Mux

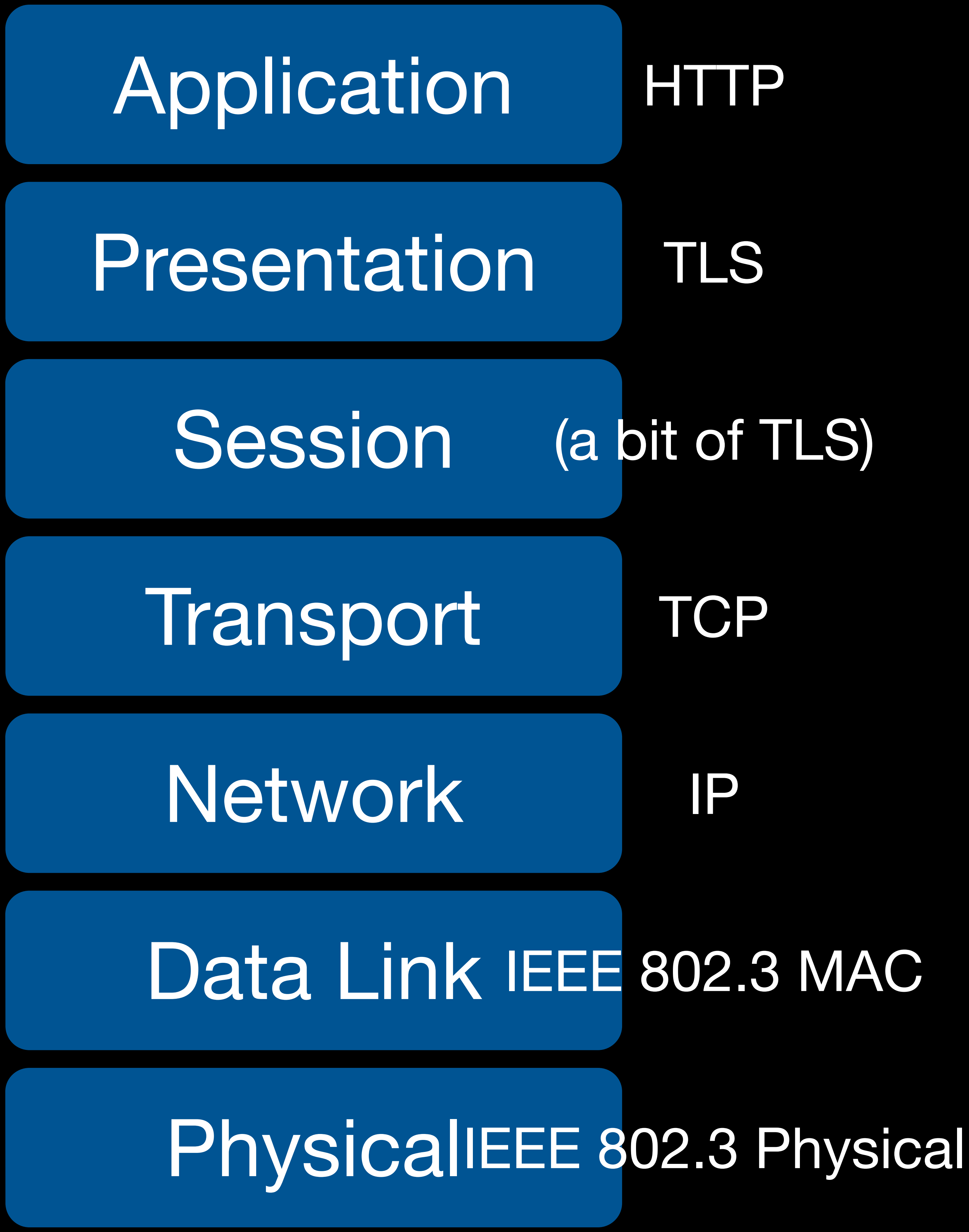
How I stopped worrying
and learned to love the multiplexing.

Berk D. Demir

@bd

Today, we’ll talk about **RPC Multiplexing**.

OSI 7 Layer Model



Application	HTTP
Presentation	TLS
Session	(a bit of TLS)
Transport	TCP
Network	IP
Data Link	IEEE 802.3 MAC
Physical	IEEE 802.3 Physical

What’s the problem?

Head of Line Blocking

Not every interaction is a single RPC.

Calls for discrete resources should not block each other

What’s the big deal?

Just open more TCP connections!

Free as in beer

New socket connections are not free as in resources and latency.

(Remember TCP 3-way handshake)

… also TCP Slow Start

In the data center
Across data centers
From the POPs

For every TCP connection

You have a separate network queue and the half of separate liveness detection logic.

TCP Keepalives.

…or a blast from October 1989

Not more frequent than one every 2 hours.

It’s the kernel, not the application.

What about?

Liveness Detection

Request Cancellation

(cost of tearing down a TCP connection is too damn high!)

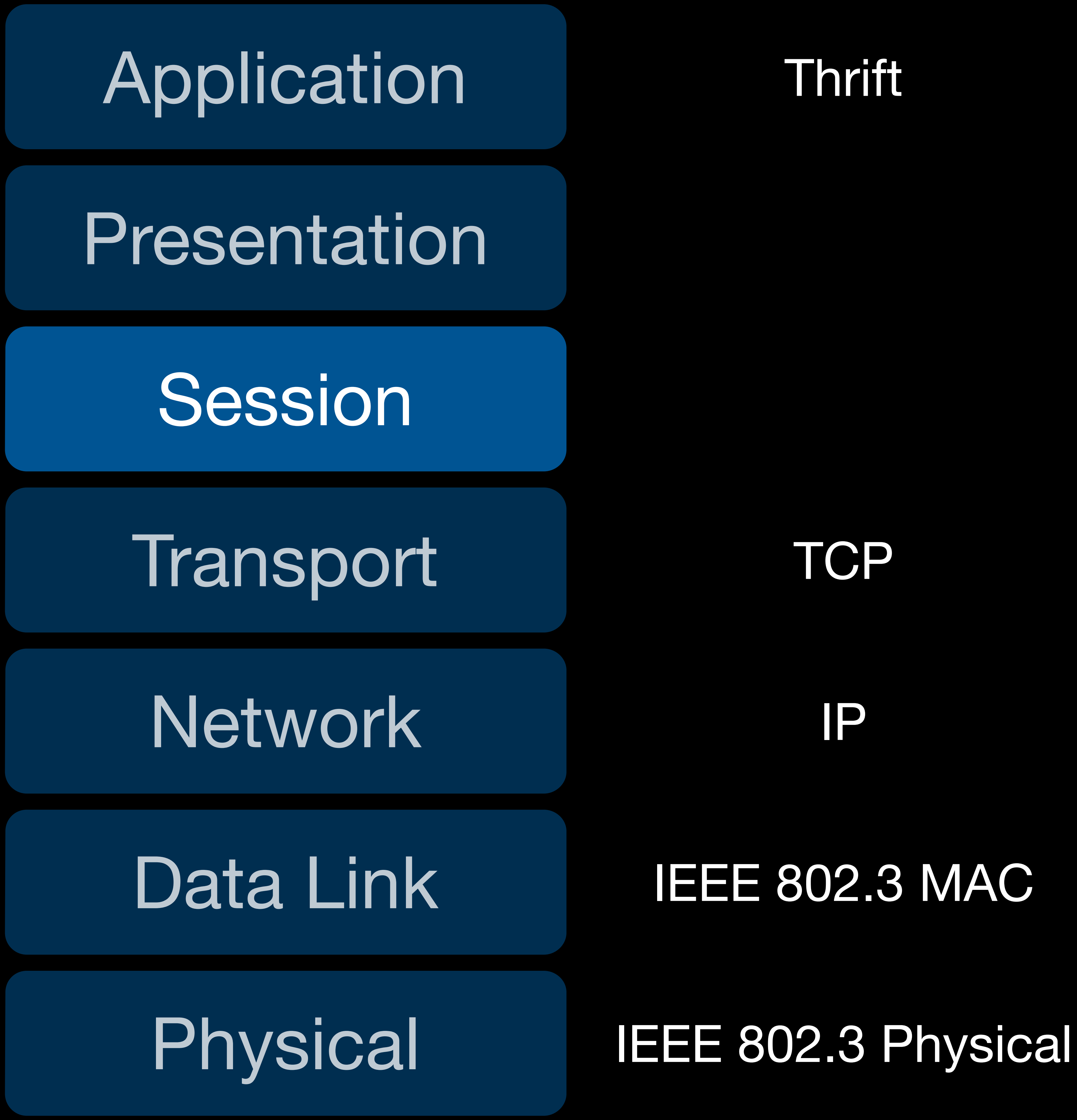
Availability Advertisement

...also

Control Plane vs Data Plane

Separate these so we can have out-of-band messaging*(node-to-node)* without affecting data plane.

Back to those 7 Layers



Application	Thrift
Presentation	
Session	
Transport	TCP
Network	IP
Data Link	IEEE 802.3 MAC
Physical	IEEE 802.3 Physical

These concerns could be addressed purely in Layer 5.

We needed a session protocol.

Explicit Queue Management

NACKs
Leases
Proper back pressure signaling

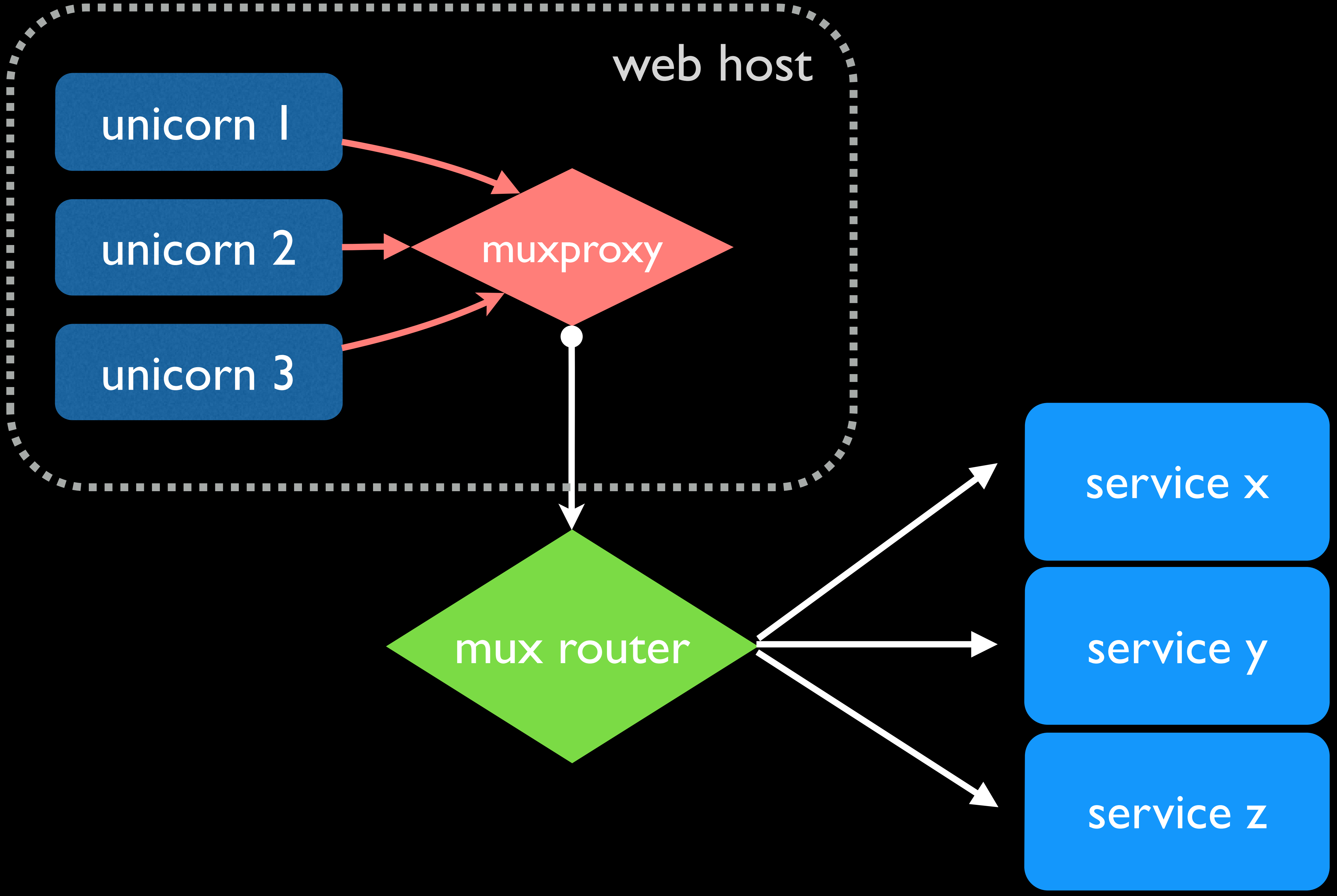
Interesting Use Cases

Destination Dispatch
GC Avoidance
Service-to-service Authentication
awk for Distributed Systems

Destination Dispatch

Intelligent routing and load balancing for less intelligent clients.

Destination Dispatch



Destination Dispatch

Mux routers at the data center edge for load, global incident status and preference aware RPC routing.

GC Avoidance

We can easily predict a young generation collection.

If we can gracefully drain all our clients via leases, why worry about GC pauses?

RPC Authentication

(Lessons from HTTP)

Authenticating *every single* RPC is expensive. Implementing AAA in application or network layer is disruptive. Let’s address the concern in the session layer.

RPC Authentication

(Lessons from HTTP)

expensive: HTTP Basic/Digest
disruptive: Modify L7, IPsec
session layer: Implement your own with X.509, Kerberos, etc.

Debugging Distributed Systems is HARD.

Production Readiness Reviews
are tough.
Production is real life and it is wild.

RPC TAP

(awk for Distributed Systems)

inject failure

simulate latency, backpressure

rewrite destinations

More

Mux is part of Finagle and it’s open source.

Related

HTTP/2 & QUIC

There must be questions!

@bd

Jeff Dean Numbers

L1 cache ref: 1ns
Branch mispredict: 3ns
L2 cache reference: 4ns
Mutex lock/unlock: 17ns
Main memory reference: 100ns
Send 2000 bytes over the network: 400ns
Compress 1K with Zippy: 2,000ns
Read 1MB from memory (seq): 12,000ns
SSD random read: 16,000ns
Read 1M from SSD: 200,000ns
RTT in the same DC: 400,000ns
RTT from SMF1-to-ATLA: 80,000,000ns
RTT from Sacramento-to-Amsterdam: 150,000,000ns

http://www.eecs.berkeley.edu/~rcs/research/