The network security impact of encrypted protocols

Tobias Mayer
Technical Solutions Architect
March 2018

Your Speaker

rchitect for Cybersecurity

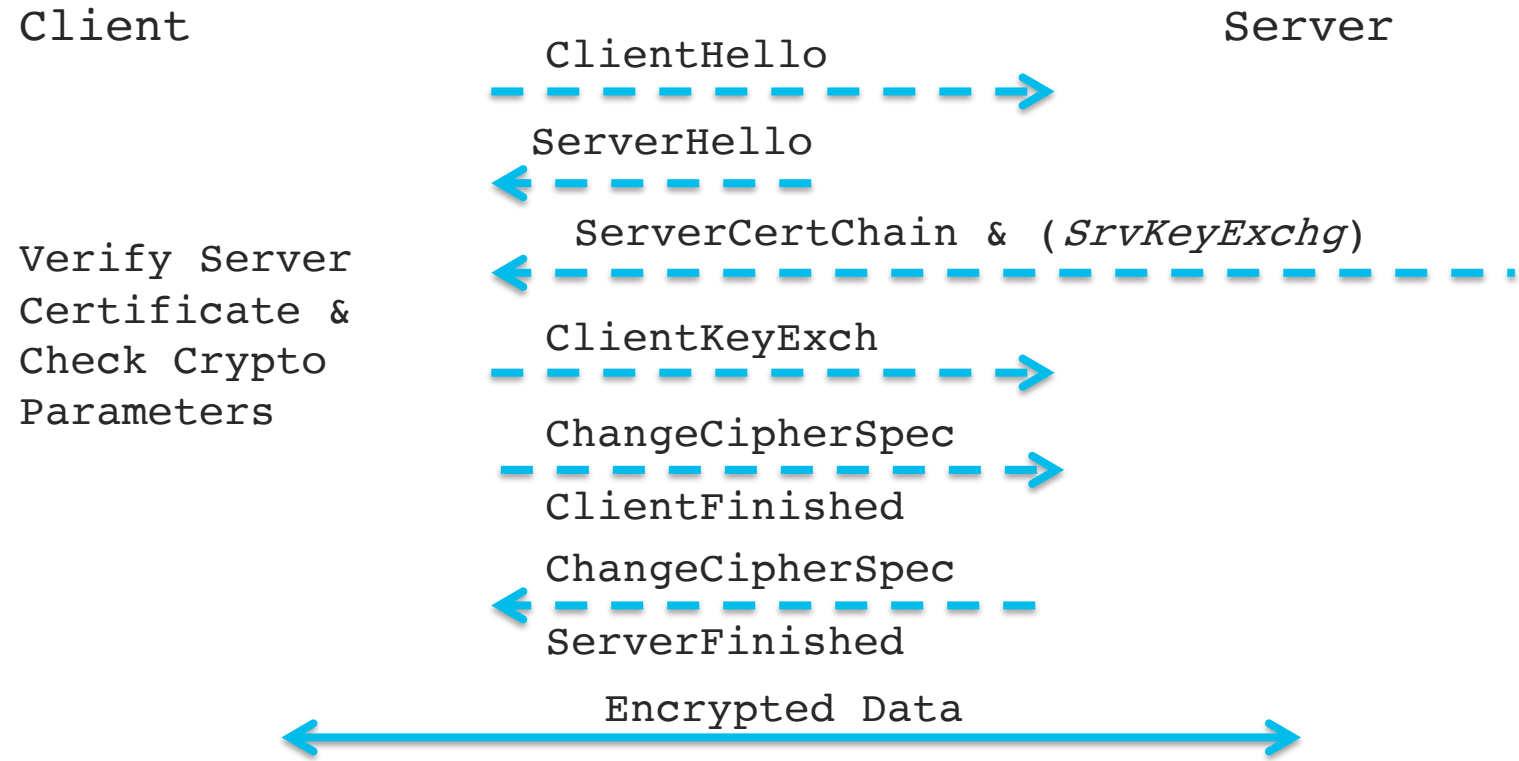
EN Certification

ork protocols

Agenda

- TLS – some stuff we need to know
- TLS Decryption Challenges with new protocols
  - Certificate pinning & HSTS
  - Impact of HTTP/2, TLS 1.3 & QUIC
- Final thoughts

TLS Handshake



Detecting the hostname…

Situation:
Hosting Provider with one external IP and several hosted domains internally

Each Domain uses its own certificate

If a client requests a connection to the IP (shared among the domains), which Server does he want to go to?

Which certificate should the server send in his Server Hello?

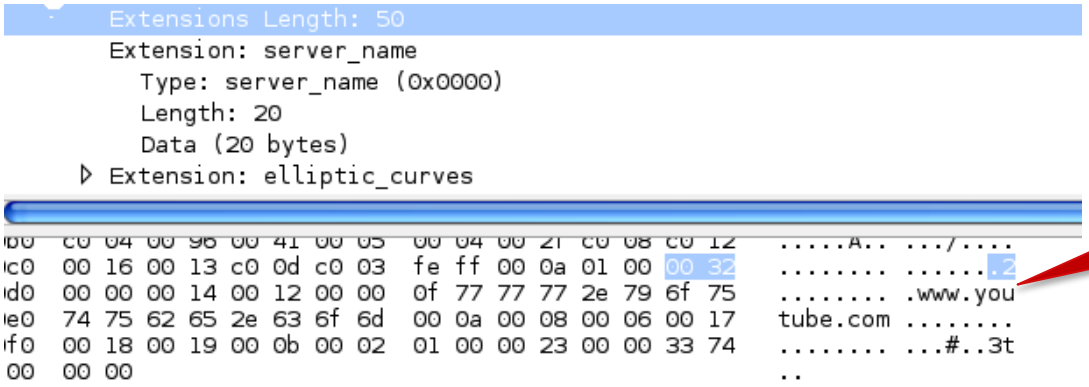
Server Name Indication

Finding out the correct destination hostname and mapping to URL Category....

Solution:
Usage of SNI (Server Name Indication) is required from Proxy side (supported in v7.7+)

Most Browser support it since many years

CLIENT HELLO during TLS sends the Host URL:



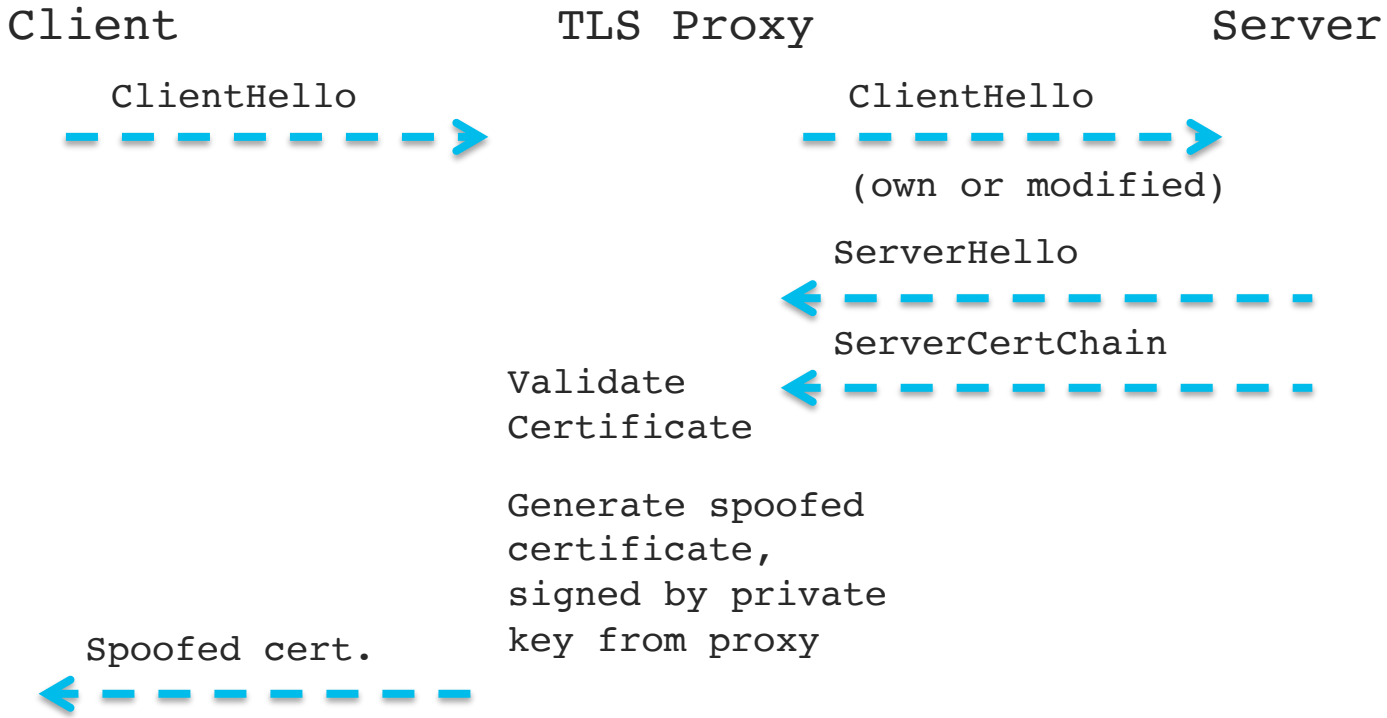
SNI = www.youtube.com

“If the traffic is encrypted, our gateways can’t see what is transferred. We need to inspect!”

Automatic reaction in most security architectures….

Breaking TLS

1. Generate pub/priv key and root cert
2. Install root cert on client device



“We are trying to break into a protocol that was not meant to be broken. Things will sometimes fail.Get used to it”

Tobias Mayer

Detecting the requested Host

The requested hostname is usually detected by one of several methods:

Transparent Request (Transparent proxy, FW)

1. Check the SNI Name in the TLS Client hello
2. Check the CN Field in the Subject of the Server Certificate

Explicit Proxy

1. Usually get the hostname from the CONNECT REQUEST
2. (Check SNI, then CN field)

HSTS

"HTTP Strict Transport Security" – http://tools.ietf.org/html/rfc6797

Protect secure HTTPS Websites against downgrade attacks

Web Server can signal to the client that only HTTPS is allowed to interact

This signal is transported using a **HTTPS** Response Header

The client behaves as follows

- Automatically turn any http:// links into https:// links
- If the secure connection cannot be assured (ex: Self Signed Certificate is used), do not allow the user to override

If you want to decrypt using a proxy, a valid CA Certificate is required!

Certificate Pinning – RFC 7469

Method to actually compare the Certificate presented from the Server to a “stored” CA Certificate on the Client. Requires a method to ensure the Client is running the latest Version of your Software

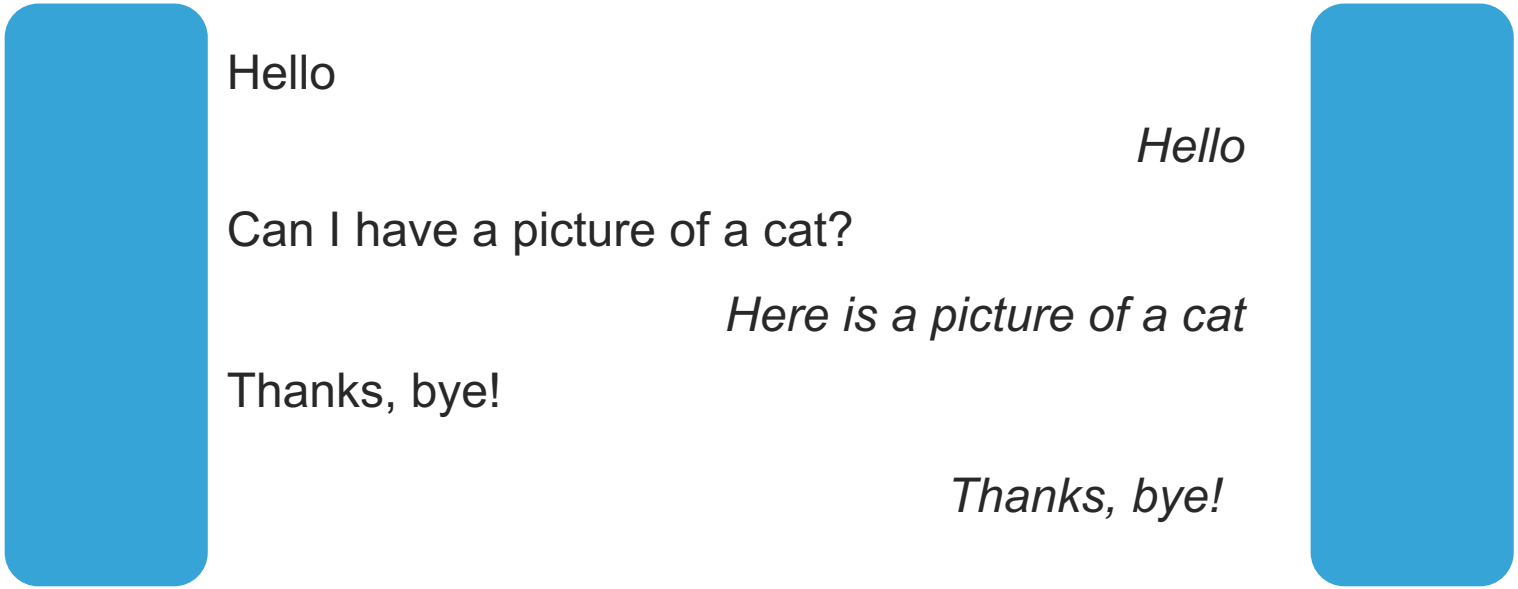
- Applies to centrally updated Applications that connect to predictable Servers

Two ways to do it:

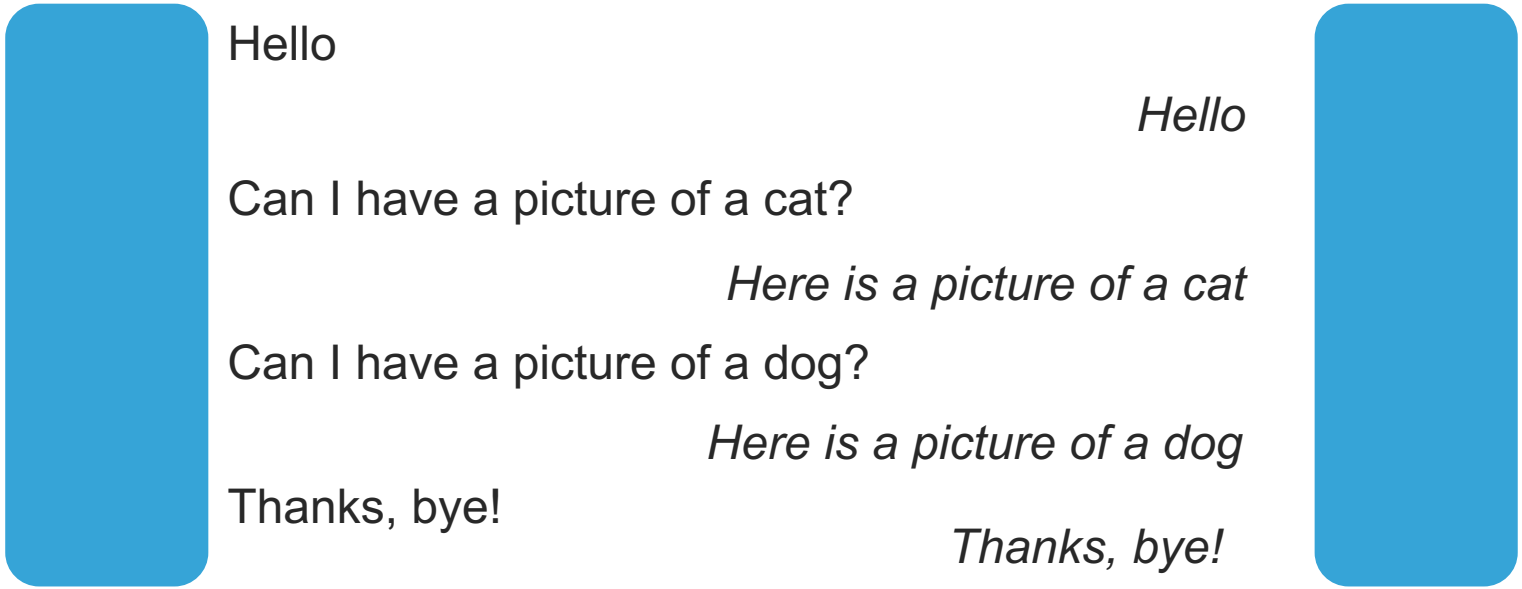
- Incorporate a static list in the application, which of the CA Certificates is expected to be used for signing the server certificate
- *Send a new “Header”(HPKP) to signal within the TLS Handshake that the client should PIN a certain public key for a certain amount of time*

Chrome connecting to gmail.com, twitter, FF connecting to mozilla.org
https://src.chromium.org/viewvc/chrome/trunk/src/net/http/transport_security_state_static.json

HTTP 1.0



HTTP 1.1



HTTP/2

SPDY & HTTP2

HTTP/2 Specification is strongly based on input from SPDY & TLS

http://daniel.haxx.se/http2/

https://www.ietf.org/blog/2015/02/http2-approved/

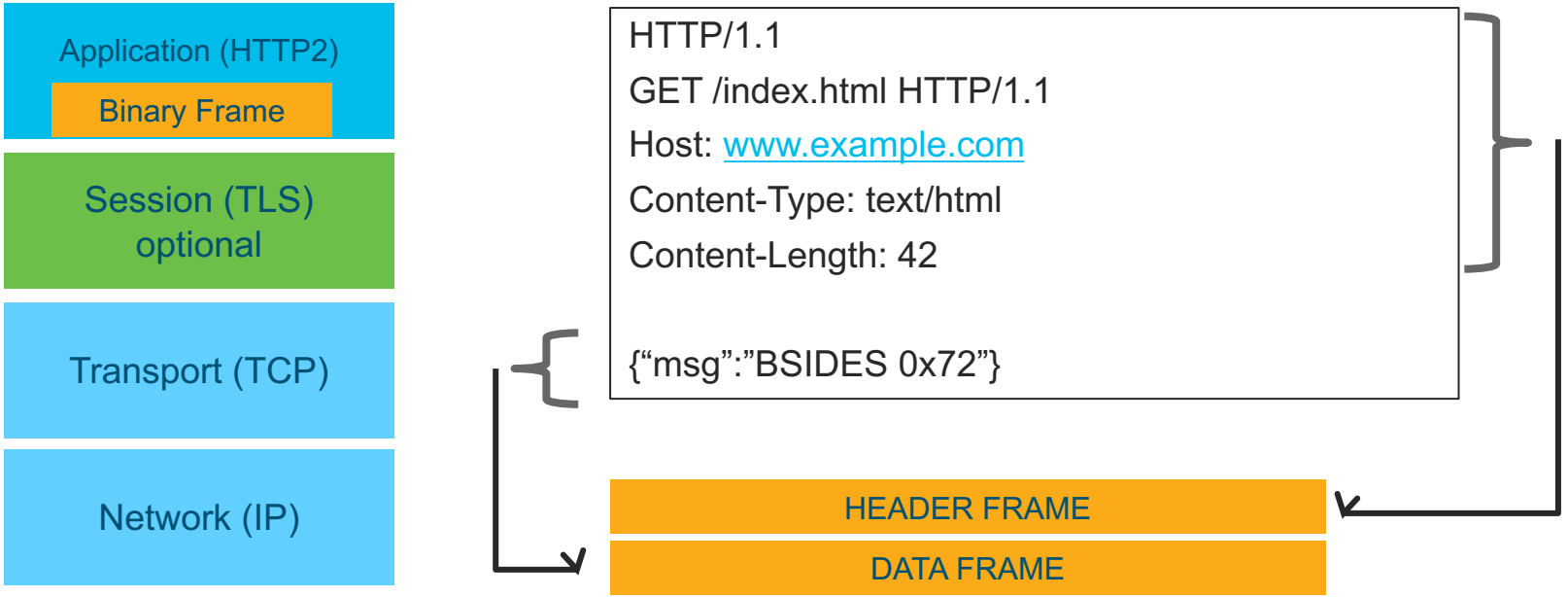
HTTP/2 – Features & Characteristics

- Header compression
- True multiplexing
- Re-Use of TCP connections

  Important with domain sharding (same ip, different domains)

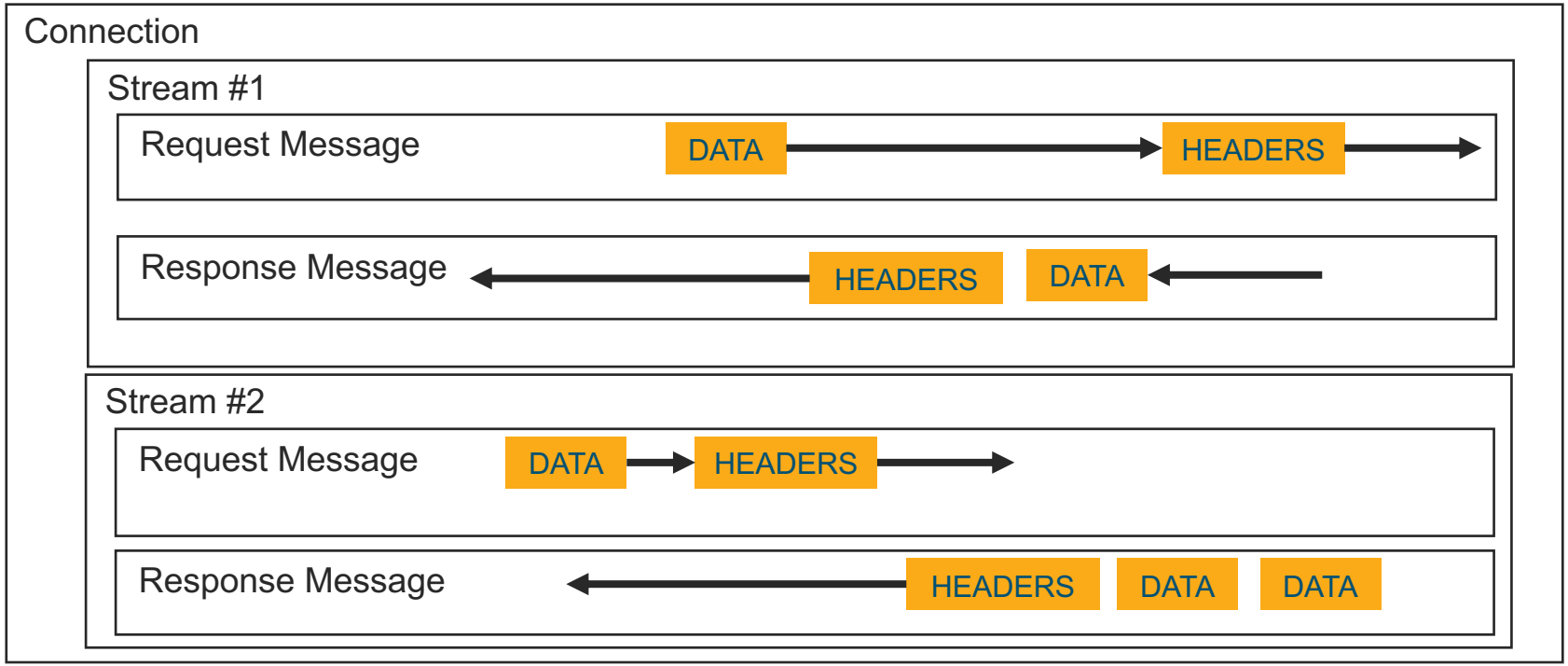
  Browser limits connection per domain -> use subdomains: i.ytimg.com for icons, s.ytimg.com for stylesheets
- Server pushing content to the client
- Prioritization of streams
- Binary Format

HTTP2 – Binary Format

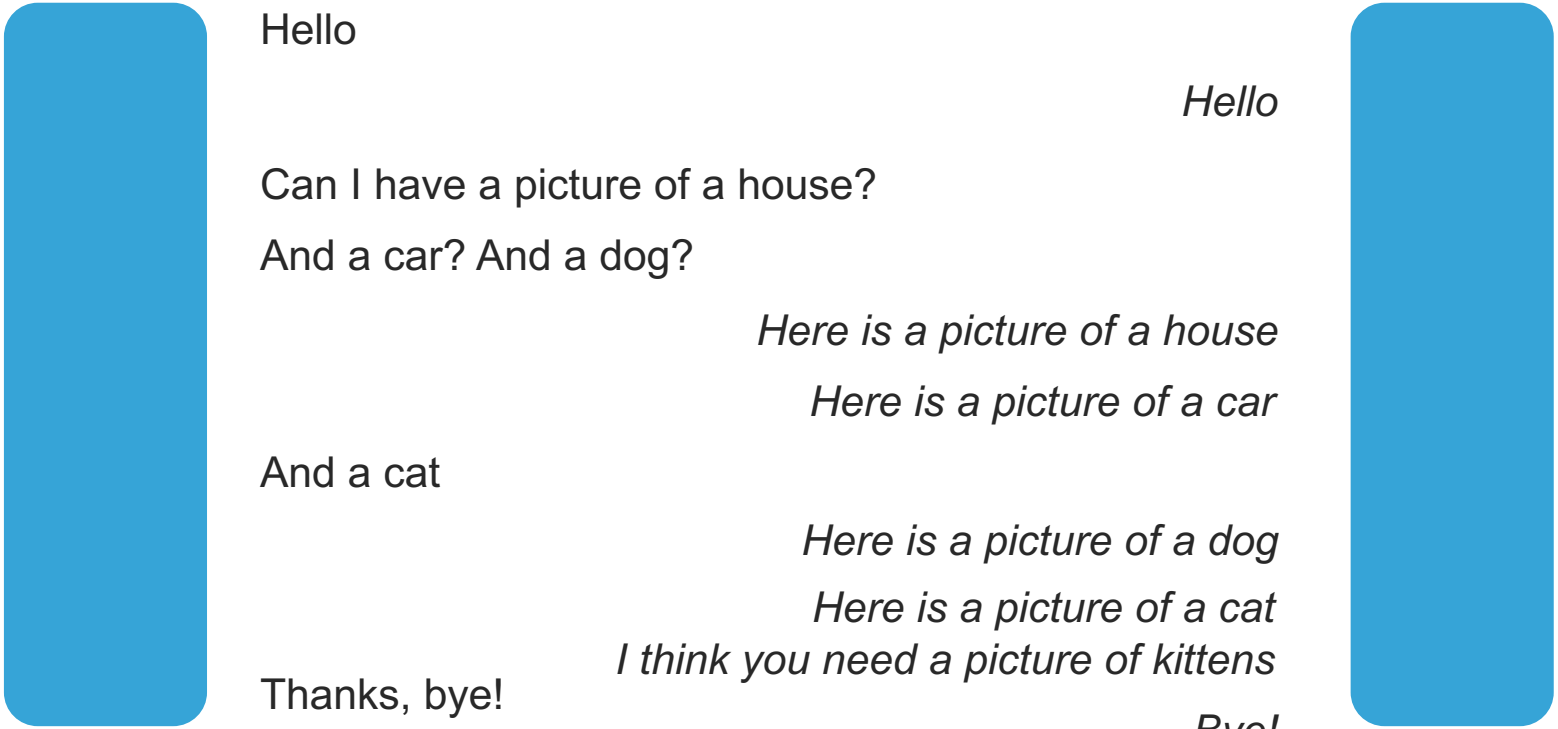


Pro: Easier to parse, more efficient use of data transfer
Con: Server, Client & Gateways need to understand the new format

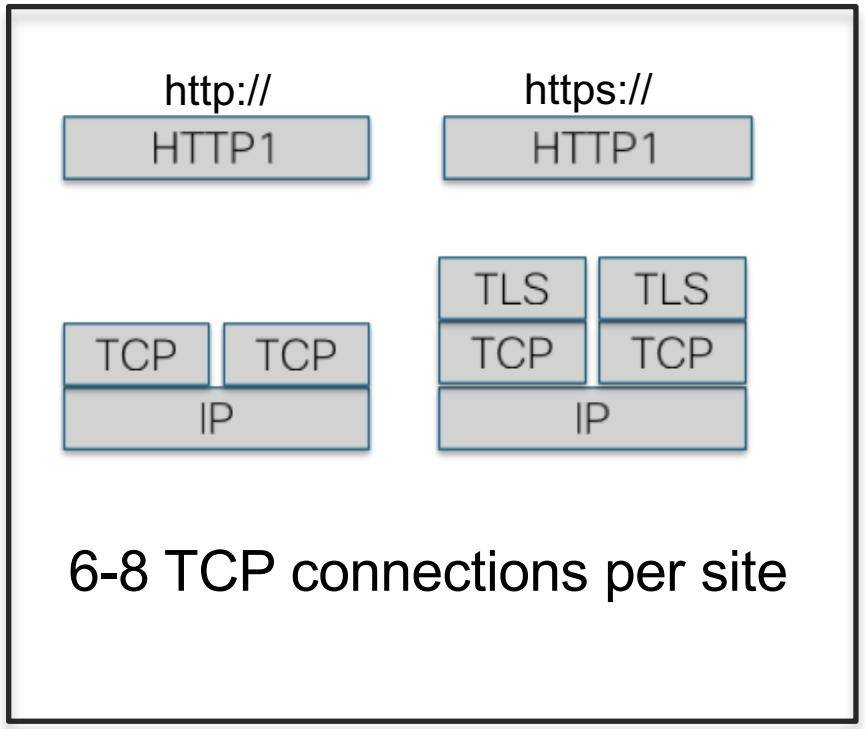
HTTP/2 – Streams & Connections

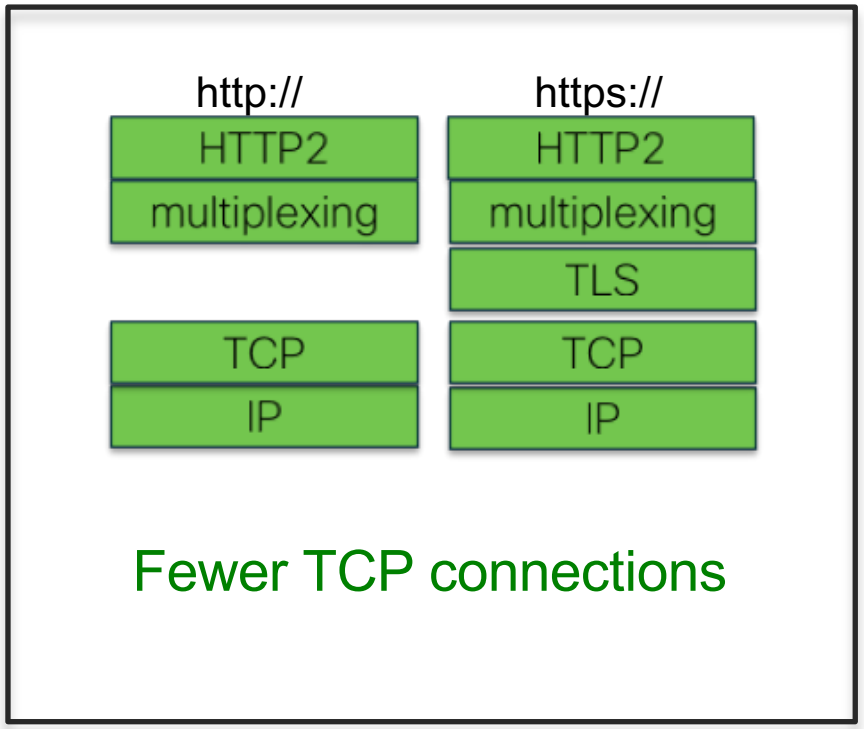


HTTP 2.0



HTTP, HTTPS, and HTTP2 Layering





Upgrade to HTTP/2

No Change in URL Structure (“http://...”,”https://...”)

Browser and server need to agree which protocol to use

Non-Secure

- Client: send “Upgrade” Header, requesting Upgrade to “H2C”
- Server: respond either with HTTP/1.1 OK (=upgrade not accepted) or HTTP 101 SWITCHING PROTOCOLS (=upgrade accepted)

Secure

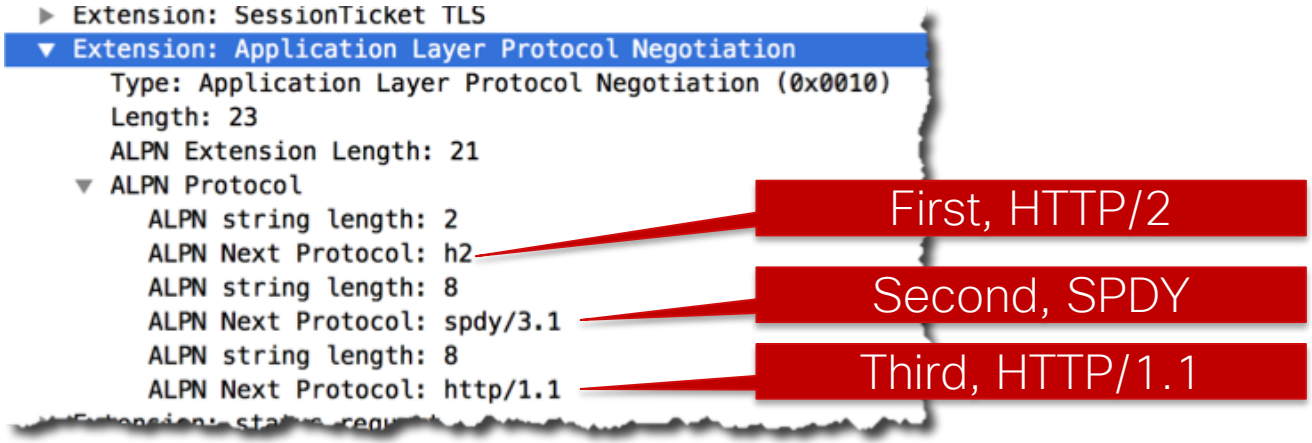
- Leveraging ALPN Extension in TLS Client Hello

HTTP/2 Negotiation over TLS

Application Layer Protocol Negotiation (“ALPN)

Client will “offer” its supported protocols and server will pick one of them

TLS Extension called ALPN as part of the TLS Client Hello



HTTP2 and TLS

Multiplex requests and responses over single TCP connection

- More efficient object retrieval
- One TCP connection to each server (avoids TCP & TLS setup delays)

All browsers only attempt HTTP2 over TLS

- Chrome, Firefox, Safari
- Avoids difficult fallback code (like was necessary with HTTP 1.1 and middleboxes)
- Upgrades to HTTP2 using TLS extension
  - Saves round trip of using HTTP's "Upgrade:" header

Page load time: **HTTP2-over-TLS** is equivalent to **(plaintext) HTTP**

- Eliminates TLS page load time penalty

http://caniuse.com/#feat=http2

HTTP/2 Facts

Pages loading faster

More usage of TLS

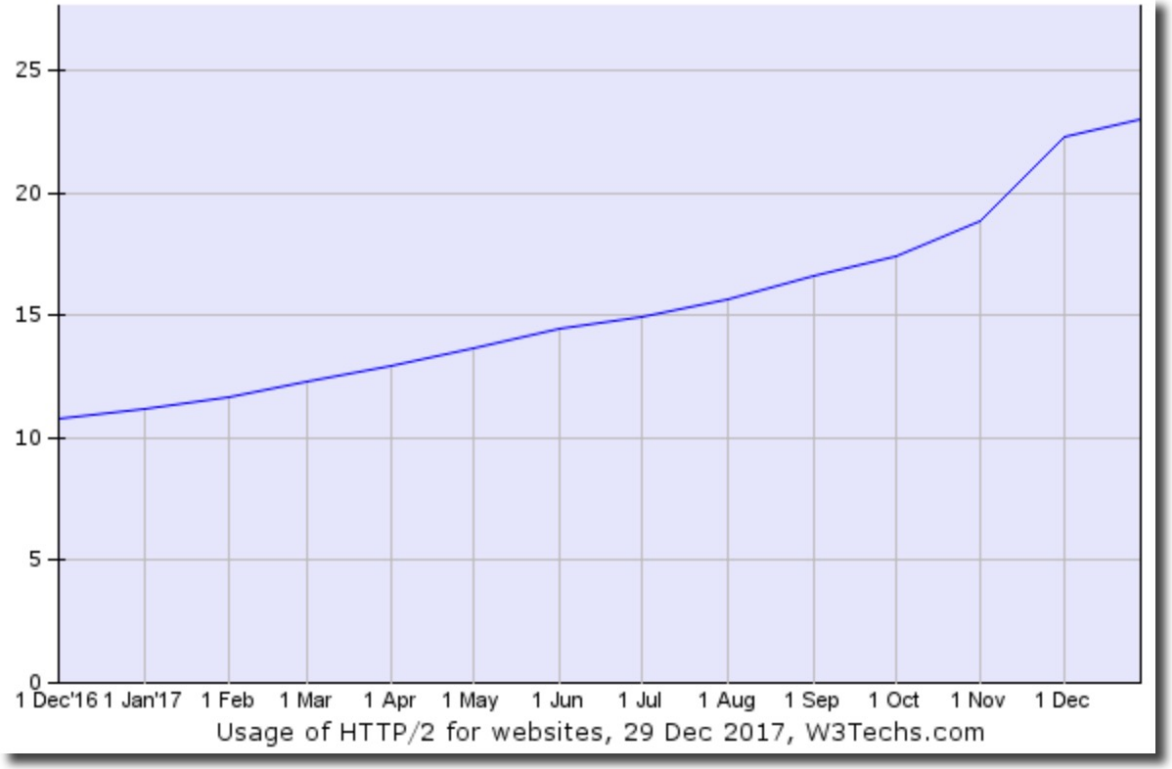
Less TCP connections but longer lifetime

HTTP/2 growing fast, websites change from SPDY to HTTP/2

SPDY has proven that a new protocol can grow fast

HTTP/2 Uptake

http://w3techs.com/technologies/details/ce-http2/all/all

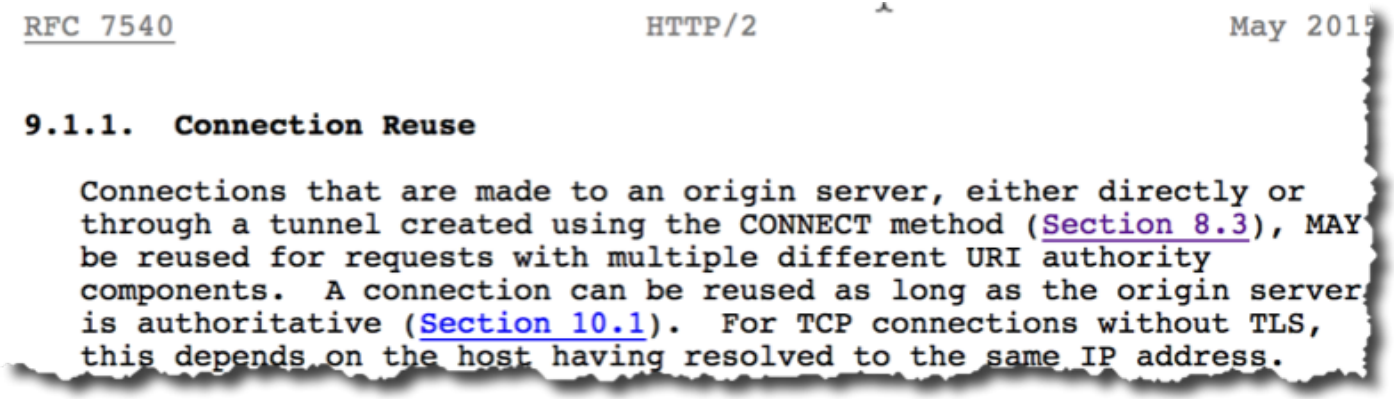


HTTP/2 challenges on proxy/ngfw/<insert nw-device here>

HTTP/2 encrypted with TLS

Binary Format & Header Compressions
need to be parsed (no more cleartext)

Single TCP connection reuse (will the gateway still filter?)



RFC 7540 HTTP/2 May 201

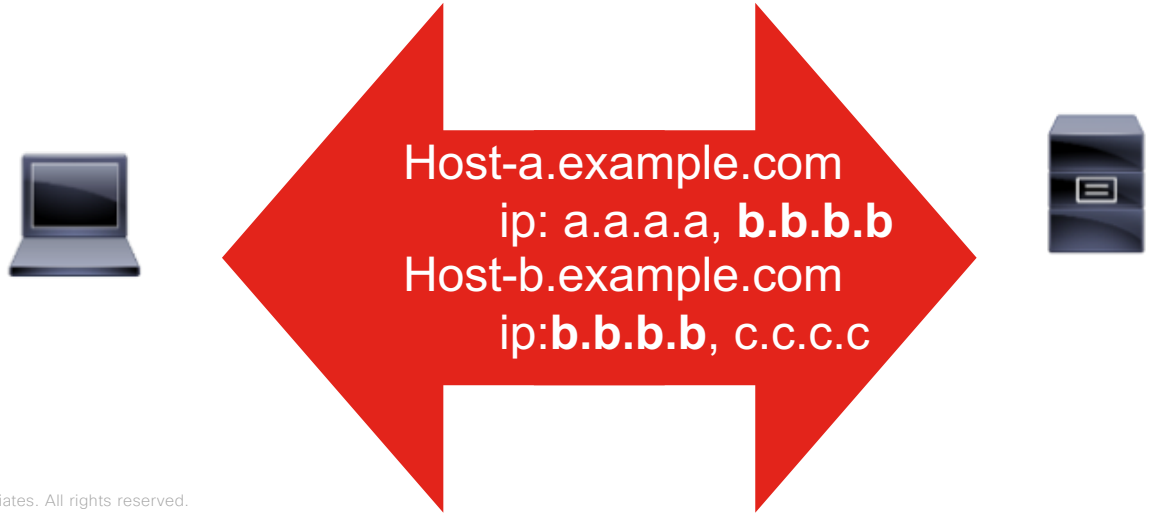
9.1.1. Connection Reuse

Connections that are made to an origin server, either directly or through a tunnel created using the CONNECT method (Section 8.3), MAY be reused for requests with multiple different URI authority components. A connection can be reused as long as the origin server is authoritative (Section 10.1). For TCP connections without TLS, this depends on the host having resolved to the same IP address.

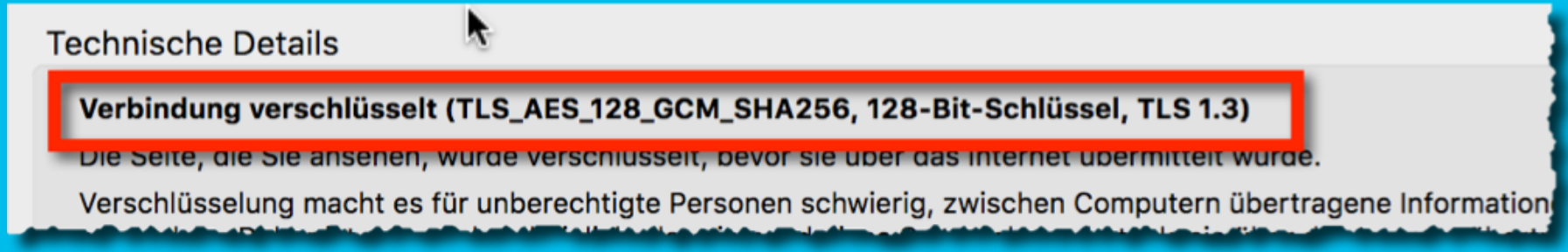
HTTP/2 challenges on proxy/ngfw/<insert nw-device here…>

Single TCP connection reuse (use same TCP connection for different hostnames but with same ip)

- Domain must be in certificate SAN and resolve to same IP



TLS 1.3


Technische Details
Verbindung verschlüsselt (TLS_AES_128_GCM_SHA256, 128-Bit-Schlüssel, TLS 1.3)
Die Seite, die Sie ansehen, wurde verschlüsselt, bevor sie über das Internet übermittelt wurde.
Verschlüsselung macht es für unberechtigte Personen schwierig, zwischen Computern übertragene Information

TLS 1.3 – draft

https://tools.ietf.org/html/draft-ietf-tls-tls13-23

Remove of static RSA authentication mode

- Using DHE / ECDHE instead for PFS

Reducing overhead by using a 1-RTT handshake

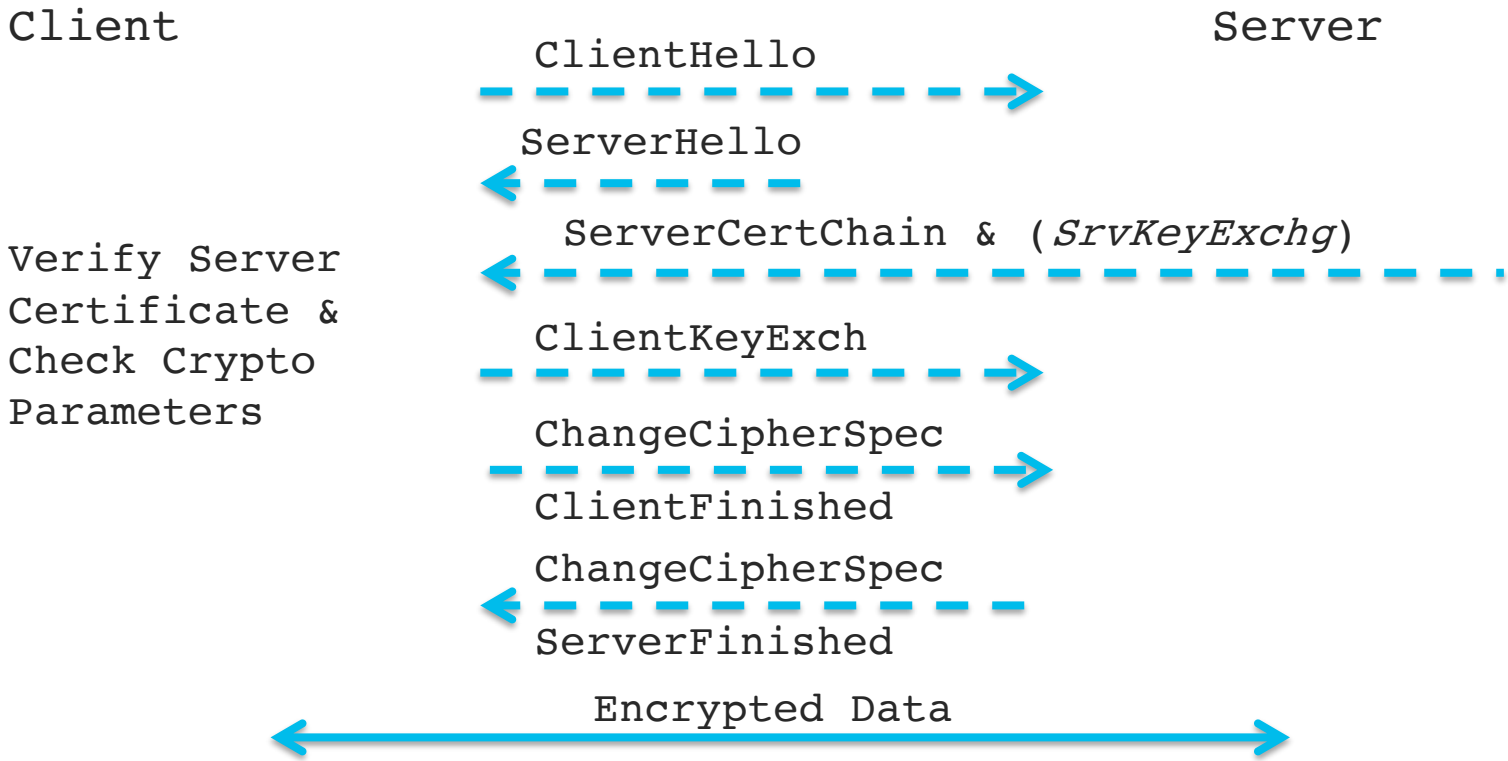
- Fallback to “legacy” handshake if client cannot handle it
- 0-RTT Session resumption -> Tickets + PSK

Remove non-AEAD Ciphers (CBC), compression, RC4, MD5, SHA224

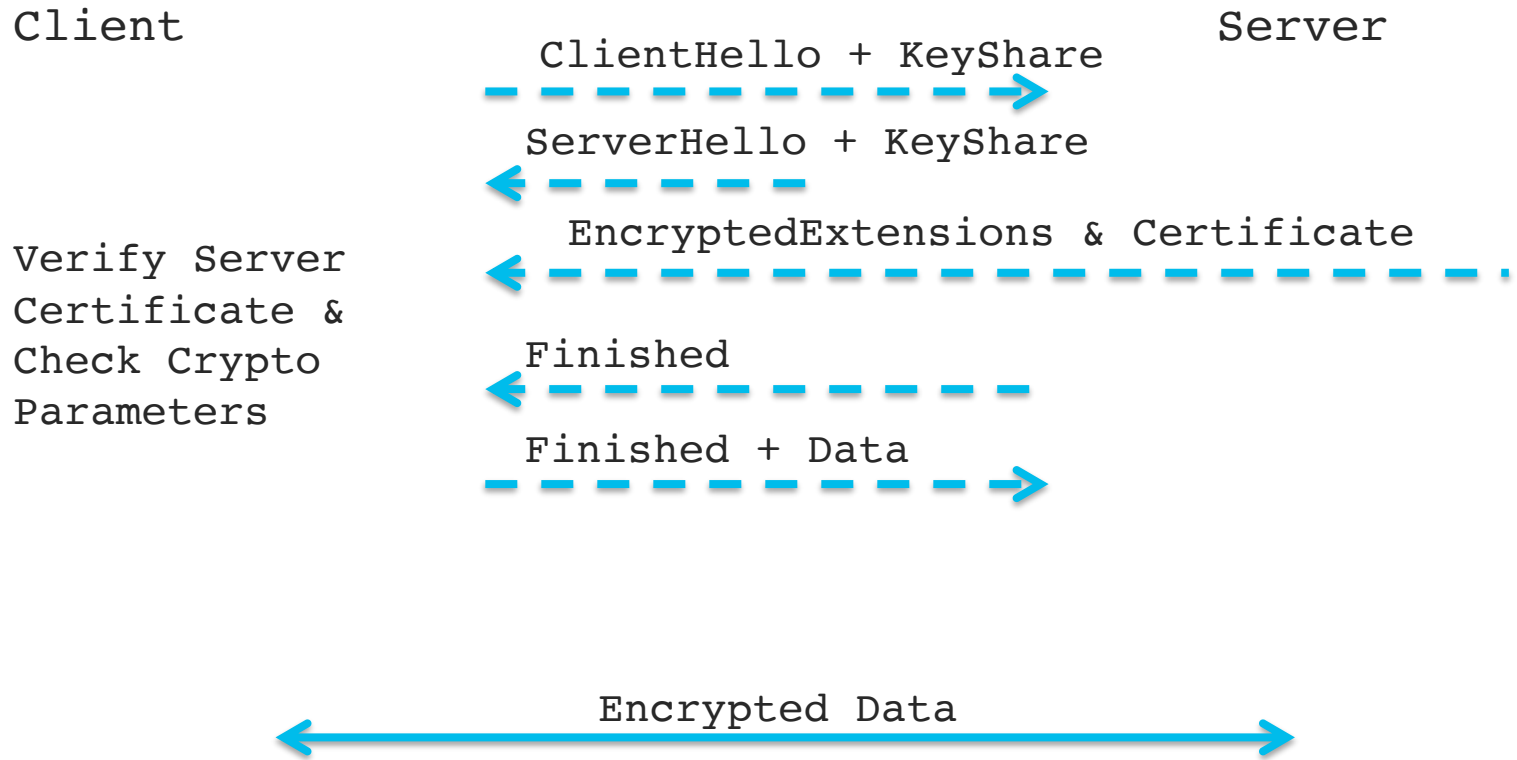
Encrypting more values in the handshake

- Certificate Extensions such as CN & SAN

TLS Handshake 1.0 – 1-2



TLS 1.3 Handshake



Modification in TLS 1.3 Client Hello

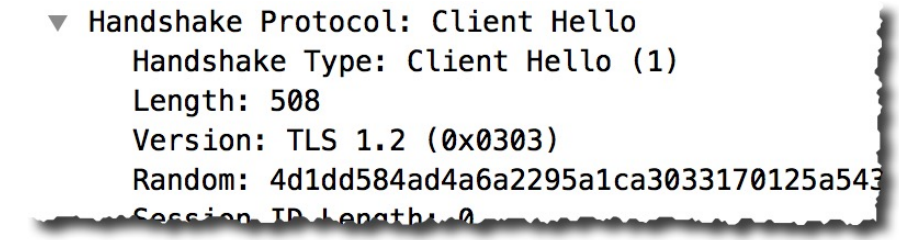
TLS 1.3 was breaking some Security Gateways due to unsupported extensions and ciphers

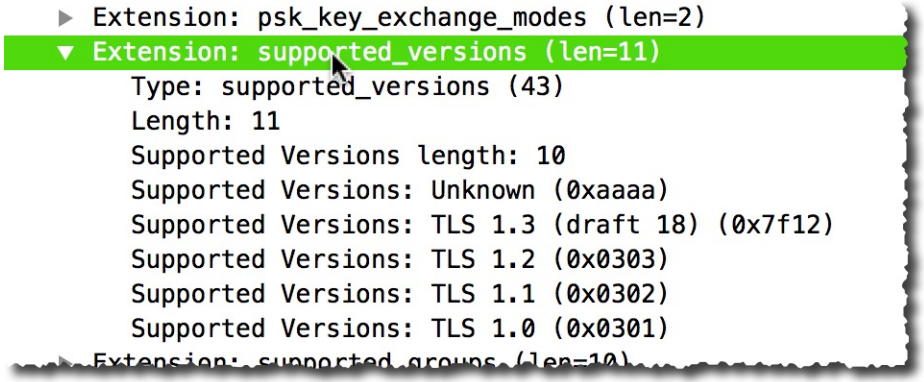
Decission of IETF:

- Signal TLS 1.2 in the “Version” field
- Signal TLS 1.3 in an additional extension

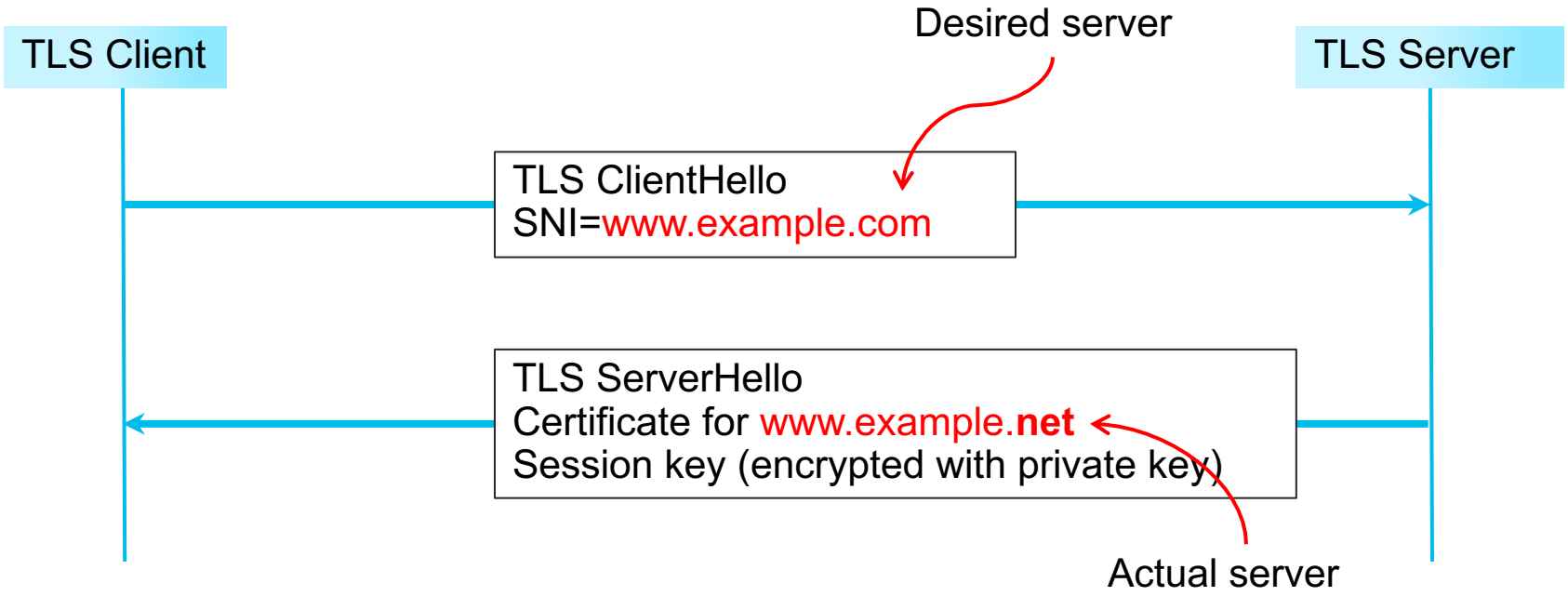
Server who don’t understand extension will negotiate TLS 1.2

Server who understand TLS 1.3, will ignore the version field and negotiate TLS 1.3



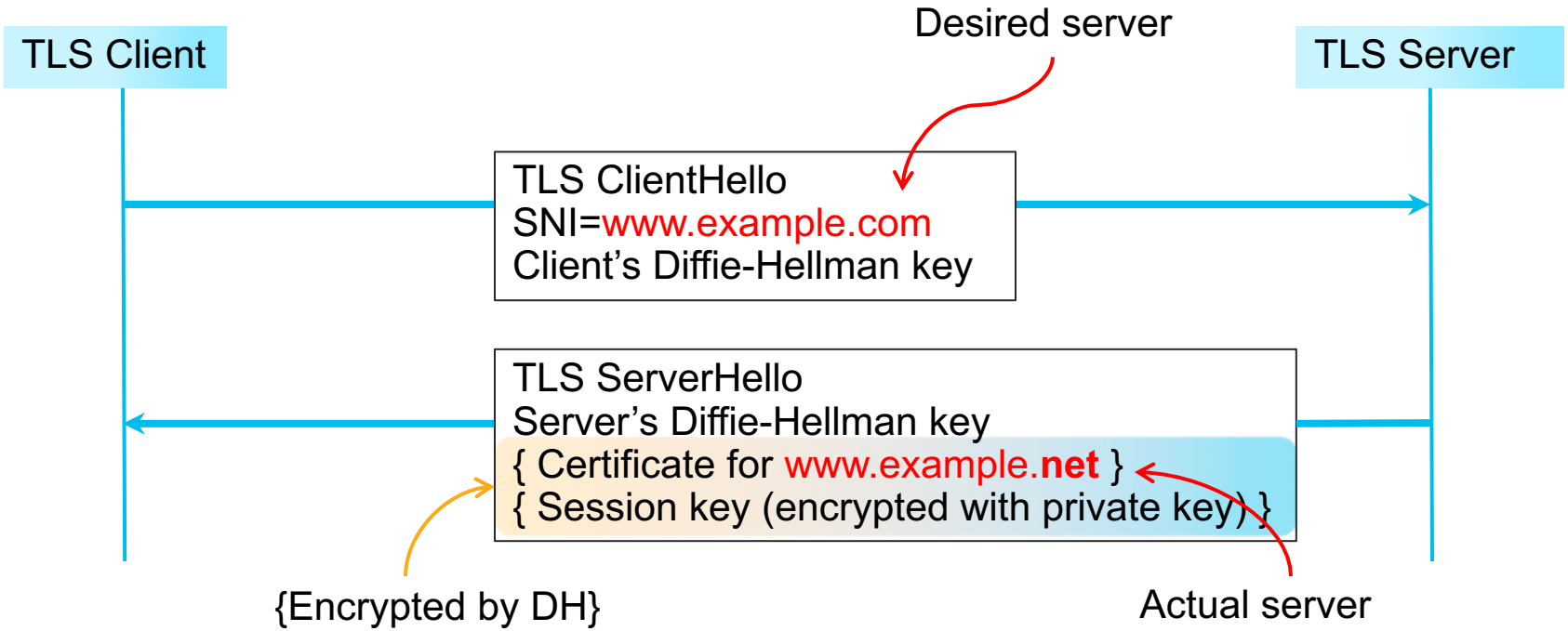


Partial TLS Handshake (TLS 1.0 – 1.2)



Server certificate can avoid decrypting if entire site is blacklisted or whitelisted

Partial TLS Handshake (TLS 1.3)



Detecting the requested Host (recap)

The requested hostname is usually detected by one of several methods:

Transparent Request (Transparent proxy, FW)

1. Check the SNI Name in the TLS Client hello
2. ~~Check the CN Field in the Subject of the Server Certificate~~

Explicit Proxy

1. Usually get the hostname from the CONNECT REQUEST
2. Check SNI ~~, then CN field~~

TLS 1.3 challenges on proxy/ngfw/<insert nw-device here…>

CN & SAN extensions are encrypted by DH

We can only rely on SNI to know if we should decrypt or not

SNI can be spoofed…

Bypassing the decrypting device

- What if we send a “Fake” SNI name?
- Example:
  Request sent to Website A but indicating a request to Website B ?
- Reason:
  - Browse to forbidden websites (boring…)
  - Malware connects C&C through decrypting device (better reason…)

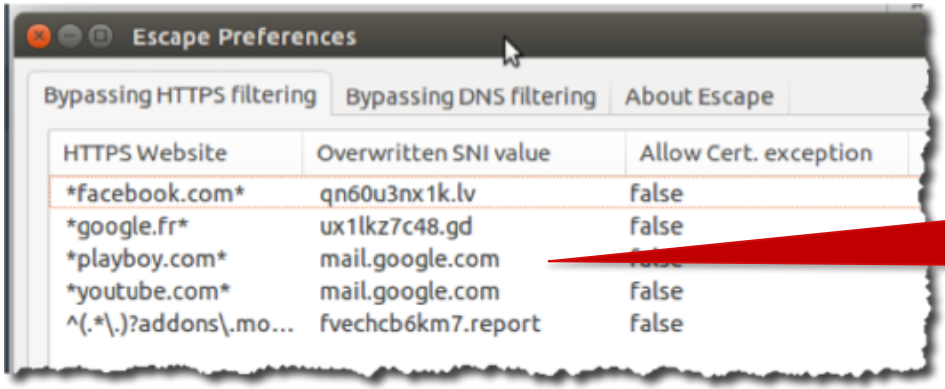
Testing

Usage of Firefox plugin called “Escape”

http://madynes.loria.fr/Research/Software

Works in UBUNTU Linux

Can override certain websites with a fake SNI name



Overriding requests for “playboy.com” with “mail.google.com”

Result of spoofed SNI strings

Logging shows wrong URL Category

Application detection shows wrong application

Selection of correct policy (decrypt/pass through) is impacted

Possible Countermeasures (thoughts…)

- Check for incorrect hostname
- Enforce policies over DNS (reduce attack surface)
- Use native IPv6 without any IP address sharing ☺

QUIC

Kate Pearce

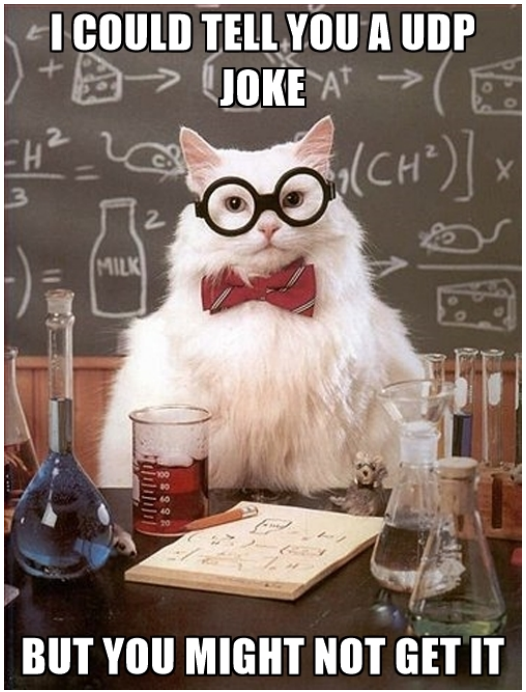
https://www.blackhat.com/docs/us-16/materials/us-16-Pearce-HTTP2-&-QUIC-Teaching-Good-Protocols-To-Do-Bad-Things.pdf

UDP vs TCP

Connection setup in TCP plus connection setup with TLS

Big latency until Data is flowing

Move to UDP for faster Session Setup

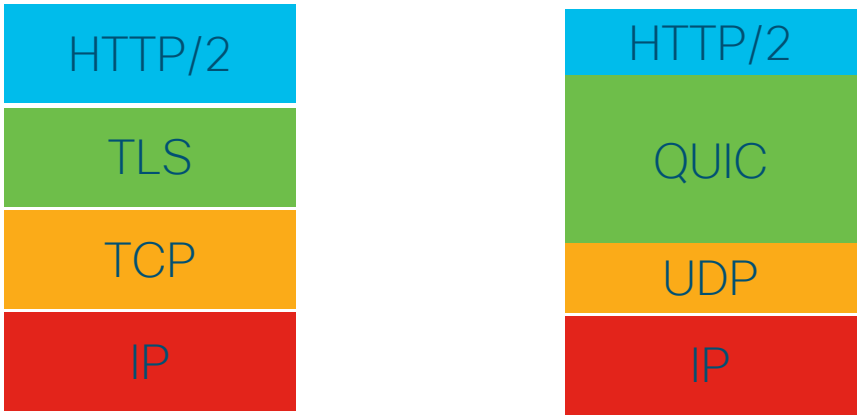


QUIC

Google championed protocol to reduce latency

UDP 80 & 443

Encryption, congestion control and some HTTP/2 things (stream handling) move to QUIC



QUIC Features

1-RTT connection handshake (Connection & Encryption negotiation)

0-RTT re-established connections

Connections survive IP address change

- Connection Identified through a unique CONNECTION UUID
- Packets can arrive in any order

Always encrypted and authenticated

Mostly fixes head of line blocking

FEC (Forward Error Correction) data recovery

- *Additional Data is transferred to eventually recreate missing packets (RAID for the network)*
- *Removed from IETF Specification Work as it had minimal positive effect*

HTTP/2 Features in QUIC

Multiplexed streams

Sharing connection across domains

HPACK header compression

Stream prioritization

Flow Control

Server initiated streams

Establishing a QUIC Connection

If using TLS, the HTTP response header will be encrypted…

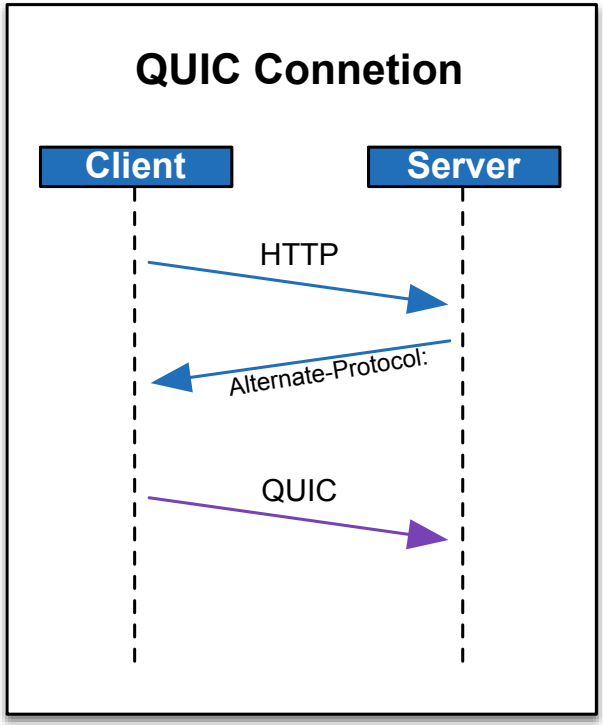
HTTP response header

Alternate-Protocol: 443:quic

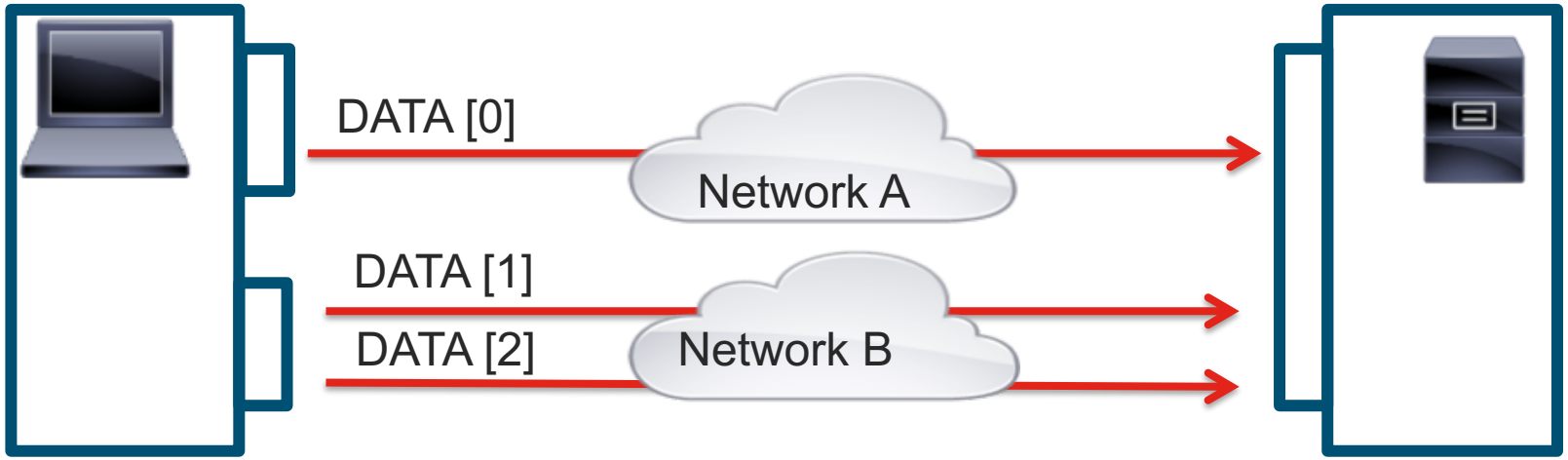
Client establishes QUIC connection in the background

Fully supported in Chrome browser

Client's can cache if server supports QUIC

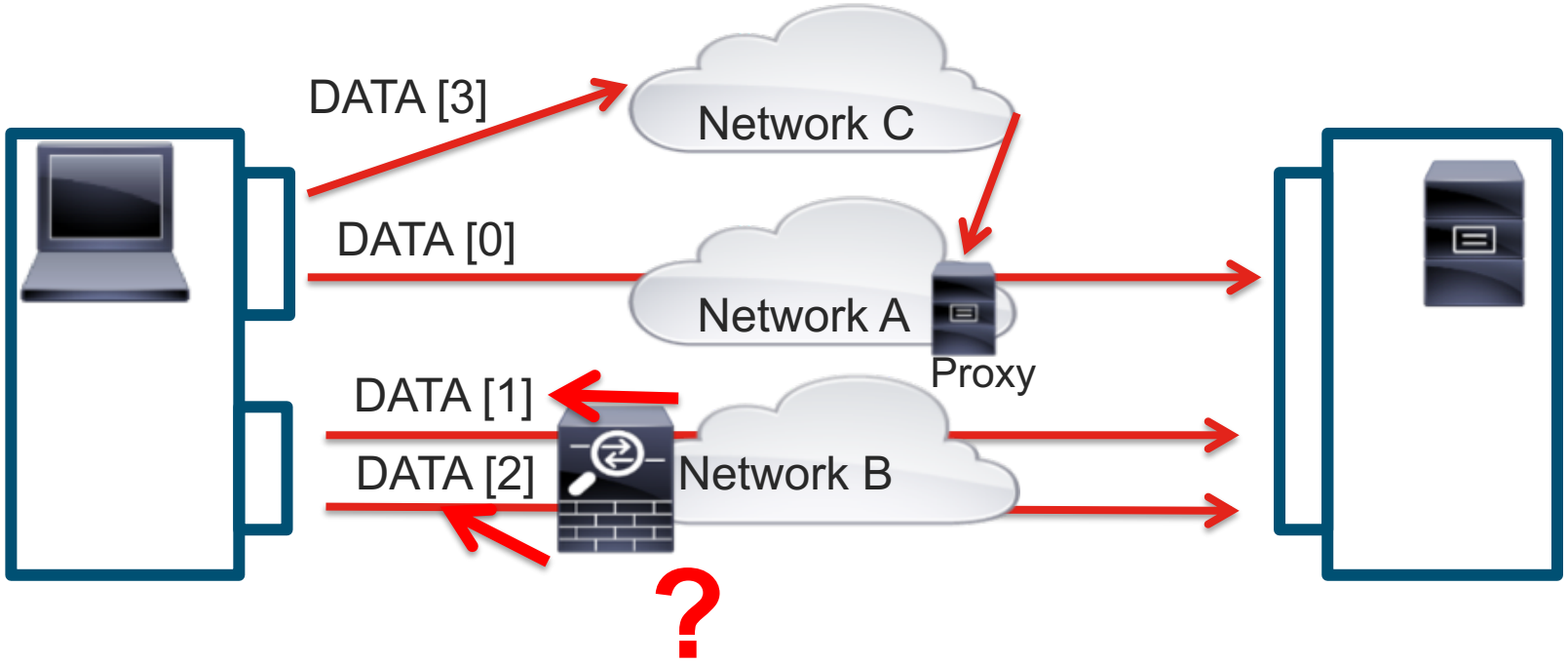


Recap: Multipath TCP

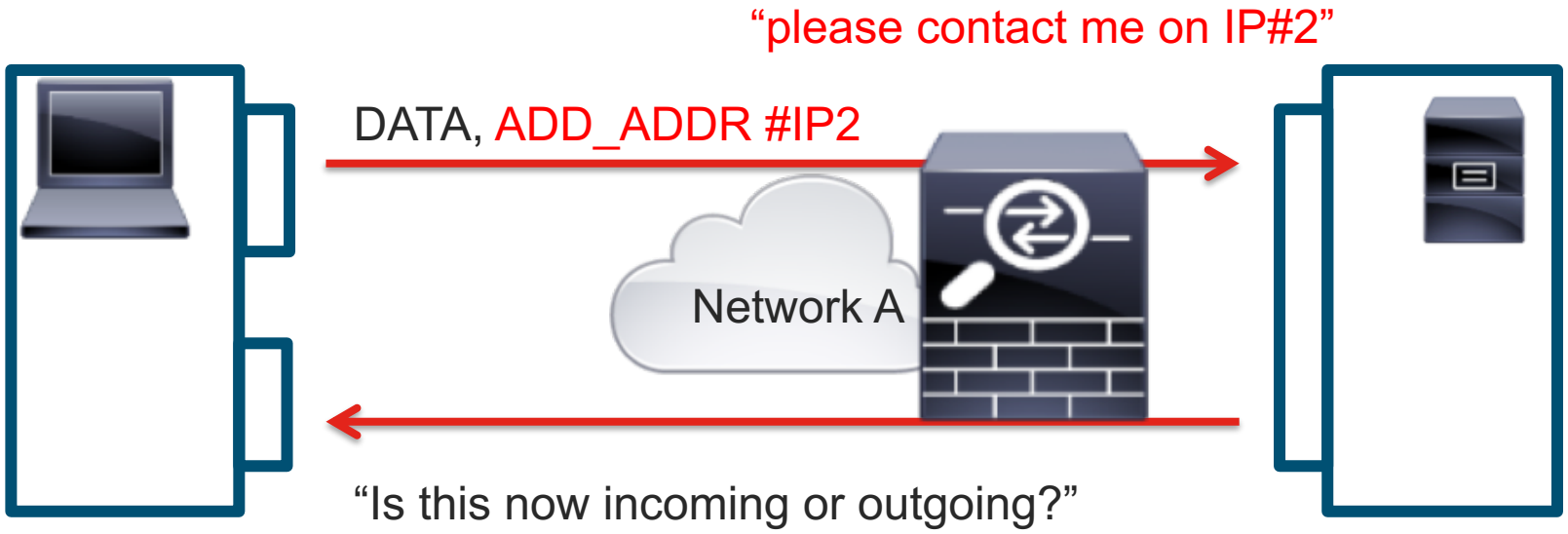


N TCP Streams contributing to ONE logical flow
Connections can be added and brought down dynamically

mTCP – IPS, NGFW challenges



mTCP – IPS, NGFW challenges (2)



mTCP – IPS, NGFW challenges (3)

QUIC Multipath

Google QUIC does not yet use Multipath….
….but IETF Working Group is discussing it:
https://datatracker.ietf.org/meeting/99/materials/slides-99-quic-sessb-first-experiments-with-multipath-quic

Flows can be distributed among several UDP connections

Can be established or torn down dynamically

Can go different paths in the network

Can use IPv4 and IPv6 on different connections

QUIC Performance

5% latency reduction on average

30% reduction in rebuffers (video pauses) on YouTube

1 second faster at the 99th percentile for Google web search

Helps more for higher latency networks

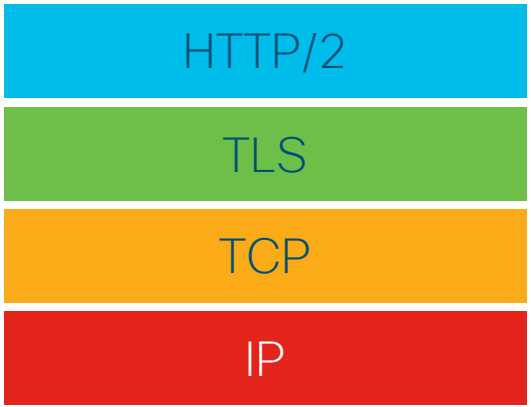
Working group within IETF has been created
https://peering.google.com/#/learn-more/quic

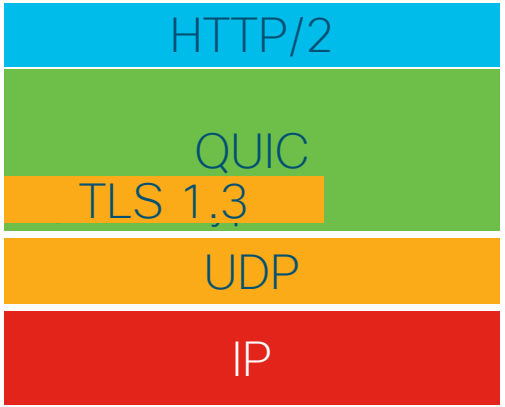
QUIC working group

Map HTTP cleanly to QUIC, make non-HTTP apps work with QUIC

Use TLS 1.3 within QUIC

https://www.ietf.org/proceedings/98/slides/slides-98-edu-sessf-quic-tutorial-00.pdf





QUIC challenges on proxy/ngfw/<insert nw-device here…>

QUIC is always encrypted, and currently can‘t be decrypted

QUIC is using multiplexed streams and most likely soon also accross multiple paths

- Will happily use IPv4 and IPv6 concurrently
- If QUIC is not understood, connections look like unrelated UDP connections

QUIC can be initiated from client and from server

- Where is now inbound and where is outbound?

Conclusion

Majority of Internet Traffic is already encrypted

Intercepting gateways are only partially decrypting

Intercepting devices that don't understand the mentioned protocols might just downgrade to known protocols -> blocking innovation

Alternative solutions need to considered to secure your network traffic

- Endpoint Security
- Anomaly detection via Netflow and/or log analysis
- DNS based Security


CISCO