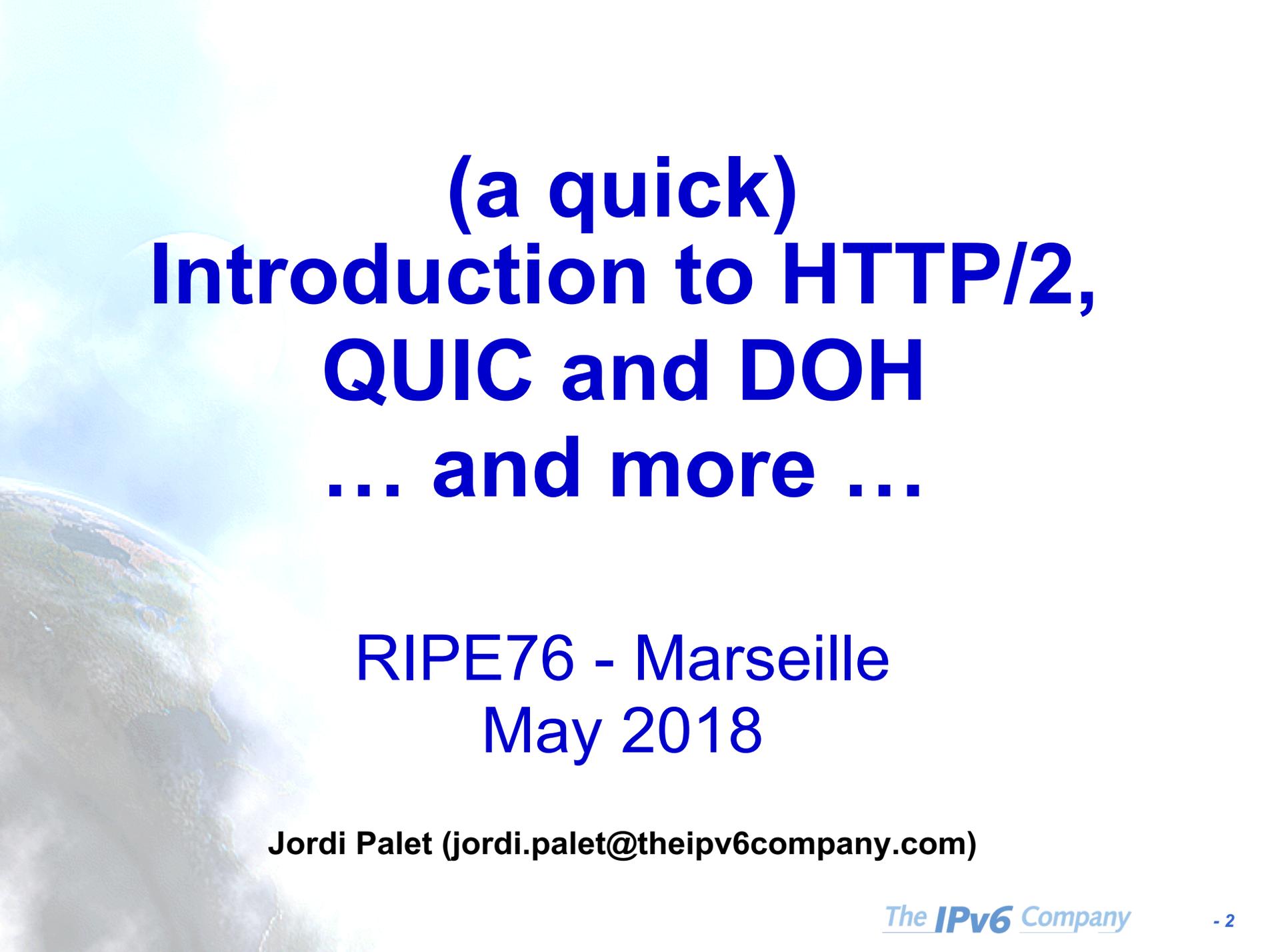


# A New Internet?

RIPE76 - Marseille  
May 2018

Jordi Palet ([jordi.palet@theipv6company.com](mailto:jordi.palet@theipv6company.com))



**(a quick)  
Introduction to HTTP/2,  
QUIC and DOH  
... and more ...**

**RIPE76 - Marseille  
May 2018**

**Jordi Palet ([jordi.palet@theipv6company.com](mailto:jordi.palet@theipv6company.com))**

# Internet is Changing

- More and more, Internet traffic is moving from many protocols and ports to all HTTP and HTTPS (ports 80 and 443)
- Only DNS is not yet using HTTP/HTTPS, however is also coming
- This change is due to many factors, including many networks filtering “what they don’t know”, so limiting the access to those protocols, which means that apps are forced to use only those
- The advantage is that by improving “only” those protocols, we can greatly enhance the Internet performance, instead of requiring improving “lots” of other protocols
- Also, there is more “perception” that security and privacy are key, so we can take the opportunity as well to secure more and more traffic

# From HTTP/1.1 to SPDY to HTTP/2

- Web sites have greatly evolved since HTTP(1.1) -> 1991(1999)
  - From few kbytes and objects, to few megabytes and hundreds of objects in a single page
  - HTTP/1.1 doesn't perform well for the actual situation
- In 2009, Google engineers posted about the SPDY project
  - Multiplexing (concurrent requests across a single TCP connection)
  - Compress and reduce HTTP headers
  - Prioritize assets (vital resources for the correct display of the page could be sent first)
  - “Server push” (the server can push resources to the browser before being asked)
- IETF HTTPbis WG, in 2012, used SPDY as starting point for HTTP/2 (RFC7540, 2015)
- Doesn't require HTTPS
  - Browser vendors only implemented HTTP/2 with TLS (HTTPS)
    - “Let's Encrypt” (<https://letsencrypt.org/>) is free, automated and open, so solves this “issue”
- With TLS, uses Application Layer Protocol Negotiation (ALPN, RFC7639) to negotiate HTTP/2 with servers
  - Earlier implementations supported NPN (Next Protocol Negotiation) because the SPDY support
  - Main difference: Who decides what protocol to speak
    - NPN -> The client makes de choice
    - ALPN -> The client gives the server a list of protocols and the server pick the one it wants

# SPDY and HTTP/2 Support

- SPDY support in 2016 was over 90% worldwide
- HTTP/2 global support for implementations
  - <https://github.com/http2/http2-spec/wiki/Implementations>
- Web sites using HTTP/2, is around 26% worldwide
  - <https://w3techs.com/technologies/details/ce-http2/all/all>
  - Because HTTPS is required “de facto”
  - However all the “top” web sites use it, so traffic is a much bigger %

# HTTP/2 Summary View

## HTTP/2

### 1. One TCP connection

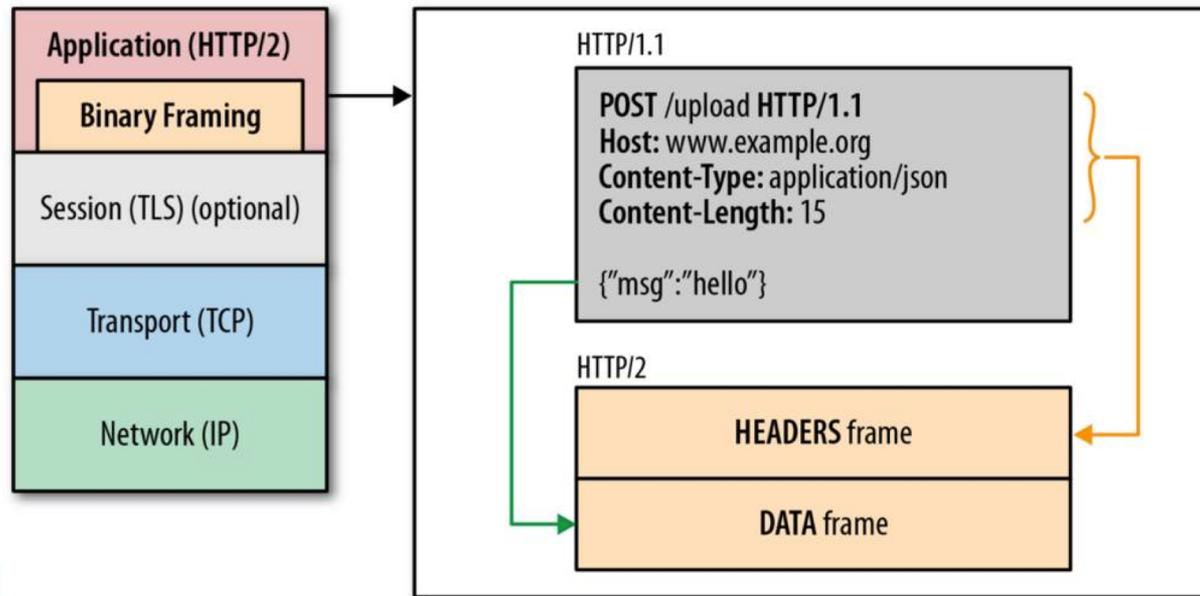
### 2. Request → Stream

- Streams are multiplexed
- Streams are prioritized

### 3. Binary framing layer

- Prioritization
- Flow control
- Server push

### 4. Header compression (HPACK)



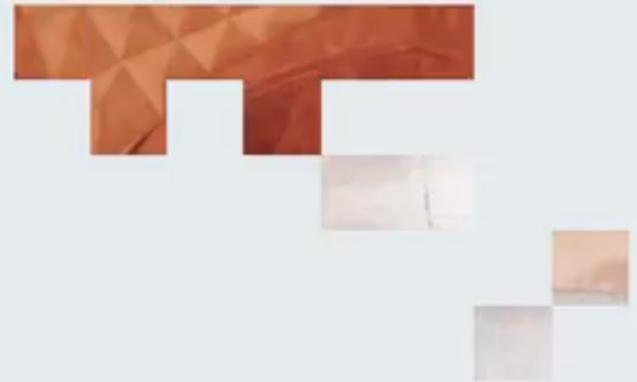
# Demo

- Typically 2.5x faster
- <https://imagekit.io/demo/http2-vs-http1>
- <https://youtu.be/QCEid2WCszM>

HTTP/2



HTTP/1.1



# Chrome Extensions

- HTTP/2 and SPDY indicator



Gmail Imágenes [Iniciar sesión](#)

Google

Buscar con Google

Voy a tener suerte

Ofrecido por Google en: [English](#)

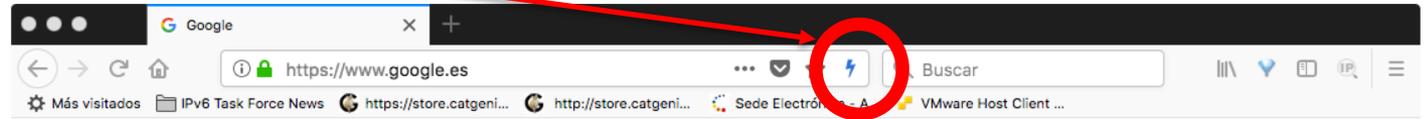
Reino Unido

[Publicidad](#) [Empresa](#) [Acerca de](#)

[Privacidad](#) [Condiciones](#) [Configuración](#)

# Firefox Extensions

- HTTP/2 Indicator



Gmail Imágenes [Iniciar sesión](#)

Google

Buscar con Google

Voy a tener suerte

Ofrecido por Google en: [English](#)

Reino Unido

[Publicidad](#) [Empresa](#) [Acerca de](#)

[Privacidad](#) [Condiciones](#) [Configuración](#)

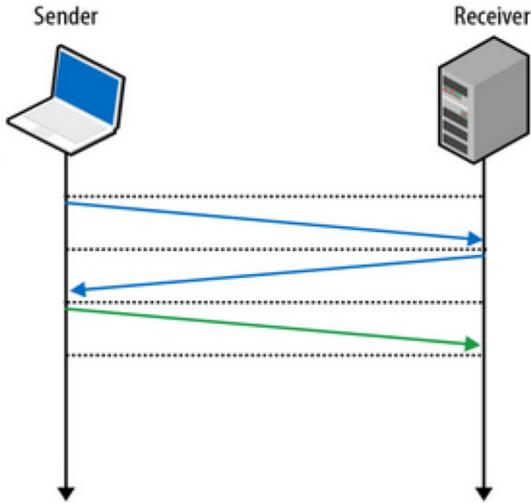
# QUIC in Short

- During the SPDY development, it was obvious that TCP is inefficient for most of the actual Internet usages, so started to work on QUIC (**Q**uick **U**DP **I**nternet **C**onnections)
- IETF QUIC WG (2016) is developing a UDP-based, stream-multiplexing, encrypted transport protocol
  - Initial use case: HTTP-over-UDP
- Already deployed by Google, so around 10% of Internet traffic uses it
- In short:
  - Transport over UDP
  - Typically implemented in Application Process (not kernel)
  - Functionally = TCP + TLS + streams
  - Includes TLS 1.3, to establish session keys and encrypt **\*ALL\*** (including ACKs)
  - Enables 0-RTT
  - In draft-ietf-quick-transport-11, only few parts of the “short header” used for all the packets except the handshake, remain unencrypted, which Disallow passive RTT measurement/packet lost (a “spin bit” proposal, draft-trammell-quick-spin, in the header flipping once per round trip, to allow estimate the RTT)

# HTTP vs HTTPS vs QUIC

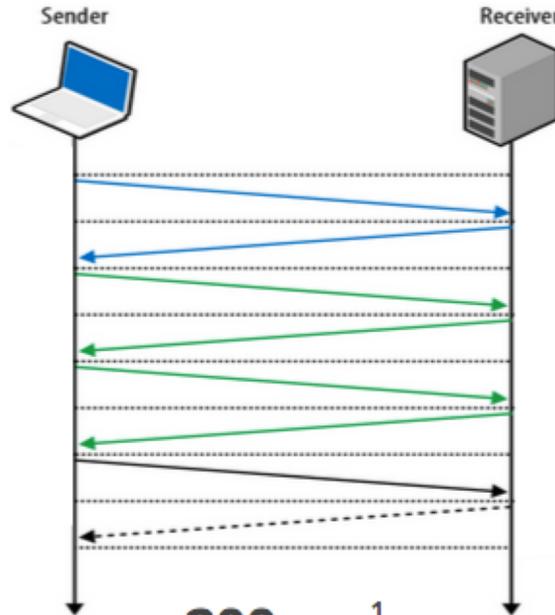
## Zero RTT Connection Establishment

TCP



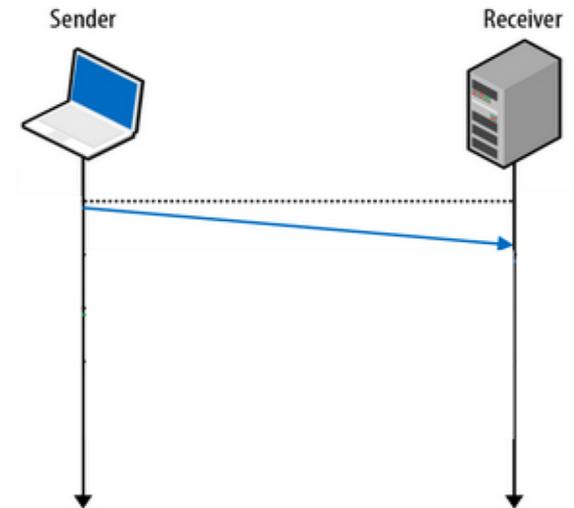
100 ms

TCP + TLS



200 ms<sup>1</sup>  
300 ms<sup>2</sup>

QUIC  
(equivalent to TCP + TLS)

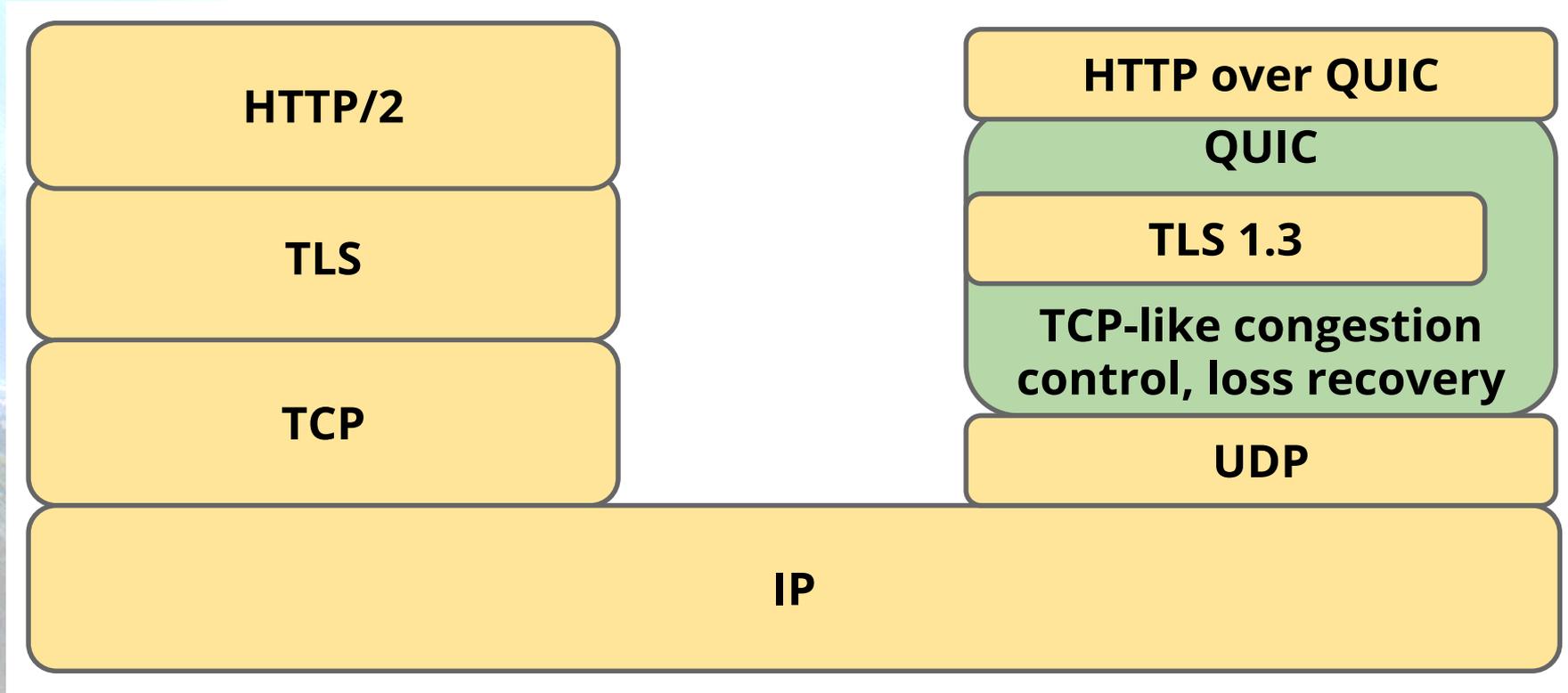


0 ms<sup>1</sup>  
100 ms<sup>2</sup>

1. Repeat connection
2. Never talked to server before

\* <https://blog.chromium.org/2015/04/a-quick-update-on-googles-experimental.html>

# HTTP/2 vs QUIC



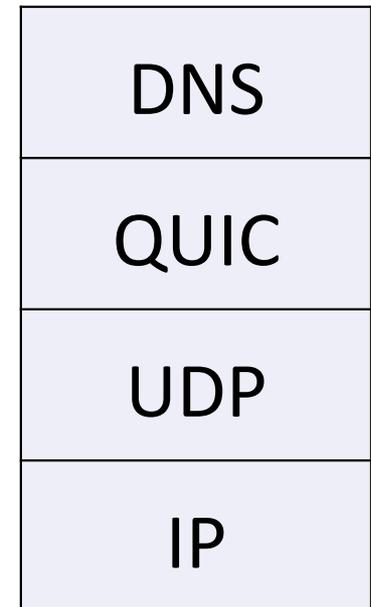
\* <https://www.ietf.org/proceedings/96/slides/slides-96-quic-5.pdf>

# DOH

- The IETF DNS over HTTPS (DOH) WG, is standardizing the encoding of DNS queries and responses over HTTPS
- Enable DNS to work over paths where existing methods have issues (UDP, TLS & DTLS)
- Transport suitable for:
  - Traditional DNS clients
  - Native web apps that use DNS
- Only using HTTP/2
- Is not “just” a tunnel over HTTP:
  - Establishes default media formatting types for requests/responses
  - Use normal HTTP content negotiation mechanism for selecting alternatives that endpoints may prefer (future new use cases)
  - Aligned with HTTP features (caching, redirection, proxying, authentication, compression)
- Avoid that authorities impose traffic discriminations or censorship
  - if they wish to do so, with DOH they will need to restrict full access to the web server providing the DOH

# DNS over QUIC

- Transport privacy for DNS
  - draft-huitema-quick-dns-over-quick-03
- QUIC
  - Transport over UDP
  - Typically implemented in Application Process (not kernel)
  - Functionally = TCP + TLS + streams
  - Includes TLS 1.3
  - Enables 0-RTT
- DNS over QUIC
  - High performance transport



# Comparing DNS “transport”

	UDP	TCP	TLS	DTLS	QUIC
Transport efficiency					
Connection set up time	✓	✗	✗	✗	<b>0-RTT</b>
Head of queue blocking	✓	✗	✗	✓	✓
Retransmission efficiency	✗	✓	✓	✗	✓
Long messages (DNSSEC)	✗	✓	✓	✗	✓
Security					
Three ways handshake	✗	✓	✓	✓	✓
Encryption & Authentication	✗	✗	✓	✓	✓

\* slides-99-dprive-dns-over-quick

# Conclusions

- HTTP/2 reduce the number of round-trips, avoid blocking by means of parallel streams and allows discarding unwanted streams, so a much faster and better web experience
  - “De facto” requires HTTPS, “Let’s Encrypt” to the rescue
- QUIC will decrease latency, avoid packet loss blocking all the streams (as in HTTP/2) and makes connections possible with different interfaces (mobility, flapping, ...)
- DOH can avoid DNS failures and some censorship
  - DNS over QUIC also provides DNS transport privacy
- How all this will impact in non-web traffic and change Internet?

# Thanks!

**Contact:**

– **Jordi Palet:**

**[jordi.palet@theipv6company.com](mailto:jordi.palet@theipv6company.com)**