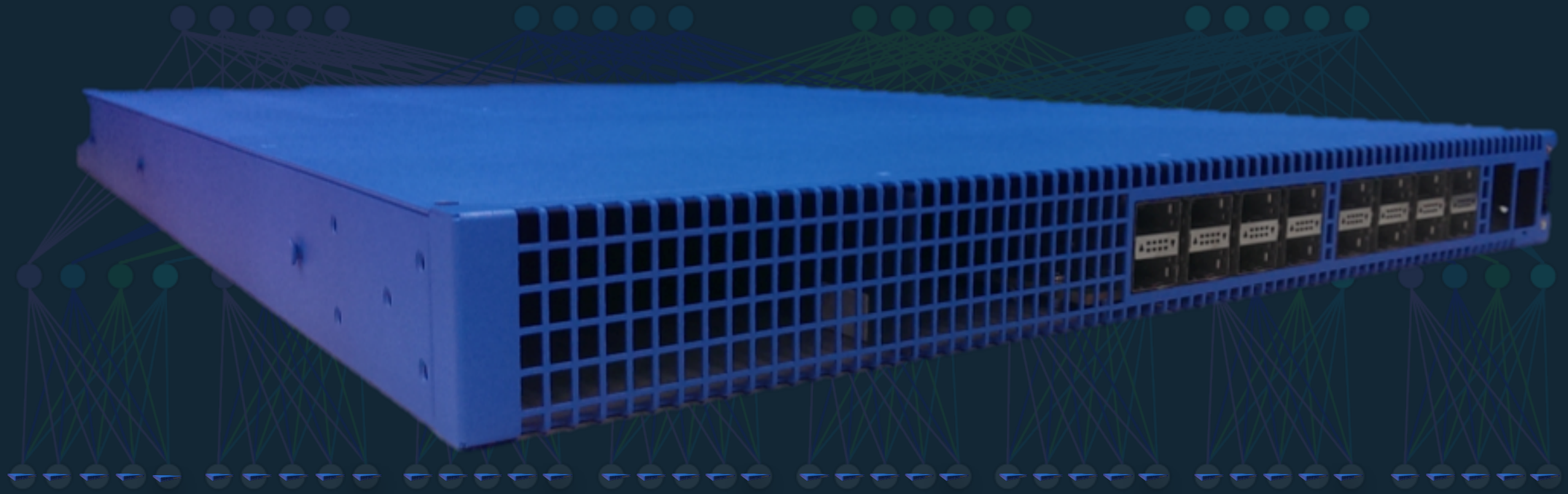
Dual power supplies
with AC and DC options

Fans

Simple enclosure
optimized for efficient cooling

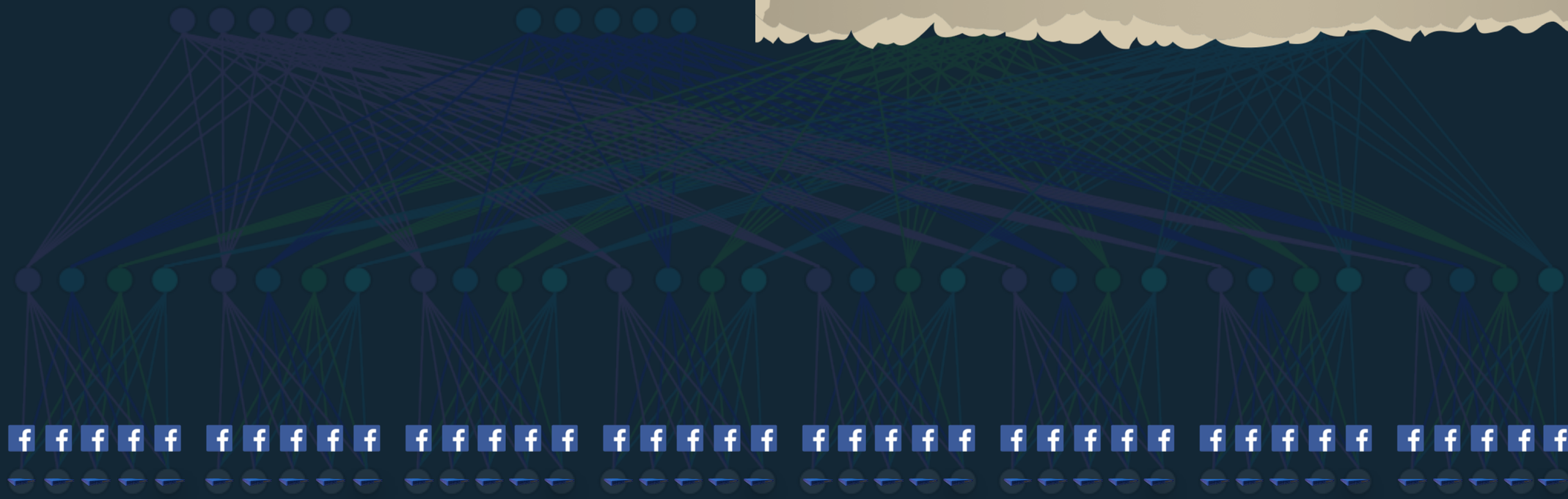
The top-of-rack switch

Facebook Wedge

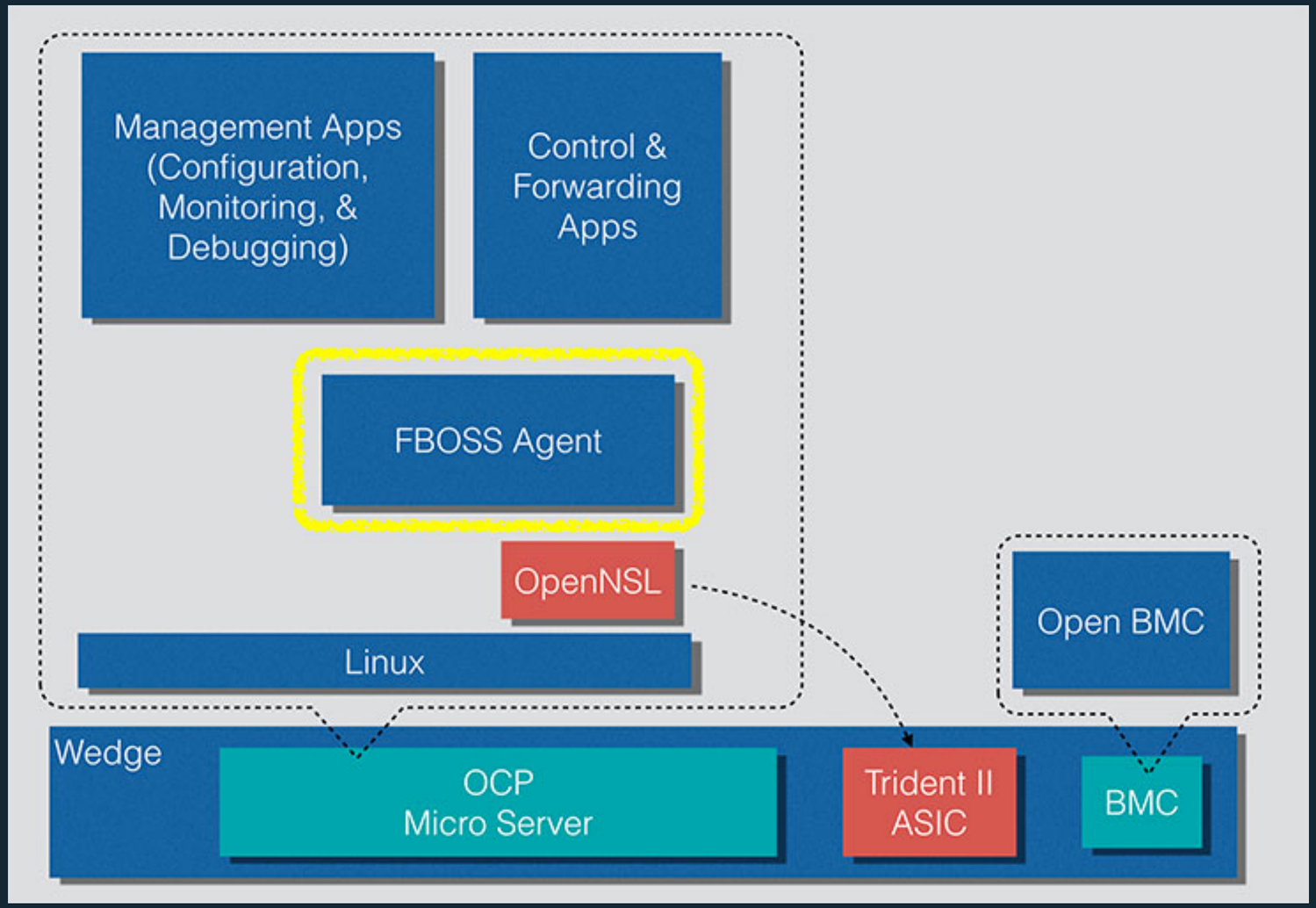


The software

FBOSS:Facebook Open Switching System



FBOSS

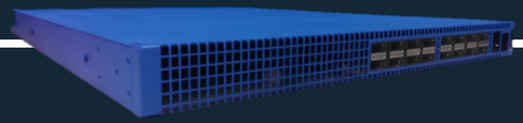

Management Apps (Configuration, Monitoring, & Debugging)
Control & Forwarding Apps
FBOSS Agent
OpenNSL
Linux
Open BMC
Wedge
OCP Micro Server
Trident II ASIC
BMC

6-pack - Core/Spine Switch

6-pack Switch

- First **open** hardware **modular switching** platform
- 128x40GE non-blocking switch
- Runs FBOSS over Linux
- Modular
  - 12 independent Wedges
  - 4 fabric, 8 front-panel
- 100G ready

Data Center Networking Summary



FBOSS
& BMC

OPEN
Compute Project

1 From Wedge

2 We built 6-pack

3 FBOSS & OpenBMC

4 OCP based eco system

5 Open hardware & software


TRUE, OPEN NETWORK SW
ECOSYSTEM

facebook

Photo Credits

http://www.flickr.com/photos/27587002@N07/5170590074

http://www.flickr.com/photos/yaketyyakyak/7001664846

http://www.flickr.com/photos/hinnosaar/3778903507

http://www.flickr.com/photos/eamoncurry/8698726494

http://www.flickr.com/photos/43158397@N02/4514113429

http://www.flickr.com/photos/nobusue/6876280595

http://www.flickr.com/photos/29487672@N07/14760573314

http://www.flickr.com/photos/joyosity/3595242078

http://www.flickr.com/photos/kyntharyn74/3262089319

http://www.flickr.com/photos/rexipe/826987087

http://www.flickr.com/photos/lablasco/6815671096