Beating the Speed of Light with Intelligent Request Routing

Sergey Fedorov
QConPlus
May 2021

Why does Network Speed Matter?

+100ms = 1%

latency

sales loss

> ***A 100 ms latency penalty implies a 1% sales loss.***

Amazon, 2009

Liddle, J.: Amazon Found Every 100ms of Latency Cost Them 1% in Sales. http://goo.gl/BUJgV

Why does Network Speed Matter?

+3s = -53%

Page load User traffic

> ***Starting in July 2018, page speed will be a ranking factor for mobile searches.***

Google, 2018

https://webmasters.googleblog.com/2018/01/using-page-speed-in-mobile-search.html

Why does Network Speed Matter?

$400M investment to reduce New York to London latency by **6ms**

That's **$66M** per ms!

The reduction in latency can improve performance, and even improve profitability and increase sales for some customers.

Hibernia Networks, 2015
https://www.submarinenetworks.com/en/systems/trans-atlantic/project-express/hibernia-express-connects-new-york-to-london-in-under-58-95ms

What about Netflix?

CDN

Static Content Delivery with Open Connect CDN

Video streaming

Small objects delivery

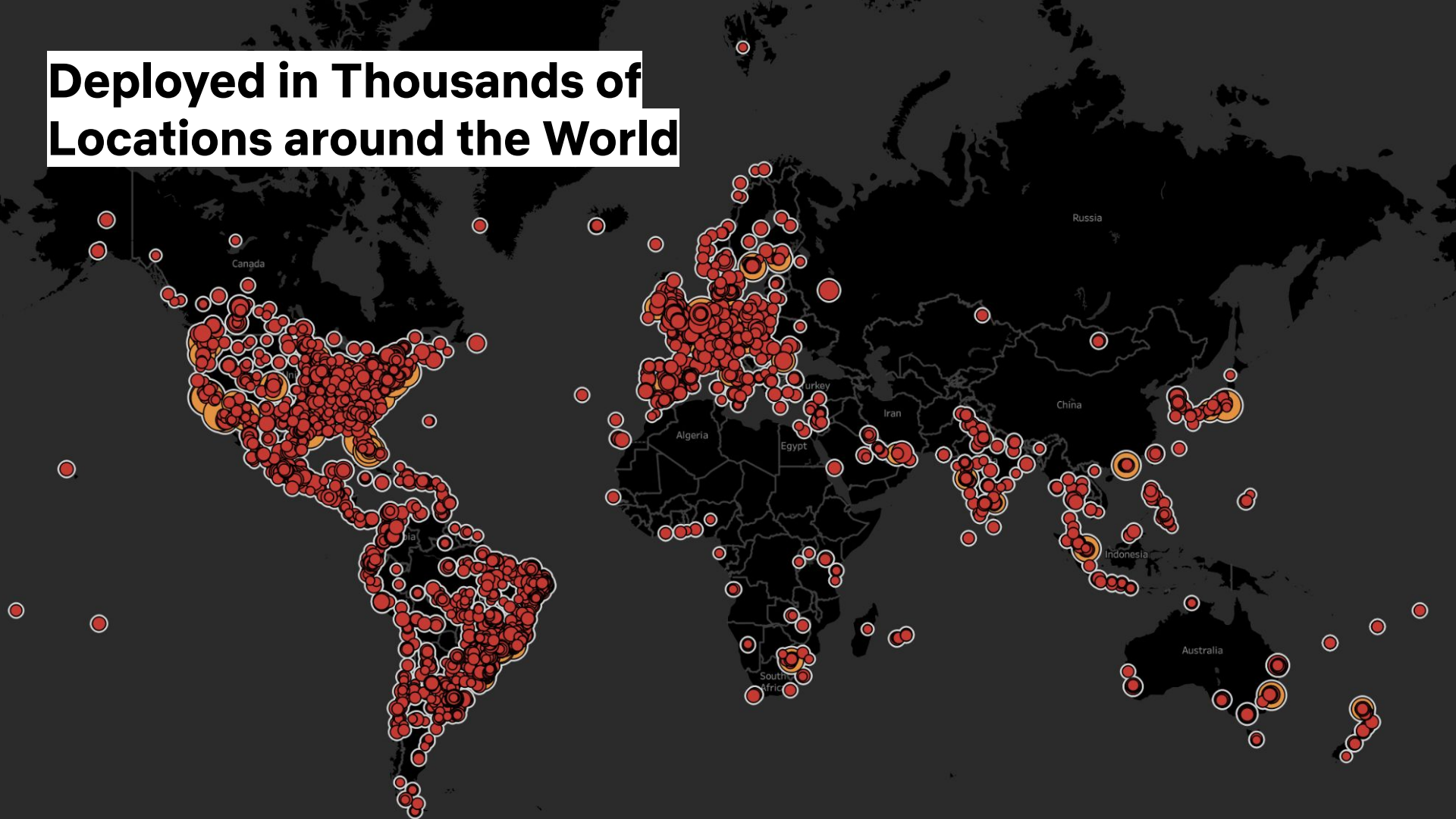
DNS

Open Connect Appliance (OCA)

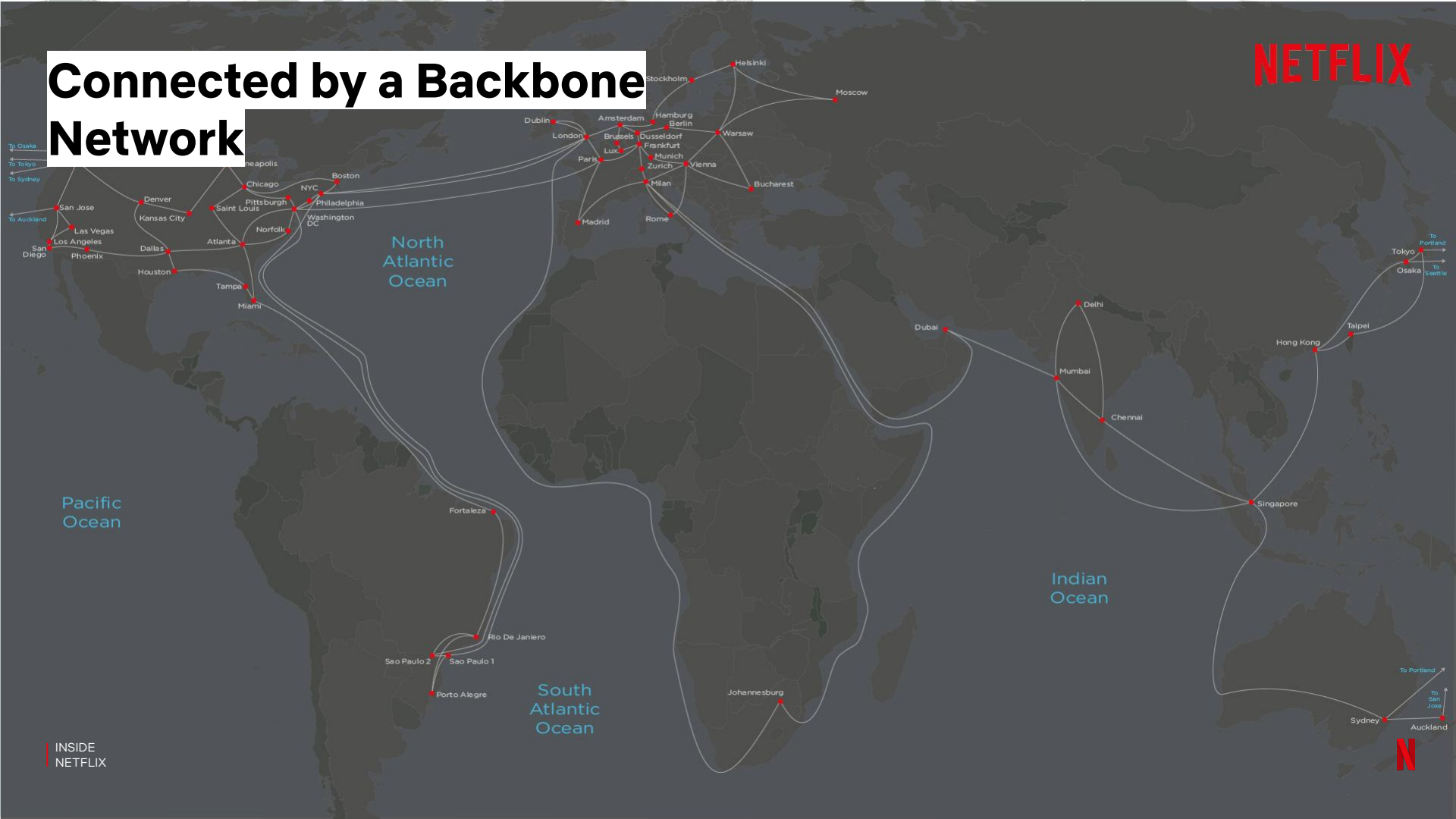
FreeBSD + Nginx
Lots of SSDs/HDDs
Optimized for max network throughput

This rack can serve around 1% of Internet traffic

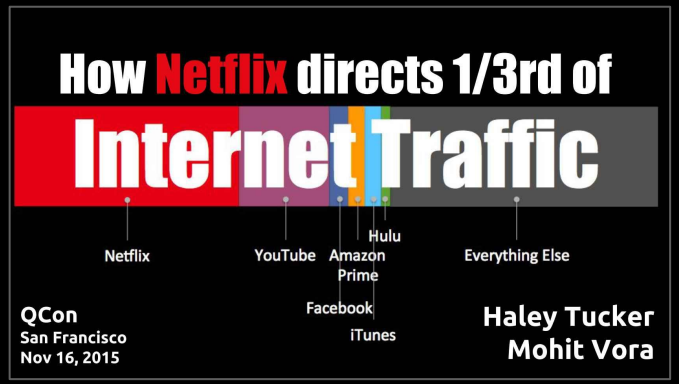
Deployed in Thousands of Locations around the World



Connected by a Backbone Network

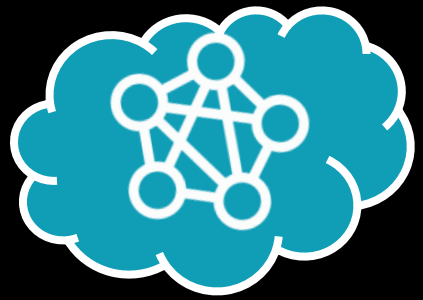


Learning More about Open Connect and Streaming Delivery



HTTP APIs

CDN



UI Personalization is Powered by the AWS Cloud

Hundreds of microservices

Personalization

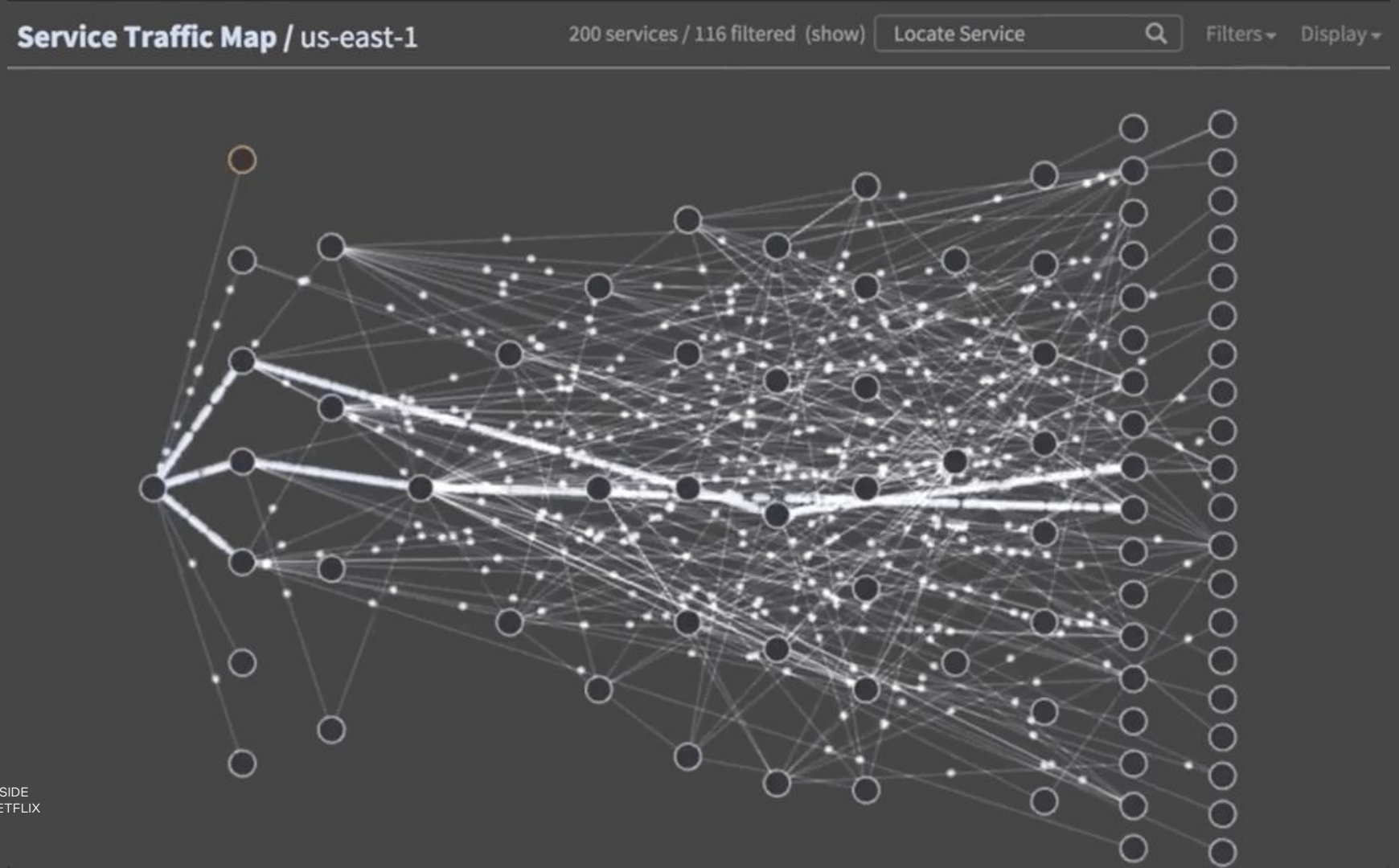
Control plane

Telemetry

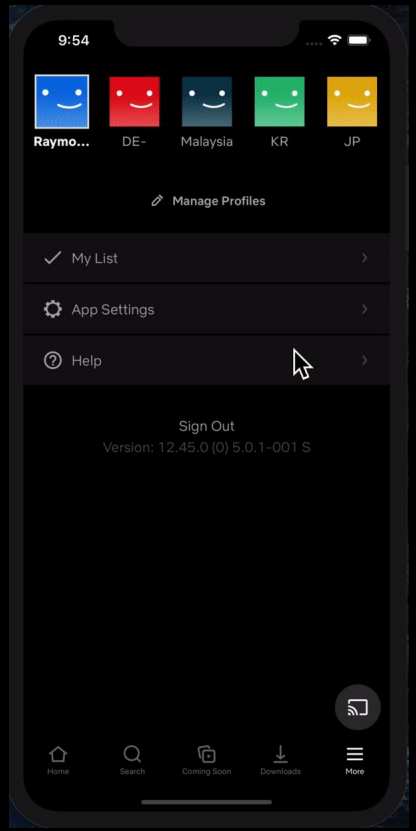
Big Data

Encoding

Cloud Services: 3 AWS Regions


Service Traffic Map / us-east-1
200 services / 116 filtered (show)
Locate Service
Filters
Display

Personalized network requests contribute up to 40% of time to render Netflix home page



How can we leverage our CDN edge infrastructure to make device-cloud requests faster?

- With **minimal impact** to device or server teams
- For **non-cacheable** API requests
- **Low** operational **overhead**

About me

Sergey Fedorov

Director of Engineering, Content Delivery

- 8+ years at Netflix
  - CDN Monitoring
  - FAST.com
  - API acceleration
  - QoE optimization
- Past: Intel/Microsoft

SPEAKER
INFO

What will I talk about?

1. How to reduce network latency using edge infrastructure
2. How we did it at Netflix
3. How **YOU** can learn from our experience

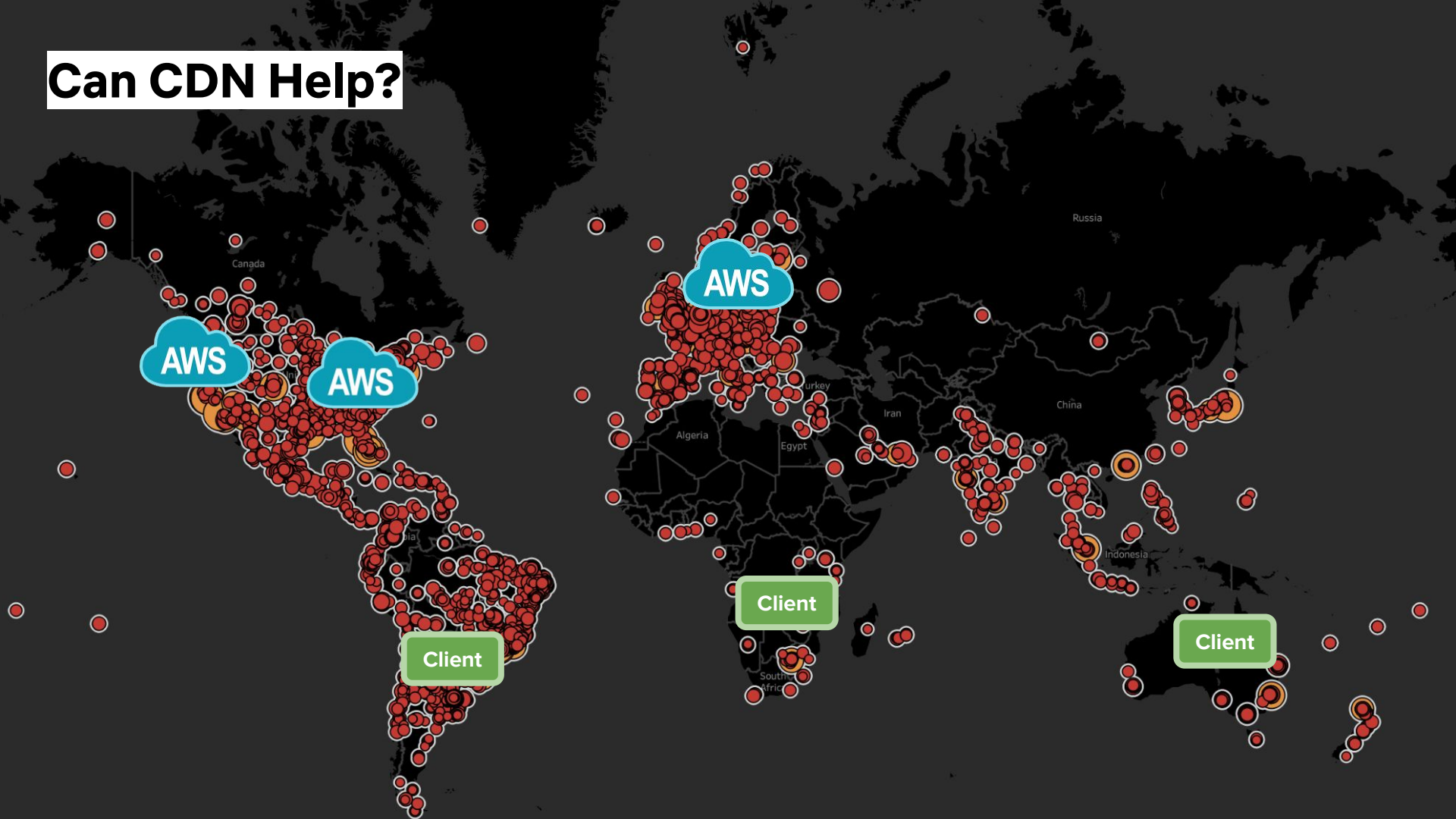
How to accelerate requests using CDN Edge

1

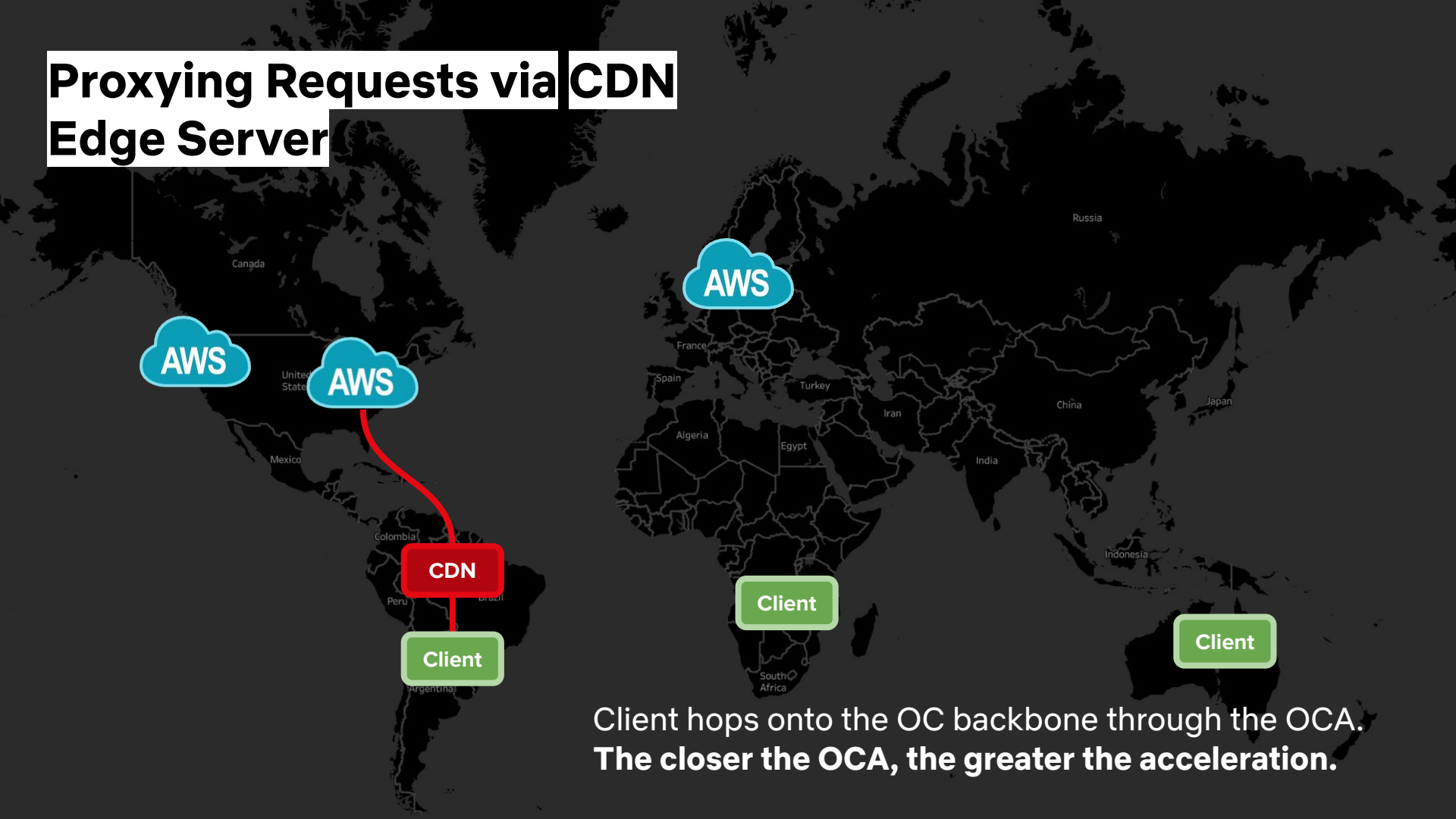
THEORY:
EDGE ACCELERATION

Cloud Services: 3 AWS Regions

Can CDN Help?


AWS
AWS
AWS
Client
Client
Client
Canada
Russia
China
Iran
Algeria
Egypt
Indonesia

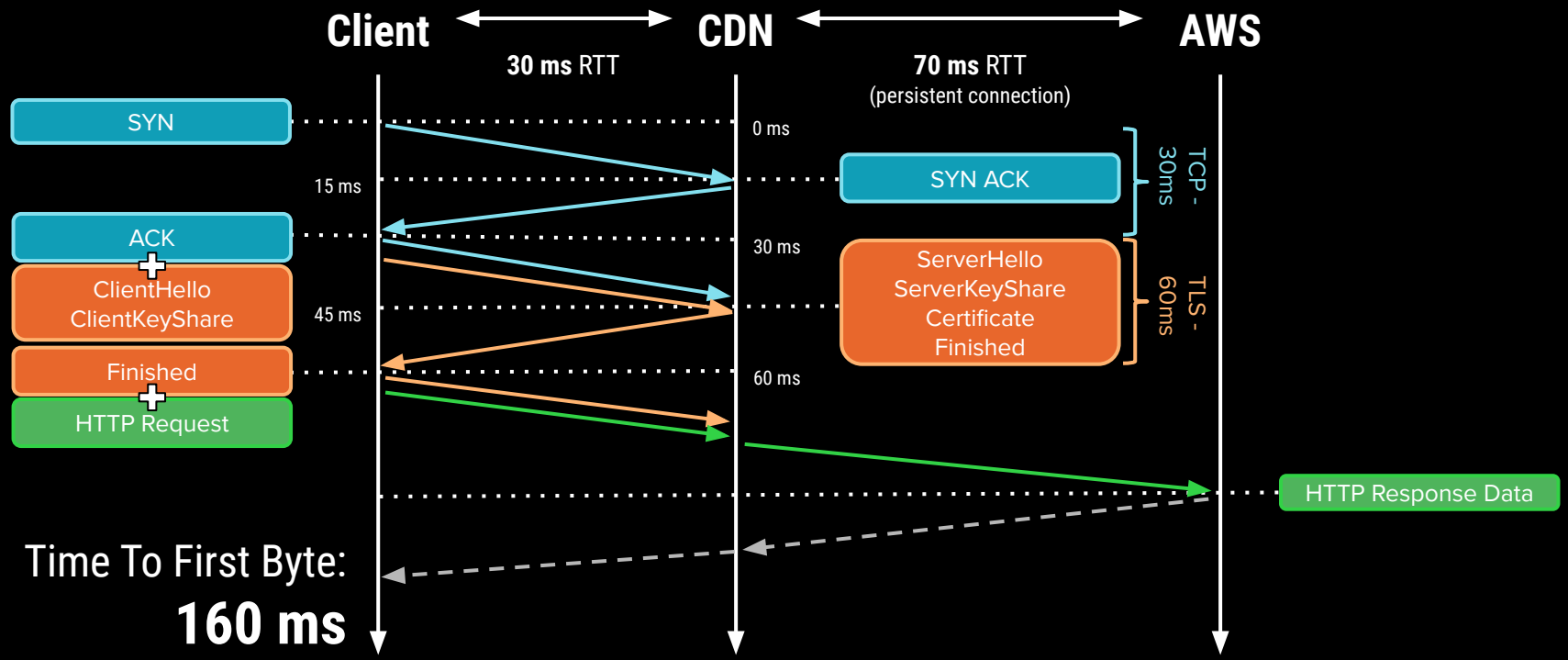
Proxying Requests via CDN Edge Server



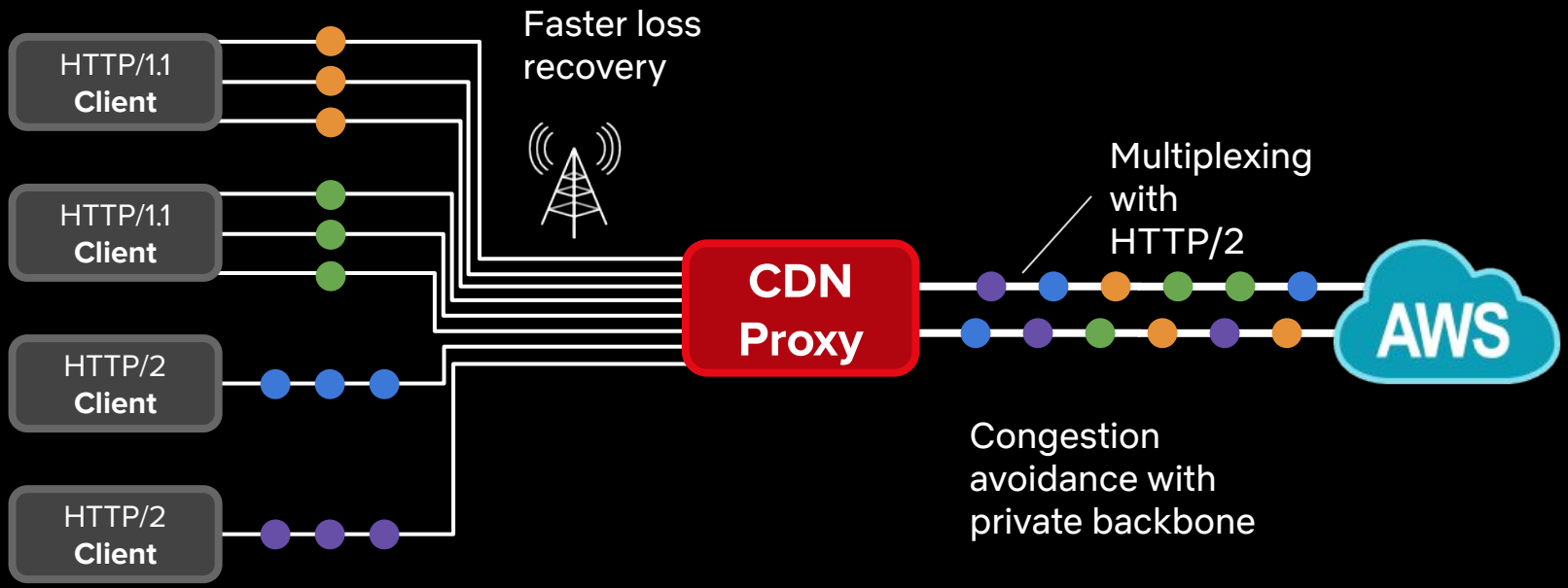
Client hops onto the OC backbone through the OCA.
The closer the OCA, the greater the acceleration.

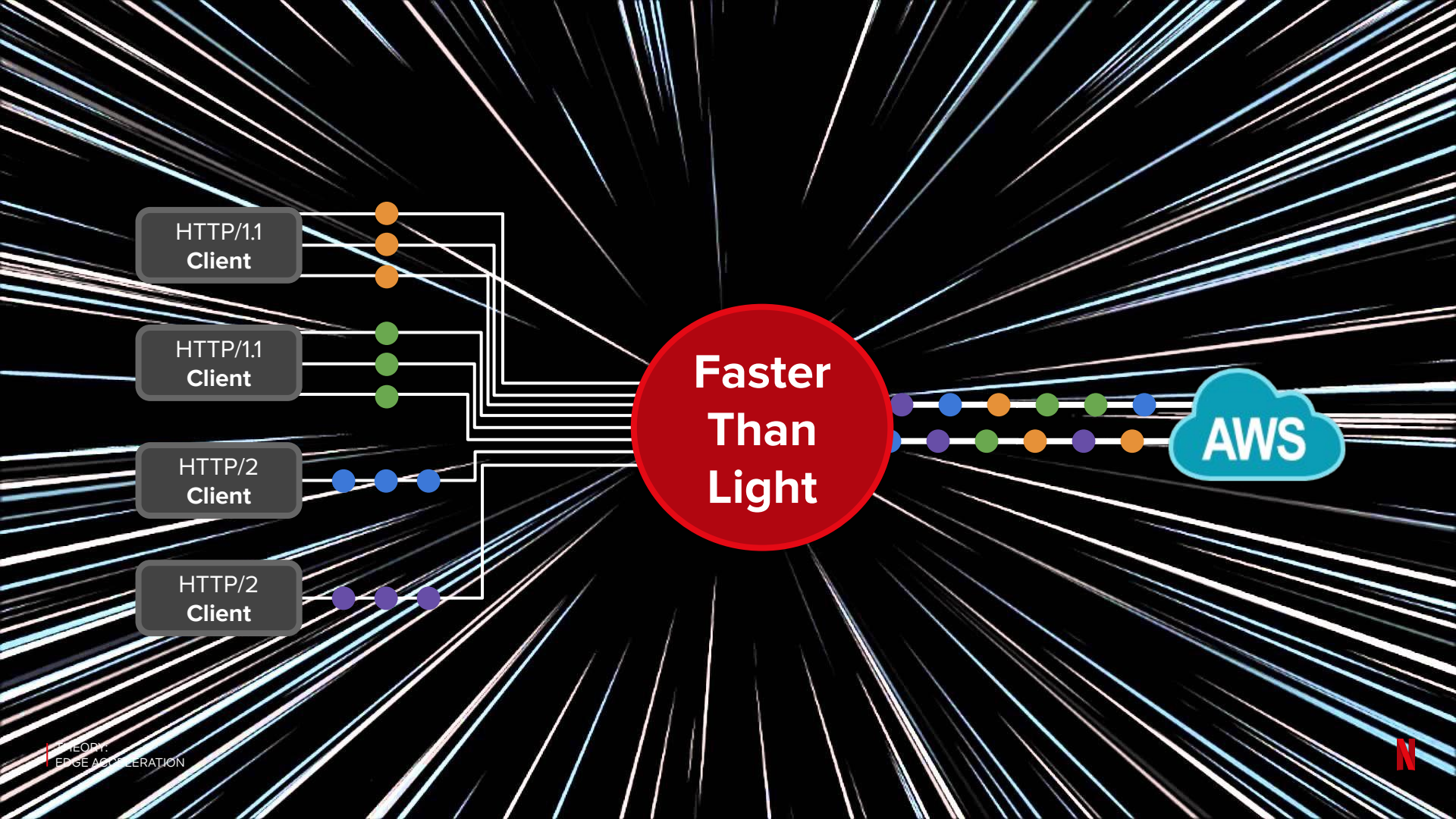
Background: TCP + TLS Connection Establishment Time Depends on Distance to a Server

Reducing Connection Establishment Overhead with a CDN Proxy



Reducing Data Transfer Times




HTTP/1.1
Client
HTTP/1.1
Client
HTTP/2
Client
HTTP/2
Client
Faster
Than
Light
AWS

Let’s launch it?!

How We did it at Netflix

2

NETFLIX
SECTION DIVIDER

Measure

Measure what you can.
Isolate the rest.

Measure

Goal: understand network connectivity between Netflix devices and servers.

aka build a latency map of the Internet for Netflix users

… and use it to compare different routing options

Network Measurement System Requirements

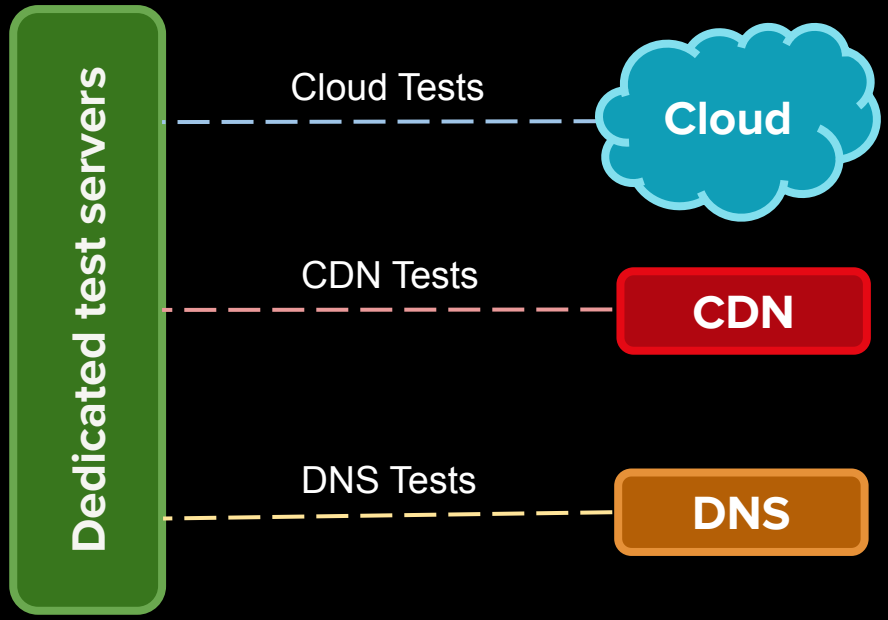
- Estimates for **our current users**.
- Full **device coverage**.
- Quickly.
- Don’t break production.

Real User Monitoring

100% geo
100% devices

Noisy signal
Only production traffic

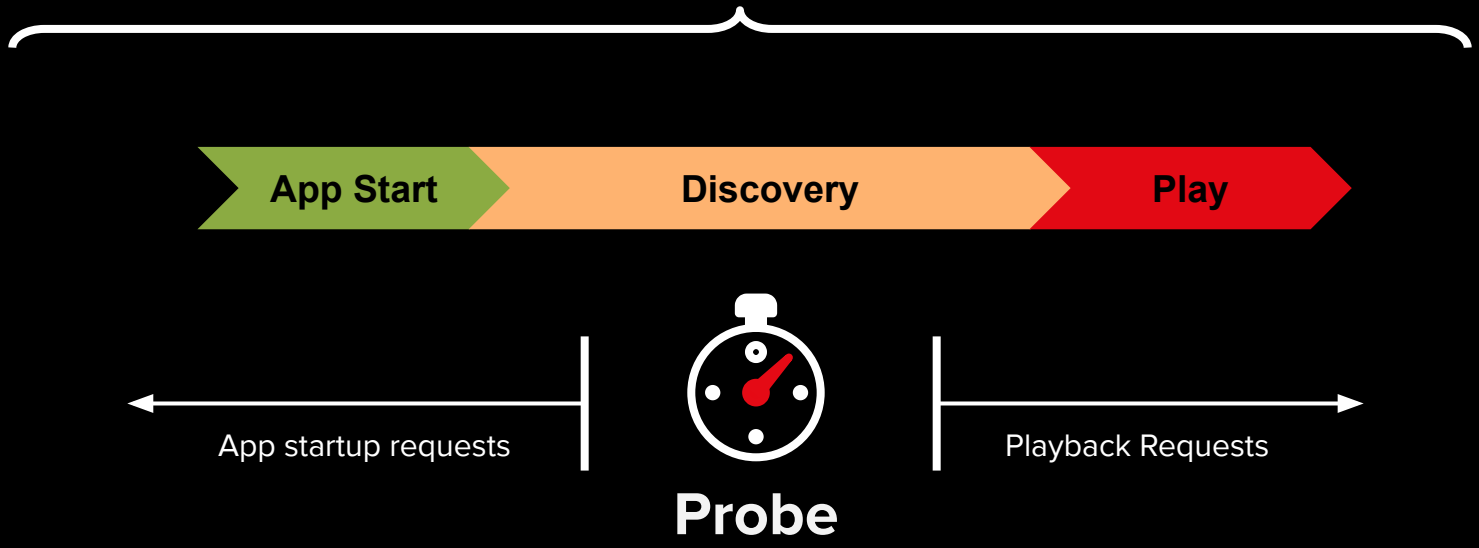
Synthetic Monitoring



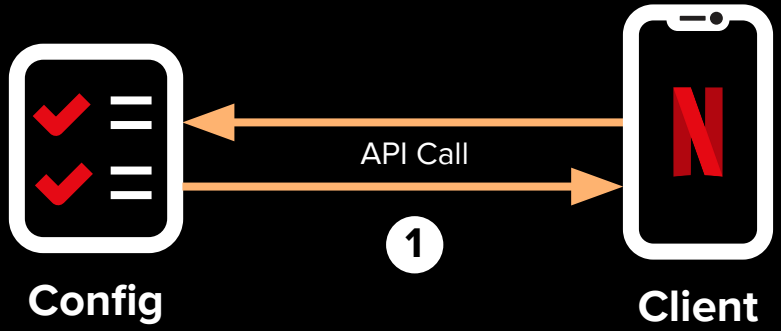
Full control
Clean signal

Partial geo
Limited device coverage

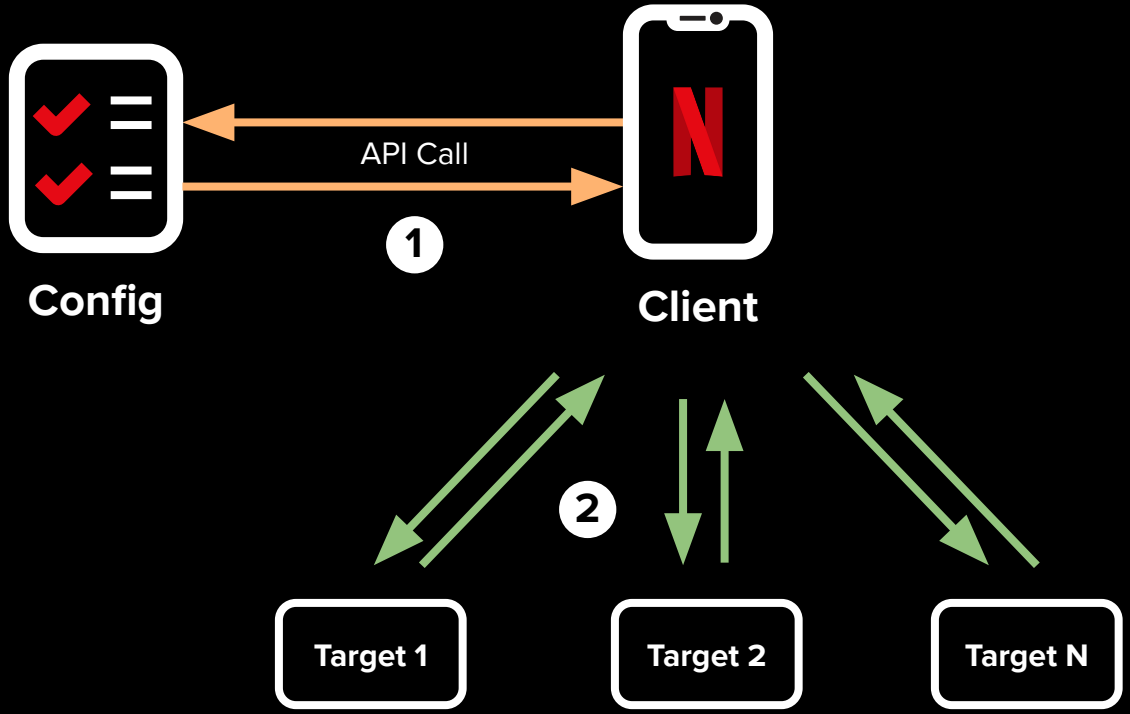
Probnik =
RUM + Synthetic



Step 1:
Client Gets a Test Recipe



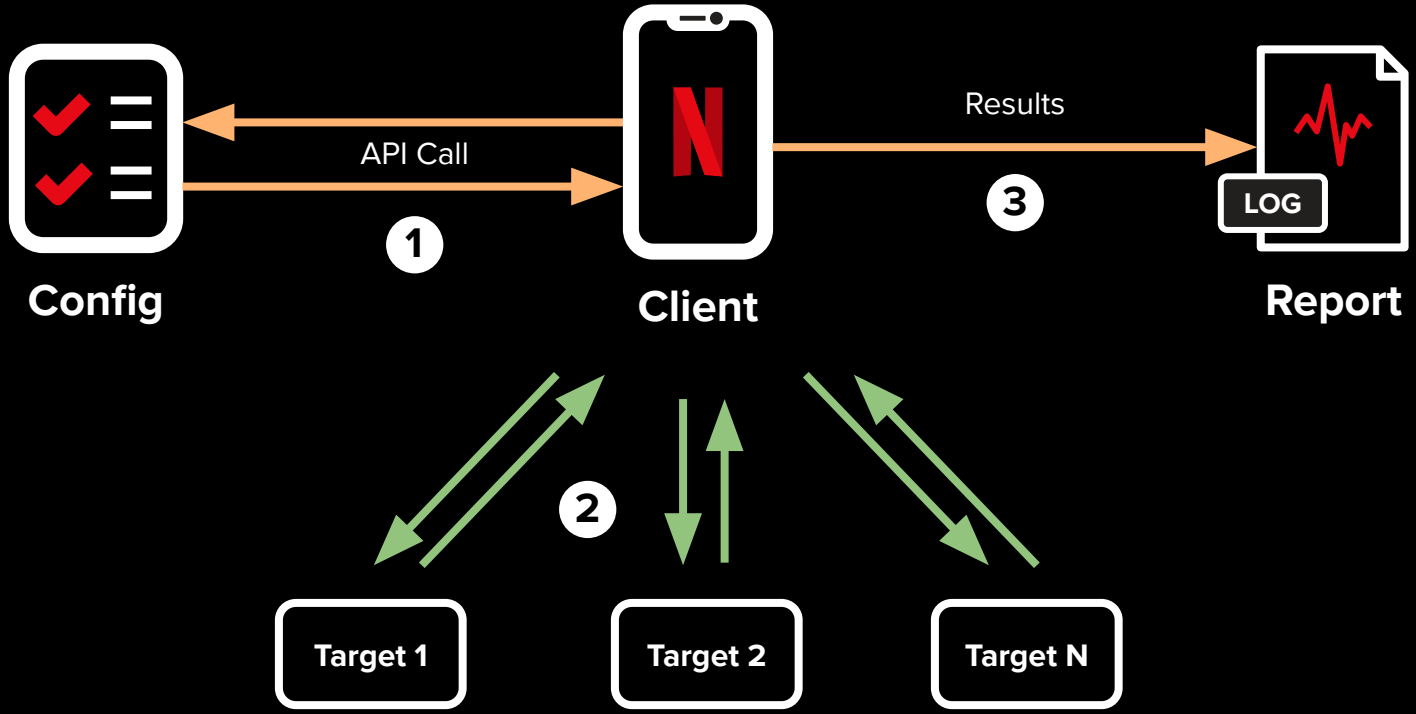
Step 2:
fetch and measure request time



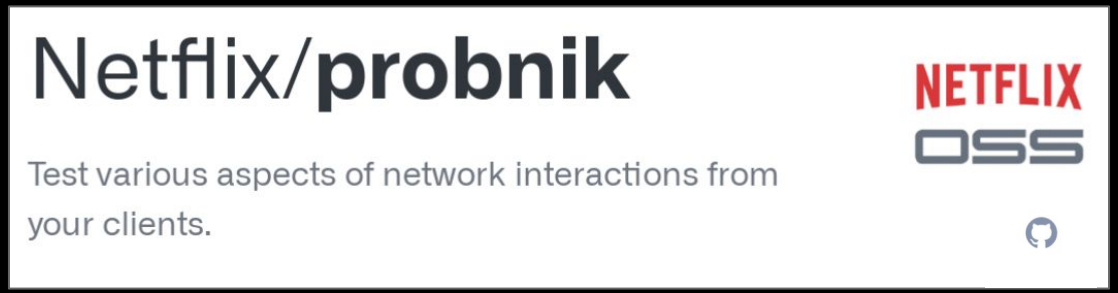
Fetch & Measure

× pulses

Step 3:
Upload Results



More Details about Probnik



Prototype

Verify performance expectations

Measure → Prototype

What to Use for Measurements?
Need a Quick Proof-of-Concept.

- **Proxy:** implementation and deployment on CDN
- **Steering:** how to choose the CDN server for a client?
- **Comparison:** Proxied requests vs AWS-direct

Client to Cloud Routing Options: Direct Link

Cloud Steering:

- 3 AWS regions
- geoDNS for server selection

Client to Cloud Routing Options: CDN Proxy

CDN Proxy Steering:

- TCP Anycast
- Single IP for all CDN locations

Compare Performance for Netflix Users

Which path is faster?

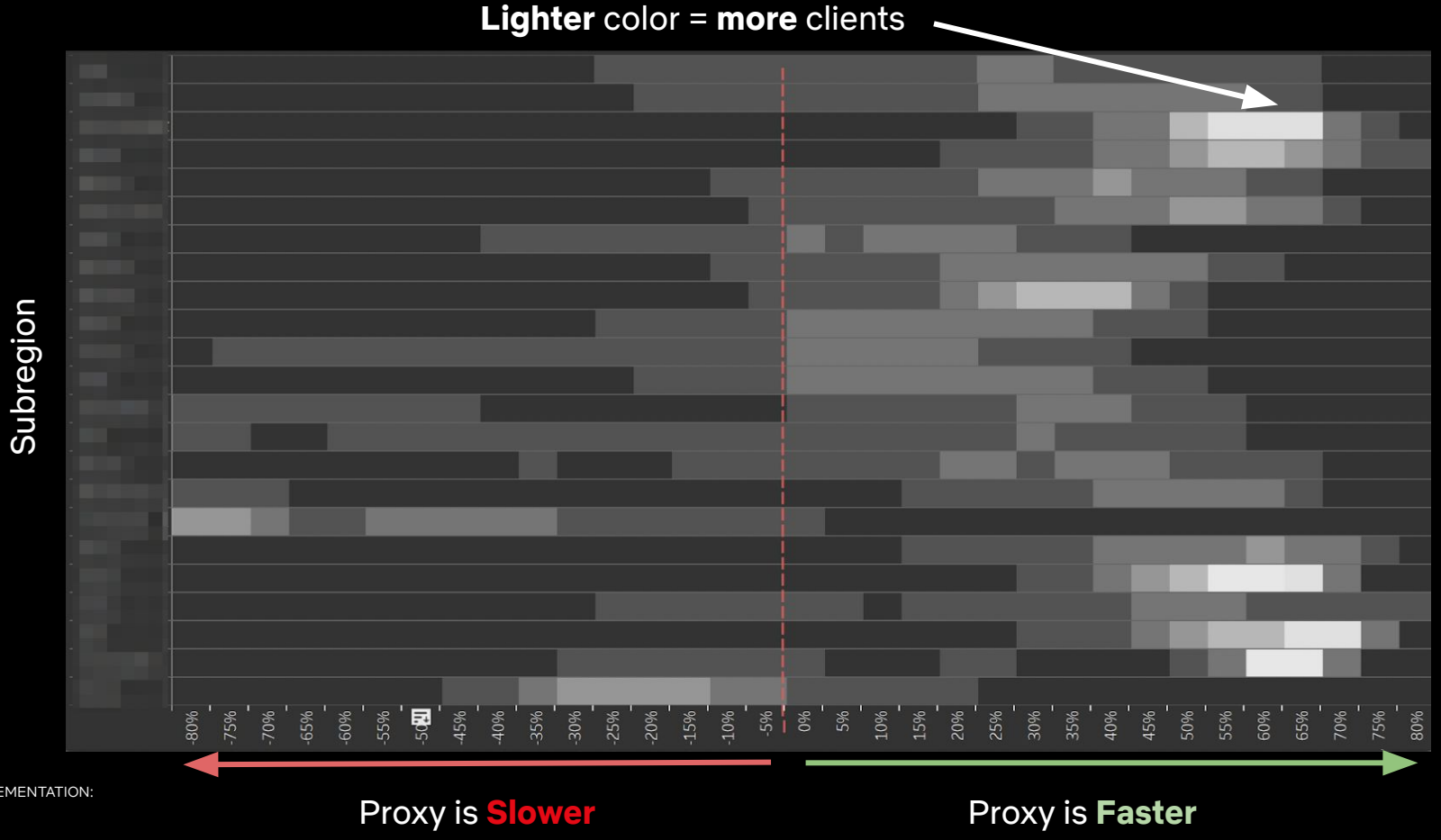
Using the Probing System to Compare


```
type: HTTP GET
name: steering test
targets:
  cdn-proxy.test.me/probe
  cloud.test.me/probe
```

→

```
cdn-proxy:    time / OK / FAIL
cloud:        time / OK / FAIL
```

Results: no Clear Winner



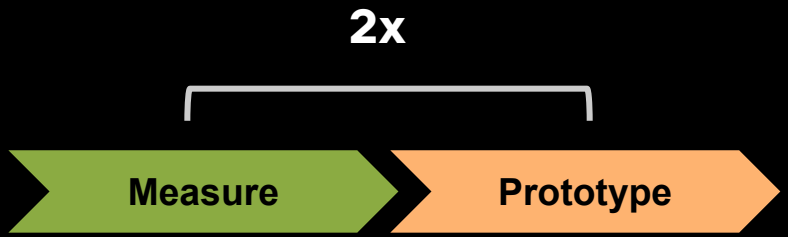

How much was that prototype?

Recap

- Accurate estimate of performance
- 100% CDN Edge proxying doesn’t work
- **No production impact**

Prototype

Verify your performance expectations

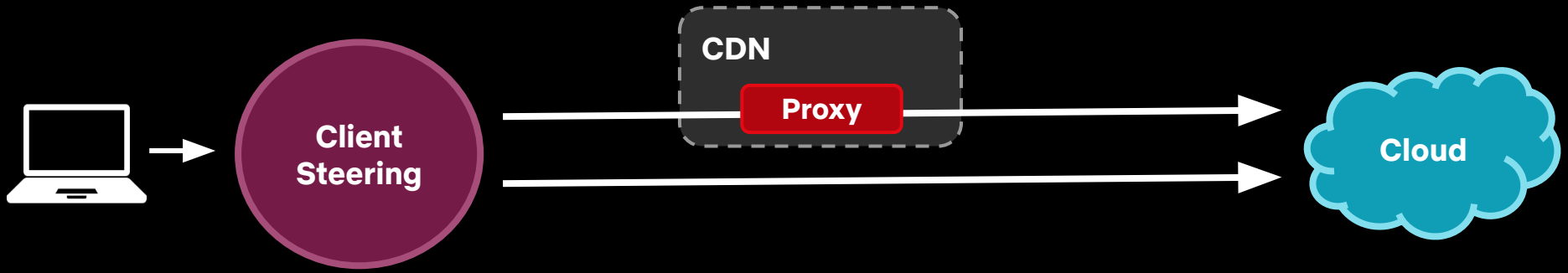


No Single Winner, Routing Based on Past Performance

Goal: find the fastest possible network path, amongst available options:

- Cloud-direct
- CDN-proxy

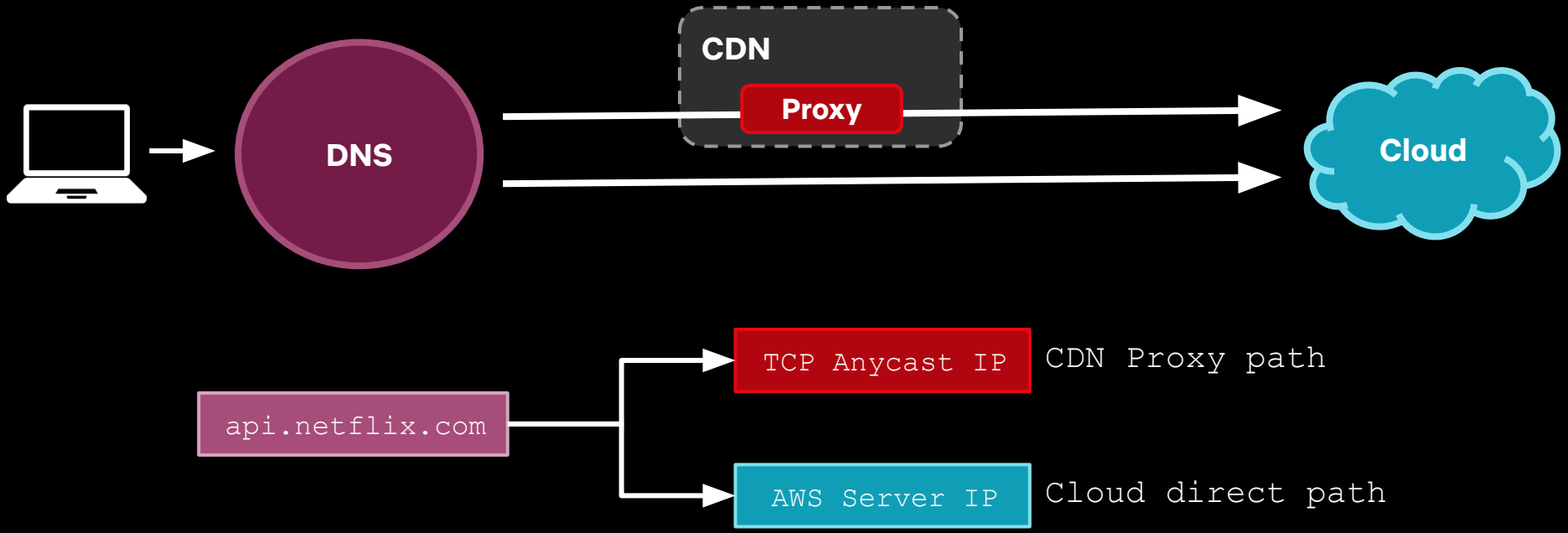
Intelligent Client Steering



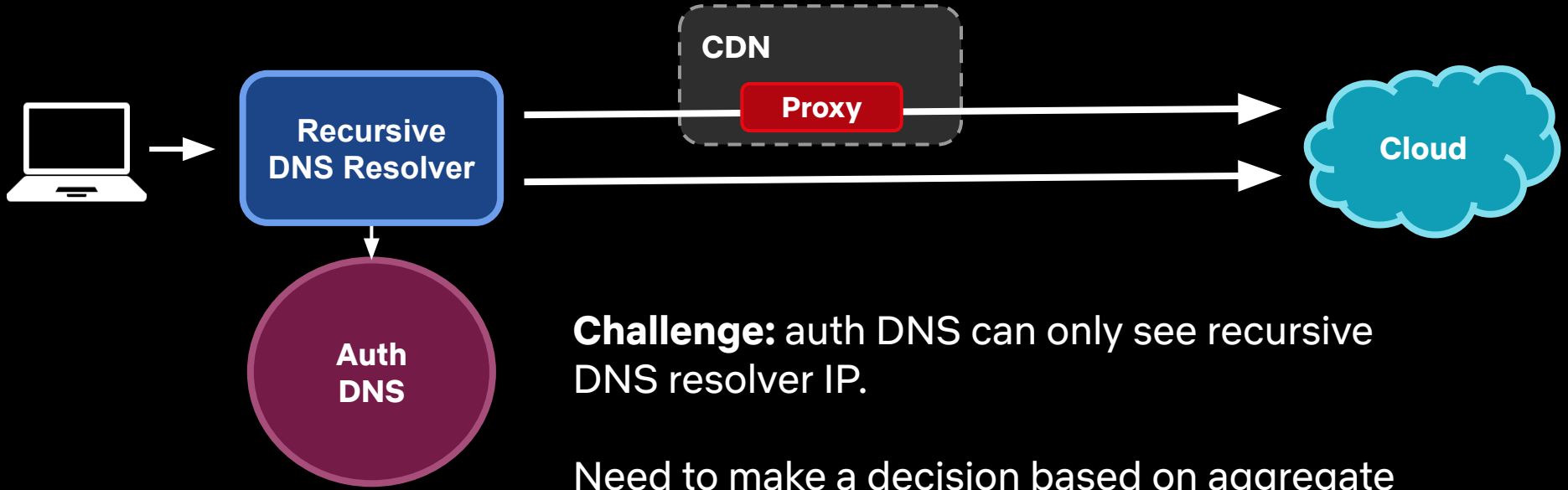
Goal: choose the fastest path

- Cloud-direct vs CDN-Proxy
- **No additional API calls**
- Easy client integration

Using DNS: Return IP of a Server on the Fastest Path



DNS Resolver Based Steering



Challenge: auth DNS can only see recursive DNS resolver IP.

Need to make a decision based on aggregate client performance.

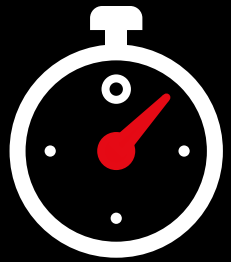
Step 1: Measure Network Performance for Each Client and Each Path


```
type: HTTP GET
name: steering test
targets:
  cloud.test.me/probe
  cdn-proxy.test.me/probe
```

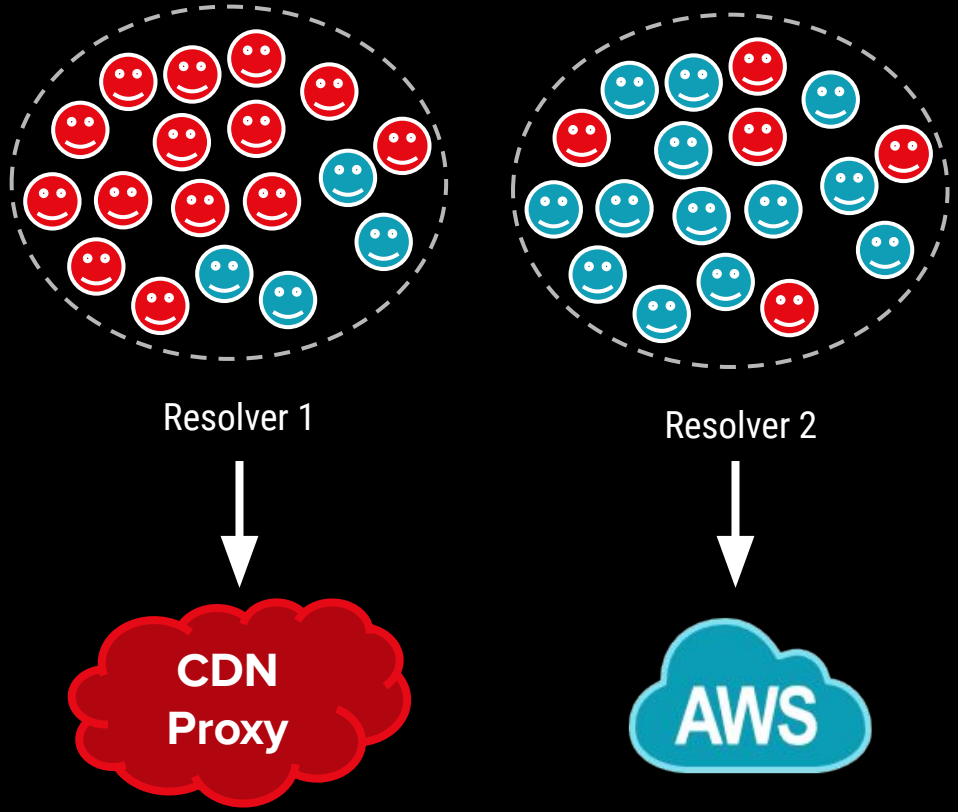
→

```
cloud:        time / OK / FAIL
cdn-proxy:    time / OK / FAIL
```

Step 2: Aggregate Device Measurements by DNS Resolver



Probe



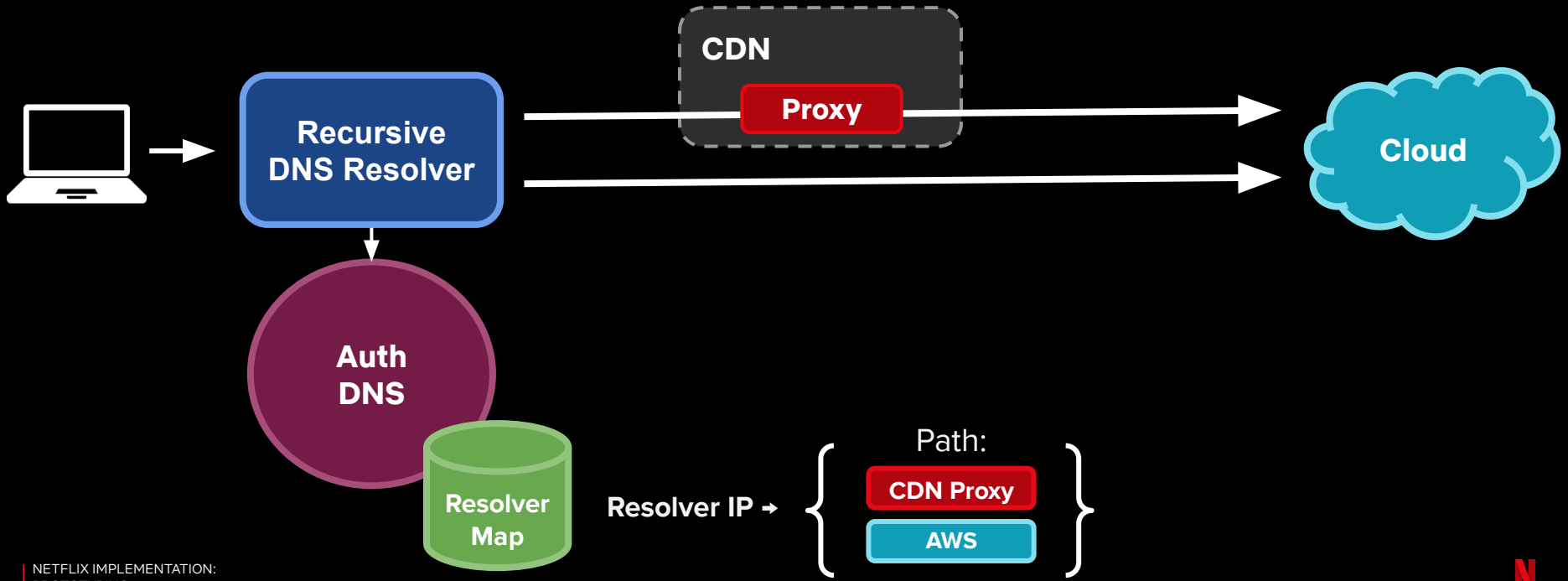
GROUP BY resolver:

resolver 1: CDN Proxy

resolver 2: AWS

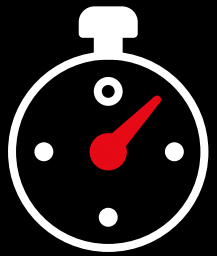
...

Step 3: Load the Resolver Map to Auth DNS and Steer Based on Resolver IP



Automatic Traffic Steering

Probes



Measure different client routing options

Data Pipeline

Choose the fastest path

Client Steering

Use for production steering policy

Use Probes to Measure Effectiveness


```
type: HTTP GET
name: steering test
targets:
  cloud.test.me/probe
  dns-steering.test.me/probe
```

→

```
cloud:          time / OK / FAIL
dns-steering:   time / OK / FAIL
```

Results: Equal or Better Performance

Recap

- Validated that the DNS-based solution works
- Tuned the resolver aggregation models
- Tested full integration of end-to-end solution
- **No production impact**

Productize

From prototype to production

Measure → Prototype → Productize

Changing Critical Path at Netflix Scale

HTTP APIs

CDN

200+ Million Users

1K+ Different Devices

10K+ CDN Servers

1K+ Locations

1M+ Requests per Second

100+ Microservices

100+ Deployments per Day

Faster Than Light

SPACE FORCE

Sergey Fedorov

Antoine Girbal

Olivier Poitrey

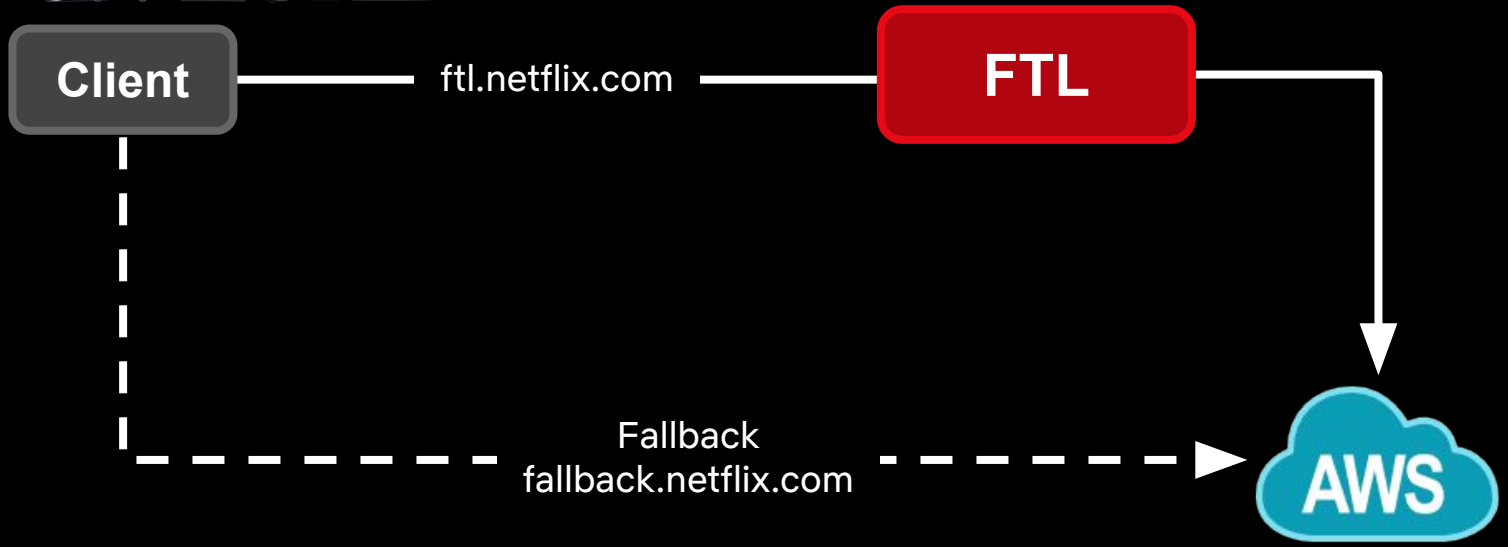
Changing Critical Path with Minimal Production Impact

Minimize the scope of failure

Embrace failure as part of your design

Graceful degradation

Embrace Failure as Part of Your Architecture



Deployment Principles

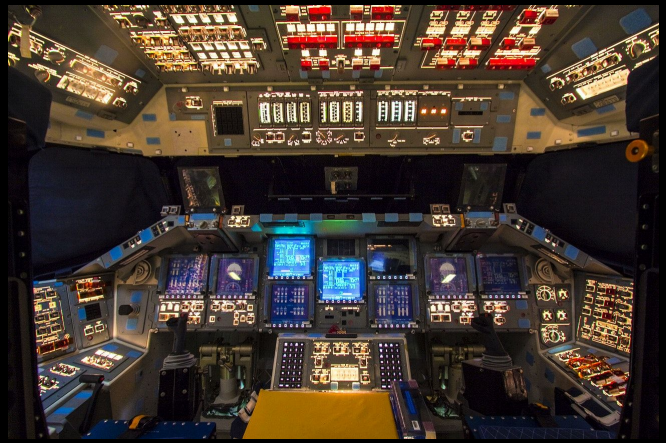
Small features.
Frequent deployments.
Clean metrics

Workflow:

1. Probe test
2. AB tests or Canaries
3. Progressive rollout
4. Done

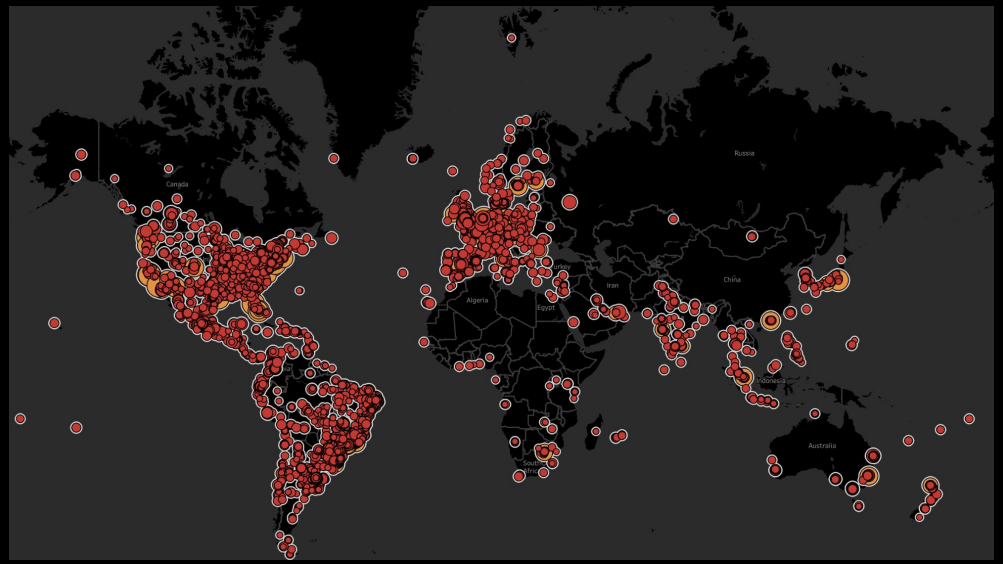
Operate

Minimize operational overhead



Measure → Prototype → Productize → Operate

At Netflix Scale Manual is not an Option



200+ Million Users

1K+ Different Devices

10K+ CDN Servers

1K+ Locations

1M+ Requests per Second

100+ Microservices

100+ Deployments per Day

Automating Response to Network Failures


```
type: HTTP GET
name: reachability test
targets:
  cloud.test.me/probe
  cdn-proxy.test.me/probe
```

→

```
cloud:       time / OK / FAIL
cdn-proxy:   time / OK / FAIL
```

Automating Response to Network Failures


```
cloud:          FAIL
cdn-proxy:      OK
```

What's broken?

- ISP's connection to AWS

Can we fix it?

- YES - route traffic via the CDN proxy

Automating Response to Network Failures


```
cloud:          OK
cdn-proxy:      FAIL
```

What's broken?

- Backbone link to AWS

Can we fix it?

- YES - Move traffic to cloud-direct path

Automating Response to Network Failures


```
cloud:          FAIL
cdn-proxy:      FAIL
```

What's broken?

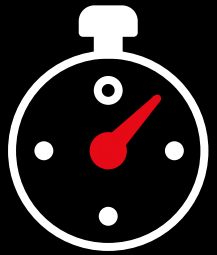
- ISP outage or client last mile

Can we fix it?

- NO (we don't have a routable path)

Automatic Traffic Steering

Probes



Measure different client routing options

Data Pipeline

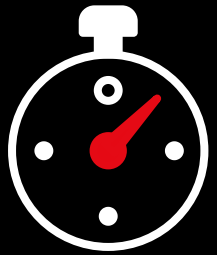
Choose the fastest path

Client Steering

Use for production steering policy

Recap: Device-Cloud API Acceleration at Netflix

Probes



100K

measurements per minute

Data Pipeline

200K

routes

Client Steering

1M+

production requests per second

Recap: Device-Cloud API Acceleration at Netflix

25%

median acceleration of requests on new connections

10%

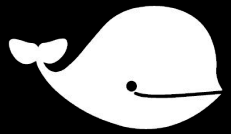
median acceleration of requests on existing connection

This is not what's available only to the unicorn companies

What can YOU Learn from it

LEARNINGS

Even if You don't Run Physical Infrastructure, there is Plenty to Choose from



BelugaCDN

STACKPATH

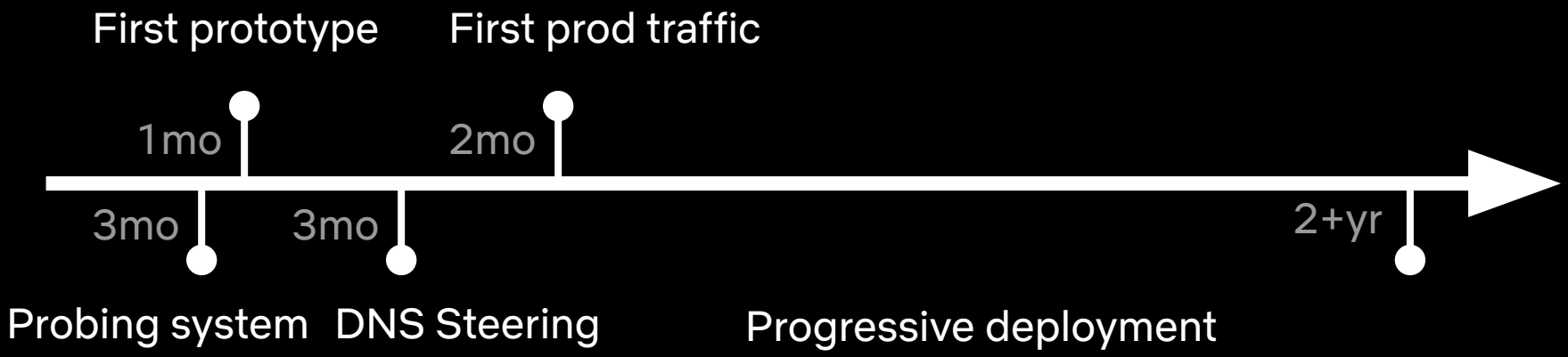
Question Your Intuition and Learn from Your Users

Request Routing is Only One of Many Optimizations You can Test

Our team of 3 ran and deployed dozens of experiments over 3 years:

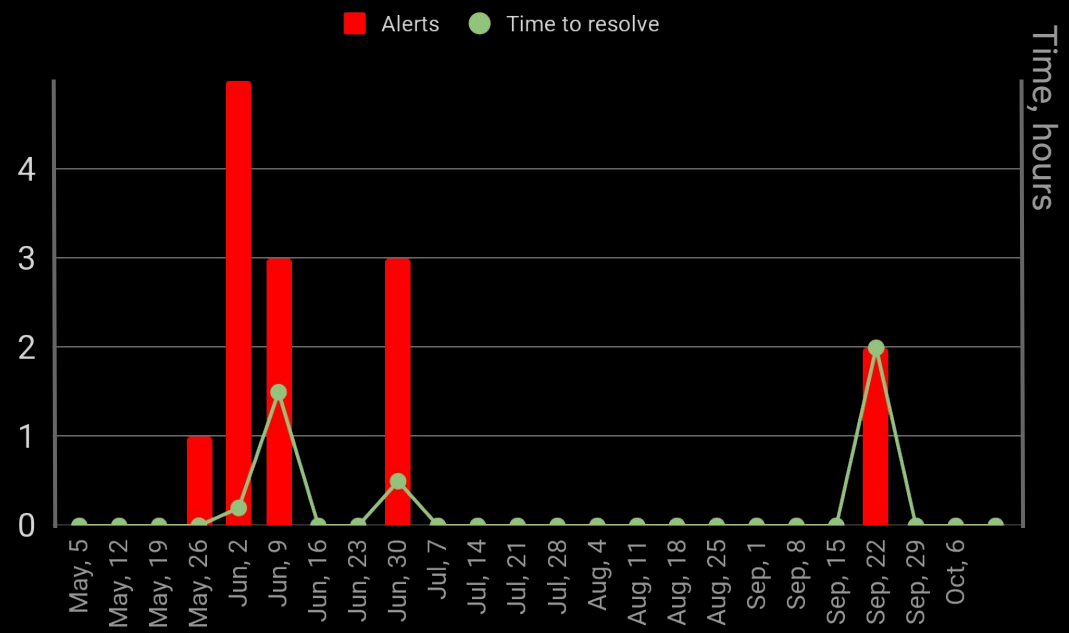
- Edge Termination + DNS Steering
- HTTP2
- TLS1.3
- TCP Fast Open
- Traffic Rebalancing
- DNS migration
- HPACK compression
- CDN chaining
- IPv6 migration
- ...

Reduce Measure-Prototype Flow, nly Productize what Works



Embrace Failure

Fallback and failure avoidance lead to much better operational experience



Our team gets **less than 1 critical alert per week**, on average

Summary

Happy Networking!

linkedin.com/in/sfedov/

@sfedov

THANK YOU