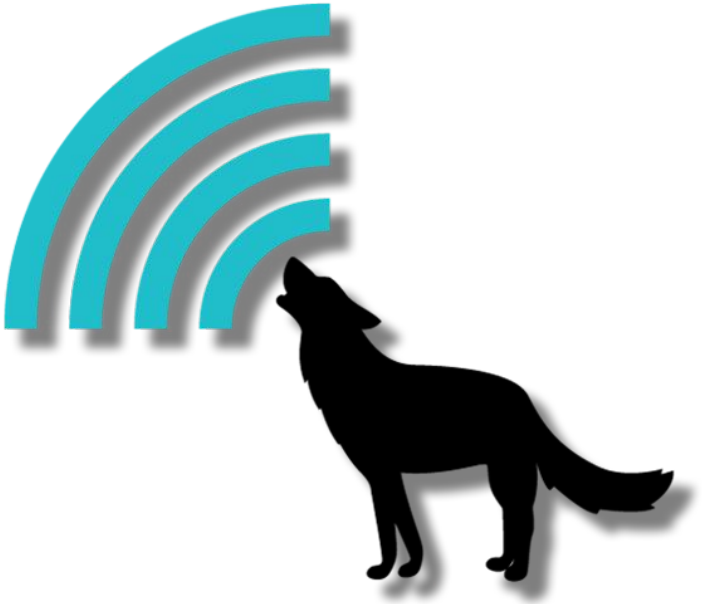

HTTP/3
For everyone!

Daniel Stenberg

https://daniel.haxx.se @bagder

Daniel Stenberg

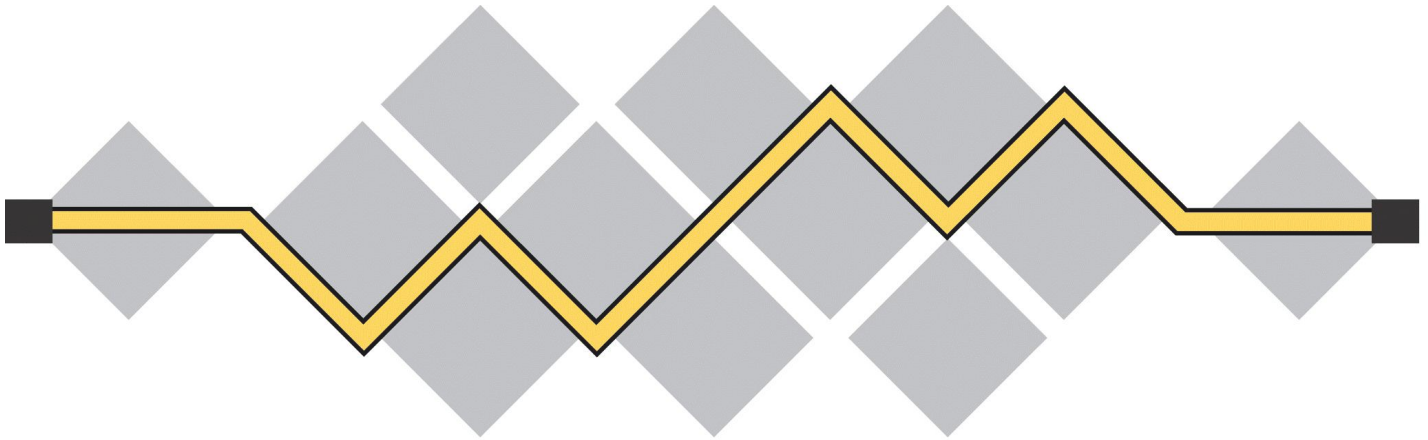
@bagder



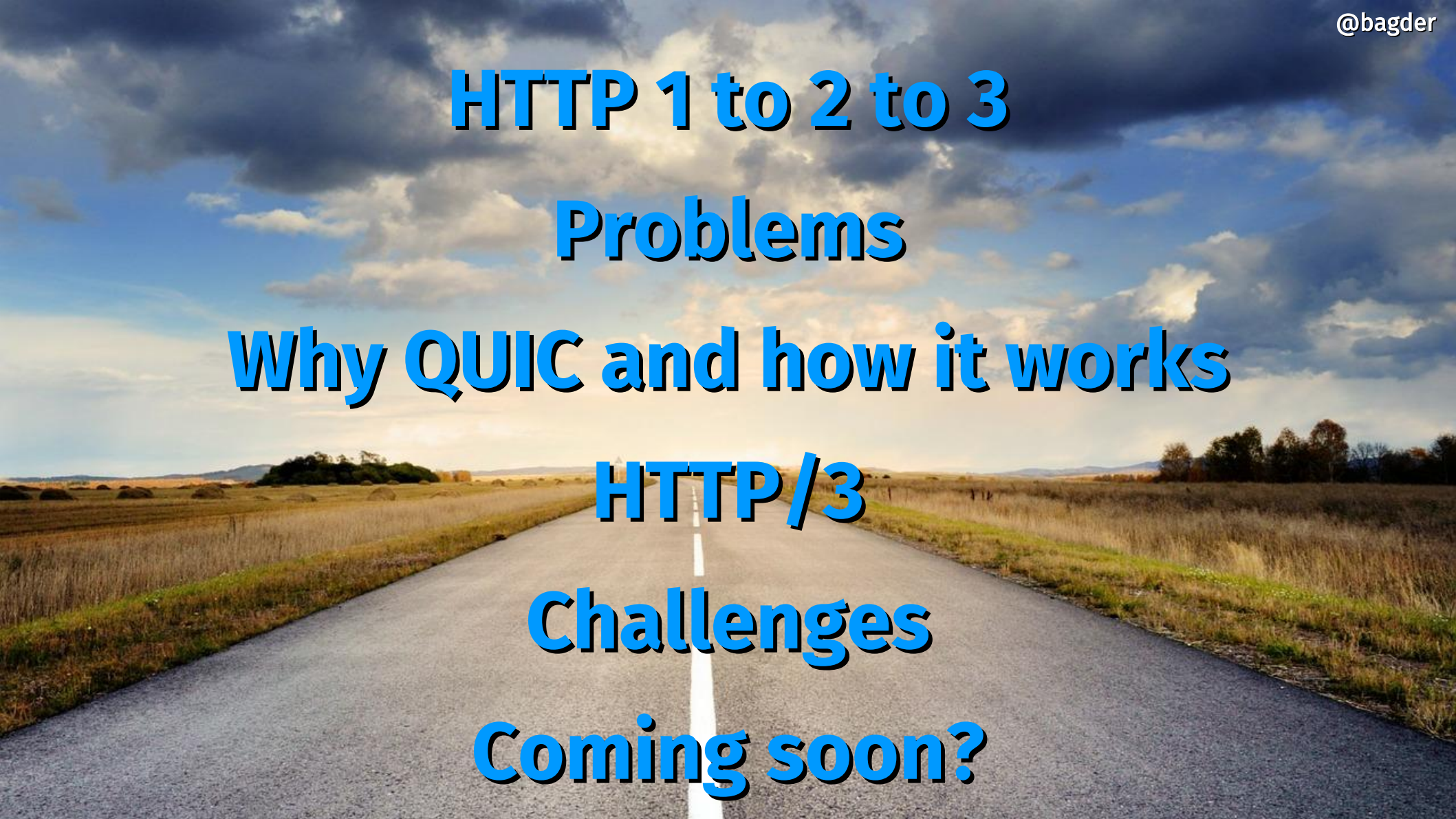
wolfSSL

Daniel Stenberg

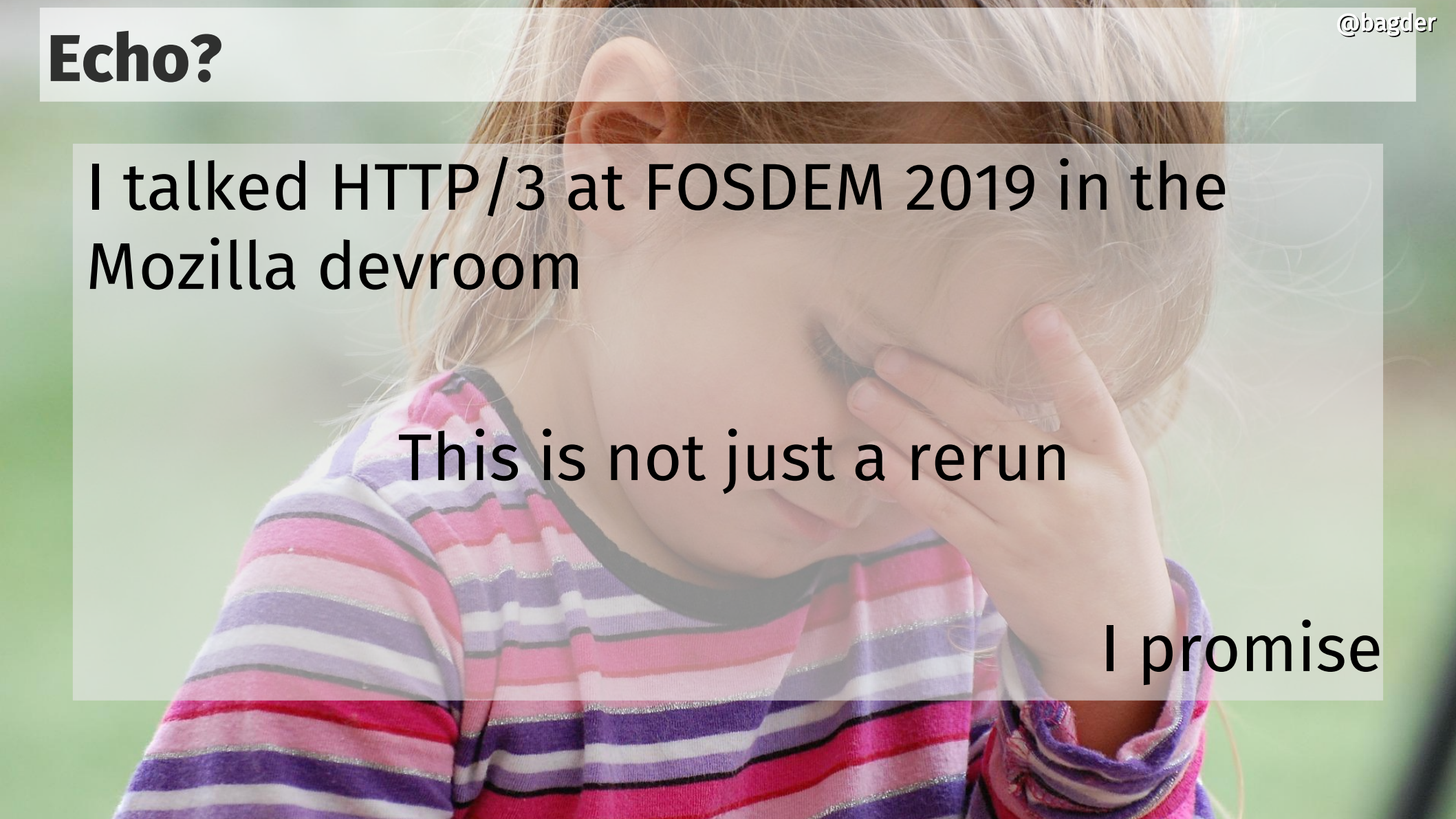
@bagder



I E T F®


@bagder
HTTP 1 to 2 to 3
Problems
Why QUIC and how it works
HTTP/3
Challenges
Coming soon?

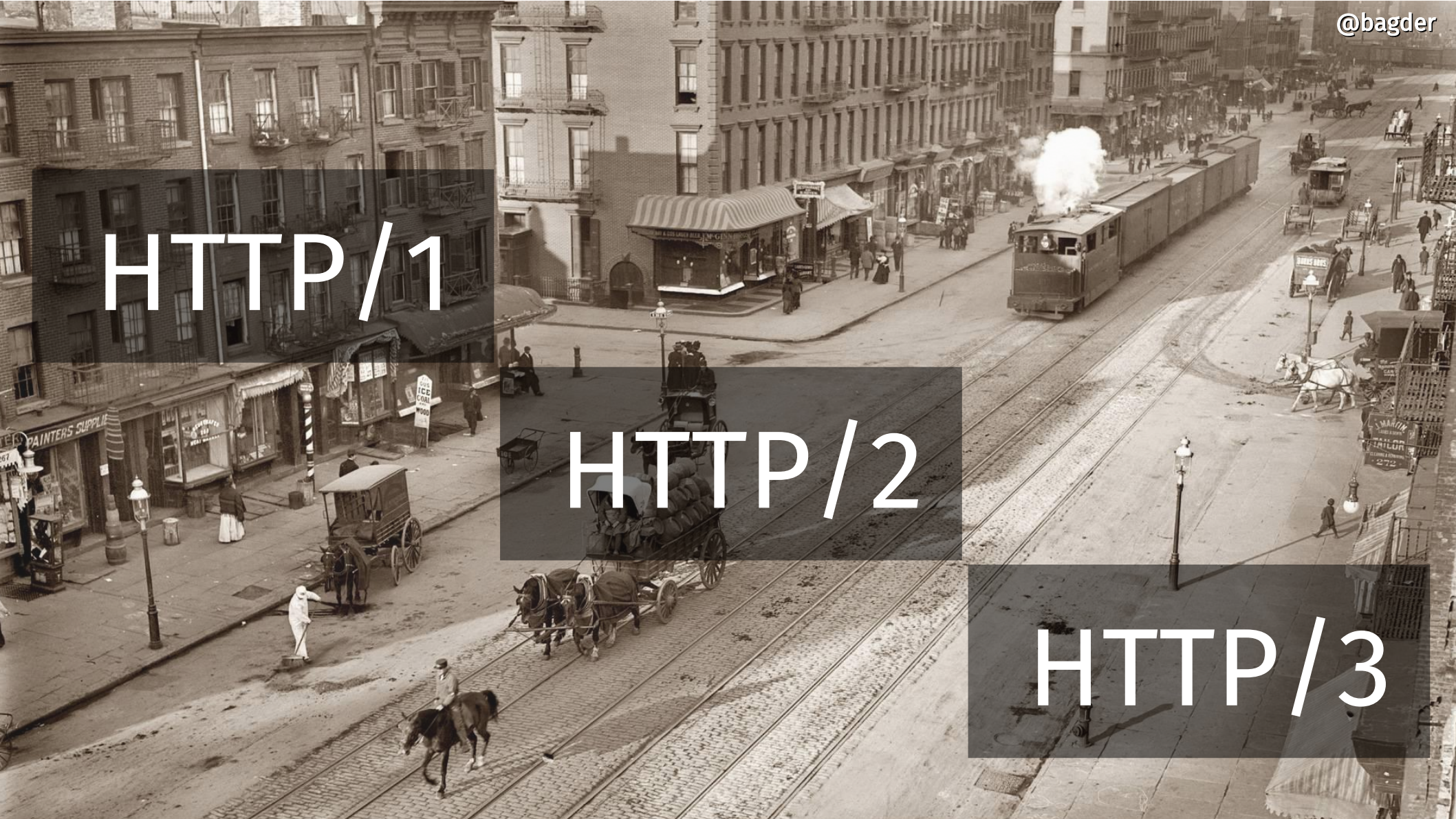
Echo?



I talked HTTP/3 at FOSDEM 2019 in the Mozilla devroom

This is not just a rerun

I promise


@bagder
HTTP/1
HTTP/2
HTTP/3

Under the hood


```
GET / HTTP/1.1
Host: www.example.com
Accept: */*
User-Agent: HTTP-eats-the-world/2020
```

```
HTTP/1.1 200 OK
Date: Thu, 09 Nov 2018 14:49:00 GMT
Server: my-favorite v3
Last-Modified: Tue, 13 Jun 2000 12:10:00 GMT
Content-Length: 12345
Set-Cookie: this-is-simple=yeah-really;
Content-Type: text/html

[content]
```

HTTP began over TCP

TCP

TCP is transport over IP

Establishes a “connection”

3-way handshake

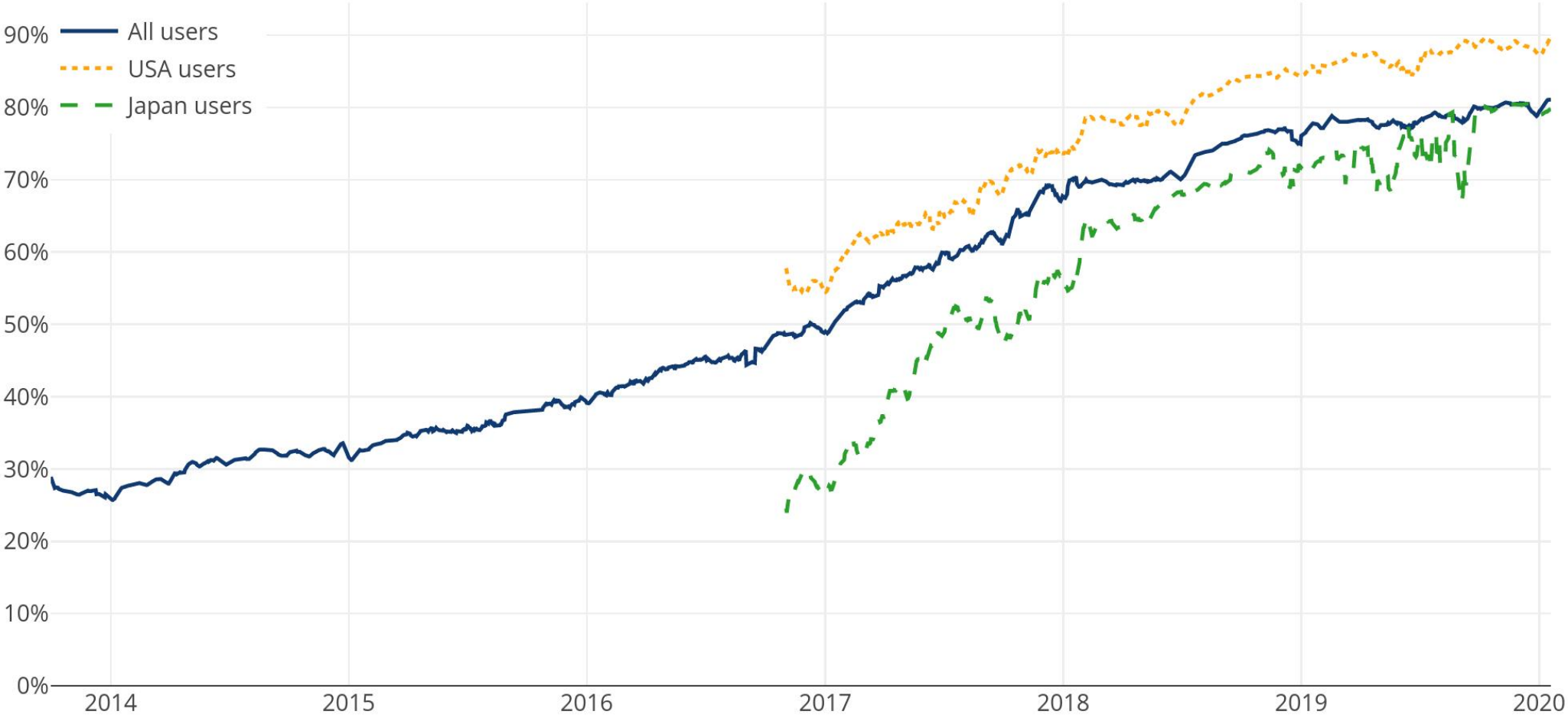
Resends lost packages

A reliable byte stream

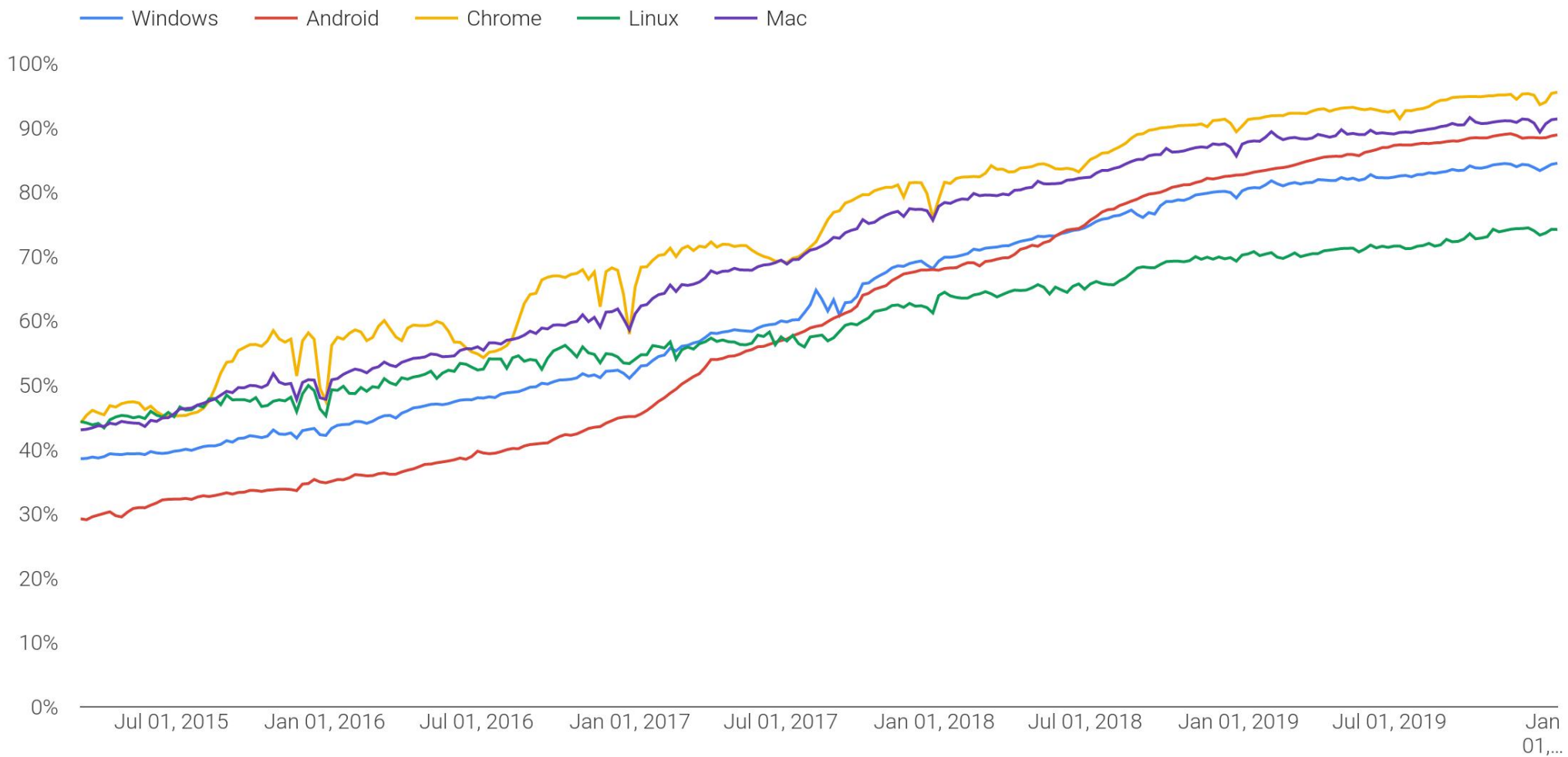
Clear text

HTTPS means TCP + TLS + HTTP

Web pages over HTTPS in Firefox



Web pages over HTTPS in Chrome



TLS

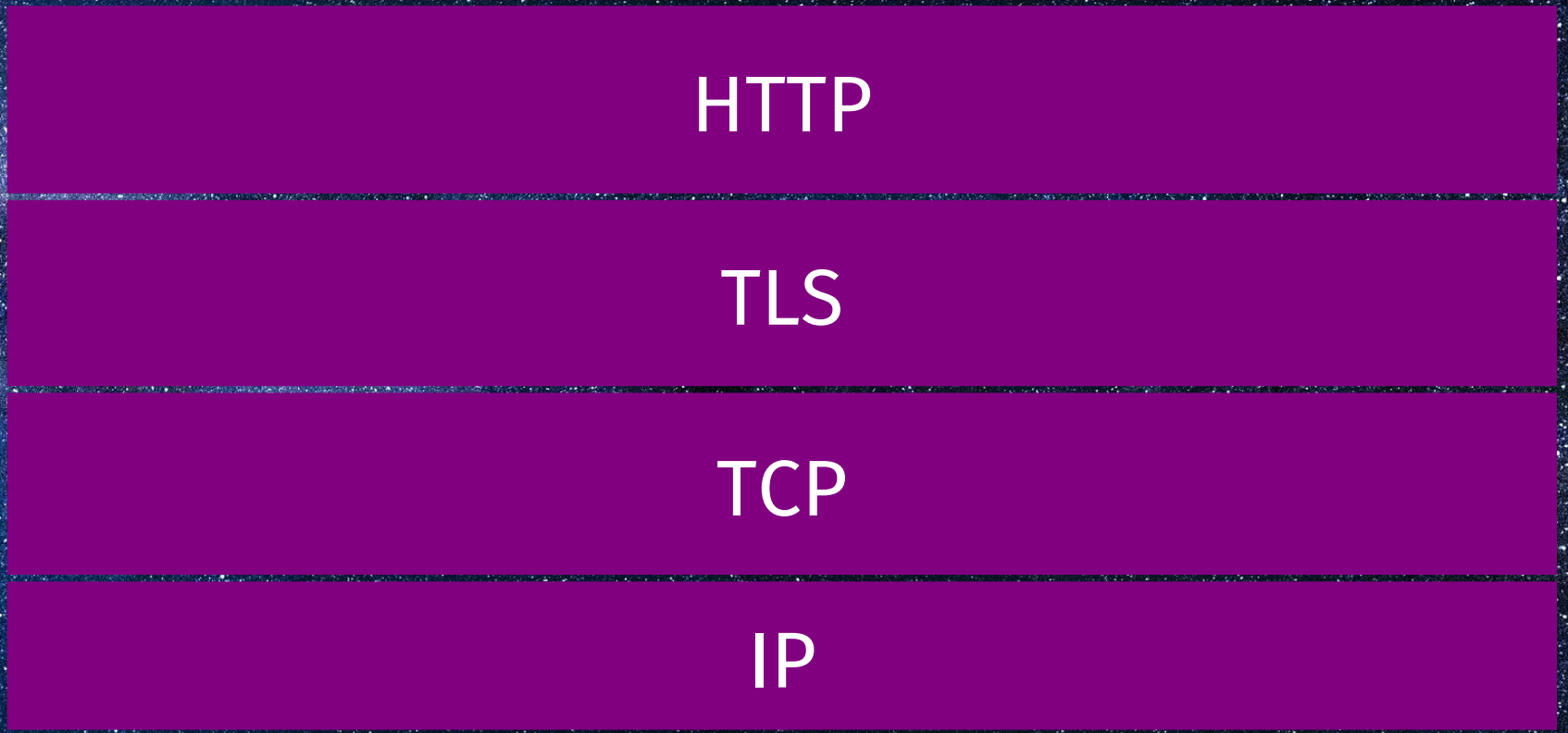
TLS is done over TCP for HTTP/1 or 2

Transport Layer Security

Additional handshake

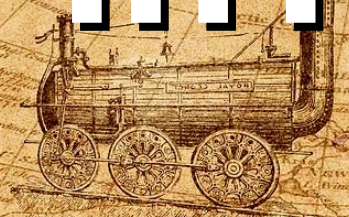
Privacy and security

Classic HTTPS stack

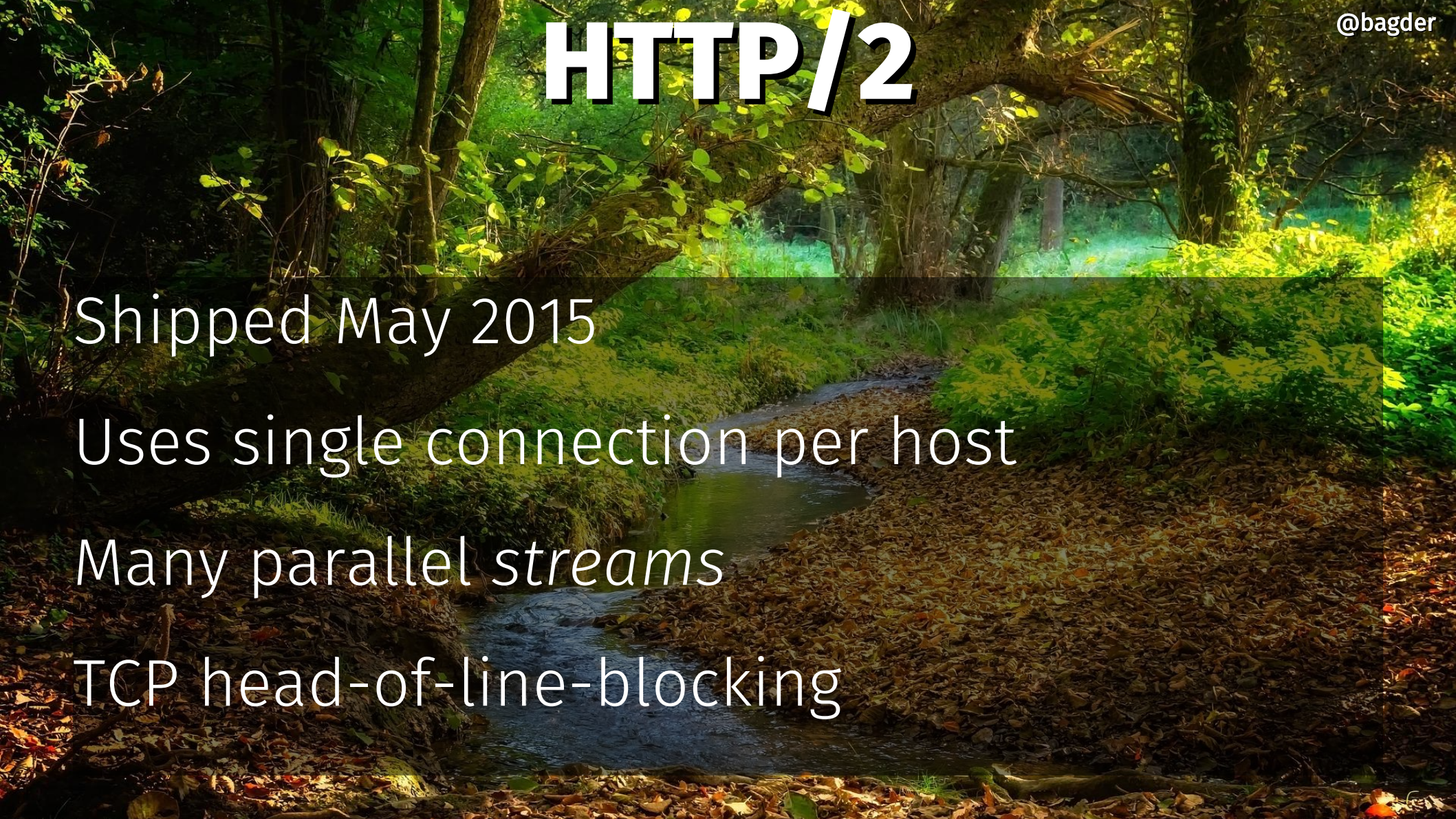


HTTP over TCP

HTTP/1.1



- Shipped January 1997
- Many parallel TCP connections
- Better but ineffective TCP use
- HTTP head-of-line-blocking
- Numerous work-arounds


HTTP/2
@bagder
Shipped May 2015
Uses single connection per host
Many parallel *streams*
TCP head-of-line-blocking

Ossification

Internet is full of boxes

Routers, gateways, firewalls, load balancers, NATs...

Boxes run software to handle network data

Middle-boxes work on existing protocols

Upgrade much slower than edges

Internet

@bagder

www

Ossification casualties

HTTP/2 in clear text

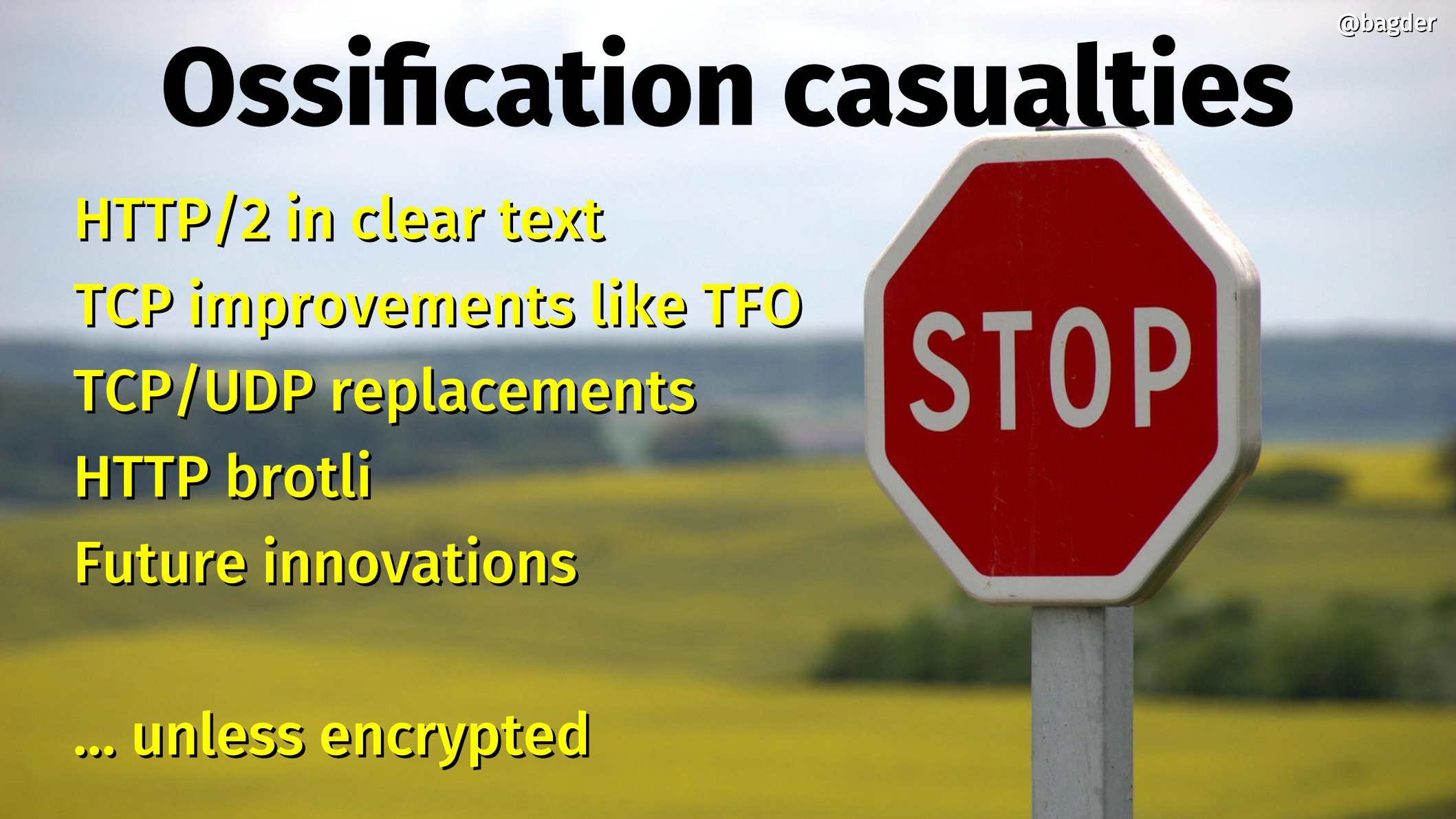
TCP improvements like TFO

TCP/UDP replacements

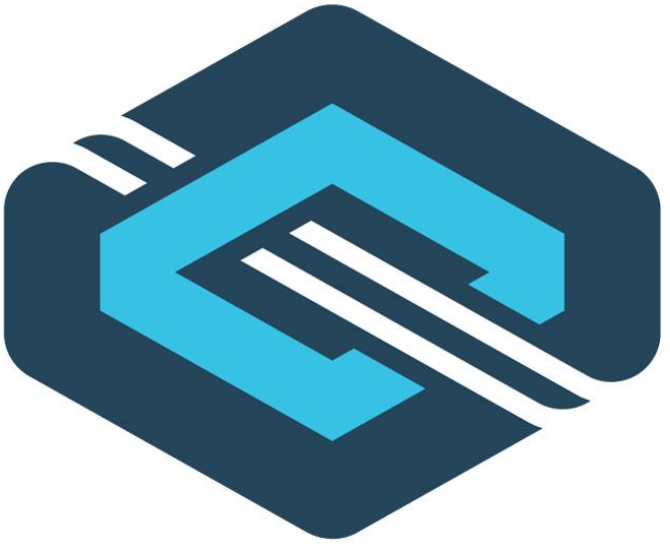
HTTP brotli

Future innovations

... unless encrypted



Improvement in spite of ossification



QUIC

QUIC is a name, not an acronym.


@bagder
f5
LITESPEED
fastly
Microsoft
traffic server
Akamai
moz://a
Google
CLOUDFLARE

A new transport protocol

Built on experiences by Google QUIC

Google deployed “http2 frames over UDP”-QUIC in 2013

Widely used client

Widely used web services

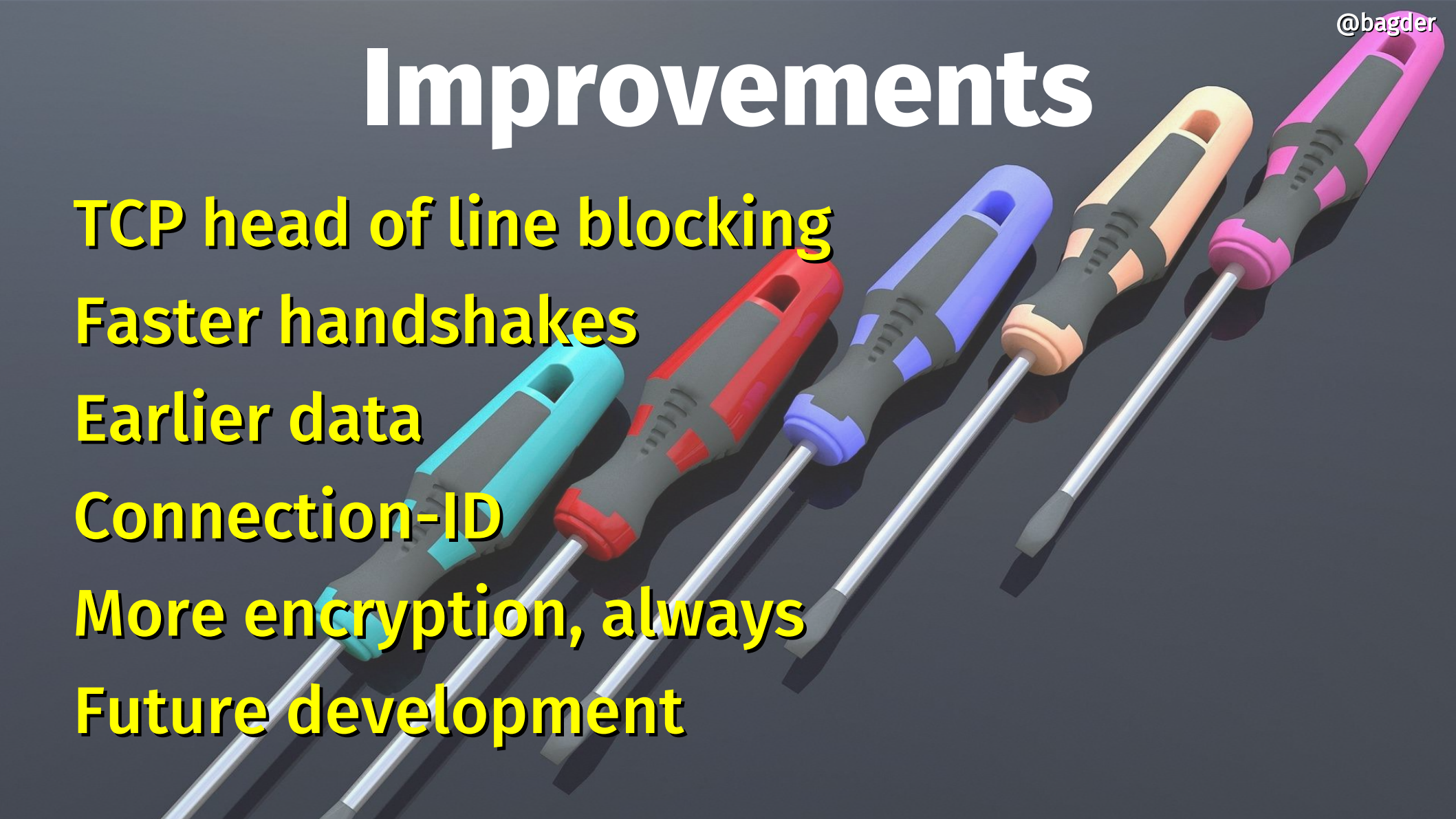
Proven to work at web scale

Taken to the IETF in 2015

QUIC working group started 2016

IETF QUIC is now very different than Google QUIC was

Improvements



TCP head of line blocking

Faster handshakes

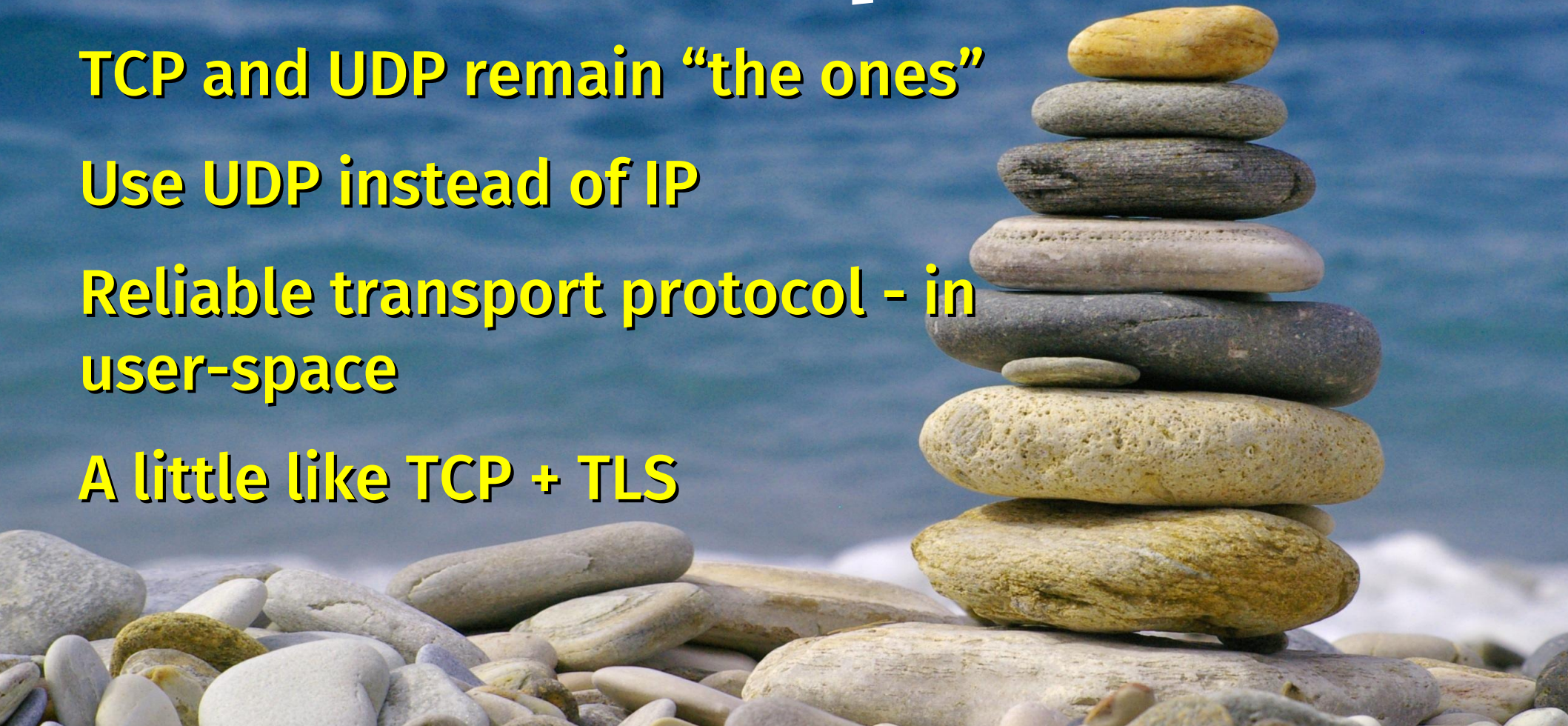
Earlier data

Connection-ID

More encryption, always

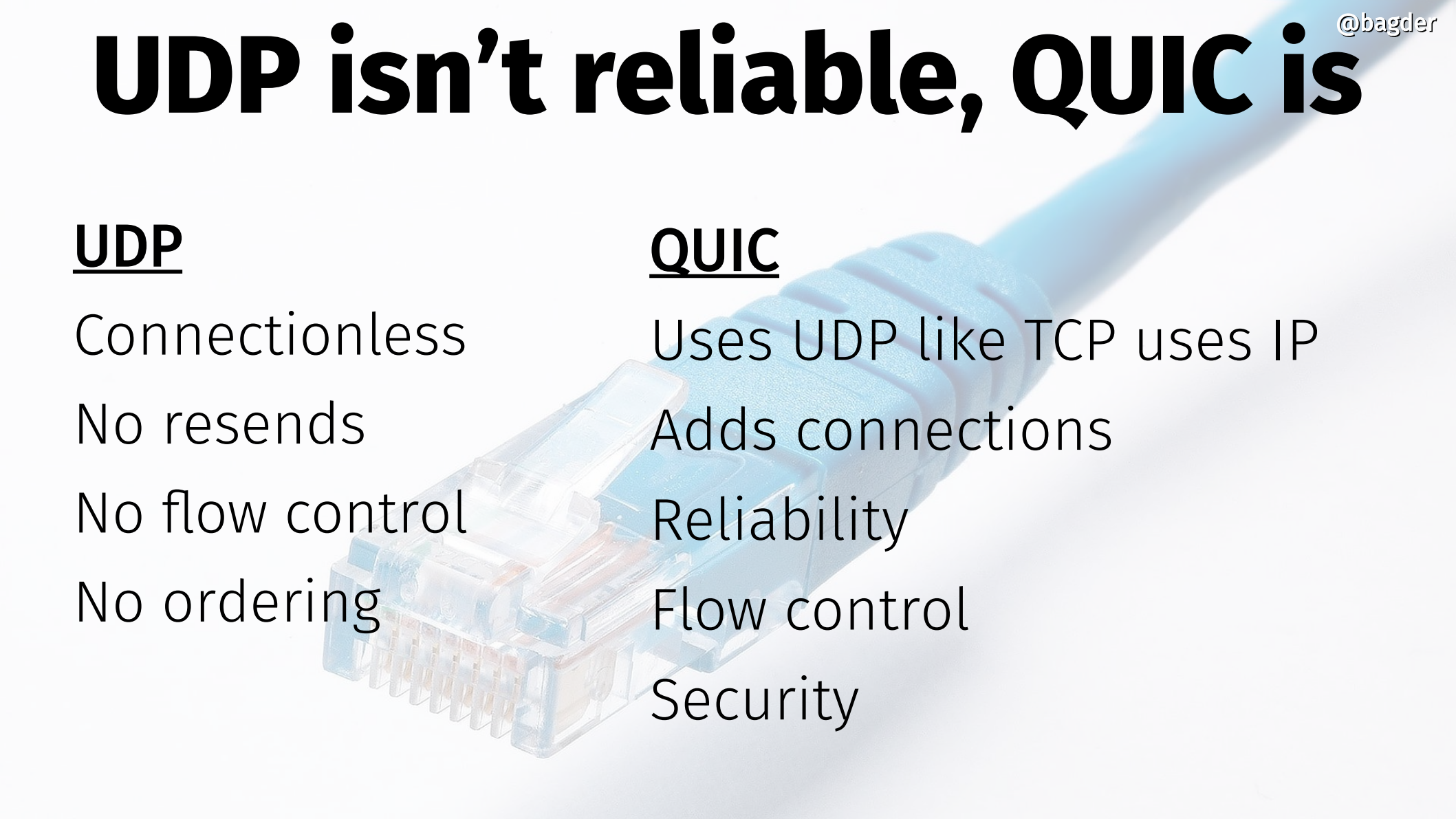
Future development

Build on top of UDP



- TCP and UDP remain “the ones”
- Use UDP instead of IP
- Reliable transport protocol - in user-space
- A little like TCP + TLS

UDP isn't reliable, QUIC is



UDP

Connectionless

No resends

No flow control

No ordering

QUIC

Uses UDP like TCP uses IP

Adds connections

Reliability

Flow control

Security

QUIC has streams

Many logical flows within a single connection

Similar to HTTP/2 but in the transport layer

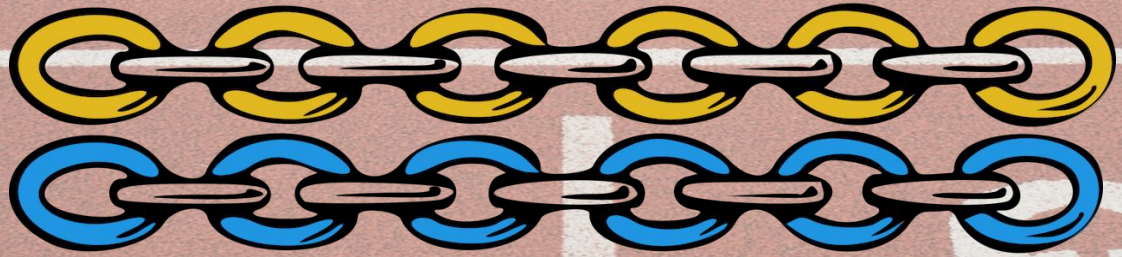
Client or server initiated

Bidirectional or unidirectional

Independent streams

Independent streams

TCP



QUIC

Application protocols over QUIC

Streams for free

Could be “any protocol”

HTTP worked on as the first

Others are planned to follow

HTTP/3 = HTTP over QUIC

HTTP – same but different

Request

- method + path
- headers
- body

Response

- response code
- headers
- body

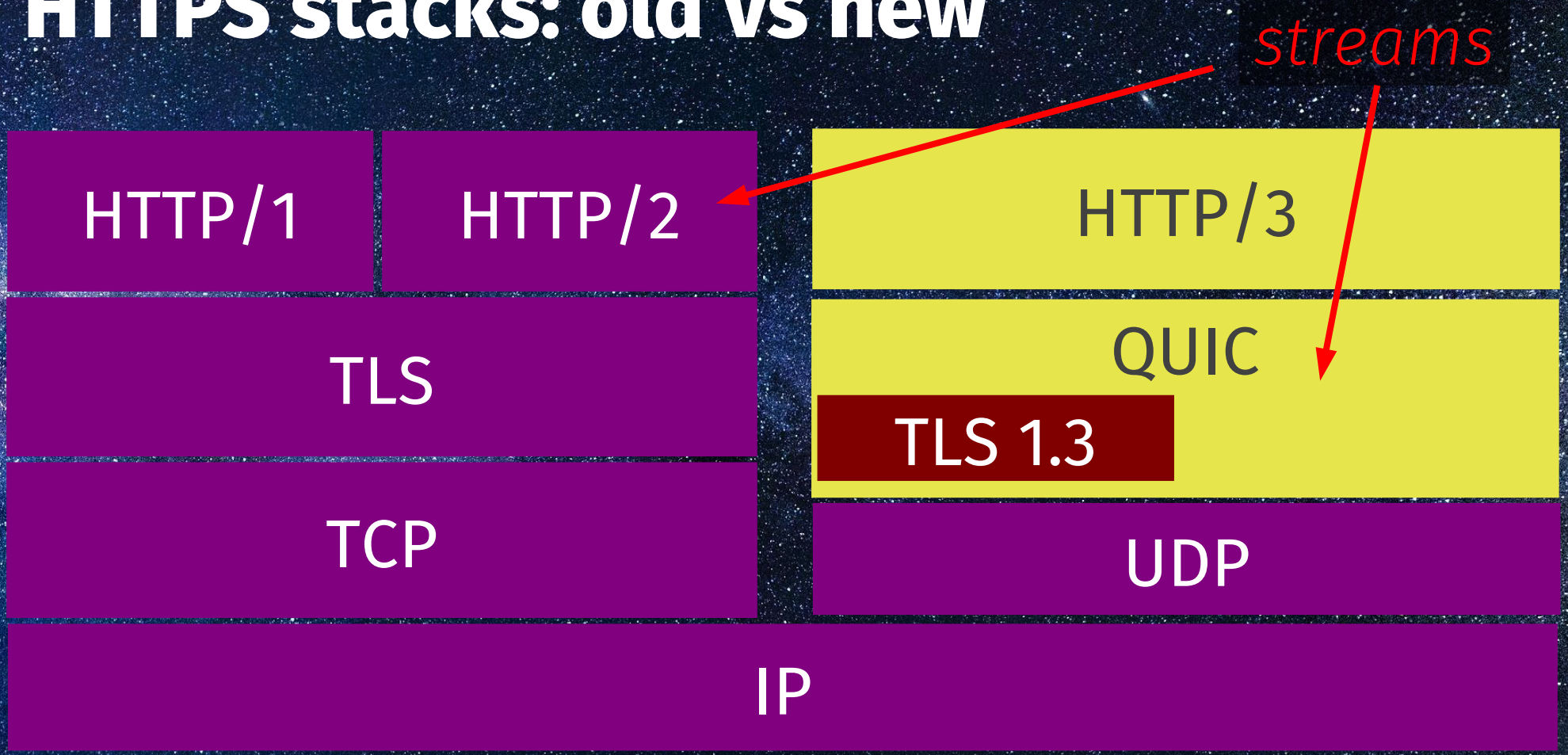
HTTP – same but different

HTTP/1 – in ASCII over TCP

HTTP/2 – binary multiplexed over TCP

HTTP/3 – binary over multiplexed QUIC

HTTPS stacks: old vs new



HTTP feature comparison

	HTTP/2	HTTP/3
Transport	TCP	QUIC
Streams	HTTP/2	QUIC
Clear-text version	Yes	No
Independent streams	No	Yes
Header compression	HPACK	QPACK
Server push	Yes	Yes
Early data	In theory	Yes
0-RTT Handshake	No	Yes
Prioritization	Messy	Changes

“The ultimate guide to HTTP resource prioritization”

Who: Robin Marx

Where: Web Performance devroom (H.1309)

When: 17:00 (today, Saturday)

(I bet it is already too late to get a seat in that room, so relax and watch the video after the fact instead.)

HTTP/3 is faster

(Thanks to QUIC)

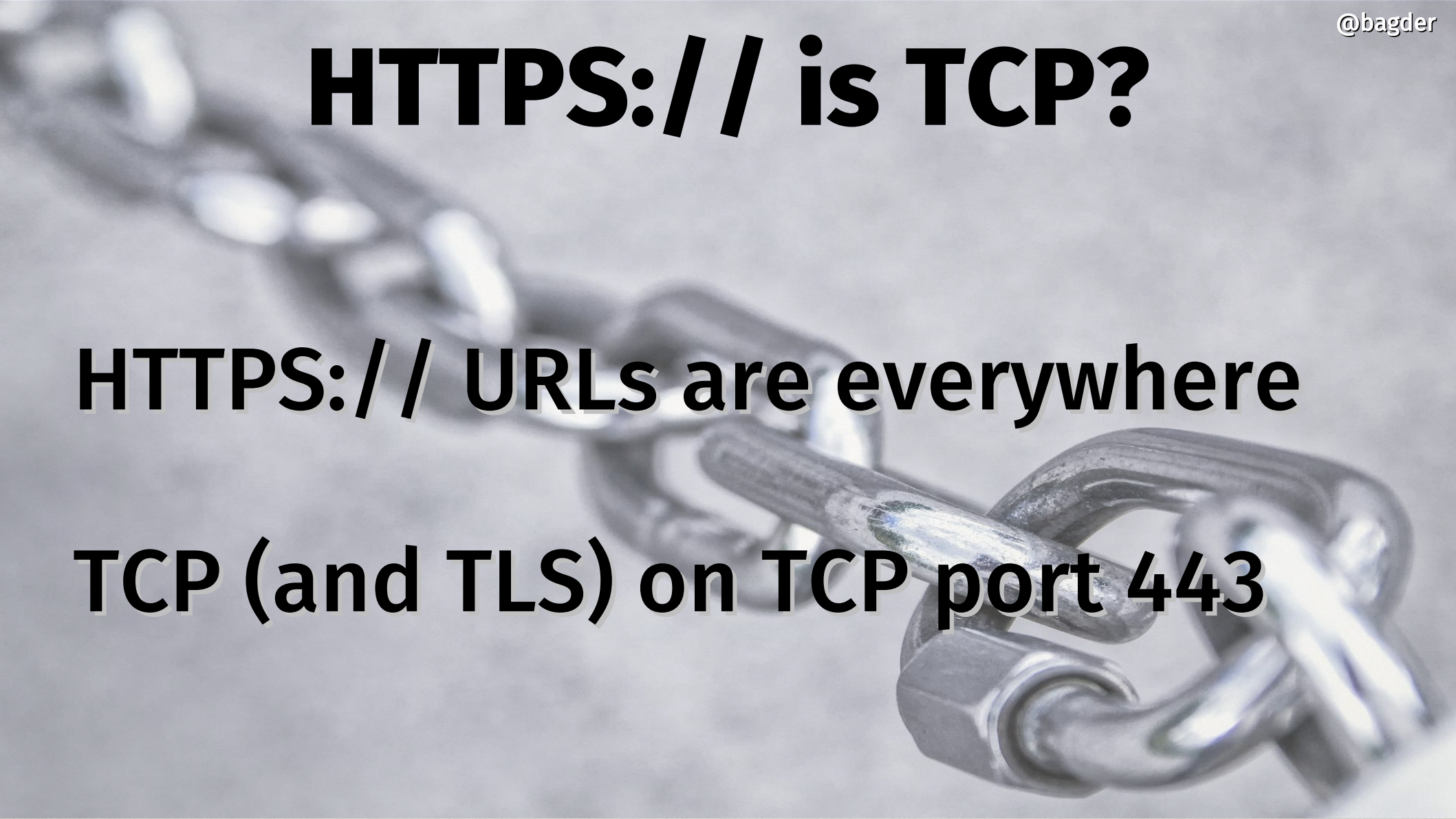
Faster handshakes

Early data that works

The independent streams

By how much remains to be measured!

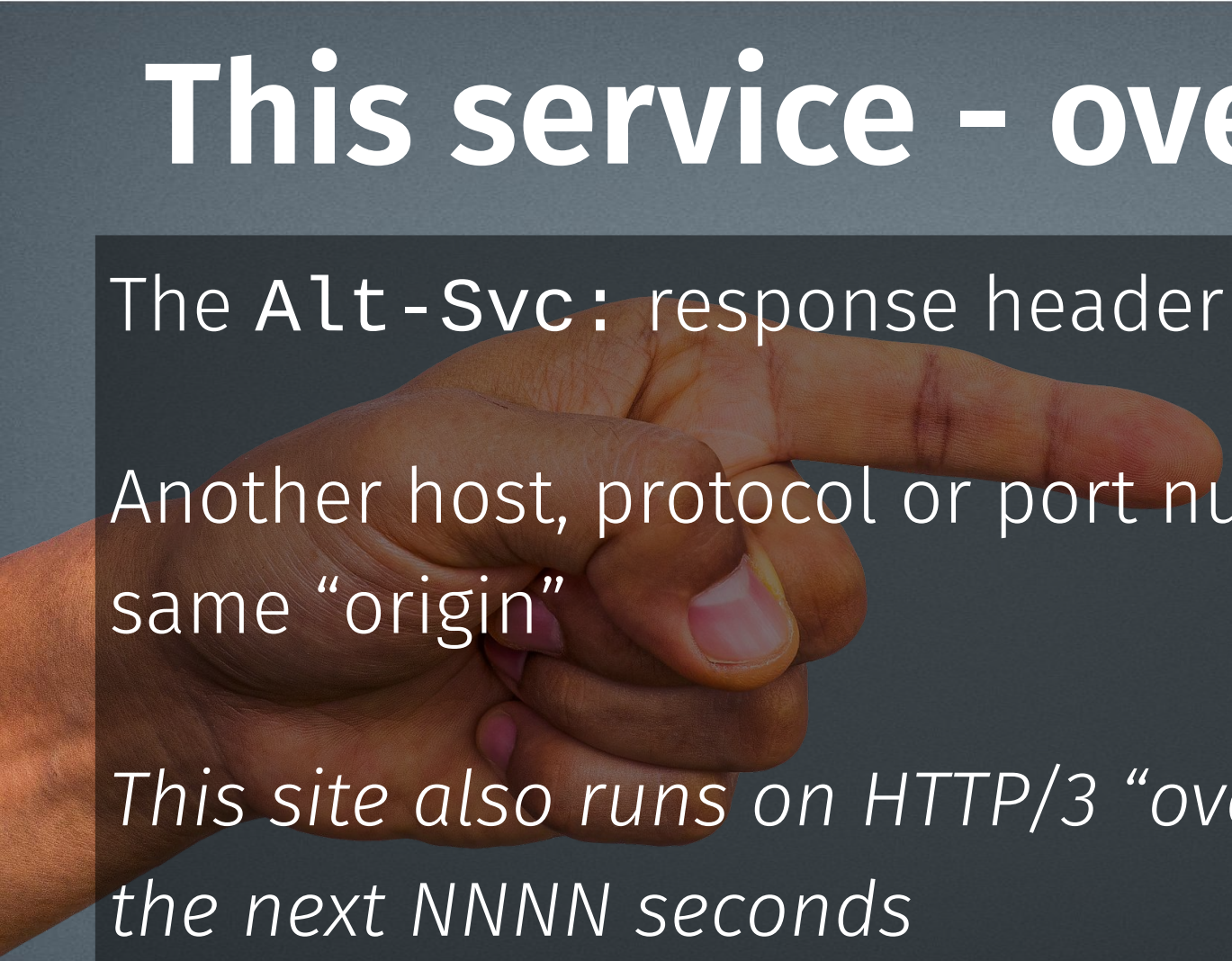
HTTPS:// is TCP?



HTTPS:// URLs are everywhere

TCP (and TLS) on TCP port 443

This service - over there!



The `Alt-Svc:` response header

Another host, protocol or port number is the same “origin”

This site also runs on HTTP/3 “over there”, for the next NNNN seconds

Race connections?

Might be faster

Probably needed anyway

QUIC connections verify the cert

HTTPSSVC – alt-svc: done in DNS

Will HTTP/3 deliver?

UDP challenges

3-7% of QUIC attempts fail

Clients need fall back algorithms

QUIC looks like a DDOS attack

CPU hog

2-3 times the CPU use

Unoptimized UDP stacks

Non-ideal UDP APIs

Missing hardware offload

The TLS situation (1/2)

TLS was made for TCP

TLS is sent over TCP as *records* containing individual *messages*

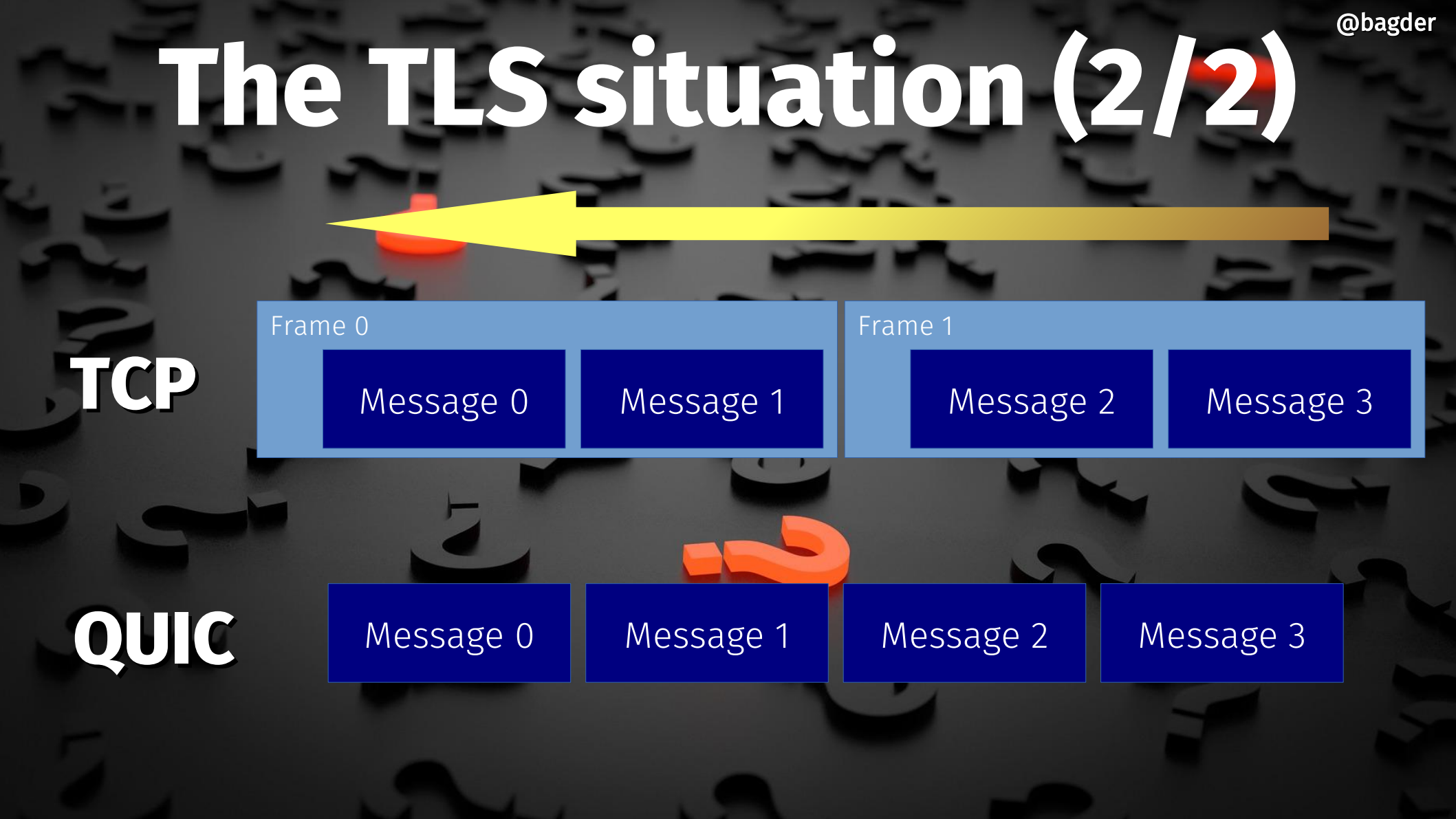
QUIC uses TLS *messages*

No TLS library support(ed) TLS messages

QUIC also needs additional secrets

The TLS situation (2/2)

@bagder



Userland

All QUIC stacks are user-land

No standard QUIC API

Will it be moved to kernels?

Tooling

Lack of tooling

Hooray for 

qlog & qvis

Ship date

Implementations

Over a dozen QUIC and HTTP/3 implementations

Google, Mozilla, Apple, Facebook, Microsoft, Akamai, Fastly, Cloudflare, F5, LiteSpeed, Apache, and more

C, C++, Go, Rust, Python, Java, TypeScript, Erlang

Monthly interops

Implementation Status

curl

Chrome and Edge Canary, Firefox Nightly

Caddy and LiteSpeed

nginx-patch + quiche

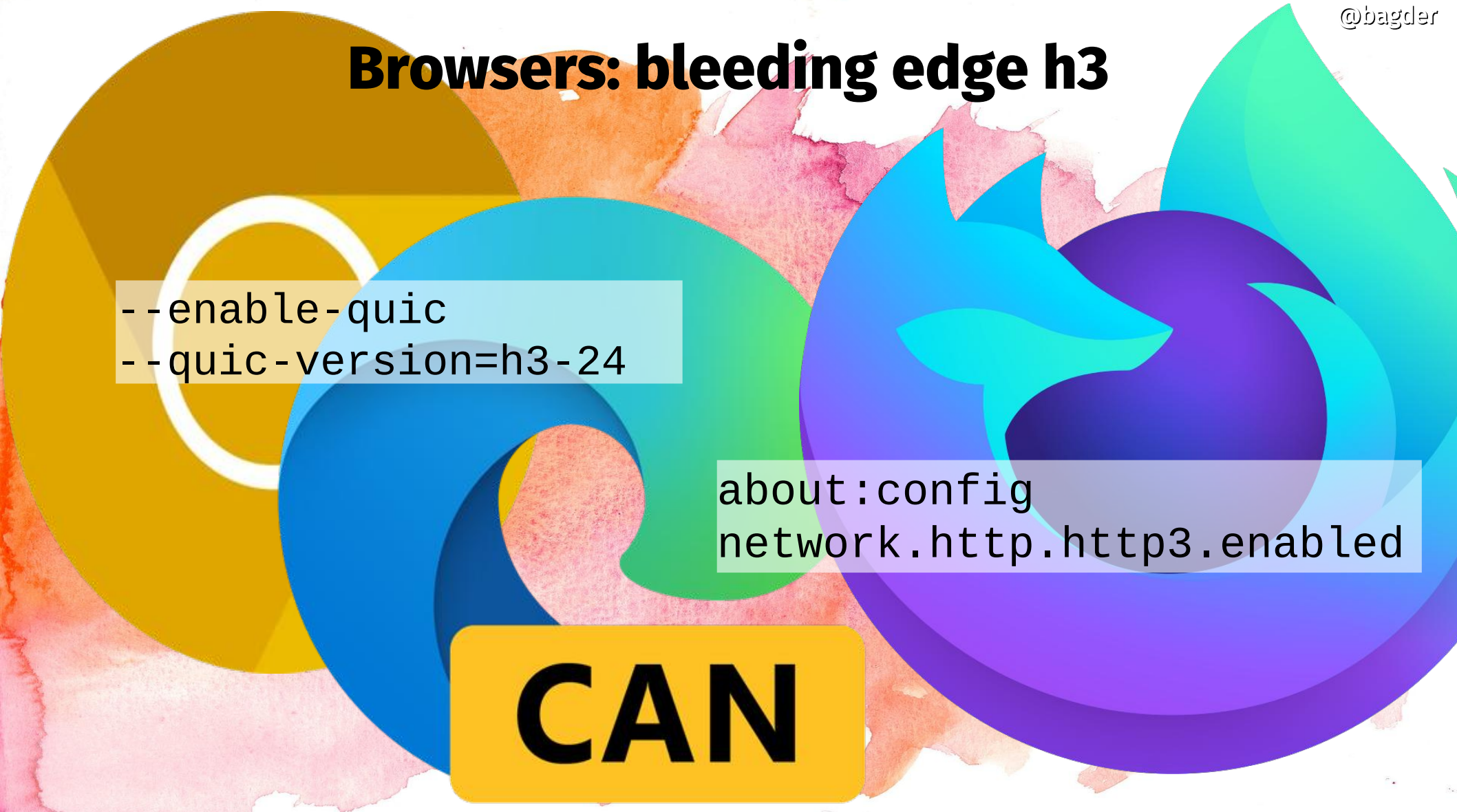
Wireshark

No Safari

No Apache httpd, IIS or official nginx

OpenSSL PR #8797

Browsers: bleeding edge h3



HTTP/3 in curl://

Experimental h3-25 works!

Alt-svc support is there

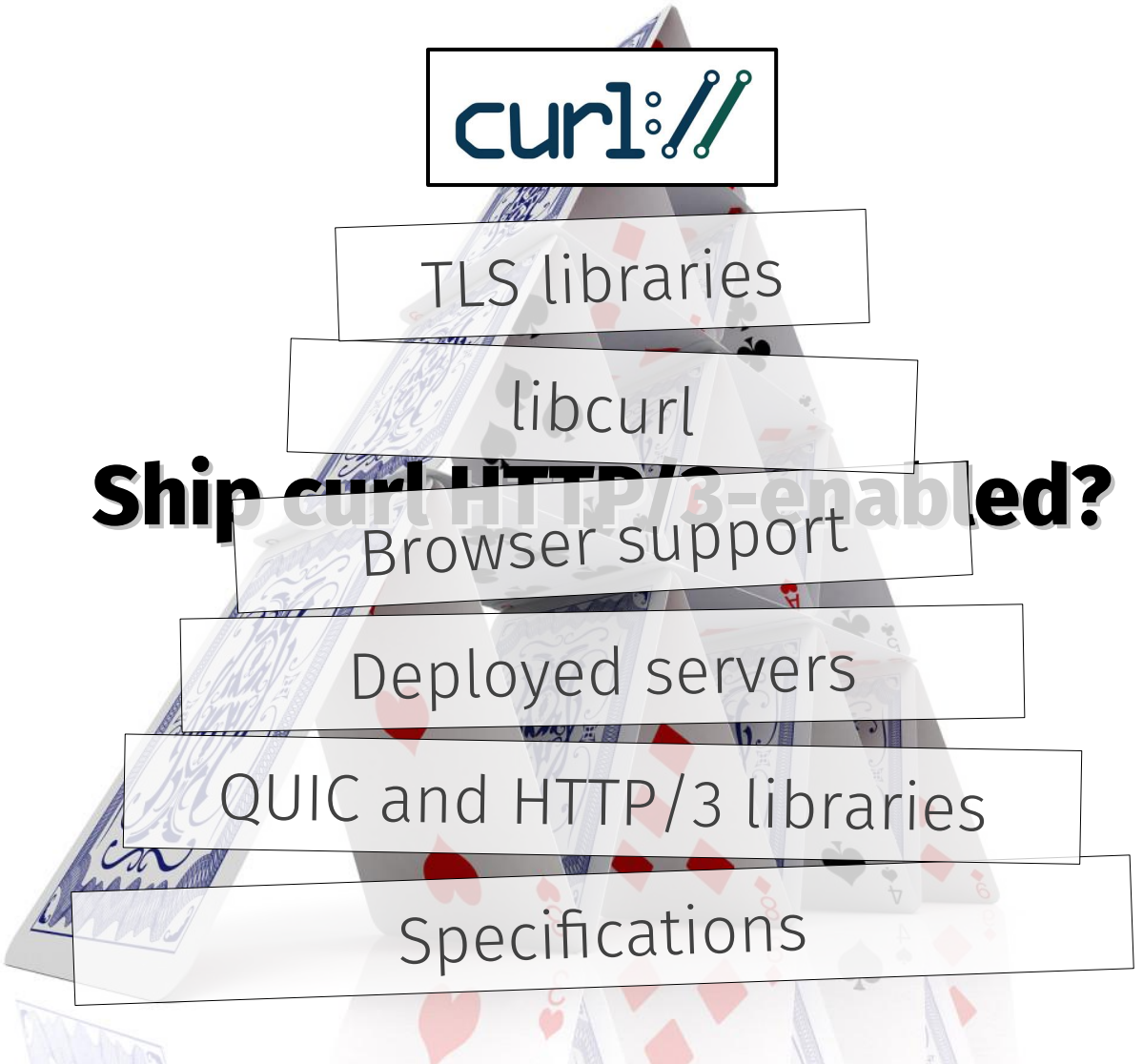
Based on ngtcp2 and QUICHE

Fallback is tricky

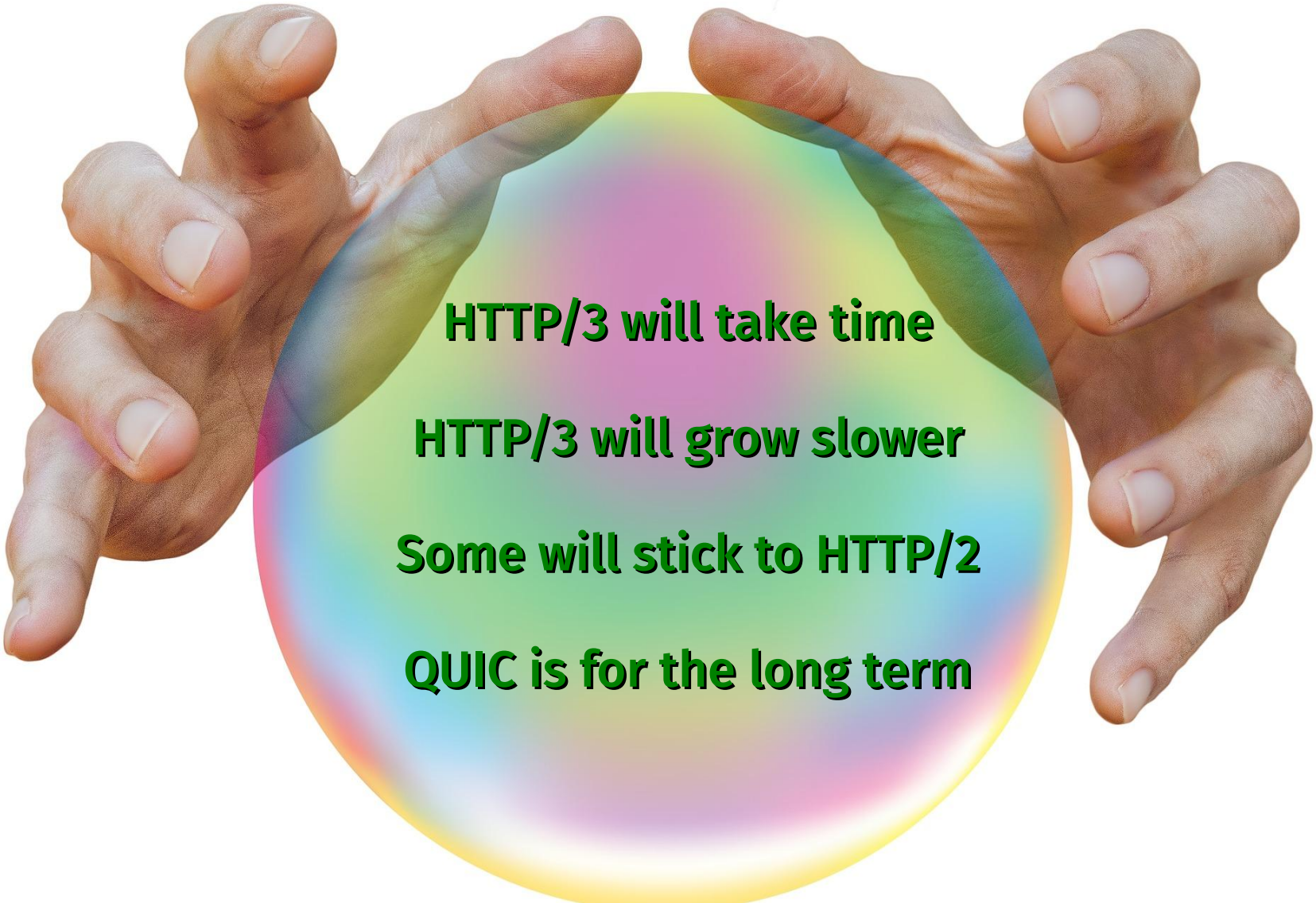
Try it!

curl HTTP/3 command line

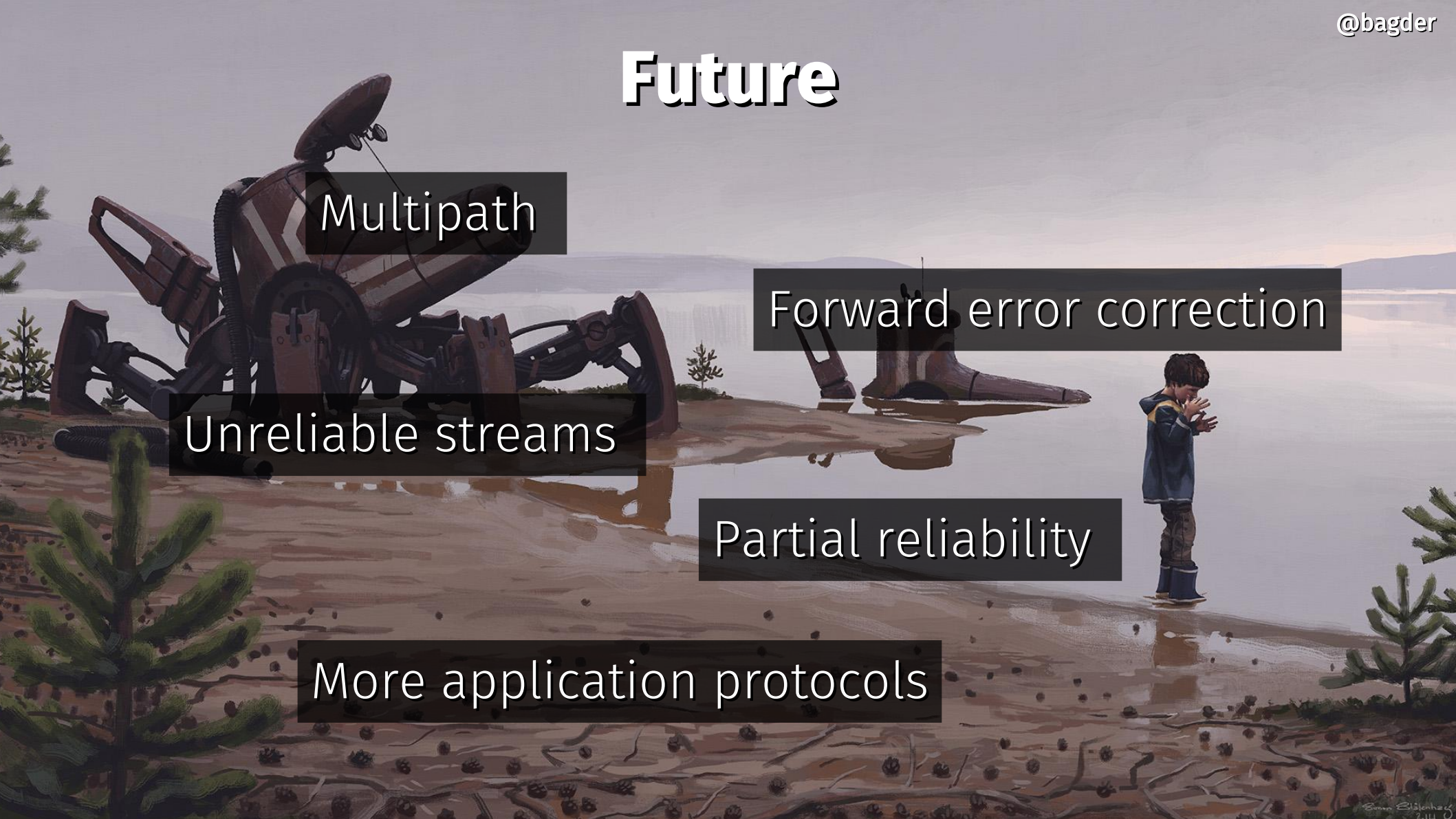
```
$ curl --http3 --head https://example.com/
HTTP/3 200
date: Wed, 09 Oct 2019 11:16:06 GMT
content-type: text/html
content-length: 10602
set-cookie: crazy=d8bc7e7; expires=Thu, 08-Oct-22
11:16:06 GMT; path=/; domain=example.com;
alt-svc: h3-24=":443"; ma=86400
```



Ship curl HTTP/3-enabled?


HTTP/3 will take time
HTTP/3 will grow slower
Some will stick to HTTP/2
QUIC is for the long term

Future


Multipath
Forward error correction
Unreliable streams
Partial reliability
More application protocols

Wait a minute, what about...

Websockets?

Not actually a part of HTTP(/3)

RFC 8441 took a long time for HTTP/2

Can probably be updated for HTTP/3

draft-vvv-webtransport-http3-01

Still in progress

Take-aways

HTTP/3 is coming

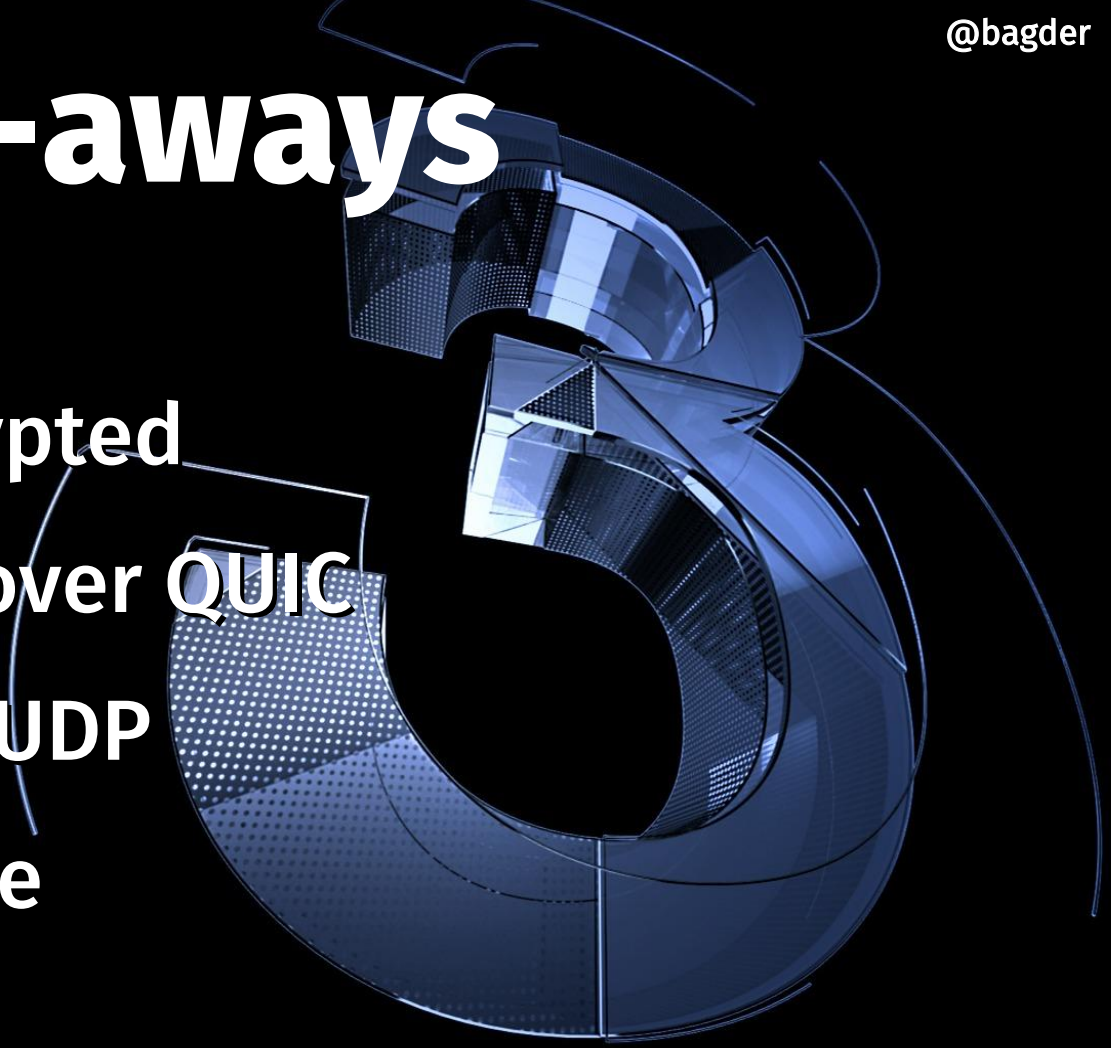
HTTP/3 is always encrypted

Similar to HTTP/2 but over QUIC

QUIC is transport over UDP

Challenges to overcome

Mid 2020?



HTTP/3 Explained

https://daniel.haxx.se/http3-explained

Thank you!

Questions?

Daniel Stenberg
@bagder
https://daniel.haxx.se/

License

Links to data and more info

QUIC drafts: https://quicwg.github.io/

DATAGRAM: https://tools.ietf.org/html/draft-pauly-quic-datagram-05

QUIC multipath: https://tools.ietf.org/html/draft-deconinck-quic-multipath-03

HTTPS stats Firefox: https://letsencrypt.org/stats/#percent-pageloads

HTTPS stats Chrome: https://transparencyreport.google.com/https/overview?hl=en

Web Transport: https://tools.ietf.org/html/draft-vvv-webtransport-http3-01

Images: http://www.simonstalenhag.se/ and https://pixabay.com/

HTTP/3 Explained: https://http3-explained.haxx.se/

QUIC implementations: https://github.com/quicwg/base-drafts/wiki/Implementations

Nginx + quiche: https://github.com/cloudflare/quiche/tree/master/extras/nginx

HTTPSSVC: https://tools.ietf.org/html/draft-ietf-dnsop-svcb-httpssvc-01

qlog: https://github.com/quiclog/internet-drafts

qvis: https://qvis.edm.uhasselt.be

Build curl with HTTP/3: https://github.com/curl/curl/blob/master/docs/HTTP3.md