A New Internet?

Introduction to HTTP/2, QUIC and DOH … and more …

ESNOG/GORE21
Barcelona
April 2018

Jordi Palet (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

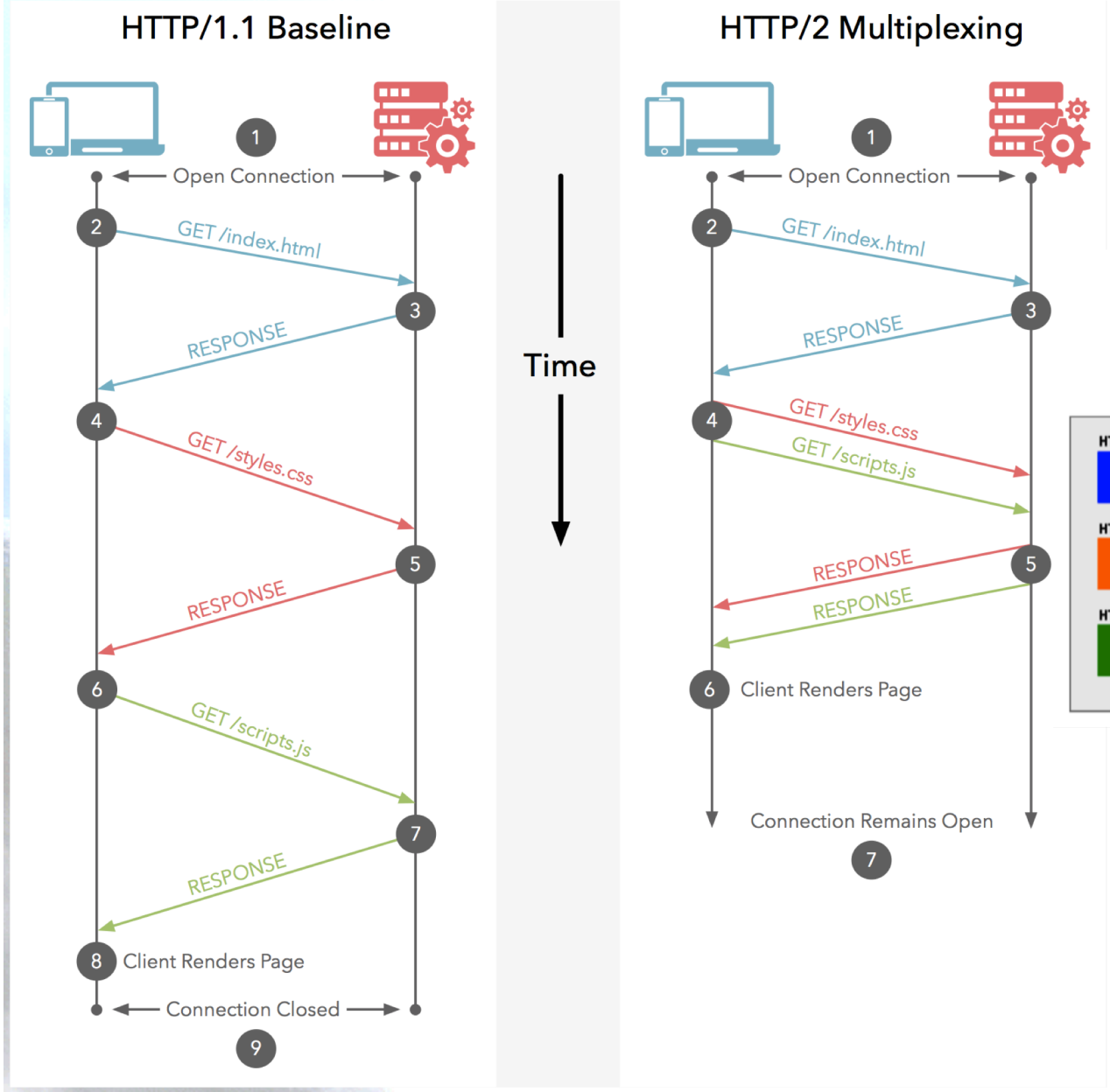
From HTTP/1.1 to SPDY

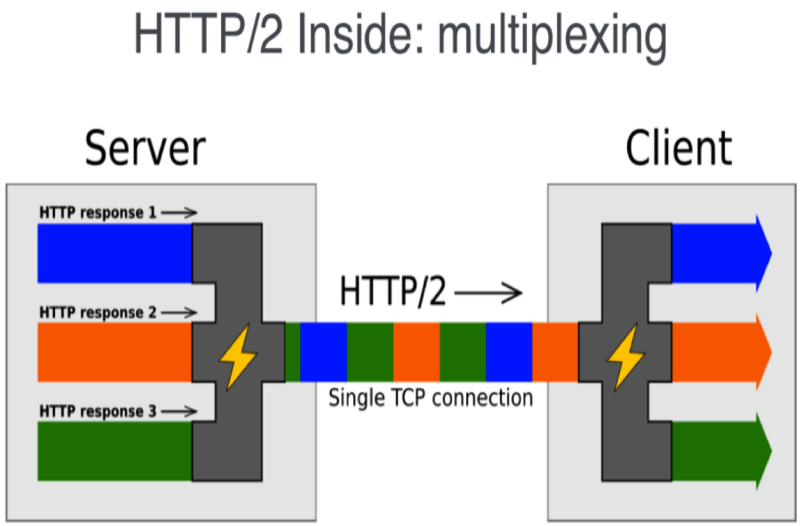
- HTTP was initially defined in 1991, revised in 1999 (HTTP/1.1)
- Web sites have greatly evolved since then
  - From few kbytes and objects, to few megabytes and hundreds of objects in a single page
  - HTTP/1.1 doesn't perform well for the actual situation
- In 2009, Google engineers posted about the 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)
- SPDY is a tunnel for HTTP in HTTPS
  - Requires support in both sides (server and browser)
  - Support in 2016 was over 90% worldwide
  - Uses Next Protocol Negotiation (NPN) to negotiate SPDY with TLS servers

From SPDY to HTTP/2

- IETF HTTPbis WG, in 2012, used SPDY as starting point for HTTP/2
- **RFC7540** (HTTP/2) approved in 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 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
- Global support for implementations
  - https://github.com/http2/http2-spec/wiki/Implementations
- Web sites using it, is around 25% 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/1.1 vs HTTP/2





* http://blog.restcase.com/http2-benefits-for-rest-apis/

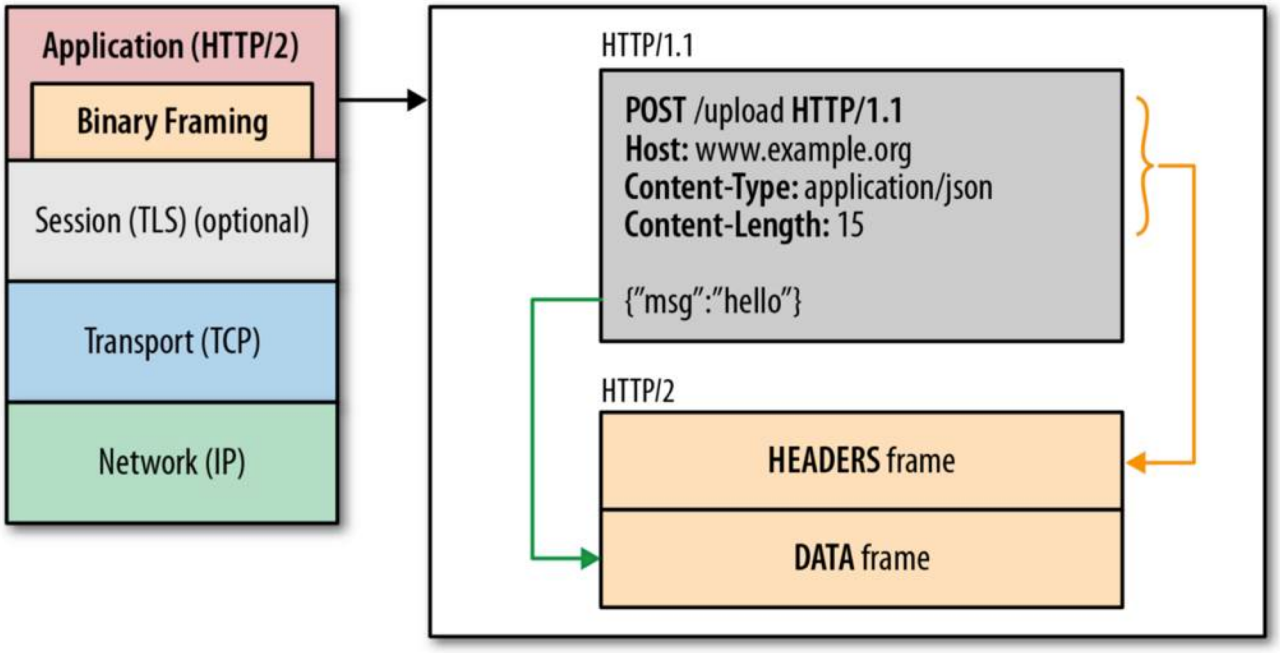
HTTP/2 (RFC7540) in Short

- Binary protocol
  - Easier framing
  - Different frame types, same setup for all
    - length, type, flags, stream identifier, frame payload
  - 10 different frame types (2 to map HTTP/1.1 features DATA & HEADERS)
- Multiplexed streams
  - A stream is an independent bi-directional sequence of frames exchanged between client and server
- Priorities and dependencies
  - Each stream has a priority (“weight”) in case there are server resource limitations, and allow building ”priority trees” with “child streams dependencies” and dynamically change those (enhances user experience while browsing)
- Header compression for HTTP/2 (HPACK – RFC7541)
- Reset
  - No need to negotiate a new TCP connection and waste bandwidth
- Server push
  - Server may try to guess what are “next resources” to be requested by client
- Flow control
  - DATA frames controlled, per stream, in the same “style” as in SSH

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)**



* http://blog.restcase.com/http2-benefits-for-rest-apis/

HTTP/2 Extensions

- Client and server can negotiate new frame types on a hop-by-hop basis
  - Those frames aren't allowed to change state and aren't flow controlled
  - Subjected to new standards

Alternative services

- Longer TCP connections, may affect load balancers, so may want to tell the client to connect to another host (performance, site brought to maintenance, …)
- Server send "Alt-Svc" header (RFC7838 – HTTP Alternative Services)
  - Another route to the same content, using another service, host and port number
  - Example: Alt-Svc: h2="new.example.org:80", h2c="other.example.org:8080"
- Opportunistic TLS
  - The Alt-Svc header allows a server with http to inform the client that the same content is available over TLS

Implementing HTTP/2 in Apache

- Example config in Ubuntu Server (>=14.04), Apache (>=2.4)
  - MPM event recommended. Never use prefork.

- Install it from Ondřej Surý PPA

```
$ sudo add-apt-repository ppa:ondrej/apache2
$ sudo apt-get update
$ sudo apt-get upgrade
```

- Enable HTTPs and related mods

```
$ cd /etc/apache2/mods-enabled
$ sudo ln -sf ../mods-available/socache_shmcb.* .
$ sudo ln -sf ../mods-available/ssl.* .
$ sudo ln -sf ../mods-available/http2.* .
```

- You need a VirtualHost with HTTPS (no changes there)
  - Your config Directive Protocols h2 h2c http/1.1

- Restart Apache2 & done

```
$ sudo service apache2 restart
```

Implementing HTTP/2 in nginx

- Example config in Ubuntu Server (>=16.04), Luckily (>=1.9.5)

- Modify your website config, should have HTTPS

```
server {
        listen 443 ssl http2 default_server;
        ssl_certificate         /path/to/server.cert;
        ssl_certificate_key     /path/to/server.key;
        # ...
        # Your HTTP server config here
        # ...
}
```

- Restart nginx & done

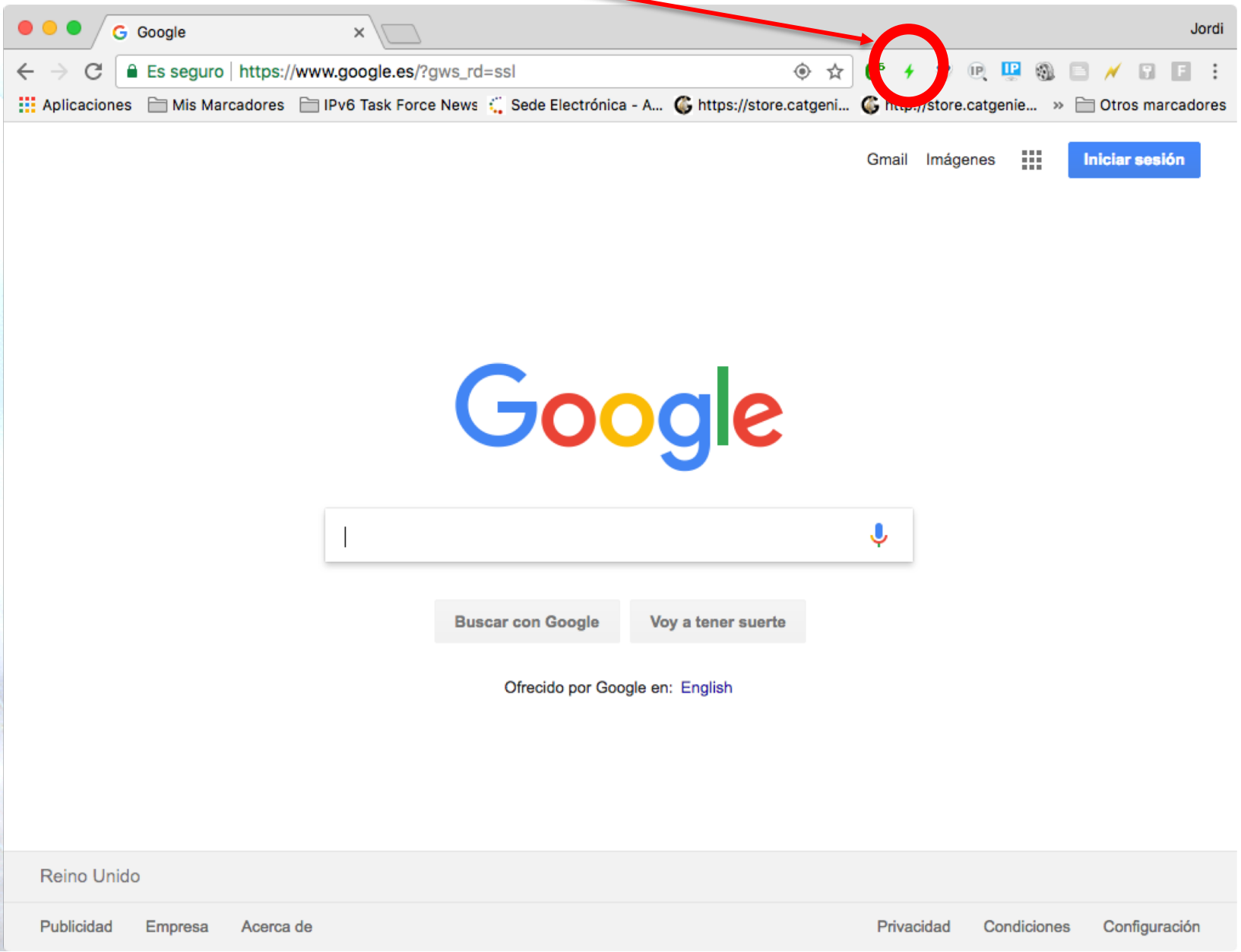
```
$ sudo service nginx restart
```

Demo

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

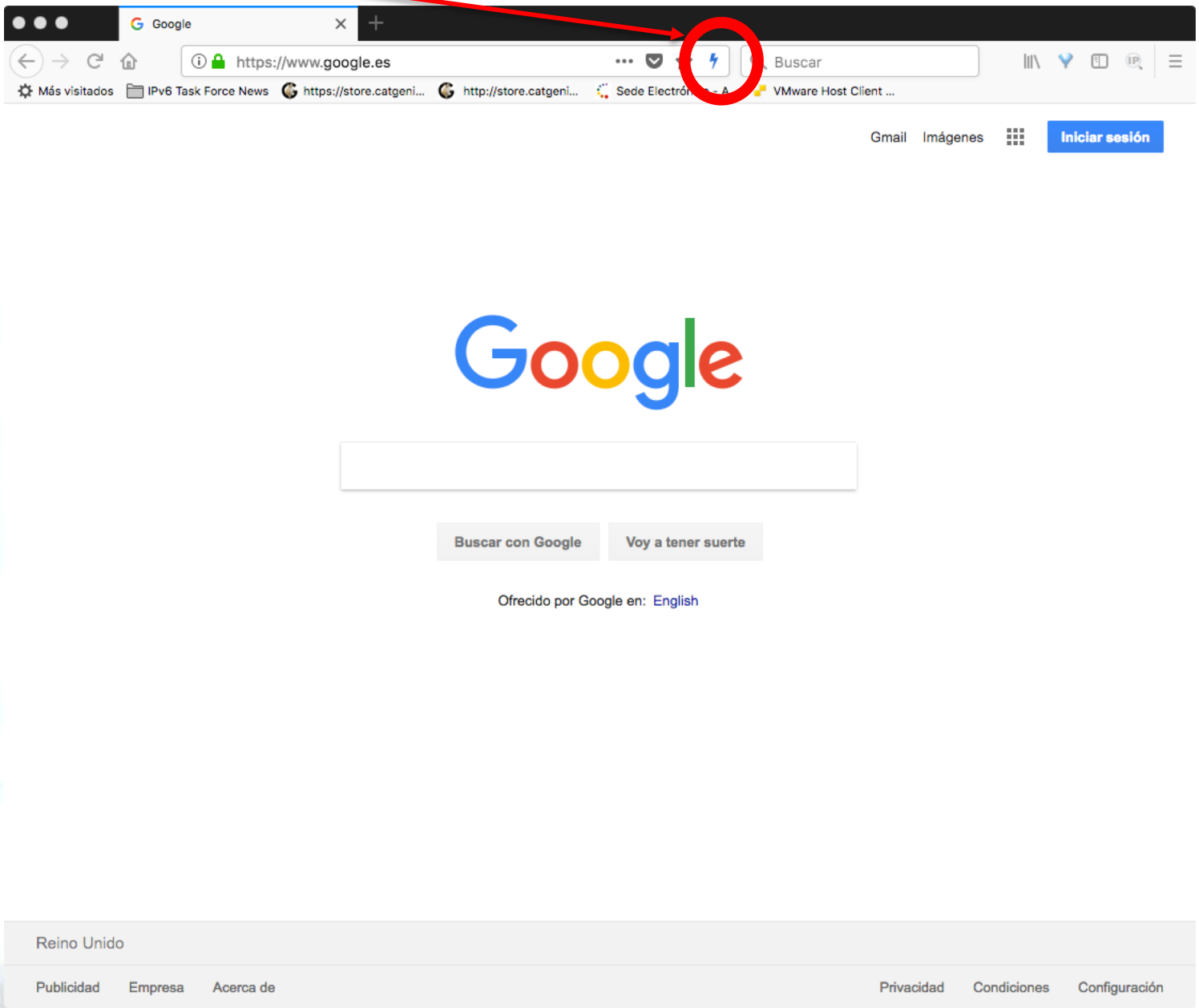
Chrome Extensions

- HTTP/2 and SPDY indicator



Firefox Extensions

- HTTP/2 Indicator

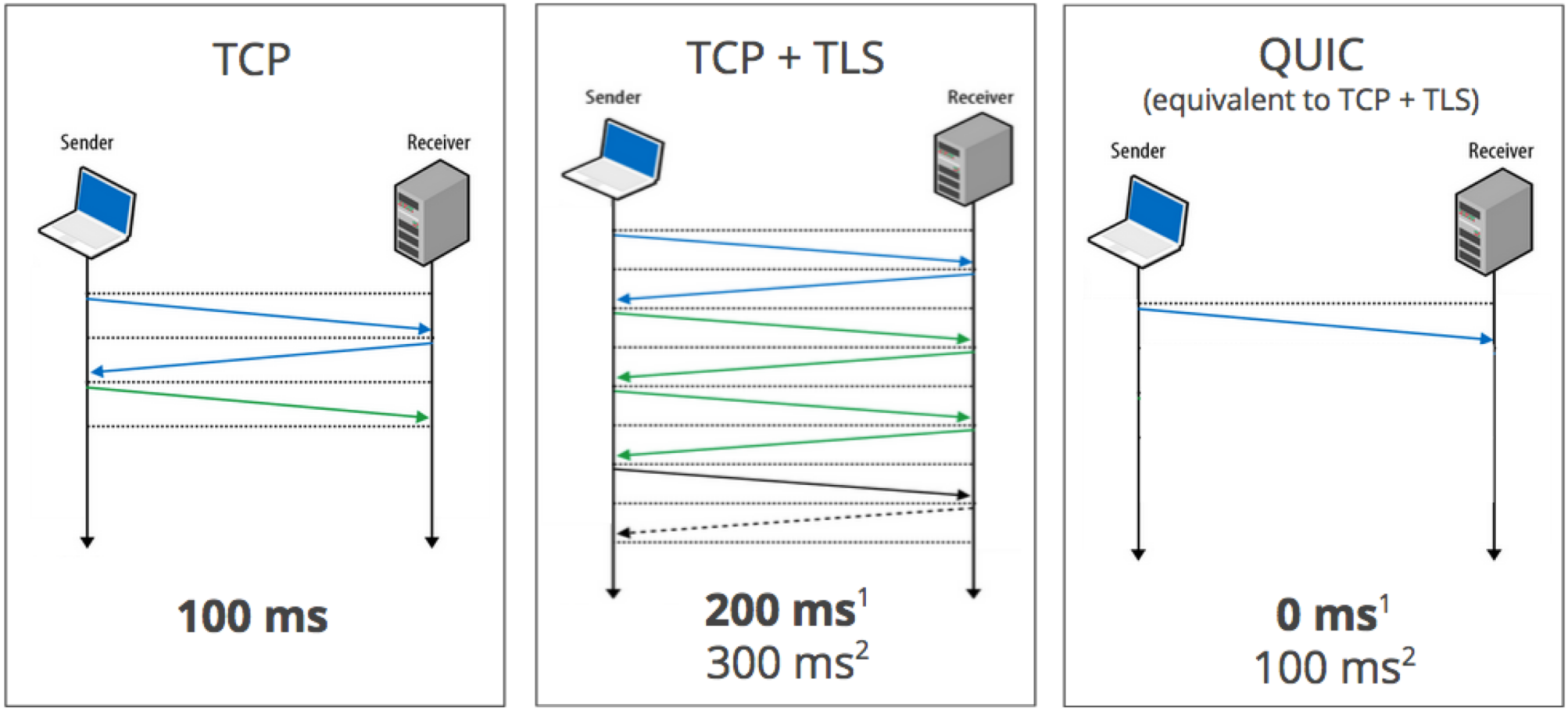


QUIC

- 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 9% of Internet traffic uses it
- QUIC standard requires encryption
  - TLS1.3 used to establish session keys and encrypt ***ALL*** the packets
    - Including ACKs
  - In actual draft (draft-ietf-quic-transport-08), only few parts of the “short header” used for all the packets except the handshake, remain unencrypted (packet number, an optional connection identifier and a byte with some flags and “packet type”)
- Disallow passive RTT measurement/packet lost
  - Proposal for a “spin bit” (draft-trammell-quic-spin) in the header flipping once per round trip, to allow estimate the RTT

HTTP vs HTTPS vs QUIC

Zero RTT Connection Establishment



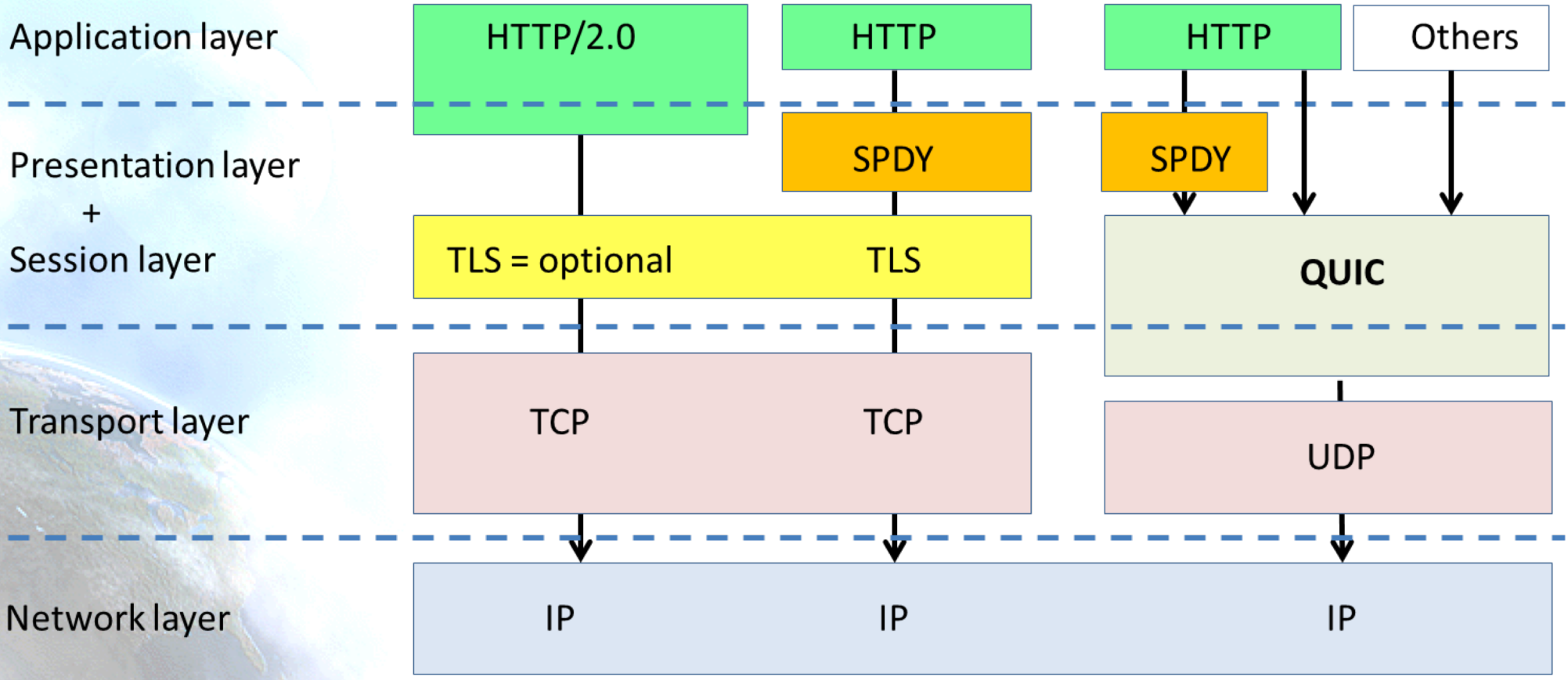
1. Repeat connection
2. Never talked to server before

* https://blog.chromium.org/2015/04/a-quic-update-on-googles-experimental.html

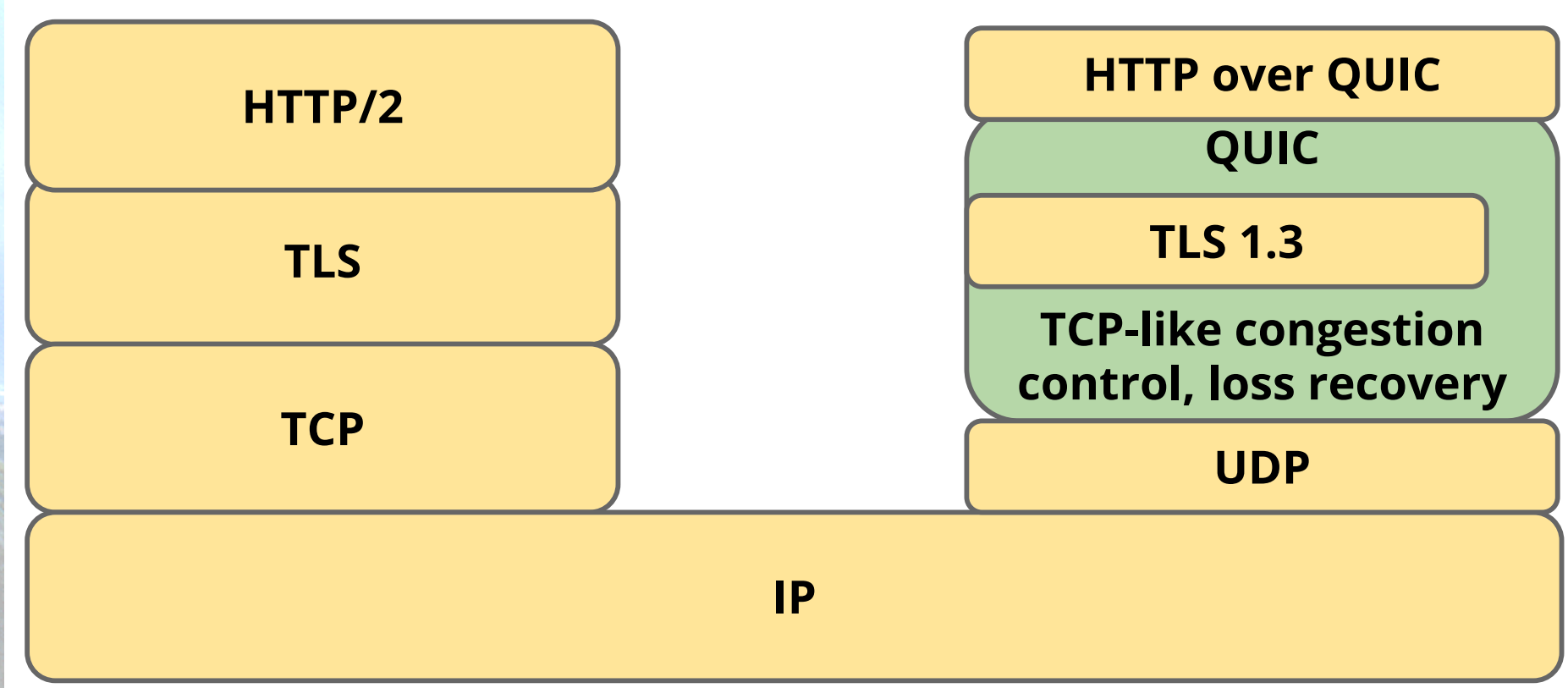
QUIC in Short

- Transport over UDP
- Typically implemented in Application Process
  - not kernel
- Functionally = TCP + TLS + streams
- Includes TLS 1.3
- Enables 0-RTT

Comparing Protocol Stacks



HTTP/2 vs QUIC



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

Implementing QUIC in servers

- Apache:
  - https://cwiki.apache.org/confluence/display/TS/QUIC
- Tracking status of many other implementations:
  - https://github.com/quicwg/base-drafts/wiki/Implementations
- Wireshark has already a QUIC decoder
- Interop test being carried out by IETF

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)

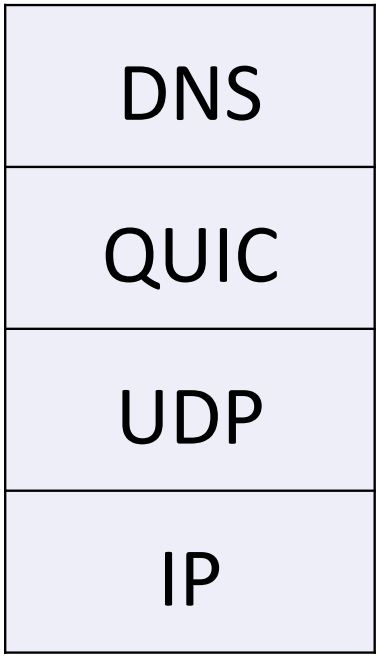
- Avoid that authorities impose traffic discriminations or censorship
  - if they wish to do so, with DOH they will need to restrict full access to the web server providing the DOH

DOH Implementations

- Generic and tools:
  - https://github.com/curl/curl/wiki/DNS-over-HTTPS
- Proxy_DNS (FastCGI endpoint + DNS proxy server)
  - https://github.com/BII-Lab/DNSoverHTTP
- Google:
  - https://developers.google.com/speed/public-dns/docs/dns-over-https
  - https://dnsprivacy.org/wiki/display/DP/Google%27s+Public+DNS-over-HTTPS

DNS over QUIC

- Transport privacy for DNS
  - draft-huitema-quic-dnsoquic
- 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	✓	✗	✗	✗	**0-RTT**
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